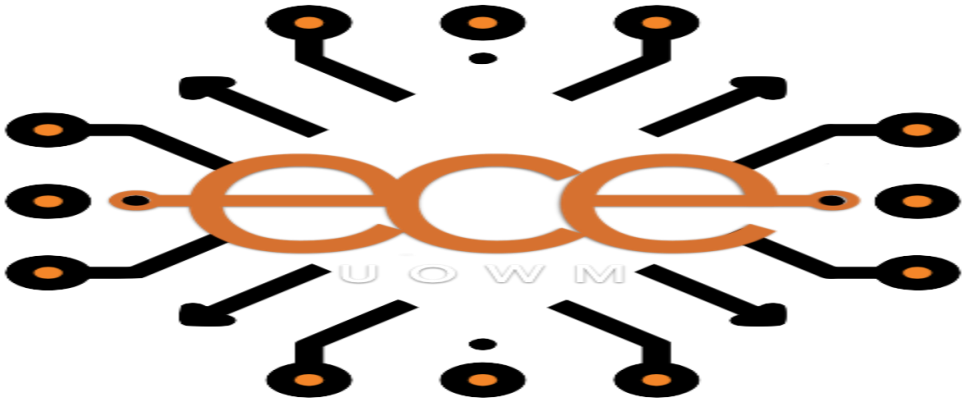


# ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

# Τεχνικές ανίχνευσης και απόκρισης κυβερνοαπειλών για ηλεκτρικά οχήματα εξωτερικής τροφοδοσίας

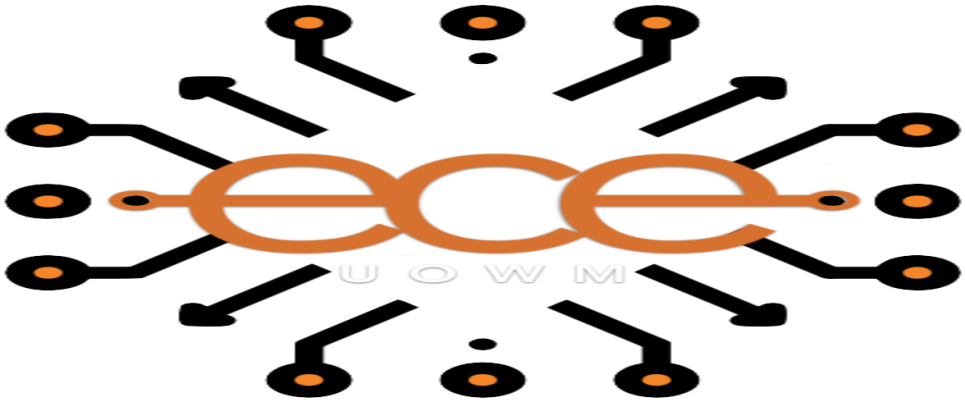
---


**ΠΑΝΑΓΙΩΤΗΣ ΚΟΥΡΤΖΕΛΛΗΣ 519**

**Επιβλέπων Καθηγητής: Επίκουρος Καθηγητής, Παναγιώτης Σαρηγιαννίδης**



**ΚΟΖΑΝΗ, ΟΚΤΩΒΡΙΟΣ 2020**




# DIPLOMA THESIS

# Cyber threat Detection and Response Techniques for Plug-in Electrical Vehicles

**PANAGIOTIS KOURTZELLIS 519**

**Supervisor: Assistant Professor, Panagiotis Sarigiannidis**

**KOZANI, OCTOBER 2020**

# Δήλωση Πνευματικών Δικαιωμάτων

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο ΄΄Cyber threat Detection and Response Techniques for Plug-in Electrical Vehicles΄΄ καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Σαριγιαννίδη Παναγιώτη, αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.



Υπογραφή Φοιτητή:

# Thanks

At the end of this dissertation, I would like to thank Prof. Panagiotis Sarigiannidis and the rest of the teaching staff for all the advice and help they have given me throughout its implementation.

Many thanks to my friends Basilis, Dimitris, Jason and Stratos for their support over the years. Finally and most important of all, I would like to thank my parents for always being there for me.

## Table of Contents

# Table of images

# Table of code snippets

| Acronyms | Descriptions |
|---|---|
| **BMS** | Battery Management System |
| **EV** | Electric Vehicle |
| **EVSE** | Electric Vehicle Supply Equipment |
| **WAN** | Wide Area Network |
| **ML** | Machine Learning |
| **ICT** | Information and Communications Technology |
| **PHEV** | Plug-in Electric Vehicle |
| **PWM** | Pulse Width Modulation |
| **KW** | Kilowatts |
| **KWh** | Kilowatts Per Hours |
| **RES** | Renewable Energy Sources |
| **DSO** | Distribution System Operator |
| **eMSP** | Electro Mobility Service Provider |
| **CPO** | Charge Point Operator |
| **CP** | Charging Point |
| **OCPP** | Open Charge Point Protocol |
| **CDR** | Charge Detail Record |
| **MV** | Meter Value |
| **AI** | Artificial Intelligence |
| **BST** | Binary Search Tree |
| **SVM** | Support Vector Machine |

# Abstract

One of the biggest problems of modern society is considered the air pollution, which has led to the discovery of alternative energy sources in combination with other factors, such as the reduction of natural resources due to their unbridled and uncontrolled exploitation. The interim solution of using catalysts to reduce pollution proved insufficient, because it simply limited the problem without leading to its final solution. The electric vehicle ensures zero emissions and frees users from dependence on liquid fuels, the sharp rise in prices and all kinds of shortages due to crises.

The various vehicles, on a case-by-case basis, are equipped with the corresponding fuels to power their engines from each of the refueling stations. Therefore, to charge electric vehicles, charging point stations are needed to supply electricity. Their supply, as mentioned, requires their connection to some kind of electricity network infrastructure. The large area of the electricity grid offers many options for potential charging facilities.

This thesis examines threats and cyber-attacks in the charging process and the targeting Battery Management System (BMS) and other parts of the car's electronics' system. More specifically, this thesis will examine whether the charging station hardware can be hacked in order to send these erroneous signals (either locally or remotely) and how the charging stations can be made tamper-proof and how cyber-attacks can be detected.

Charging Point Stations have many functions, such as, providing and controlling the energy to the Electric Vehicle (EV) using the Electric Vehicle Supply Equipment (EVSE) component, collecting the measurements from the meter for each charge of an Electric Vehicle, identifying and authorizing EV users via user authentication component, enabling remote capabilities (e.g., adjustment of the maximum current allowed) to the Charge Point via the local Controller component over the Wide Area Network (WAN).

The protection of the national electric grid should become a priority for all the organization/entities that are getting engaged in the EV ecosystem. The output of this thesis is aiming at increasing the cyber security of a standard EV charging enterprise's platform through the integration of Machine Learning (ML) techniques for identifying anomalies in the charging patterns, and therefore minimize the exposure both enterprises' database and the stability of the electric grid. The thesis covers both the Information and Communications Technology (ICT) and the electric engineering domain on an effort towards increasing the cyber security on what is called Energy Internet.

In the implementation part of this thesis, we will use dataset in CSV format obtained from a standard EV charging enterprise's database to apply anomaly based algorithm, in order to discover if any abnormal functions of charging happens. Many different evaluation methods will be applied in order to ensure high quality to the findings of the ML techniques. The applied evaluation methods will contain qualitative (visual inspection, manual investigation) metrics offering a validation framework wide enough to cover different aspects of cyber security in the area of EV smart charging.


## Keywords:

- electric vehicles,
- electric vehicles smart charging,
- electric vehicle supply equipment,
- anomaly detection,
- machine learning,
- unsupervised algorithms,
- supervised algorithms

# Chapter 1 - Introduction

## 1.1. Research Problem

The evolution of car and battery technology has now made electric propulsion a tangible reality, which is radically changing car data. Addressing the major environmental and economic challenges associated with climate change and dependence on fossil fuels creates new conditions for the automotive industry and for our daily lives. Electricity, like other alternative fuels, is constantly gaining ground [1].

The more electric vehicles expand and evolve, the more their refueling becomes a point of concern for drivers, a point of superiority over cars with internal combustion engines. The electricity grid ensures the widest possible availability of supply sources, while the technology makes charging electric vehicles in addition to being affordable and an extremely simple and easy process.

Main object of thesis is to examine threats and cyber-attacks in the charging process and the targeting BMS and other parts of the car's electrics' system. More specifically, a Plug-in Electric Vehicle (PHEV) communicates with and is controlled by a charging station. This means that if an attacker could intrude the software of the charging station, it might be possible to influence the charging behavior of the vehicle. Therefore, some threats and challenges arise in the security of smart charging of EVs:

• Whether the charging station hardware can be hacked in order to send these erroneous signals (either locally or remotely).

• How the charging stations can be made tamper-proof and how cyber-attacks can be detected.

## 1.2. Contribution of Thesis

The main contribution of this thesis is the new knowledge of examining possible threats and abuse of smart charging in EVs. In other words, applying specific algorithms in real collected database of a standard EV charging enterprise, to test the possible abnormal activity on the Charging Station can trigger designers and programmers of the networks of those smart Charging Stations, to reconsider the possible security issues that can arise, by third malevolent parties like industrial saboteurs.

The algorithms that have been applied in the context of this thesis have proven their significance in the detection of such abnormalities during the process of smart charging. Metrics procured at the completion of the program have very high accuracy scores in many occasions even 100%, meaning that finding those abnormal chargings is a very easy task, if the data are handled properly and the algorithms are given the right parameters to operate on that given data. With the detection completed the system employing these metrics can easily then trigger an alarm that some abnormal activity is taking place, which then subsequently can give an insight to the developers, as to how the cyber-attack can be prevented adequately. This is more important than trying to solve the problem in later time, because the damage may very well be already done to the charging points and even worse to the charging stations and worst case scenario to the whole grid of the EV charging enterprise. Problems like these are easier to occur with the rapid increase of the EVs distribution globally, so addressing this sooner is of great importance to all parties involved in their creation and usage.

## 1.3. Structure of Thesis

This thesis is divided into nine chapters. First chapter (current chapter) presents the main research problem and its objectives. Second chapter, examines the concepts and main functions of EVs. In more detail, second chapter presents principles of operation of EVs, positive impacts of the introduction of the use of electric cars, smart Charging ways in Electrical Vehicles. Third chapter focuses on issues, vulnerabilities, threats and challenges of Smart Charging. More specifically, this chapter presents vulnerabilities in vehicle communication, possible attacks intervention in smart charging, and protocols that participate on smart charging function. Next, in the fourth chapter we analyze the data given to us per their values and importance in our scenario. Chapter five is an introduction to the algorithms which we have used during the context of the thesis, in order which will be presented, Isolation Forest, KMeans, and Standard Deviation for the unsupervised division of the program, and Support Vector Machine, Gaussian Naive Bayes, Decision Trees for the supervised one. Chapter six is the explanation of the code that took part in it, on data collected of charging cases, in order to conclude whether it is possible to detect abnormalities in a smart charging function. Finally, in the conclusion chapters we summarize the most important findings of this research and provide some analytical observations based on the results, we procured.

# Chapter 2 - Electrical Vehicles

One of the biggest problems of modern society, air pollution, in combination with other factors such as the reduction of natural resources due to their unbridled and uncontrolled exploitation, has led to the discovery of alternative energy sources. In the spirit of the new data, vehicle manufacturers have been led to design and build electric vehicles. The intermediate solution of using catalysts to reduce pollution proved to be insufficient, because it simply limited the problem without leading to its final solution. The electric vehicle ensures zero emissions and frees users from dependence on liquid fuels, the vertical increase in their prices and any kind of shortages due to crises (e.g., Gulf War). Thus, ecological sensitivity, the realization that conventional vehicle pollutants are a major factor in air pollution and the knowledge that a clean environment is equivalent to quality of life have led the automotive industry to "listen" to new needs and adapt accordingly. [2]

## 2.1. Principle of operation of an electric car

Electric cars simply depend on batteries. As a result their mechanical parts differ significantly from the parts of a car with an internal combustion engine. An electric battery car usually consists of three main components: the controller, the battery and the electric motor. In an electric battery car, the accelerator pedal is connected to a potentiometer that measures the power applied to the pedal by the driver. The potentiometer then sends a signal to the controller telling him how much power the battery should distribute to the electric motor. The batteries used in electric cars are rechargeable and usually come in these forms or variants [3]:

- Nickel-Cadmium (NiCd)
- Lead-Acid (and adjustable lead acid valve or otherwise)
- Nickel Metal (NiMH)
- Metal Hydride
- (LiON) Lithium-ion polymers

The battery's energy output is measured in kilowatts per hours (KWh), which shows how much energy storage it possesses.

## 2.2. Electric vehicles

Vehicles belonging to this category operate exclusively using an electric motor, controlled by an electronic power converter. Electricity is provided by batteries, photovoltaic batteries or fuel cells. The electric car has zero carbon dioxide emissions as it moves. But if the electricity for charging the batteries comes from the conventional power grid, then carbon dioxide emissions are not significantly reduced. However, pollutants are concentrated and can be reduced by using filters in production stations. The biggest environmental benefit, however, is if electricity comes from alternative sources [4]. The self-sufficiency of electric cars is generally lower than that of petrol dependent cars. Normally, one charge for an electric vehicle means it can cover a distance in the range between 200 and 320 kilometers. The charging process typically needs around 3-4 hours to complete, which makes it difficult to cover long distances. A quick charge of 80% can take approximately half an hour. In an EV, the electric motor is the sole source of movement. In the electric vehicle industry, two types of engines have prevailed: the permanent brushless motor and the three-phase induction motor.

In “Prius” and “Civic” hybrid vehicles, the brushless motor solution has been chosen, while in purely electric vehicles, such as the high-performance Tesla Roadster, the inductor is used. Less common, and in lower power applications, is the use of DC, foreign or parallel excitation motors, while in the past such motors have been used in electric public transport.

## 2.3. Effect of the introduction of the use of electric cars

**Energy consumption**

It is commonly known, that EVs possess the advantage of thermal use in the city, on the account of that they halt their energy consumption as long as they are do not move, i.e. at the red stop signal. But how is this advantage important to us in a mixed use? In most studies, the kilometer by energy ratio consumed by an electric car in comparison with a thermal one is almost half. Although, in practice, the user of the electric car will be forced to drive at speeds lower in that of a thermal and may not enjoy the comforts offered by its counterpart (e.g., air conditioning system and other electrical subsystems). Lower driving speeds are meant for saving energy, reducing the number of road accidents and if they occur possibly reducing the damage dealt. As a result on the subject of energy consumption the application of electric cars will be positive on a short-term basis, but it may cease to be of significance in the context of electricity consumption at some point later on. [4]

Each electrical fast charging station has a capacity of 10 to 300 KW. The main question here is what comes from the unrestrained usage of numerous stations? The answer is it will cause energy demand at suddenly even higher levels. As a result more power plants will have to be constructed to cover that kind of demand. Perhaps even power plants of nuclear energy may have to be created, which are a serious safety

concern these days. In the scenario where all thermal cars are ultimately replaced by electrical cars, all the energy currently distributed in the transport field would be in the form of electricity. This energy in many cases is between 20% and 40% of the total energy consumed by a country. This will no doubt cause what we have mentioned. One way to control this is to construct special type of charging terminals and special charging systems for electrical so they connect to a power supply system made exclusively for this use. [5]

This network of newly constructed smart stations will provide electricity to a vehicle depending on the availability of electricity. Depending on the intensity and time period of the charging the costs will differ, high intensity charging voltages for short periods of time will cost more and low intensity ones during the night will cost less. In that way we will be able to avoid unwanted user demand peaks. It is estimated that in 10 to 30 years the number of electric cars will be similar to that of thermal cars. A frequently asked question is what happens when a million tourists start to use the country's energy grid? The addition of all these electrical cars will probably cause the collapse of the entire country's electricity distribution or at least generate frequent power outs, which in the long run will damage the system. [5]

Power outs or black outs such as these are extremely dangerous to the infrastructure of the energy grid. The primarily concern is the damage done to the electrical devices that are connected at the time of the black out. It can have an effect at the time it takes for the device to charge, making it much longer as time passes, meaning an increase in the energy consumption which negates one of the main advantages of the EVs. Or it can even destroy the electronics of the device, for example the battery of an EV making completely unusable. Most importantly, they can affect the CP stations. They can damage their systems and as an effect start sending wrong output signals to the vehicles connected, damaging many cars simultaneously, and further increasing the magnitude of that concern.

## 2.4. Plug-in Electrical Vehicle

A plug-in hybrid Electrical Vehicle is a hybrid vehicle with rechargeable batteries connecting the vehicle to a socket at an electrical source. The main components of an EV that are responsible for its operation are depicted in Figure 1. Namely:

- **Battery:** In a PHEV, the battery provides an auxiliary power source of electricity to start the EV before the traction battery begins to engage and additionally powers the rest of the vehicle's accessories.

- **Charger:** It is responsible for the receival of the incoming AC electricity supplied via the charge port and its conversion to DC power for charging the traction battery. It monitors battery measurements such as voltage, current, temperature, and state of charging.

- **Charge port:** The port is the means for the PHEV to connect to an external power source so it can charge the traction battery.

- **Fuel storage (gasoline):** The fuel storage stockpiles gasoline on board the PHEV until it's required by the engine.

- **Lightweighting materials:** Lightweighting is the process of weight reduction of an EV by replacing materials with a higher strength per weight than traditional materials. For example, when we replace heavier iron or steel parts with High Strength Steel (HSS), aluminum, magnesium or composite materials such as glass and carbon-fiber-reinforced polymers.

- **Power electronics controller:** This particular component controls the flow of electricity delivered by the traction battery and regulates the electric traction motor's speed and torque output.

- **Electric traction motor:** Using the traction battery pack's generated power, the motor drives the PHEV's wheels. Some PHEVs use motors like these to perform both the drive and regeneration actions.

- **Radiator:** This component maintains the proper operating temperature range of the rest of the PHEV's systems for its safe function: the engine, electric traction motor, power electronics controller, and other.

- **Internal combustion engine:** For the engine to operate, gasoline fuel is provided into either the intake manifold or the combustion chamber, where it will be combined with air, and the mixture of fuel and air will be ignited by a spark plug.

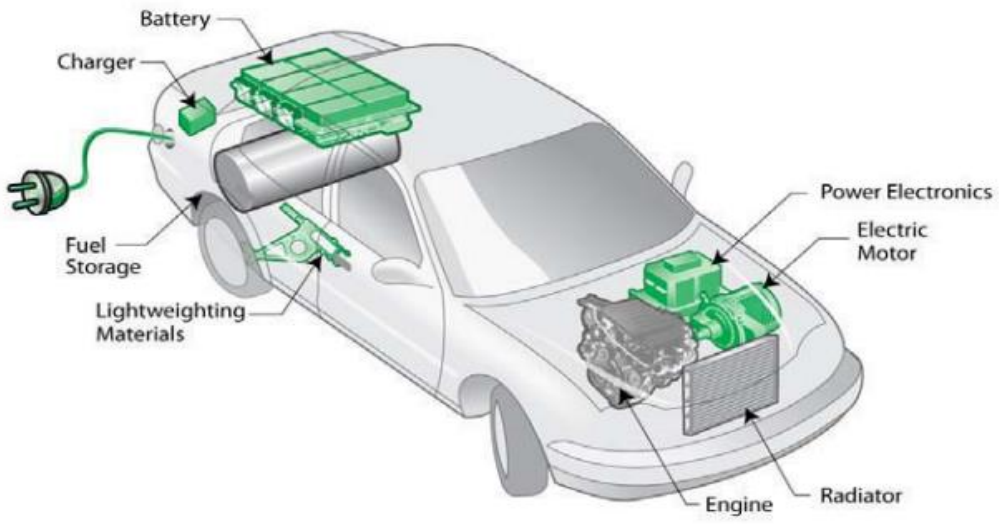



*Figure 1- Parts of a plug-in Electrical Vehicle [7]*

Hybrid plug-ins have the features of both conventional hybrid electric and purely electric vehicles. While PHEVs are expected in the form of passenger vehicles, they can also be commercially light trucks, business trucks, school buses, scooters and military vehicles. PHEVs are also referred to as “network-connected vehicles” or GO - HEVs in their conventional form. Compared to conventional cars, PHEVs can help reduce pollution and dependence on oil and reduce greenhouse gas emissions that lead to global warming. Plug-in hybrids do not use any natural fuel during their electrical operation, unless their batteries are recharged from Renewable Energy Sources. PHEVs have not yet entered mass production, but Toyota, General Motors and Ford have announced their intention to produce such vehicles.

PHEVs are the evolution of what is today a “fully” hybrid vehicle. A fully hybrid car has the ability to start and accelerate at low speeds without the use of the engine, with the battery being charged, however, exclusively by the engine and the power recovery system during braking. A PHEV operates in the same manner but has a larger battery and gives the driver the option to charge it at home using a power source so he can only move his vehicle using electricity. Usually, the car will be charged at night, which will be stationary for a long time. So PHEVs and HEVs use batteries powered by batteries and M.E.K., to save fuel, but PHEVs can further delay the use of fuel by charging the vehicle from home [6].

Moreover, PHEVs have an advantage over purely electric vehicles, because their drivers do not have to worry about the possibility of “discharging” their vehicle. This is because when the battery is discharged, PHEVs operate like conventional ones and use their engine and power recovery system when braking to charge the battery and promote the vehicle. Because they use both an engine and an electric motor, they have smaller and cheaper battery packs than their purely electric vehicles. Today hybrid commercial vehicles use, as mentioned NiMH batteries, which can offer short distances with the exclusive use of electricity in the respective plug-in hybrids. For PHEVs, then, greater

power storage and greater demands will be achieved with Lithium - ion (Li ion) battery technology, as expected.

## 2.5. Smart Charging in Electrical Vehicles

The electricity grid is in the middle of a mutation, a radical change, to harmonize with the needs of the sustainable economy. The 2020 European targets and the 2030 ones designed to reduce emissions increase the penetration of Renewable Energy Sources (RES) and improve the efficiency that enhances decentralized production, the use of storage systems and EVs. For the smooth operation of the new production and demand elements that arise, the electricity grid should become smarter.

Similarly, EU traffic is changing, bringing significant changes to meet the demand for environmentally friendly traffic. Electric traffic is constantly gaining ground as it has zero emissions, is quiet and 3 times more efficient than the corresponding petrol engines.

As the use of EV increases, the Electricity Distribution Networks will face local problems. Even at low levels of penetration, EV can easily overload the local network and alter the mains voltage, with negative effects on local consumers. Faced with this problem, the classic approach is through the construction of new lines and transformers, to meet the new demand. However, this approach is not the best financial solution and will burden network costs, creating a serious barrier to the penetration of electricity [8].

But there is another solution. What we call "smart charging". Smart charging includes the wise charge of EV batteries: charging them in a way that avoids overloading the network and in the future by offering support to the network in times of need and in a way that the EV battery will support the maximum intrusion of RES into the local

network. Smart charging can offer multiple benefits to users, the grid and society as a whole:

1. Customer participation in smart charging is only possible if there are financial benefits to attracting them. Studies have shown that 90% of EV charging is done at the user's home or at work. With smart charging, users will be able to charge their car at home without differentiating the needs of their electrical installation. With this approach, users will be able to take advantage of low prices in the morning with low demand.

2. Smart charging gives EV the ability to be flexible loads scattered across the network, which can be used by the Network Operator in a way that meets the needs of the network and thus avoid costly network enhancements. Studies have shown that the electricity needs that arise if all traffic were electric (i.e., all cars moving today were electric) will be only 25% of the total energy consumed today by a country like Cyprus, to satisfy all needs of society / economy. With this finding, it is easy to conclude that the electric infrastructure, as it is today, could satisfy the entire additional load that will result in 100% of the traffic being electric, without the need for any amplification. This presupposes that the charging of EV will be done with smart management for the benefit of all [9].

3. Smart charging can offer sustainable electric propulsion with great benefits to society. The low cost of charging achieved through smart charging along with the efficiency of electric cars will drastically reduce the use of primary energy, which will lead to a drastic reduction in emissions. With the flexibility offered by smart charging, the utilization of scattered RES systems is facilitated and allows for increased penetration with multiple benefits for all users.

All of the above can be done in the environment of developed operation of smart networks, which under the conditions of their development is imposed as soon as possible, with proper regulation and with a vision on the part of the Transmission and Distribution Network Administrators [9].

As research progresses more and more types of potential threats and attacks are devised and given proper attention they can be identified before they take effect. For example, many attacks exploit weaknesses in network services, such as SSH, HTTP, SOAP etc., that have very poor infrastructure. These same services are also available for the network interface of mobile phone. Hence, these services can be attacked even without physical access via the mobile network interface. Attackers will try to bypass these systems using these services with no or very outdated encryptions and access the system to install malicious code which will be used to damage the infrastructure of the electrical grid or more commonly access the payment system to extrapolate money from it. If it is done during a charging session the attacker can transfer the flow of money from the enterprise owning the charging station to its own specified destination, without anyone noticing. This can propagate to the entire grid and take hold of many procedures at once. Another scenario is the attacker gaining access to the database of the charging company, with an SQL injection type of attack. He then could possess full access to all the recorded data of the enterprise with the intent of corrupting them to become unusable and damage it irreversibly.

## 2.6. Ways of charging Electrical Vehicles

### Method 1 - Slow charging from a common electrical outlet (single-phase, three-phase)

The vehicle is connected to the mains using common power receivers (usually 10 A) located in homes. For the owner to perform this charging method the electrical installation of the house must meet all the safety requirements and there must be a grounding system as well as insurance devices, in order to protect against overload and protection against current leakage, which can still be caused inside the vehicle. This way of charging is the most common, thanks to the simplicity and cheap cost it requires,

however it carries risks in case it is used incorrectly, and it is distinguished by many imitations. As for its misuse, it is important to note that although in most countries the existence of an escape relay is mandatory, several homes have older electrical installations without an escape relay, and it is usually difficult for the electric vehicle user to be aware of this. On the subject of the constraints encountered in this way of charging, these are:

- The available energy. In order to avoid overheating of the socket and cables, in case of using more hours than the allowable limit, and to avoid fire, the electric shock in case the electrical installation is outdated or the appropriate protection measures have not been taken.

- Energy management. If the socket that supplies the vehicle is not in a separate circuit or the total consumption exceeds the safety limit (usually 16 A), it will interrupt the circuit, interrupting the charging. The usual charging time is 10-15 hours and a 10 A circuit is usually used. Power receivers do not exceed 16 A-250 VAC although this is different in some countries (Figure 2).

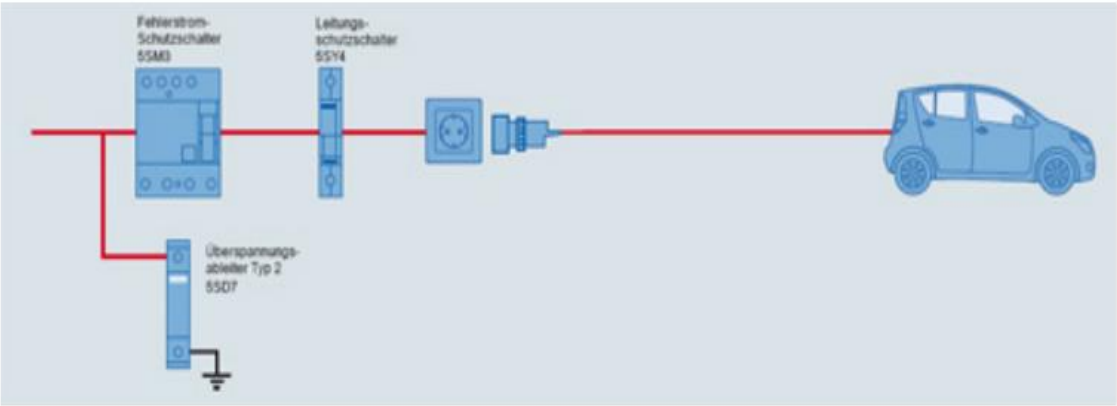



*Figure 2- Charging according to Method 1: Slow charging from a common electrical outlet [10]*

**Method 2 - Slow charging from a common electrical outlet (single-phase, three-phase) with internal cable protection**

The vehicle is connected to the mains using common power receivers as in the above case the charging is done via single-phase or three-phase supply and ground pipe installation. However, this method provides additional protection by adding a control system inside the cable, which allows communication between the electric vehicle and the couple. Charging 2 was originally intended mainly for the US, but recently gained a lot of interest in Europe, with the aim of replacing Method 1. Nevertheless, in addition to the obvious disadvantage of having a control device inside the cable, the main disadvantage is the lack of protection of the coupler, one of the most likely fault points, by the control system. The charging time ranges from 3 to 8 hours (Figure 3).

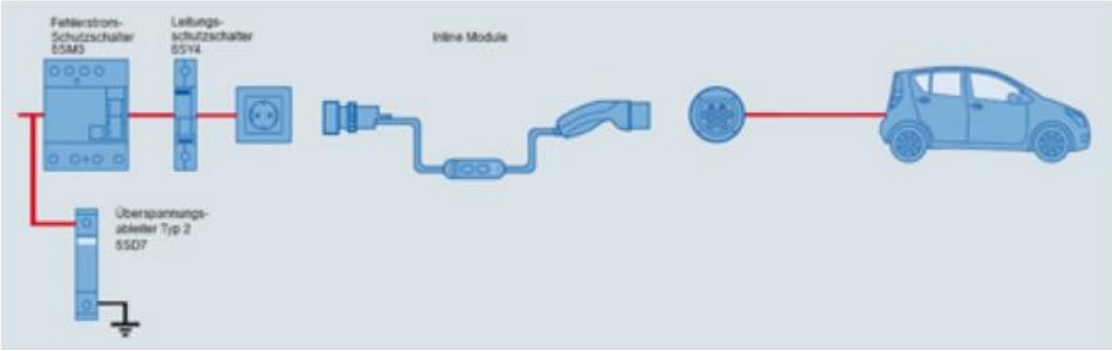



*Figure 3- Charging according to method 2: Slow charging from a common electrical outlet (single-phase, three-phase) with internal cable protection [10]*

**Method 3- Slow charging using a specific current receiver with an installed control and protection system.**

The vehicle is connected directly to the mains via a socket of specific specifications and a separate circuit. This is the only way to charge the standard electrical installations. According to the international standard IEC 61851-1, the device / control system, between the supply equipment and the EV, instructs the following functions [11]:

- Confirmation that the vehicle is properly connected
- Continuous control in case of power leakage
- Activate and deactivate the system
- Charging rate selection
- The control system is usually installed as an additional duct in the wiring of the charging cable, together with the phase, neutral, and ground. It therefore requires the use of special components.

It also allows the distribution of loads, so that the electrical appliances of the house operate during the charging of the vehicle or otherwise improve the charging time. Finally, pairs for this type of charging according to international standards require a range of control and signal nozzles at both ends of the cable, as seen below in detail in Figure 4.

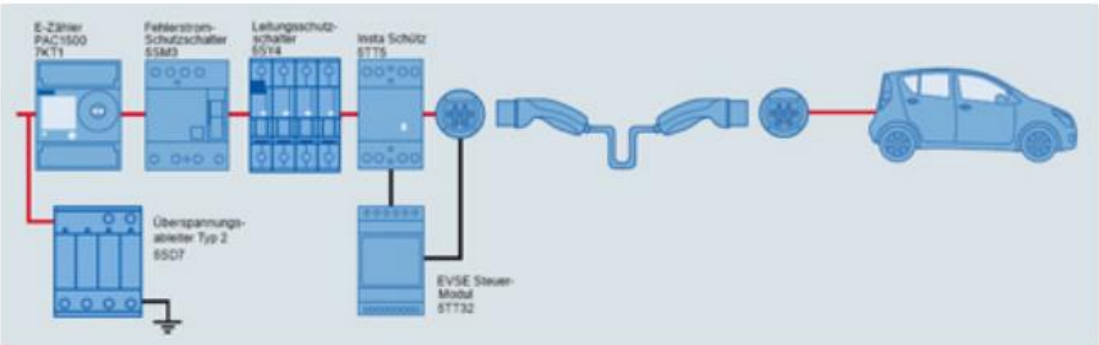



*Figure 4- Charging according to method 3: Slow charging using a specific current receiver with an installed control and protection system. [10]*

**Method 4 - Fast charging with the usage of external charger**

This charging mode is related to the connection of the electric vehicle to the mains using an external charger that has a control and protection system installed. The alternating current of the network is converted to a continuous charging station and the plug type ensures that only if the vehicle fits, the connection can be made possible. Using fast charging with direct current, an intensity of up to 400 A is achieved [12].

**Charging Levels**

While in Europe the IEC 62196 standard is used, which separates charging modes to categorize charging equipment, in the United States charging modes are classified as Charging levels (Table 1).

*Table 1- Charging times and the relevant requirements of the various charging levels*

| | **Requirements** | **Voltage (V) /Amber (A)** | **Time of charging** |
|---|---|---|---|
| **LEVEL 1** | ------ | 120 / 13 | 7-8 hours |
| **LEVEL 2** | Special connection | 240/32 | 3-4 hours |
| **LEVEL 3** | Special wiring and external charger | 500/200 | < 45 minutes |

**Level 1**

The transfer of alternating current to the internal charger of the 120 Volt electric vehicle, either 15 A (using 12 A) or 20 A (using 16 A), through a common power socket, located either in homes or in commercial buildings in USA. However, because the power provided (maximum 1.44 kW) is insufficient, this results in prolonged charging times. So obviously, and it is an inefficient, but accessible and cheap option. At level 1, a new separate circuit is recommended as necessary to avoid overcharging. The charging equipment is installed inside the electric vehicle. While on the connection cable, a switch has been installed in case of power leakage.

**Level 2**

The transfer of alternating current to the internal charger of the electric vehicle: 208 to 240 Volts, single-phase or three-phase. The maximum current is set at 40 A. At this level, the equipment is divided into inductive and wired, to which reference has been made above. Regardless of the equipment, a separate circuit is required to charge the vehicle. Usually the charge ranges from 15 A, with charging power at a maximum of 3.3 kW.

**Level 3**

The transfer of direct current from an external charger to the electric vehicle. The maximum current intensity is set at 400 A. At this level, also known as Fast Charging, an external charger is used, installed on a three-phase 480 V AC circuit. The purpose of level 3 is to achieve a charge rate of 50% for a charge time of 10 to 15 minutes. Charging power ranges from 60 to 150 kW.

## 2.7. Charging Points/Stations for Electrical Vehicles

With the growing negative impact of climate change globally by poor energy fossil fuels, many countries have started taking the necessary steps to reduce their carbon emissions. In particular, for the section of public transportation the passage from fuel to electrification as means of energy, is considered to be one of the main ways to achieve a significant reduction in $CO_2$ emissions. In recent years, electric cars have gained popularity, with more than 180.000 of them being built to this date worldwide. Despite only presenting 0.02% of all road vehicles so far, a much desired objective is to have more than 20 million electric cars on the roads by the year 2020. In order to ensure the widespread development of electric cars, actually leads to the anticipated reduction of $CO_2$ emissions, it is important to be charged from renewable energy sources (e.g., wind, solar). [12].

Electric cars could possibly help with energy storage when there is a surplus and supply power back to the grid. The ability of electric cars to store energy while being used for transportation represents a huge untapped potential for the development of energy systems [13]. On the one hand, since vehicles only drive for a small percentage of the day and a percentage of vehicles remain unused in parking spaces and given the fact that electric vehicles are equipped with large batteries, they could be used as storage devices when parking (process Vehicle-to-Grid (V2G)) and thus increase the energy storage capacity of the network. Indeed, there are studies that have shown that if a quarter of vehicles in the US were electric, this would double the current storage capacity of the network. On the other hand, since a large number of electric cars will need to be charged daily, if electric cars charge when needed, the network load may be overloaded. Grid-to-Vehicle (G2V) - in real time, is taking into account the limitations of distribution networks within which electric cars must be charged.

In addition, electric car navigation systems must consider the ability of vehicles to recover energy when braking and / or when driving downhill and choosing routes that make full use of this capability. By doing so, it may be possible for vehicles to charge less frequently, thus maximizing energy efficiency, reducing costs for their owners, and minimizing the stresses they cause on power grids. In this context, a number of techniques and mechanisms for the management of electric vehicles, either individually or collectively, have been developed.

For example, mobile-based applications have been developed to provide to the drivers information about charging sites, where charging time slots are unoccupied. In addition, original systems have been developed for energy-efficient routing, while new types of chargers that can fully charge an EV's battery in less than an hour. Although many developments have taken place in terms of physical infrastructure and technologies for electric vehicles, they may not be sufficient to manage the anticipated excess number of EVs. Such problems will require algorithms involving many different entities, each with its own goals, needs and motivations (e.g., energy for charging, maximization of profit), while operating in an adjoined scenario and try to deal individually as well as together with a number of uncertainties. [13].

Charge points as already mentioned in section 2.5 are susceptible to a large range of attacks. One example of them being the Denial-of-Service Attack. In this particular attack the CP system becomes completely unavailable for any EV trying to connect to it. Small scale attacks of that type are not very important and can be dealt with eventually, but a large scale one could cause the entire grid to shut down, blocking all users from charging and given enough time without being dealt, halting all vehicle related activities, causing mass traffics and many kind of damages.

# Chapter 3 - Threats and Challenges of Smart Charging

## 3.1. Vulnerabilities in vehicle communication

Smart Charging may put into risk the reliability and security of the power network, as neither the charging stations have deployed security mechanisms for identifying and preventing security threats and attacks, nor the Distribution System Operator's (DSO) have implemented security mechanisms for mitigating potential disturbance of the network due to a break-down (or a hack) of the smart charging stations [18]. Smart charging is complex process which needs the organization of a number of services such as metering and payment for energy, communication between the EV battery management system and the charge point, followed by a communication mechanism between the CP and a central management system, and finally the establishment of a communication channel between the CS and the energy suppliers (DSO, Transmission System Operators (TSO), smart girds, etc.). Because these services are offered from a variety of entities, this communication scheme creates an environment susceptible to a number of security threats on different levels [17].

The co-existence of an electrical system monitored and controlled from an ICT infrastructure is an open challenge due to the heterogeneity of the cyber-physical systems that are get engaged that require the standardization of protocols and the implementation of two primary interfaces, one for electricity and another for the management of the system. In the case of the smart charging scenario, the ICT system is related to the status, authorization, metering, and billing of the EV that interact with the system [20].

A higher level depiction of the entities that are getting engaged in the smart-charging use case are depicted in Figure 6. The DSO is responsible for the distribution of the electric power and ensures the functionality of the electricity network, the CPO takes care of the customer-end services (authentication, billing, etc.) alongside with the management of the charging points, the Electro Mobility Service Providers (eMSP) is responsible for setting the billing mechanism, the CP acts as the open gate to the system, and eventually the EV which is the end-user to the infrastructure. The roles/entities of the smart-charge that are described briefly above are presented in the form of a table (Table 2) and provide a more detailed description about their functionalities.

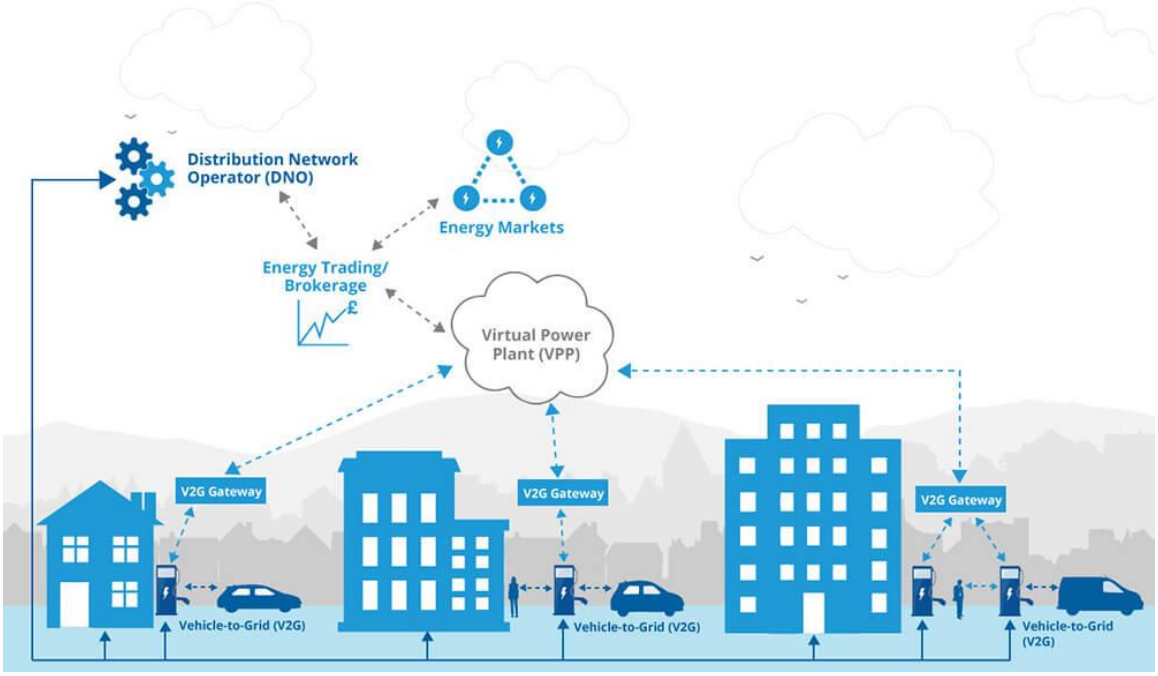



*Figure 5- A high level depiction of the entities getting engaged in the smart charge scenario [21]*

*Table 3- A high level depiction of the entities getting engaged in the smart charge scenario*

| Entity | Description |
|---|---|
| **Distribution System Operator - DSO** | Manages the electrical grid. |
| **Electro Mobility Service Provider - eMSP** | An eMSP is a market role that offers charging services to EV drivers. |
| **Charge Point Operator - CPO** | The CPO is responsible for the management, maintenance and operation of the charging stations (both technical and administrative) Managing and maintaining the physical electrical network – the DSOs – and all other entities: the energy providers, the customers and the eMSPs. |
| **Charging Point – CP** | Charge Points are devices where EVs are connected to get charged. Each CP contains at least one meter per socket (Measuring Instruments Directive (MID)) owned and controlled by the CPO. |
| **Electric Vehicle - EV** | Gets charged through a CP. |

## 3.2. Protocols that participate on smart charging

### IEC 61851-1

Different charging topologies need to be considered for conductive AC- and DC-based dedicated charging equipment. Such EV charging equipment is defined in the IEC 61851 standards series (Figure 7). Initially, this protocol defines necessary requirements for applying conductive charging. IEC 61851 standard provides two ways of application a) on-board and b) off-board, for both AC and DC charging. Furthermore, if it is necessary, it can be applied to other additional equipment of the vehicle that needs to be charged. Moreover, IEC 61851 standard offers to user four ways to apply the charging system. More specifically, user can select either slow charging or fast charging. Finally, this standard describes some basic requirements that have to be applied, for user's safety and normal operation of the electrical network. [23].

Concerning the user's safety, IEC 61851-1 has been created using Pulse Width Modulation. This protocol has the ability to control and handle changes in electricity network. Consequently, IEC 61851-1 standard is an integrated protocol which is used in vehicle communication and charging systems of Electrical Vehicles.

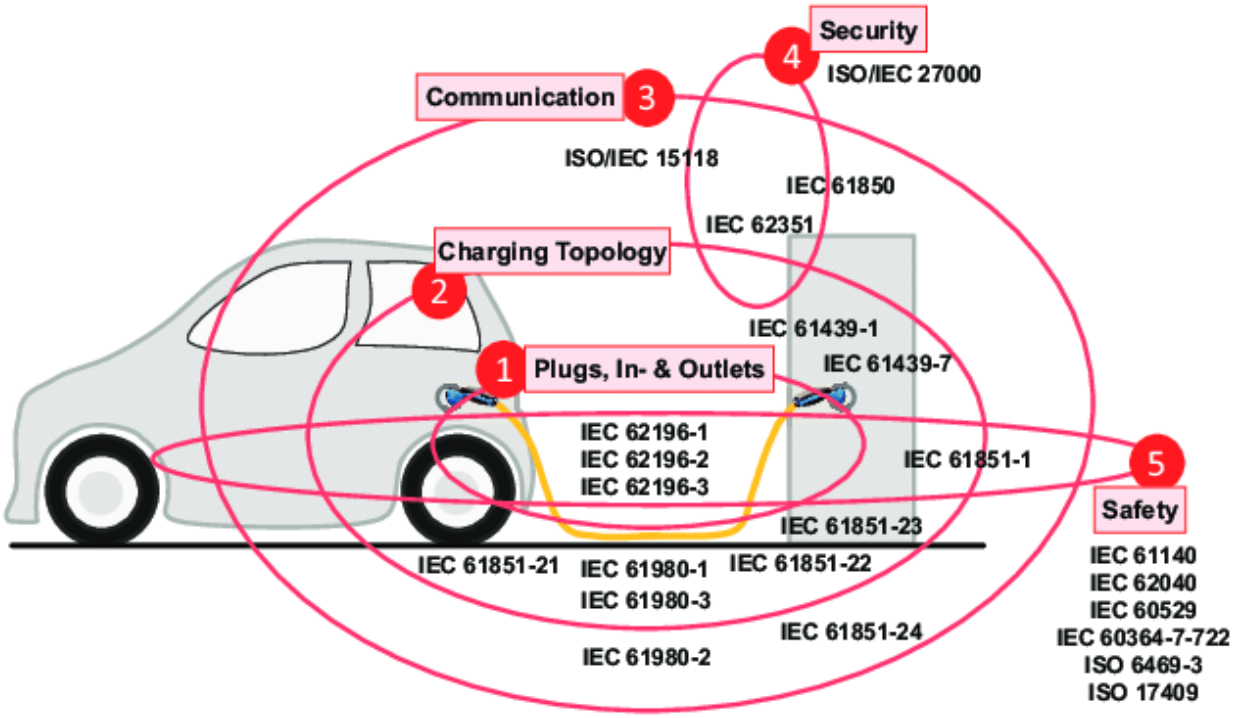



*Figure 6- Protocols that participate on smart charging of an Electrical Vehicle [23]*

## 3.3. Smart Charging Abuse

A cloud-based back office of a CPO communicates with a charge point via the Open Charge Point Protocol (OCPP). The charge capacity of a charging station can be set from the cloud by means of OCPP requests. Version 1.6 and 2.0 of this protocol support smart charging. This means that one platform can connect to a wide range of charging stations and still be able to provide smart charging services to all of them.

The charging station then in turn communicates with the charging station via the IEC 61851 protocol (for high speed DC charging other standards are used, but these are usually not used for smart charging).

There are a few important observations to be made here:

- Via OCPP, the maximum charge rate for a charge point/socket can be set for a specific period.
- The charge point imposes this maximum on the EV.
- The EV can choose its own charge rate, as long as it is below the maximum.

It is therefore *not possible* to set a specific charge rate for an electric vehicle, only the maximum charge rate can be set.

On average, a charging station can charge at around 11 kW. This means that someone with access has control over charging stations with a combined capacity of around 220 MW, equal to the power output of a medium-sized power plant. It is expected that around 200,000 charging stations will be connected in 5 years, which corresponds to a potential capacity of around 2 GW. Simultaneous switching on or off of all these charging stations can lead to a pan-European blackout.

Here we indent at increasing the cyber-security of a standard EV charging enterprise's platform through the integration of ML techniques for identifying anomalies in the charging patterns, and therefore minimize the exposure to both the enterprises' database and the stability of the electric grid. The scenario covers both the ICT and the electric engineering domain on an effort towards increasing the cyber security on what is called Energy Internet [22].

# Chapter 4 - Data Analysis

This chapter presents the data that will be applied in our scenario that contain real recorded charging processes, which took place during the years 2018, 2019 and 2020. Main concept of the implementation is to examine possible threats and cyber-attacks on smart charging network that used by charging point stations, where electrical vehicles are connected in other to be charged.

## 4.1. Research Questions

As mentioned in sections 2.5 and 2.7, in a charging process, an attacker could exploit vulnerabilities of the network of a charging station and affect the behavior of the charging process in plug-in electrical vehicles that are connected. Therefore, the main object of thesis is to examine threats and detect cyber-attacks in the charging process. In this scenario, PHEV communicates with and is controlled by a charging station.

In order to answer the research questions made above in chapter 1, it is necessary to run specific anomaly detection tests using the algorithms already mentioned and other data manipulation techniques. Consequently, if a charging process behaves abnormally, in other words, if a connected EV requests high voltage, we could be able to detect it. Therefore, it is crucial for the charging station owner to monitor, manage and restrict the use of their devices remotely to optimize energy consumption. Otherwise, the smart network of the charging station might break down.

## 4.2. Presentation of data for the scenario

In the context of this thesis, we used a dataset in CSV format taken from the database an EV charging enterprise. The database consists of different tables, each one of them representing a unique entity in the EV scenario, namely: the Charge Detail Records (CDRs), the Connections and the Meter Values (MVs). Each table existed in its own file separated by year. For the proper implementation of the code all three tables needed to be included as our input. Because they were in separate files this was not possible. So the files had to be merged together per year acquired. Specifically the datasets were manually merged at two separate columns: the ChargePoint_ID and the Connection_ID. Now that all our data were complete, we could begin our analysis process. The EV charging datasets comprised of thousands of charge sessions hosted on a cloud platform dating back to 2012.

*Table 3 - Charge Detail Records (CDRs)*

| Column | Data type | Description |
|---|---|---|
| **ID** | **PK, lint** | **ID for CDR** |
| **Duration** | **Nvarchar (50)** | **Duration of session** |
| **Volume** | **Nvarchar (50)** | **Volume in kWh** |
| **AuthenticationId** | **Nvarchar (50)** | **Unique charge card ID** |
| **ChargePoint_ID** | **FK, int** | **Unique Charge Point ID** |
| **ConnectorId** | **Nvarchar(255)** | **Charge Point Connector Identifier** |
| **Calculated Costs** | **Float** | **Costs of charging processes** |
| **dStart** | **datetime** | **Session start time** |
| **dEnd** | **datetime** | **Session end time** |

Table 3, describes the necessary details of each charging attempt such as the duration and the volume, but it also includes features from other tables as foreign key in order to express the correlation with the other entities of the grid. Therefore, every record to the database includes the unique ID of the charge card used by the EV driver, and the unique ID of the charging station. The features of duration, volume and session start/end time have the highest value for the Artificial Intelligence (AI) algorithms, as they can offer a useful insight for the pattern of a charging session.

*Table 5- Meter Values (MVs)*

| Column | Data type | Description |
|---|---|---|
| **ID** | **PK, int** | **Meter Value identifier** |
| **Timestamp** | **datetime** | **In-session timestamp** |
| **Value** | **Decimal(18,2)** | **Measured Value** |
| **ValueType** | **int** | **Specifies unit of measurement (0 = Wh, 1 = kWh, 8 = Amp)** |
| **ReadingContext** | **int** | **Specifies measurement or Instruction (3 = measurement, 6 = instruction (6 is used to send max amperage))** |
| **Connection_ID** | **FK, int** | **Connection Identifier** |

Table 5 describes the measurements monitored by each individual charging meter such as its value in in what type of measurement was recorded, the time it took place. Each meter possesses an ID, which serves as a key in the database to maintain uniqueness. Also, each connection made to this meter comes with a unique ID, for better data distinctiveness.

*Table 6- Connections*

| Column | Data type | Description |
|---|---|---|
| **ID** | **PK, int** | **Connection identifier** |
| **ConnectorId** | **int** | **ChargePoint Connector** |
| **ChargePoint_ID** | **FK, int** | **Unique ChargePoint ID** |

Finally, Table 6 is comprised by the distinct identifiers of each connection made between an EV and a CP.

Our purpose in this implementation is to run some algorithms, in order to see if smart charging system can secure the enterprise's grid, and to prevent potential blackouts in the national electrical grid. Different users are synchronized (either on purpose either unintentionally) and proceed timely in connection/disconnection actions, causing an unexpected load to the electrical grid. Such actions can be prevented if AI/ML techniques are integrated into the EV charging enterprise's software.

The dataset in CSV format, which has been collected from an EV charging enterprise, can be used as a starting point for getting an insight of the charging stations' behaviour, extracting the attributes of a "normal" charging action and identifying suspicious actions as outliers.

# Chapter 5 - Algorithms Exploration

## 5.1. Algorithm

An algorithm is a predetermined sequence of computer-implementable commands, designed to solve a class of problems or to complete a computation. Anomaly detection refers to the problem of finding patterns in a set of data that do not agree with the expected behavior. Extreme pricing detection has a variety of applications such as credit card fraud detection, security fraud detection, security and medical care systems, and even military systems for detecting hostile activities. The importance of extreme price detection stems from the fact that extreme data values translate into important information in a wide range of application areas.

The first attempts to detect extreme values date back to 1970, when researchers tried to elicit erroneous measurements from their data to ensure that the data matched best with the proposed models [23]. Detection of anomalies or extreme prices has been researched in the field of statistics since the beginning of the 19th century. Over the years, a wide variety of techniques have been developed in various fields of research. Many of them have been created for more specific applications while others are more general. There are also cases where, although a technique has been developed for a specific problem, it is then applied to areas that were not originally intended.

**Types of extreme anomalies**

In a time series and more specifically in the graphical representation of a time series a spike upwards or a steep draft, which stands out from the rest of the graphic behavior is defined as an abnormal behavior or anomaly. Mathematically, this happens when the value that the data is receiving at that particular time is far from what it was before. But there are many types of anomalies.

1. Point anomalies: When data is far from the rest.

2. Contextual anomalies: Abnormalities very common in time series and have to do with the general context. For example, it makes sense to spend 200 euros a day on vacation, but not on holidays like Easter.

3. Collective anomalies: Anomalies exist in large groups.

**Point anomalies**

We encounter these extreme values if an object in the data (a point) shows a different behavior from the rest of the data. Although it is the most easily detectable type of extreme value, an important problem is the appropriate measure of the deviation of one point from the rest. In Figure 8, we see an example of extreme point values, where it is clear that the two points that are in a circle and have been named V1 are much further away from the set of points V2 and are characterized as extreme values [26]. As an example from real life let's talk about credit cards. We assume that all the data refers to the transactions of an individual and more specifically to the amounts spent per transaction. A transaction in which the amount allocated is much larger than the average normal spent by that particular individual is characterized as a point extreme value.

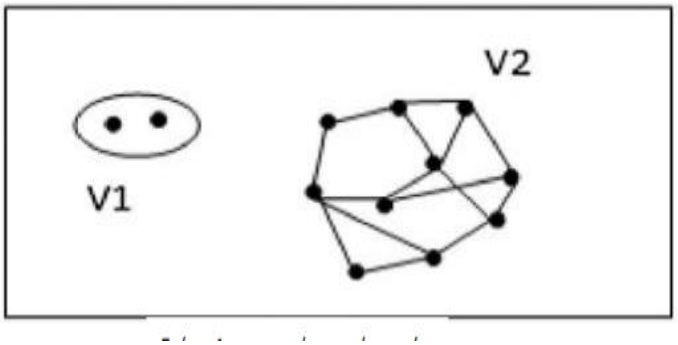



*Figure 7- Extreme point values (anomalies) [26]*

**Environmental values related to contextual anomalies**

This type, we find it if one point of the data deviates significantly from the rest in a particular environment, and only in that. The concept of environment arises from the structure of data and is part of the wording of the problem. Each fact is defined on the basis of two characteristics.

A) The environmental characteristics, e.g., those that determine the environment and

B) The behavioral characteristics, e.g., those that determine the points that are outside the specific environment. [27]

**Collective extreme anomalies:** This type of extreme value refers to a set of data, which as a group, show a different behavior from the general set of data, while as independent units may not be extreme values. In Figure 9, we see an example of a collective extreme value. In the cardiogram, while the values that are in red alone are not an extreme value, as a set of values they differ from the usual and are characterized as abnormal. [27]

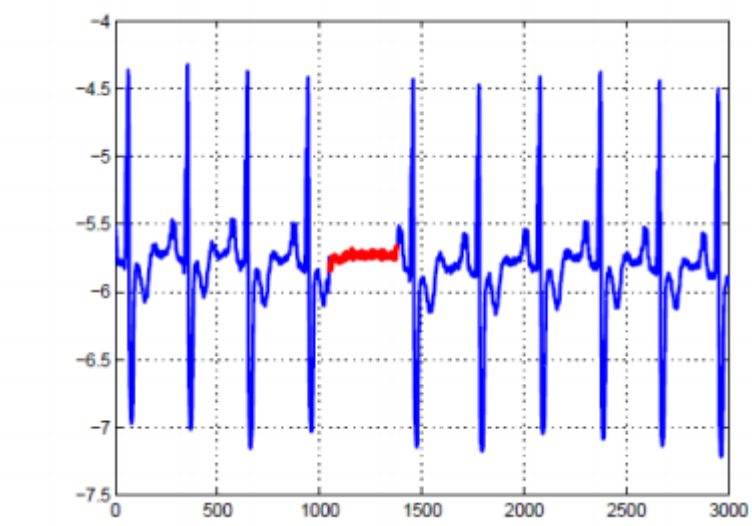



*Figure 8- Collective anomaly corresponding to an Atrial [27]*

### Detection and analysis of anomalies

Many different techniques can be used to identify unusual patterns in datasets that are not in line with the expected behavior. Points that belong in these types of patterns are called outliers, i.e. excessive-extreme prices. It has many applications in our daily life, such as detecting irregularities in bank account transactions (credit card fraud), unusual patterns in the movement of a network that can be a sign of attack by a hacker, or even in medical science and in particular reading and detecting a tumor on an MRI or CT scan. Sometimes when it comes to time series analysis and finding anomalies, there are some challenges that we need to work on to get a good result. For example in many cases the

boundaries between abnormalities and normal data are not accurate. In this case, normal observations could be considered anomalies and vice versa. Even more often, approaches to detecting anomalies in one field more often than not can be used in another.

That is, it is possible that we cannot follow the same way of analysis and detection, when the data of one way is completely different from the way of the methodology we follow, or when the method we have as a model is different from the one we want to follow.

For example we cannot rely on a statistical method of analysis if we want to work with the method of categorization, or when we intend the analysis to be done with supervised machine learning we will not help if we choose to study an experiment implemented with unsupervised. Many can be helpful but will not be the rule for our implementation. Finally, a problem is the availability of training and validation data for training a model, because there may now be an extremely large number of data sets on the Internet, but most of the times it will need some configuration to meet the operating needs of our model.

## 5.2. Machine learning

Machine learning is the concept of allowing computers to learn solving problems without being explicitly programmed by a person. In another perspective, ML teaches computers to function as any does so: learn by usage experience. Machine learning is but a small domain within the vast field of what is called Artificial Intelligence.

In the security field of study, ML continuously studies and learns by analyzing given data to find logical patterns between them, so it can more effectively detect possible threats, such as malware in encrypted traffic, predict where certain sites should be classified as “bad regions”, meaning the contain phishing code to steal data from the user connected or protect cloud based data, which nowadays is a rampant field of study and continuous technology advancement, by uncovering suspicious behavior.

In the scenery of cyber threat detection, many organizations globally are tasked to continuously track millions of external and internal data points across the internet and correlate them accordingly. Of course such a task is not easy. Controlling such a vast volume of information with only a team of people can prove almost impossible. This is where ML is introduced, because it can recognize patterns and predict threats in data sets which can be considered massive by a common analyst, but for it are fairly easy to go through, moreover at a machine speed of calculation. By automating the process of analysis, cyber threats can be easily isolated from that information and given deeper human analysis.

## 5.3. Ways to operate extreme price detection techniques

### Supervised problems

Supervised problems are those that the computer does not solve on its own. That is, the computer is given a set of data, and there is the human factor, which tells the computer how to sort this data. The behavior of the data, whether normal or not, should be predetermined. This can be done in two ways, either by saying what is normal and anything that does not go with it is considered an extreme value, or by determining what is abnormal and anything that is contrary to it is considered normal. This technique requires the human factor to know all possible extreme values or that it can be considered normal in the data, something that is not so feasible since the goal is for the computer to be able to detect extreme values on its own. Theoretically, this type of methodology provides a better detection of extreme values as there is access to more information, but keeping accurate data labels is a major challenge that rejects this theory. [29]

### Unsupervised

In unattended techniques, there is no pre-classification by the human factor, and the computer must detect for itself that there are extreme values, if of course they exist. In these methods, it is assumed that data that behaves normally often follow a pattern, while extreme values do not behave in this way. However, this assumption is not always correct as there are cases where the similarity is not enough to determine the regularity or not of some data as in the case of collectively extreme values. That is why this technique is often ineffective and leads to wrong extreme values.

**Semi-supervised**

This kind of approach is something between the two previous ones. It is used when from all the data, there are a few that have been pre-characterized as normal. Based on this, we try to characterize what is left. This approach essentially sets a limit to normalcy, where a given value is called an extreme value if it is outside it and normal if it is within it.

**Clustering**

A simple definition for clustering: clustering is the process of organizing patterns (observations, data, or attribute vectors) into clusters, where the members of a group are similar to each other according to a criterion. The aim is to identify the groups that belong to different amounts of data, based on some criteria of homogeneity. The grouping technique falls into the broader category of unsupervised learning techniques. The difference between data clustering and data classification is that in classification the groups to which the data will be placed are predefined.

This means that the number of groups, their names and identities are known in advance. This is also a learning system since the labels given by the available standards are used so that the classification system learns the description of each class and is able to classify a new standard. In contrast, data grouping emphasizes that groups do not pre-exist but are decided by the algorithm in a dynamic way. In data grouping, a given set of data must be managed in order for the groups to emerge dynamically according to the algorithm. The aim is to create groups, each of which will gather homogeneous elements. Each of these groups maintains a center, usually its most central element. An example of

grouping is given in the following image, where on the left is the original set of items before grouping and on the right the entry of items in clusters.

## 5.4. Isolation Forest

Isolation Forest utilizes the concept of isolation to detect anomalies in the dataset. It takes advantage of two quantitative properties that anomalies have:

- Anomalies are the minority, consisting of fewer instances, and
- They have feature values which are very different from those of normal instances.

These two characteristics make anomalies susceptible to isolation, meaning that they are more likely to be isolated from other instances when the dataset is randomly partitioned. This algorithm works by recursively randomly partitioning the dataset until it reaches a particular depth or isolates a point. To represent the partitions, it uses a special kind of Binary Search Tree (BST), called iTree.

The idea is that anomalies, since they lay further from the rest observations, will require a lower number of random dataset partitions to become isolated, whereas normal observations will need a higher number, as they are close to other normal points. This translates to respectively shorter and longer path lengths (or distances from the root node) in each iTree. The anomaly score that is inferred for an example during the evaluation stage is based on this path length value. For example, in Figure 10, we can see that point xo requires on average fewer “cuts” to be isolated, than point xi . The model of Isolation Forest is composed of an ensemble of iTrees. Each iTree is built on a subsample of the original dataset. The use of subsamples has some very useful properties. Each subsample is formed by randomly picking instances from the whole dataset without replacement.

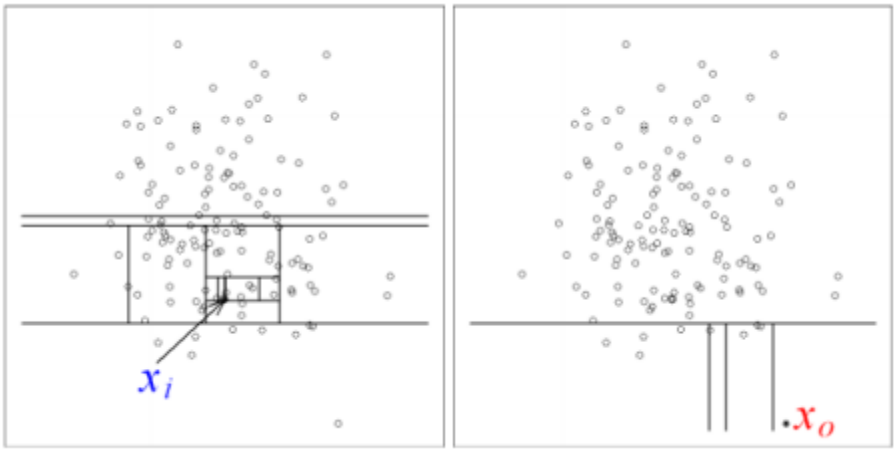



*Figure 9 - Isolation Forest anomaly score [24]*

For the Isolation Forest algorithm to work, it requires an anomaly score for decision making. It is defined with the following mathematical equation:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

*Figure 10 – Isolation Forest equation*

h(x): Defines the length of the observation path x.

c(n): Defines the average length of the failed search path in a binary search tree.

n: Defines the number of external nodes. The path length, calculated on average in a forest of random trees, is a measure of consistency. Each observation made by its usage is given an anomaly score and based on this one of the three following decisions can be derived:

- If the score is closing to a value of 1 it indicates the existence of anomalies.
- If the score is less to a value of 0.5 it indicates normal observations.

• If all scores tend closer to 0.5 then the sample as a whole does not appear to have any noticeable abnormalities.

The anomaly score for a data point is the average value of the path lengths acquired by “passing” the data point from each iTree this approach often causes a high computational complexity for data of higher dimensionality, and, because the model is optimized to profile normal points, and not to detect anomalies, it often ends up with too many false positives, or few true positives (not to mention that most of the times a labeled dataset is mandatory for the training phase). Isolation Forest is different from such algorithms because its model isolates anomalous instances instead of profiling the normal ones, requires no labeled dataset (it is an unsupervised algorithm) and is also robust when applied at data with high dimensionality.

Moreover, most distance-based and density-based methods do not handle the effects of swamping and masking well, having poor performance in such cases. Swamping is the when normal instances are too close to anomalies, causing them to get incorrectly flagged, and masking is the situation in which too many similar anomalies form a small cluster, concealing their presence. Isolation Forest alleviates the effects of these two situations by operating on random subsamples of the original dataset.

## 5.5. KMeans

The KMeans partitioning algorithm is one of the simplest and most popular grouping algorithms belonging to the broader category of unsupervised learning techniques. This algorithm is popular because of the simplicity of its implementation and its linear complexity which is of the order n (O (n)), where n is the set of elements. The process of grouping a data set based on k-means is easy, as long as the number (k) of the resulting clusters is predetermined. The main idea is to first identify k centroids, one for each cluster. These initial centroids must be skillfully chosen, because different starting positions for centroids give different results. That is, the initial position of the centroids affects the result that the algorithm will give. Thus, it is often considered better to choose those centroids so that they are as far apart as possible. The next step is to select each item from the data set and associate it with the nearest centroid. When this is done for all data elements, the first step is complete and a first and "rough" grouping has already occurred. Next, k new centroids need to be recalculated, which will be the center of gravity for each cluster resulting from the previous step. So once the new k centroids are defined, the same procedure is followed to assign each of the data set elements to the nearest centroid. Thus, a repetition of the same process takes place. The result of this iteration is that at each step the centroids change position (new ones are defined) and the elements are assigned to the appropriate cluster each time based on the nearest centroid. When in some iteration no data shifts occur, then the execution of the algorithm ends. The result is the grouping of the data set into k clusters.

**Weaknesses of K-means**

• The algorithm converges to local optimal and not to universal optimal.

• The way in which the original centroids are defined is not clearly defined. A fairly popular way of choosing the original centroids, is to be chosen at random. This

method is also applied in the present diploma. The result depends on the original centroids. An unsatisfactory solution often arises due to the "poor" initial selection of centroids. For this reason it is recommended to do several execution tests with different initial centroids each time [30].

• Another cluster may be left without members and thus a centroid may not be renewed. This is the known problem of remote components that are often not included in the process.

• The results also depend on the distance measure used. Many times it is necessary to normalize the data set data in order to apply some distance measure. So for example when the algorithm performs clustering based on Euclidean distance, it assumes that the cluster data is all spherical. Various studies have been able to extend k-means so that it can work not only on spherically shaped data but also on elliptically shaped data. Another weakness of the algorithm is that it is difficult to identify groups with different formation and size. The problem is mainly exacerbated in very large data sets. Usually the level of difficulty that the algorithm faces in such large data sets has to do with the density of the data, which can be large elsewhere and small and generally varied [30].

• The results depend on the value of k, which is the input for the algorithm. Although there are many ways to estimate k and many efforts have been made in this direction, unfortunately the problem still remains unsolved. The algorithm fails to find the optimal k on its own and make this widely accepted. Of course by optimal k, we mean that k which best describes the separation of each data set so that the resulting groups make sense. The various ways of estimating k that exist today in the literature have been mentioned in the previous chapter. This last weakness of k-means can be more easily understood through an example. It is often a nuisance as it is not always possible to know how many clusters there are when it comes to real-world clustering [30].

## 5.7. Standard Deviation

The standard deviation of a populace and the standard blunder of a measurement got from that populace, (for example, the mean) are very unique yet (related by the backwards of the square foundation of the quantity of perceptions). The detailed safety buffer of a survey is processed from the standard mistake of the mean (or on the other hand from the result of the standard deviation of the populace and the reverse of the square base of the example size, which is something very similar) and is normally about double the standard deviation - the half-width of a 95 percent certainty stretch. In science, numerous scientists report the standard deviation of exploratory information, and by show, just impacts in excess of two standard deviations from an invalid desire are considered measurably huge - typical arbitrary blunder or variety in the estimations is in this route recognized from likely real impacts or affiliations. The standard deviation is additionally significant in fund, where the standard deviation on the pace of profit for a speculation is a proportion of the unpredictability of the venture. [31]

## 5.8. Support Vector Machine

Support Vector Machines (SVMs) are a set of learning methods used for classification and regression analysis problems. The main idea of SVMs is to construct a superplane so that the separation distance between positive and negative examples is maximized. The vectors of the elements closest to this superplane are the support vectors. This desirable property is achieved following the principle of Structural Risk Minimization by the theory of Industrial Learning. [32]

The idea of minimizing structural risk is to find a case h for which we can guarantee the lowest actual error. The real error of h is the probability of h making a mistake in a randomly selected example that it has not seen before. The advantage of this technique is that good performance is achieved in the classification problems without incorporating knowledge from the problem area. Seeing the input data as two sets of vectors in an n-dimensional space, the SVM will construct a dividing superplane in this space, which will maximize the distance between the two sets. To calculate this distance, two parallel superplaces are constructed, one in each side of the separator superplane, which are "pushed" onto the two data sets. Intuitively, a good separation is achieved by the superplane having the greatest distance from adjacent data points of both sets, since in general the greater the distance the better the generalizer error of the classifier. Data classification is a common need in the field of Engineering Learning. Suppose some data points belonging to the two sets are given, and the goal is to decide which set a new data point will say.

## 5.9. Gaussian Naive Bayes

There is a large family of algorithms based on Bayes' theorem. The theorem assumes that the variables are independent of each other. This has several advantages as well as disadvantages. The algorithms of this family are fast and can easily predict large volumes of data, they can be used for binary categorization (classification) but also for multi-class categorization, they are excellent in text categorization problems and we use them in our work. But Bayes' theorem is based on the fact that variables are independent of each other, which can make it difficult to perform a model in cases where there is a direct correlation between the variables, as in our case where the interpretation of natural language depends on the set of words in a sentence. Hence the name Naive.

$$P(A|B) = \frac{P(A) * P(B|A)}{P(B)}$$

*Figure 11 – Gaussian Naïve Bayes equation*

[33]

The Bayes categorizer is used to estimate the probability that a new set / snapshot belongs to one of the predefined categories. The efficiency of this algorithm is quite high while at the same time it achieves high speeds. Bayes' theory, which, as he said, is a probabilistic approach to knowledge mining, aims to find the most likely hypothesis from a set of hypotheses or a set of training D, but also the knowledge that is possible in advance about the possibilities of the various h ∈ H.

Depending on the distribution function used by Naive Bayes we may see it under different names. For example with a Gaussian distribution (or Normal distribution), we have the Gaussian Naïve Bayes, with a polynomial distribution the Multinomial Bayes, Bernoulli Naïve Bayes for a Bernoulli distribution and others. Algorithms can be used for both binary categorization and multi-class categorization. [33]

## 5.10. Decision Trees Classification

Categorization and forecasting are two types of data analysis that can be used to build models that will describe important data classes or predict future data. While categorization predicts discrete values, forecasting models continuous value functions. So the forecast actually helps us make a decision. A predictive model for discovering knowledge from a database makes predictions of unknown or future values of some features, based on the other values that the features in the database have. Various methodologies have been used to categorize and generate predictive models. Some of these are:

- Decision Trees
- Bayesian (Bayesian Classifiers)

A widely used technical knowledge extraction technique used to solve categorization problems is that of Decision Trees, in which an attempt is made to approach a categorical objective function using the Divide and Conquer technique. The problem area is divided into areas of snapshots that have the same value for a feature, and the process is repeated retrospectively, thus representing the model produced as a decision tree. Figure 13 gives an example of a decision tree Decision trees are considered to be one of the most practical and simple approaches to Engineering Learning. A decision tree is essentially a structure of cubes. Each cube signifies a choice between different alternatives, and each leaf cube represents a classification or decision. The decision tree approach is very useful in categorization problems. [34]

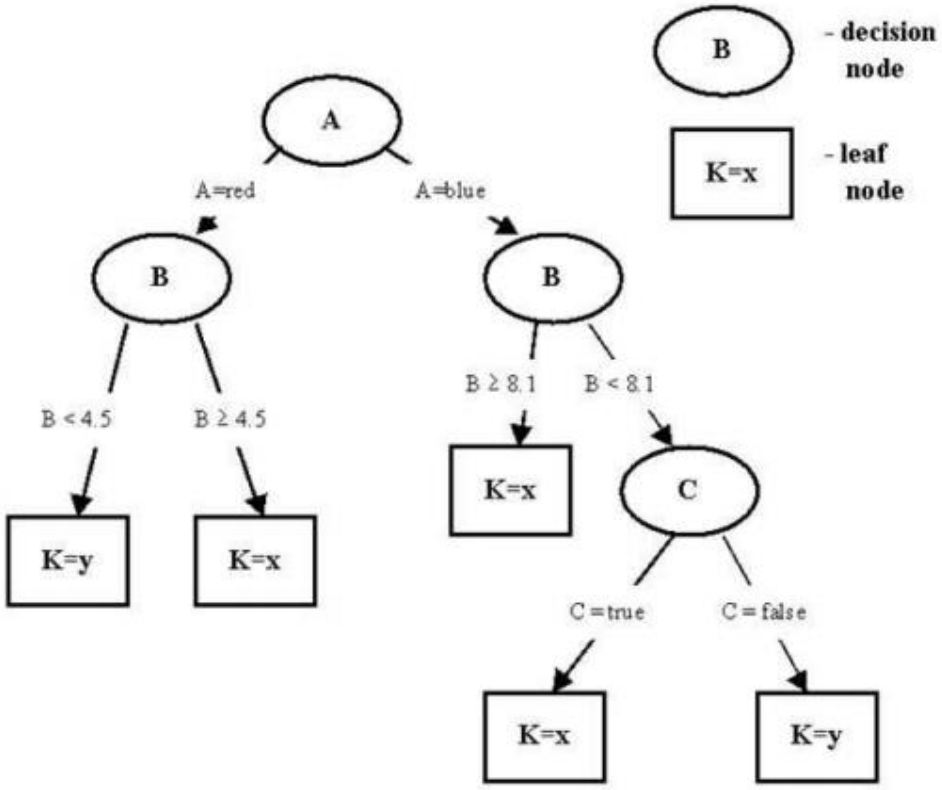



*Figure 12 - Decision Tree [34]*

## 5.11. Scoring functions

After we have obtained from our supervised algorithms their respective prediction outputs it is crucial to calculate somehow how accurate they were. To do that we will employ many different metrics which accomplice that final part of our program. There are many tools for assessing the accuracy of a model's predictions. They belong in three basic Application Programming Interfaces (APIs) categories:

- **Estimator score method**: Estimators have a score method providing a default evaluation criterion for the problem given by the user.
- **Scoring parameter**: Model selection and evaluation tools that use cross-validation take a scoring parameter that controls what metric they apply to the estimators evaluated.
- **Metric functions**: The metrics module implements functions measuring prediction error.

The module **sklearn.metrics** provides a fairly simple set of functions which measure a prediction error given a data and a prediction grid:

- functions from the set that end with **_score** in their name return a result, where the higher the value the better the measurement.
- functions from the set that end with **_error** or **_loss** in their name return a result, where in contrast with the previous ones the lower the value the better the measurement.

In our thesis we will implement the following six metrics. All of them are designed to operate given two sets of data array/sparse matrixes: **y_true** which is the set of Ground truth (correct) target values and **y_pred** which is the set of estimated targets or anomalies as returned by a classifier.

- **Accuracy classification score**: In a multilabel classification procedure, this metric function computes the subset accuracy: the set of labels predicted for a given sample must exactly match the corresponding set of labels in the y_true parameter.

- **Precision:** The precision metric is the ratio tp / (tp + fp) where tp being the number of true positives and fp the number of false positives. The precision analyzes the ability of the classifier not to label a negative sample as positive.

- **Recall:** The recall is the ratio tp / (tp + fn) where tp being the number of true positives and fn the number of false negatives. The recall analyzes the ability of the classifier to find in the given input all the positive samples.

- **F-measure score:** The F-beta score can be interpreted as a weighted mean of the precision and recall scores, with value from 0 for worst to 1 for best.

- **Support**: The amount of occurrences of each class in the array y_true.

- **Confusion matrix:** This function evaluates the accuracy of the classification. [35]

The **accuracy_score** function returns back a float value, which represents the accuracy percentage. The **precision_recall_fscore_support** function on the other hand is more robust, because it contains a tuple with all the above metrics combined. Finally, the **confusion_matrix** function returns a matrix where the true labels are stored in the first class and the predicted labels in the second class [[ tn fn] [fp tp]].

# Chapter 6 - Implementation

## 6.1. Tools and Programs Used

### Python

Python comes with packages and modules that are pre-installed and can be imported directly into our application and implemented. We will present the most relevant to the object we are interested in, and we will show their use with some simple examples. For a detailed presentation the reader can refer to the official documentation of the Python language and particular in the official documentation of sklearn library from where the content presented in the following subsections has been drawn.

### Anaconda

In order to run tests in the records of the EV charging enterprise's database, we used Python language, which is a very powerful tool in data analysis, and the execution was implemented in the Anaconda environment, where we used the Jupyter Notebook (Figure 14).

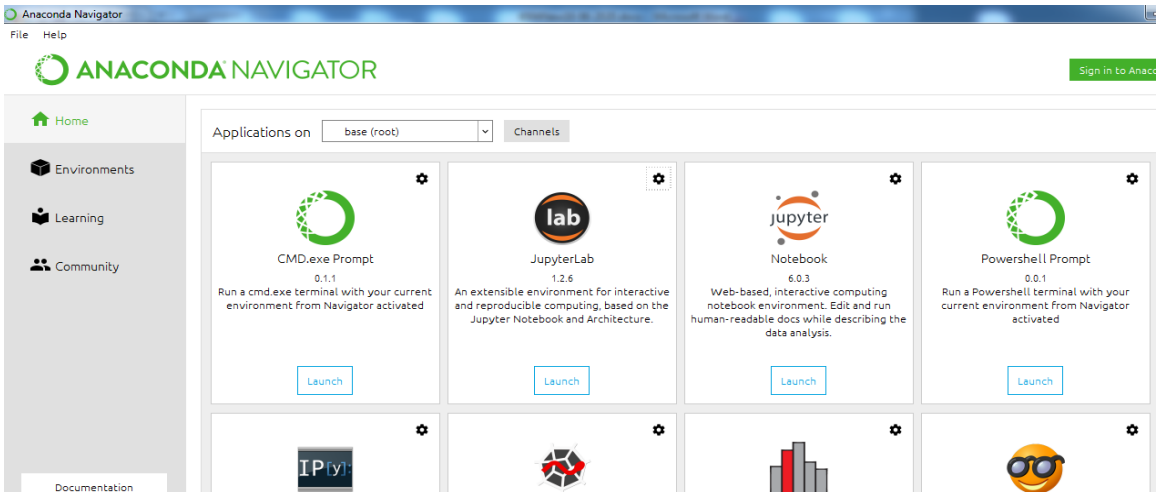



*Figure 13 - Anaconda development enviroment*

### 6.1.1 Necessary Python libraries

For scientific computing and computational modeling, we need additional collections of Python modules called libraries or packages. They are not part of the Python standard distribution. These allow us, for example, to create plots, operate on matrices, and use advanced numerical methods:

- **NumPy** (NUMeric Python): This library offers the ability to create matrixes and functions for linear algebra

- **SciPy** (SCIentific Python): This library includes many functions that are necessary for the implementation.

- **Matplotlib** (PLOTting Library): Necessary for creating plots and visualize data that will be extracted from isolation forest algorithm.

- **Pandas** The Python / pandas ecosystem available through the 'Anaconda' distribution (for version 3.x of the language) will be used and the interface will be the Jupyter Notebook which is being installed as part of Anaconda.

### 6.1.2 Plotting

For plotting our output images, we used in the case of Isolation Forest and KMeans the pyplot library, which is an excellent tool for this purpose. It has plotting capabilities of two and three dimensions. In the case of standard deviation we experimented with three available options Box, Scatter and Violin. After trying them on all three datasets with numerous modifications, we concluded that the Box option presented the best imaging. Standard deviation comes with its own library of tools for data presentation.

```
import matplotlib.pyplot as plt
import plotly.graph_objs as graph_objects
from plotly import figure_factory as FF
```

*Code snippet 1- Plotting libraries*

## 6.2. Code analysis and results examination

The data below is for a user case at a charging process that was recorded in the EV charging enterprise's database. The defined columns that we will use to plot our diagrams are described in section 4.2. From them some will not be included due to lack of values or repetitive context. The eleven columns that will be used are:

- Volume
- Duration
- Authentication ID
- Charge Point ID
- Calculated Cost
- Connector ID
- Value
- Value Type
- Reading Context
- Transaction ID
- Connection ID

### 6.2.1 Isolation Forest

IsolationForest.ipynb file contains the necessary code, in order to run the isolation method and find possible anomalies from the charging processes that have been recorded in the dataset of the EV charging enterprise. Anomalies are data patterns with different characteristics from normal occurrences. The discovery of such anomalies has significant relevance and often provides critical information in various cases. For example, fraudulent in the use of credit cards can be detected by found anomalies in credit card transactions. The discovery of a new star can be signified by an anomalous spot in an astronomy image. An unusual computer network traffic pattern could indicate the attempt of an unauthorized access from a third party. These examples use anomaly detection algorithms with high detection performance and fast execution to achieve their goals. [35]

In this section we introduce the parameters which were used in our program by the specified algorithm and break down the code piece by piece, to explain how it works.

| Parameters | Data type (Default value) | Short Description |
|---|---|---|
| **n_estimators** | **Integer (100)** | The number of base estimators |
| **max_samples** | **Integer/Float (auto)** | The number of samples to select to train each base estimator |
| **max_features** | **Integer/Float (1.0)** | The number of features to select to train each base estimator |
| **bootstrap** | **Boolean (False)** | True: Random subsets of individual trees are sampled with replacement.<br>False: sampling without replacement is performed. |
| **n_jobs** | **Integer (None)** | The number of jobs to run in parallel. |
| **verbose** | **Integer (0)** | Controls the verbosity of the tree building process |
| **random_state** | **Integer/ RandomState (42)** | Calculates the pseudo-randomness of the selection and split values for each branching step and each tree. |

*Table 7 - Isolation Forest parameters*

Through many trials with our algorithms optimal parameters have been set for our specified data. Some of them were left at their default values, as they work better with them, others had to be tweaked several times, like the number of iterations needed or the number of estimators necessary, to find the precise ones for the better predictions.

For this prediction model most of the default parameters remained the same, with the exception of "**contamination**" with value 0.2 instead of "auto" and **random_state** with value 42. The **random_state** parameter is a very significant one for many algorithms and tools. How you choose to define it affects the behavior of the program greatly. Every time you run something without specifying **random_state**, you will get a different result, this is the expected behavior. On the other hand if you use **random_state**="some_number", then you can guarantee that the output of the first execution will be equal to the output of the second one, i.e. your split will always remain the same. It doesn't matter what the actual number is i.e. 42, 0, 21. The important thing to carry in mind is that every time you use it with a specified number, you will always get the same output.

**Explanation of code**

The implemented code is presented below. First of all, it is necessary to link our project with the appropriate libraries:

```
import pandas as pd
import numpy as np
import scipy
import time
import sys
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

```
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import IsolationForest
```

*Code snippet 2- Libraries imported for Isolation Forest*

Next, we define a function named **time_convert**. This function was used in order to convert time format into seconds. In csv the dataset Duration is string type so it is formatted like "10:08:00". Consequently, this function takes as parameter time duration and returns in the main program the total seconds.

```
def time_convert(x):
  h,m,s = map(int,x.split(':'))
  return (h*60+m)*60+s
```

*Code snippet 3- Duration column data conversion function*

After that we must read the CSV file with the essential records. The necessary code is:

```
year = '2020'

data = pd.read_csv('dataset'+year+'.csv', error_bad_lines=False, nrows = 250000)
data.Duration=data.Duration.astype(str)
data.AuthenticationId=data.AuthenticationId.astype(str)
```

*Code snippet 4- File containing data reading*

We create a global variable year so it can be used everywhere in our code, which is much easier. The parameter **nrows** specifies how many records will be read from the given file.

Next, we convert all data columns in **numpy ndarrays** (n dimensional arrays) and we use **reshape**(-1,1) for a single feature / column. We also define the arrays with type float (**astype(np.float)**).

Since the AuthenticationID column is an array of strings it can't be used in our dataset, so we will use the **CountVectorizer** tool from **sklearn.feature_extraction.text** library. It converts a collection of texts/strings to a matrix of token counts. This implementation will produce a sparse representation of the counts using **scipy.sparse.csr_matrix.** Similarly we use this tool on the TransactionID column, because its values do not provide in their current form any relative significance.

```
    duration = np.array(data['Duration'])
    duration_array = []
    for charging_time in duration:
        try:
            duration_array.append(time_convert(charging_time)) # convert dataset value to
seconds
        except: # in case of faulty data input
            duration_array.append(0)
    duration_array = np.array(duration_array).reshape(-1, 1).astype(np.float)
    #============================================================

    volume_array = np.array(data['Volume']).reshape(-1, 1).astype(np.float)
    #============================================================

    authentication_id = data['AuthenticationId']
    vectorizer = CountVectorizer()
    authentication_id_vector = vectorizer.fit_transform(authentication_id)
    authentication_id_vector.shape
    indicies = authentication_id_vector.indices.tolist() # collect indicies
    del indicies[len(authentication_id):] # remove extra rows if produced
    authentication_id_array = np.array(indicies).reshape(-1,1).astype(np.float)
    #============================================================

    charge_point_id_array = np.array(data['ChargePoint_ID']).reshape(-1,
1).astype(np.float)
    #============================================================
    connector_id = np.array(data['ConnectorId'])
    connector_id_array = np.where(np.isnan(connector_id), 0, connector_id) # replace
null values with zeros
    connector_id_array = np.array(connector_id_array).reshape(-1, 1).astype(np.float)
    #============================================================
    calculated_cost_array = np.array(data['CalculatedCosts']).reshape(-1,
1).astype(np.float)
    #============================================================
    value_array = np.array(data['Value']).reshape(-1, 1).astype(np.float)
    #============================================================




    # For the columns below we check the unique values by converting the array to a set.
    # if unique values are more than one we can use this column in our data plotting
    # all columns had more than one unique value so they will be included in the dataset
    # print(set(data['ValueType'])) / print(set(data['ReadingContext']))
```

```
    # print(set(data['TransactionId'])) / print(set(data['Connection_ID']))

    value_type_array = np.array(data['ValueType']).reshape(-1, 1).astype(np.float)
    #============================================================

    reading_context_array = np.array(data['ReadingContext']).reshape(-1,
1).astype(np.float)
    #============================================================

    transaction_id = data['TransactionId']
    vectorizer2 = CountVectorizer()
    transaction_id_vector = vectorizer2.fit_transform(authentication_id)
    transaction_id_vector.shape
    indicies = transaction_id_vector.indices.tolist()
    del indicies[len(transaction_id):]
    transaction_id_array = np.array(indicies).reshape(-1,1).astype(np.float)

    connection_id_array = np.array(data['Connection_ID']).reshape(-1, 1).astype(np.float)
```

*Code snippet 5 – All columns data manipulation process*

Our next step is controlling more efficiently our data. We stack them all together. The **hstack** function is used to stack the sequence of input arrays horizontally (column wise) to make a single array.

```
    dataset_11_features = np.hstack((duration_array,
                                     volume_array,
                                     authentication_id_array,
                                     charge_point_id_array,
                                     calculated_cost_array,
                                     connector_id_array,
                                     value_array,
                                     value_type_array,
                                     reading_context_array,
                                     transaction_id_array,
                                     connection_id_array
                                    ))
```

*Code snippet 6 – Data unity*

Using the Isolation Forest algorithm every point in the dataset is separated. In the case of 2D dimensions a line is created randomly and it attempts to single out a point. An anomaly point can be separated fairly easy in a few steps, while normal points can take

significantly more steps to be segregated. Recently h2o's Isolation Forest was also made available which is more scalable on high volume datasets, something that could be worth exploring. [30]

PCA is used to create a data frame for 3D and later 2D plotting. The fit_transform is then used to apply the dimensionality reduction on dataset_11_features. Pandas DataFrame is two-dimensional size-mutable, data structure with labeled axes (rows and columns).

```
pca = PCA(n_components=3, random_state=None)  # 3 dimensions
dataset_11_features[np.isnan(dataset_11_features)] = 0 # replace nan values with 0
dataset_3_features_after_pca = pca.fit_transform(dataset_11_features)
pca_3_columns_dataframe = pd.DataFrame(dataset_3_features_after_pca)
```

*Code snippet 7 – PCA used for 3 dimensions plotting*

Next, after we have imported the Isolation Forest library, we give the appropriate parameters shown in Table 7:

```
IF = IsolationForest(n_estimators=100, max_samples='auto', contamination=float(.20), \
                     max_features=1.0, n_jobs=-1, random_state=42, verbose=0)
# fit function create estimator
IF.fit(pca_3_columns_dataframe)
```

*Code snippet 8 – Isolation Forest parameters*

A sudden spike or fall in a metric output is an irregular behavior and both cases need to be handled accordingly. Detection of these anomalies can be solved by utilizing supervised algorithms if we have information on anomalous behavior, but before that without feedback it's difficult to identify those points. So we model this as an unsupervised problem using algorithms like Isolation Forest, KMeans, etc.

```
# predict function finds if a particular sample is an outlier or not.
prediction = IF.predict(pca_3_columns_dataframe)
pca_3_columns_dataframe['Anomaly'] = pca_3_columns_dataframe[0] > 110000
outliers = pca_3_columns_dataframe.loc[pca_3_columns_dataframe['Anomaly']]
```

```
    outlier_index = list(outliers.index)
    print("Outlier indexes:\n")
    print(outlier_index)
    print(pca_3_columns_dataframe['Anomaly'].value_counts())
```

*Code snippet 9 – Prediction of outliers with Isolation Forest*

Now here we have metrics on which we have classified anomalies based on the prediction of the algorithm. We will try to visualize the results and check if the classification makes sense. We also normalize the metrics with the help of **StandardScaler** for plotting purposes.

Plotting of three dimensions.

```
    scaler = StandardScaler()
    x = scaler.fit_transform(pca_3_columns_dataframe)
    x_dimensional_reduction = pca.fit_transform(x) # 3 dimensional reduction

    figure = plt.figure()
    axis = figure.add_subplot(111, projection='3d')
    # Plotting of the compressed data points
    axis.scatter(x_dimensional_reduction[:, 0], x_dimensional_reduction[:, 1],
zs=x_dimensional_reduction[:, 2],
            lw=1, s=5, label="inliers", c="blue")
    # Plotting of outliers
    axis.scatter(x_dimensional_reduction[outlier_index,0],
            x_dimensional_reduction[outlier_index,1],
            x_dimensional_reduction[outlier_index,2],
            lw=2, s=40, label="outliers", c="red", marker="x")
    axis.legend()
    plt.title("Isolation Forest")
    plt.show()
```

*Code snippet 10- Isolation Forest 3D plot*

The same procedure is used for the plotting of a two dimensional grid by passing in the PCA the n_components parameter with a value of 2.

```
    pca = PCA(n_components=2)  # 2 dimensions

    pca.fit(pca_3_columns_dataframe)

    pca_2_columns_dataframe =
pd.DataFrame(pca.transform(pca_3_columns_dataframe))

    Z = np.array(pca_2_columns_dataframe)

    figuresize = (10, 5)

    plt.figure(figsize=figuresize)

    plt.contourf( Z, cmap=plt.cm.Blues_r)

    b1 = plt.scatter(pca_2_columns_dataframe[0], pca_2_columns_dataframe[1], s=30,
label="normal points", c='blue')

    b1 = plt.scatter(pca_2_columns_dataframe.iloc[outlier_index,0],
pca_2_columns_dataframe.iloc[outlier_index,1], s=30,

                label="predicted outliers", c='red', edgecolor="red")

    plt.legend(loc="upper right")

    plt.xlabel('P1')

    plt.ylabel('P2')

    plt.show()
```

*Code snippet 11- Isolation Forest 2D plot*

### *Results and Plotting Outputs*

### *Year 2018*

In this section we analyze the predicted points by our algorithm in the dataset of the year 2018 with the insertion of 250000 charging records. In the course of the thesis, all plotting images will be visualized by two colors. Blue will be used for the indication of normal points and red, being an alarming type of color, will be used for the outliers predicted. Our data frame with eleven features, meaning 11 dimensions, cannot in its state be represented in a two or three dimensional grid. But with the help of PCA the values of the given data frame are formed in a way that they can.

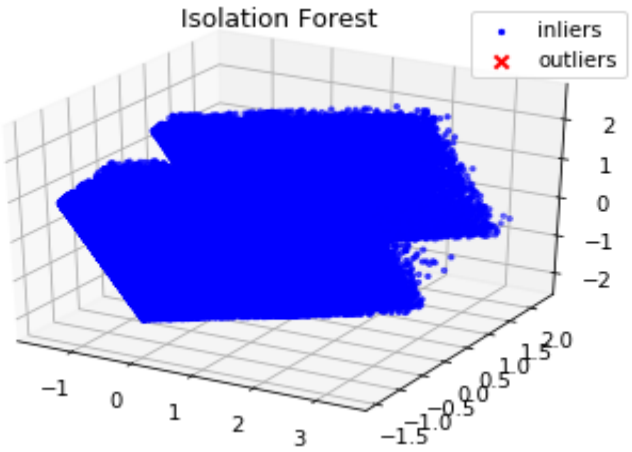



*Figure 14 – 3D plotting output of Isolation Forest in year 2018 dataset*

As shown in Figure 15, produced by three dimensional plotting all the points given, every point in our display is blue, which means that no anomalies were found. The points are very close to each other and form a unity. That gives us a conclusion that their metric values for all columns were very similar in their total.

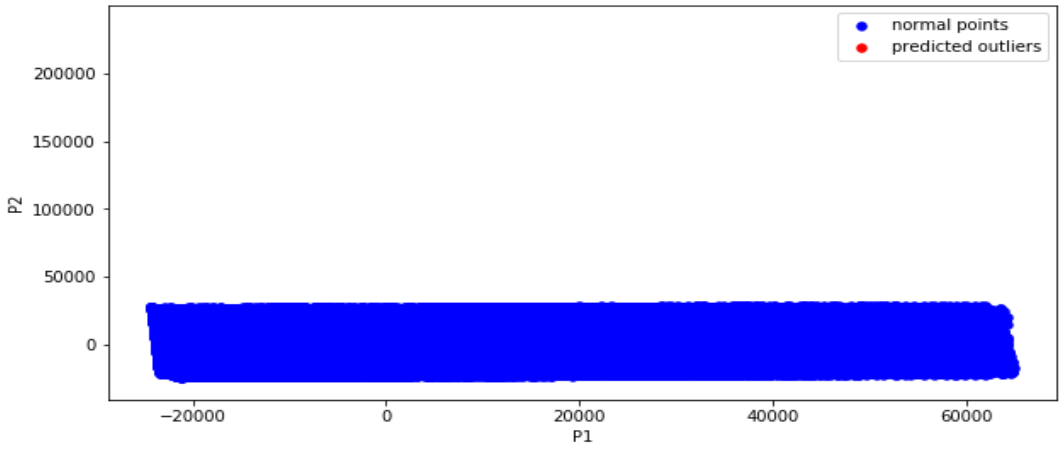



*Figure 15 – 2D plotting output of Isolation Forest in year 2018 dataset*

Similarly in the case of two dimensional plotting as seen in Figure 16, the same conclusion can be derived. All points can be seen more clearly, that have the same range of values. The program was terminated early due to lack of anomalies.

### *Year 2019*

In this section we analyze the predicted points by our algorithm in the dataset of the year 2019 with the insertion of 250000 charging records.

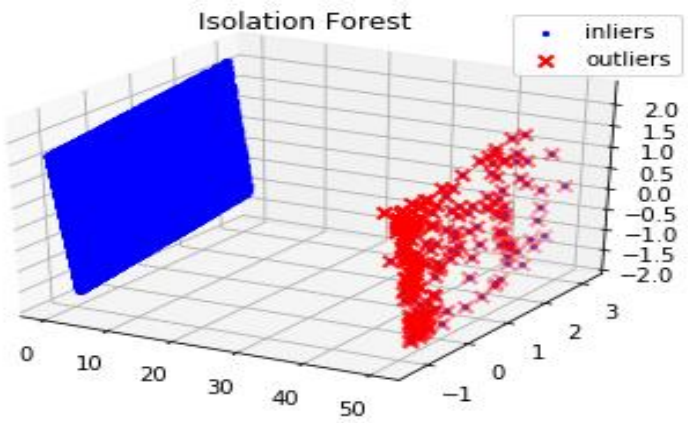



*Figure 16– 3D plotting output of Isolation Forest in year 2019 dataset*

Finally, in Figure 17 we can observe some outliers and have an insight for how far these points are separated from each other. The gap is big enough to press the two groups at the sides. The normal points are extremely close to the initial positions.

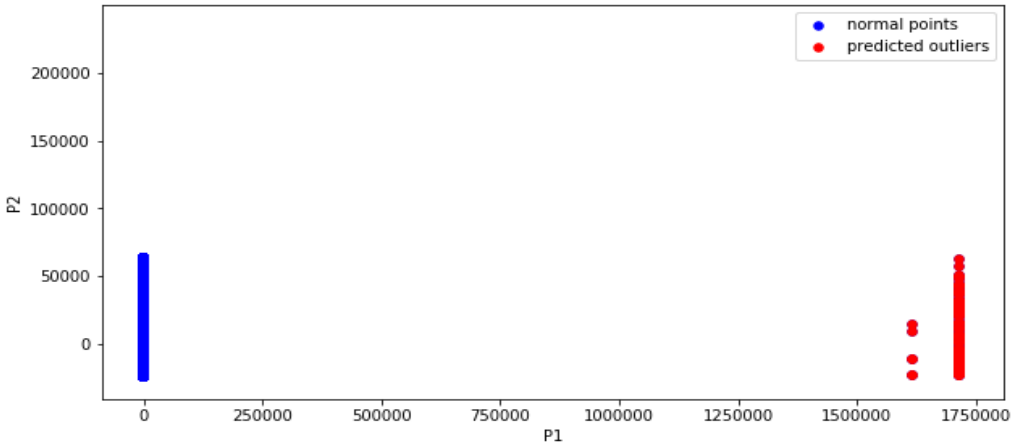



*Figure 17– 2D plotting output of Isolation Forest in year 2019 dataset*

Likewise, as visualized even better in Figure18 in 2D plot the anomalies create at the othe r end of the grid two collections. Each of these outliers is a record that its value for some reason was pushed significantly away from the sphere of what is considered normal in the complex of recorded processes given to us for this year. It should be noted that what is co nsidered normal for one year is not the same as the rest ones. Parameters can be altered in the future or eventually the set of features might increase to increase the charging security stability.

### *Year 2020*

Finally, in this section we analyze the predicted points by our algorithm in the dataset of the year 2020 with the insertion of 250000 charging records.

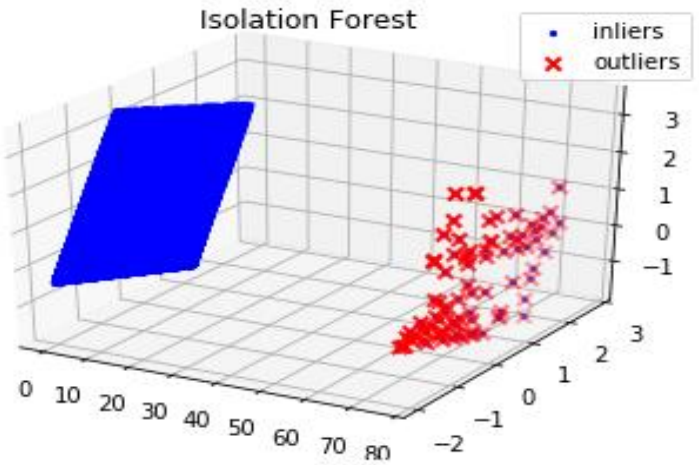



*Figure 18– 3D plotting output of Isolation Forest in year 2020 dataset*

In Figure 19, the dataset of year 2020 also predicted a fair number of outliers. Their group is less than the year's 2019 and sparser, but still is pictured in a matching way, which is something reassuring in the capabilities of our tools in plotting.

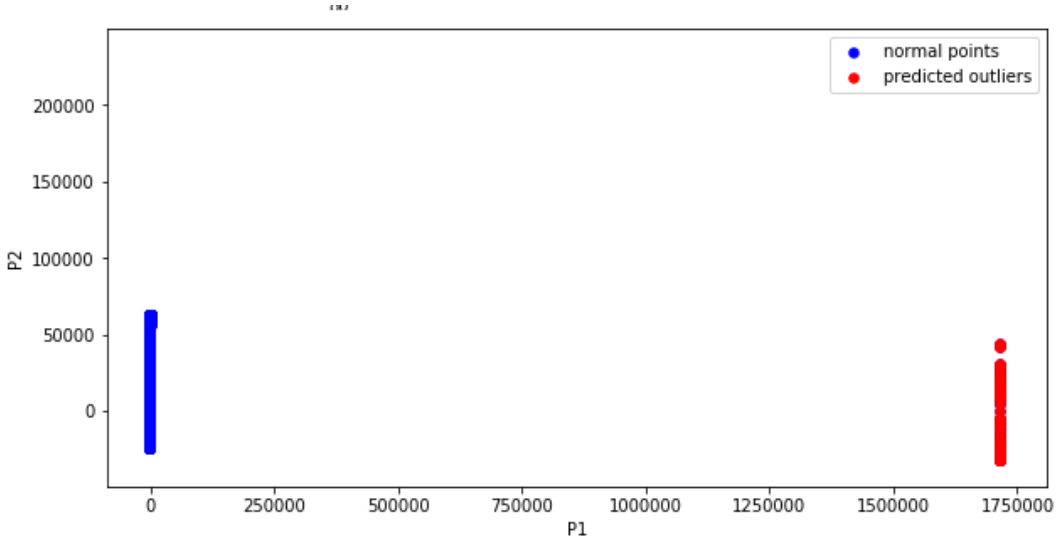



*Figure 19– 2D plotting output of Isolation Forest in year 2020 dataset*

In Figure 20 the outliers, probably because they are less in number than before create a single group in the grid. On the other hand, perhaps we should consider that the abnormal points in the year's 2019 dataset that break from the bulk and form two separate sets hide something of importance.

This concludes the section of the Isolation Forest and our remarks on his outputted results and we will continue with the next algorithm.

### 6.2.2 KMeans

In this section we introduce the parameters which were used by KMeans and examine the code lines.

| Parameters | Data type (Default value) | Short Description |
|---|---|---|
| **n_clusters** | **Integer (8)** | The number of clusters to form as well as the number of centroids to generate. |
| **init** | **Method (k-means++)** | Method for initialization |
| **n_init** | **Integer (10)** | Number of time the KMeans algorithm will be run with different centroid seeds. |
| **max_iter** | **Integer (300)** | Maximum number of iterations of the KMeans algorithm for a single run. |
| **n_jobs** | **Integer (None)** | The number of jobs to run in parallel. |
| **verbose** | **Integer (0)** | Controls the verbosity of the tree building process |
| **random_state** | **Integer/ RandomState (None)** | Calculates the pseudo-randomness of the selection and split values for each branching step and each tree. |

*Table 8 - KMeans parameters*

In the case of Kmeans also many of the parameters stayed at their default state. We changed the "**init**" parameter from "k-means++" to "random", because with the first one the program would choose for us the number of clusters by its own. For the program though to be more accurate we decided to check the proper number by using the elbow heuristic method which runs KMeans clustering for a range of values to determine the best one. Also, **random_state** was given a specific number.

**Explanation of code**

Code found in KMeans.ipynb

Most of the steps taken to ensure a working program for the case of Isolation Forest algorithms are repeated in the context of KMeans.

The appropriate libraries must be imported.

```
import pandas as pd
import numpy as np
import scipy
import time
import sys
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

*Code snippet 12- Libraries imported for KMeans*

After that we read from the file the records that we wish to insert in our dataset and we manipulate our data columns in the manner presented above in the Isolation Forest analysis section of the thesis.

Using KMeans means we will operate with clusters. To find the optimal number of clusters to control our data, we employ the **Elbow** heuristic method. The **Elbow** heuristic method runs KMeans clustering on the dataset for a range of values for x (i.e. 1 to 14) and then for each value of x computes the averages score for all clusters. The distortion score (default) is computed, the sum of square distances from each point to its assigned center.

```
    cluster_range = range(1, 14)

    kmeans = [KMeans(n_clusters=i) for i in cluster_range]

    score = [kmeans[i].fit(dataset_3_features_after_pca).score(dataset_3_features_after_pca) for i in range(len(kmeans))]

    plt.plot(cluster_range,score)

    plt.xlabel('Number of Clusters')

    plt.ylabel('Score')

    plt.title('Elbow Curve')

    plt.show()
```

*Code snippet 13- Finding optimal number of clusters for KMeans using Elbow heuristic*

Next, we give our algorithm the appropriate parameters shown in the Table 8:

```
kmeans = KMeans(n_clusters=cluster_num, init='random', n_init=10, max_iter=300, \
                    n_jobs=-1, random_state=42, verbose=0)
# fit function computes k-means clustering.
kmeans.fit(pca_3_columns_dataframe)
```

*Code snippet 14- KMeans parameters*

The predict function finds the closest cluster each sample in data frame belongs to.

```
    # predict function finds if a particular sample is an outlier or not.
    prediction = IF.predict(pca_3_columns_dataframe)
    pca_3_columns_dataframe['Anomaly'] = pca_3_columns_dataframe[‘A’] > 110000
    outliers = pca_3_columns_dataframe.loc[pca_3_columns_dataframe['Anomaly']]
    outlier_index = list(outliers.index)
    print("Outlier indexes:\n")
    print(outlier_index)
    print(pca_3_columns_dataframe['Anomaly'].value_counts())
```

*Code snippet 15- Prediction of outliers with KMeans*

Finally, it is time to plot our clusters in three dimensions.

```
    scaler = StandardScaler()
    x = scaler.fit_transform(pca_3_columns_dataframe)
    x_dimensional_reduction = pca.fit_transform(x) # 3 dimensional reduction

    figure = plt.figure()
    axis = figure.add_subplot(111, projection='3d')
    # Plotting of the compressed data points
    axis.scatter(x_dimensional_reduction[:, 0], x_dimensional_reduction[:, 1],
zs=x_dimensional_reduction[:, 2],
            lw=1, s=5, label="inliers", c="blue")
    # Plotting of outliers
    axis.scatter(x_dimensional_reduction[outlier_index,0],
            x_dimensional_reduction[outlier_index,1],
            x_dimensional_reduction[outlier_index,2],
            lw=2, s=40, label="outliers", c="red", marker="x")
    axis.legend()
    plt.title("Isolation Forest")
    plt.show()
```

*Code snippet 16- KMeans 3D plot*

And here for a two dimension representation.

```
    pca = PCA(n_components=2)  # 2 dimensions
    pca.fit(pca_3_columns_dataframe)
    dataset_2_features_after_pca = pca.fit_transform(pca_3_columns_dataframe)
    pca_2_columns_dataframe = pd.DataFrame(pca_3_columns_dataframe)
    Z = np.array(pca_2_columns_dataframe)
    figuresize = (10, 5)
    plt.figure(figsize=figuresize)
    plt.contourf( Z, cmap=plt.cm.Blues_r)
    b1 = plt.scatter(pca_2_columns_dataframe['A'], pca_2_columns_dataframe['B'],
s=30, label="normal points", c='blue')

    b1 = plt.scatter(pca_2_columns_dataframe.iloc[outlier_index,0],
pca_2_columns_dataframe.iloc[outlier_index,1], s=30,
                     label="predicted outliers", c='red', edgecolor="red")
    plt.legend(loc="upper right")
    plt.xlabel('P1')
    plt.ylabel('P2')
    plt.title(str(cluster_num) + ' Cluster KMeans')
    plt.show()
```

*Code snippet 17- KMeans 2D plot*

### *Elbow Curve Plotting*

Here the Elbow heuristic is explained in the following plot. The curve begins to bend afte r it reaches the optimal number of clusters that it calculated, between the range of values it was given. When testing with specific number of records, the Elbow heuristic found mo re preferable with less data the two cluster solution and as they increased the three cluster one. Specifically when records were 100000 and more. It should be noted that finding the right amount of clusters is very time consuming task, especially when working with so m any data and takes several seconds. That is why this time will not be taken under consider ation when comparing the execution times of each model, as it would not be fair for the KMeans clustering model.

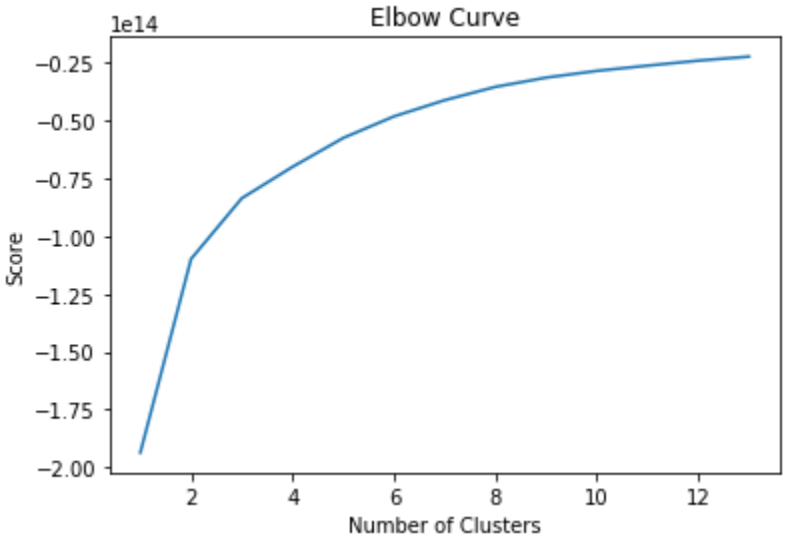



*Figure 20– Elbow heuristic curve output*

When we graph the plot, we see that the graph levels off rapidly after 3 clusters, implying that addition of more clusters will not explain much more of the variance in our relevant variable.

### *Results and Plotting Outputs*

### *Year 2018*

Here we will discuss the resulting imaging created by our algorithm in the year 2018 with the insertion of 250000 charging records and try to compare them with the images created by the Isolation Forest algorithm. The more these plots are similar the easier for us to be certain as to how accurate the where.

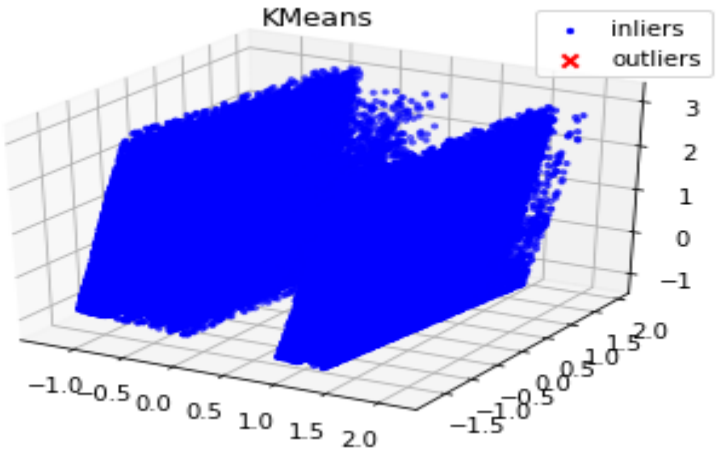



*Figure 21– 3D plotting output of KMeans in year 2018 dataset*

KMeans like its counterpart Isolation Forest does not project any outliers in Figure 22. This is good because our models appear to be in sync. The plot is different here with the creation of three clusters, as requested, with two of them aligned very closely to each oth er.

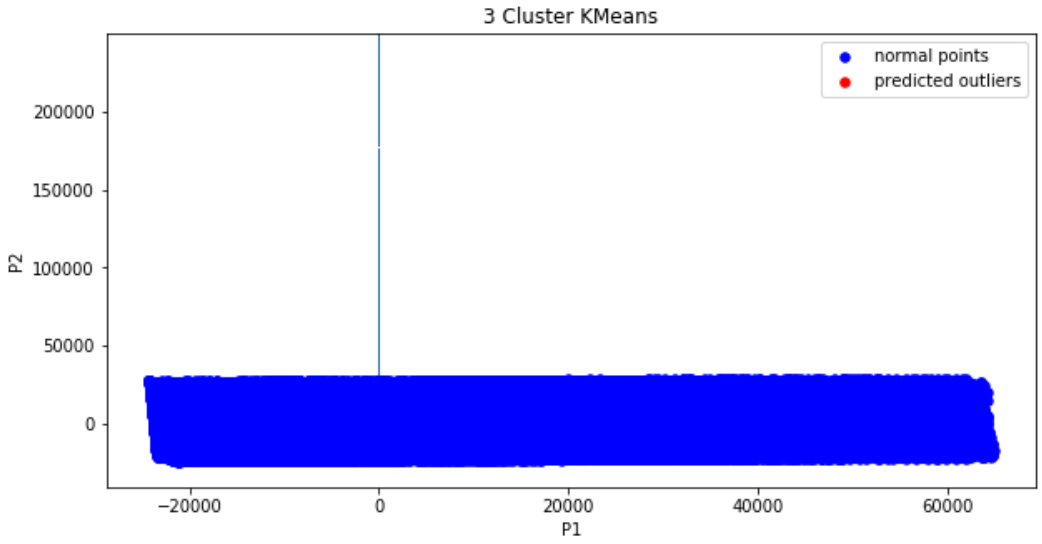



*Figure 22– 2D plotting output of KMeans in year 2018 dataset*

The resulted plot also matches the previous case by further pointing how closely aligned are the three clusters shown in the 3D example. There seems to be a single line originating from the zero point in the P1 axis, but that probably is no importance and is from how the algorithm handles the data.

### *Year 2019*

Resulting plots created by our algorithm in the year 2019 with the insertion of 250000 charging records.

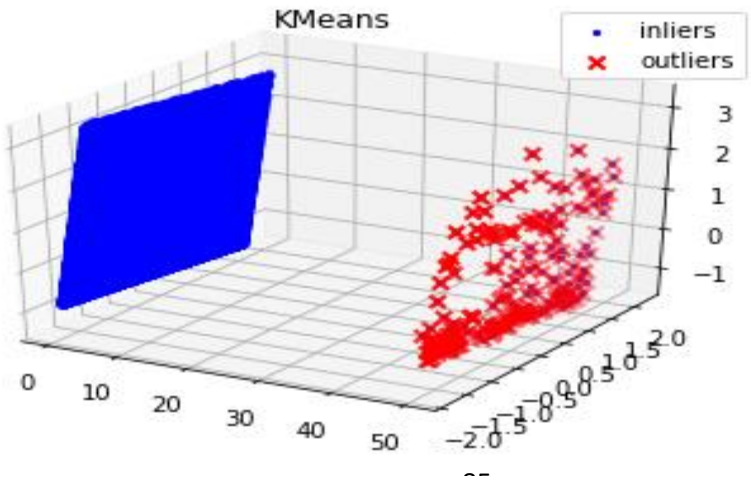



*Figure 23– 3D plotting output of KMeans in year 2019 dataset*

By reviewing the resulted 3D grid in comparison with one predicted by the previous algor ithm we can safely say that the results are quite defining. The grid is split in two clusters. The normal points operate as one cluster entity as their metrics appear to be in the same s pecter and the other cluster of course being the outliers. One discernable difference is that in this projection the outliers tend to be drawn more to the bottom of the grid, as opposed to the Isolation Forest instance where they tend to the left.

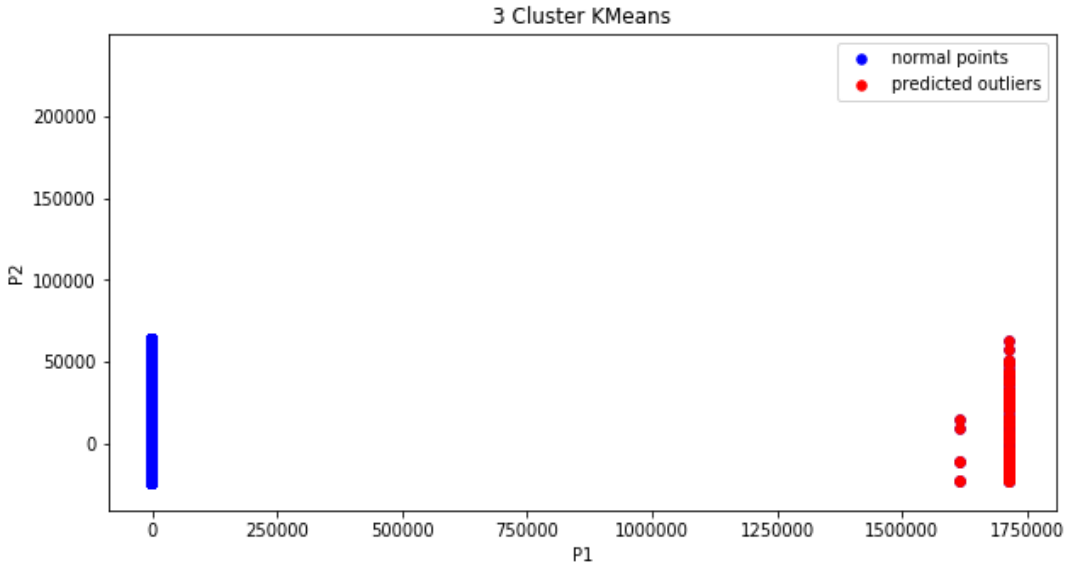



*Figure 24– 2D plotting output of KMeans in year 2019 dataset*

Unfortunately, the 2D imaging does not provide any more insight than we have already acquired beforehand. The outliers here also appear to be forming their own cluster. Most of them have values in unison with the normal points in the P2 axis, with some outcasts o f their own. Specifically, the two outliers peaking above the cluster, which can also be see n in the 3D plot grid and the other small group with smaller values in the P1 axis, which c an also be seen in the 3D grid with some difficulty, formed right above the center of the c luster.

### *Year 2020*

Finally, here we can observe the resulting plots created by our algorithm in the year 2020 with the insertion of 250000 charging records.

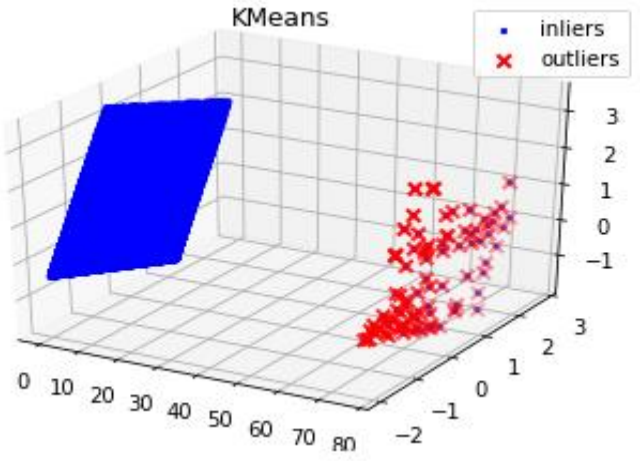



*Figure 25– 3D plotting output of KMeans in year 2020 dataset*

In 3D imaging like in our previous model, the red points tend to the bottom of the grid, followed by a small gap. Since they tend aligning fairly close, we can assume that there is a common characteristic that causes the majority of them. Another small cluster can be seen as we go up, probably caused by some other characteristic or perhaps a combination of two.

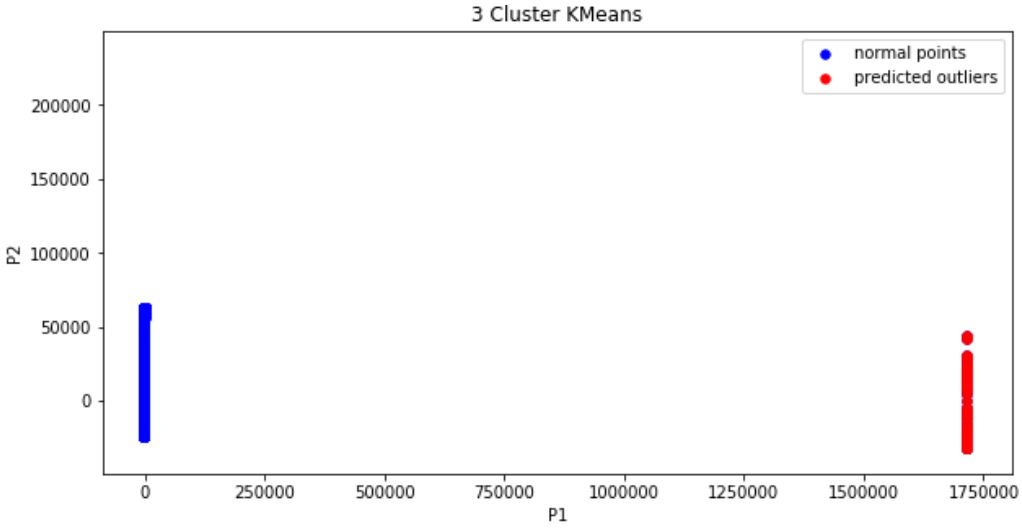



*Figure 26- 2D plotting output of KMeans in year 2020 dataset*

In 3D plot, similar conclusion can be made. In our normal area the cluster seems to inflate slightly to the end of the P2 axis and the outlier cluster, again seems to have a few outcasts at the top, like the year 2019. So it is definitely something that should be looked further into. Since, outliers are rare in what we have procured each distinct characteristic of them can be used as a tool to determine its true nature. Whether it was simply a power out during the charging process, a transaction not concluded due to an error in the system or a malicious intent to cause a short circuit.

This concludes the section of the KMeans and our remarks on his outputted clustering res ults and we will continue with the next and final model.

### 6.2.3 Standard Deviation

Code found in StandardDeviation.ipynb

As is the case for the previous two unsupervised models the flow of execution regarding the data fetching and manipulation remains unchanged.

The appropriate libraries must be imported as usual.

```
import pandas as pd
import numpy as np
import scipy
import time
import sys
import plotly.offline as py # version 3.10.0
import plotly.graph_objs as graph_objects
from plotly import figure_factory as FF
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import CountVectorizer
```

*Code snippet 18 - Libraries imported for Standard Deviation*

Here we will give our model the anomalies as shown above and try to extrapolate the standard deviation, a measure of the spread of a distribution, of the array elements and also the average, median, max and minimum of the array elements.

```
pca = PCA(n_components=2, random_state=None)  # 2 dimensions
# fit_transform to apply the dimensionality reduction on dataset_11_features
dataset_11_features[np.isnan(dataset_11_features)] = 0 # replace nan values with 0
dataset_2_features_after_pca = pca.fit_transform(dataset_11_features)
# Pandas DataFrame is two-dimensional size-mutable,
# data structure with labeled axes (rows and columns)
pca_2_columns_dataframe = pd.DataFrame(dataset_2_features_after_pca)
pca_2_columns_dataframe.columns = ['A', 'B']

pca_2_columns_dataframe['Anomaly'] = pca_2_columns_dataframe['A'] > 110000
outliers = pca_2_columns_dataframe.loc[pca_2_columns_dataframe['Anomaly']]
outlier_index = list(outliers.index)
print("Outlier indexes:\n")
print(outlier_index)
```

*Code snippet 19 – Standard Deviation anomalies process*

```
mean = np.mean(pca_2_columns_dataframe[['A','B']])

standard_deviation = np.std(pca_2_columns_dataframe[['A','B']])

print(("The mean is %r") %(mean))
print(("The standard deviation is %r") %(standard_deviation))

median = np.median(pca_2_columns_dataframe[['A','B']])
maximum = np.max(pca_2_columns_dataframe[['A','B']])
minimum = np.min(pca_2_columns_dataframe[['A','B']])

print(("The median is %r") %(median))
print(("The maximum is %r") %(maximum))
print(("The minimum is %r") %(minimum))
```

*Code snippet 20- Calculation of mean, standard deviation, median, maximum and minimum*

In the case of Standard Deviation the plotting is done with the use of other tools. Here the **plotly** library is in charge of creating the desired outputs. We will use the Box option to plot our points. Unfortunately, this library is not ideal for 3D plotting, so only 2D plotting will be shown afterwards in the results section. The library creates instead of the standard icons, html files to display its graphs. The convenient aspect of this is that files can be stored for posterity and be viewed again, whenever the analyst wants. One drawback is that the size of the html files it creates is too large and takes some time to render especially with so many records.

```
trace = graph_objects.Box(
    x=x,
    y=y,
    name="All Points",
    jitter=0.3,
    pointpos=-1.8,
    boxpoints='all',
    boxmean='sd' # represent mean and standard deviation
)


layout = graph_objects.Layout(
    height=700,
    width=700,
    yaxis=dict(
        title='Data deviation',
        zeroline=False
    ),
)

data1 = [trace]

figure = graph_objects.Figure(data=data1, layout=layout)
py.plot(figure, filename='BoxOutput'+year+'.html')
```

*Code snippet 21- Standard Deviation 2D plot*

### *Results and Plotting Outputs*

### *Year 2018*

Resulting plots created by our algorithm using **plotly** in the year 2018 with the insertion of 250000 charging records.

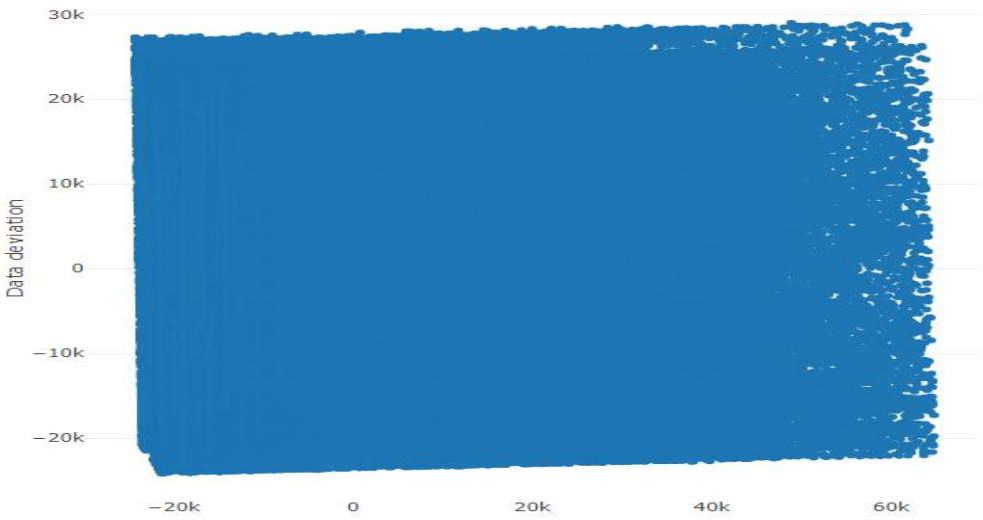



*Figure 27– 2D plotting output of Standard Deviation in year 2018 dataset*

For the third and final time the plot produced by our model and tools for this particular year was null in outliers. The **plotly** library sets the graphical limits accordingly with the maxim values of its given set that is why it is seen here as a block or a sheet of points, but it does not deviate much from the previous resulted plots of this year.

| Metrics calculated | Column ‘A’ value (float) | Column ‘B’ value (float) |
|---|---|---|
| **Mean** | 3.586858e-12 | -1.212847e-11 |
| **Standard deviation** | 20841.358262 | 14352.629523 |
| **Median** | -3813.7450044019415 | -3813.7450044019415 |
| **Maximum** | 64791.914162 | 29016.413739 |
| **Minimum** | -24254.771485 | -24204.020309 |

*Table 9 – Standard Deviation model metric results for year 2018*

The production of these values for the standard deviation metric in accordance with minimum and maximum values suggests that the points spread on the grid are not very far between them, so their metrics are similar which is correct based on what we have already seen.

### *Year 2019*

Resulting 2D plot created by our algorithm in the year 2019 with the insertion of 250000 charging records.

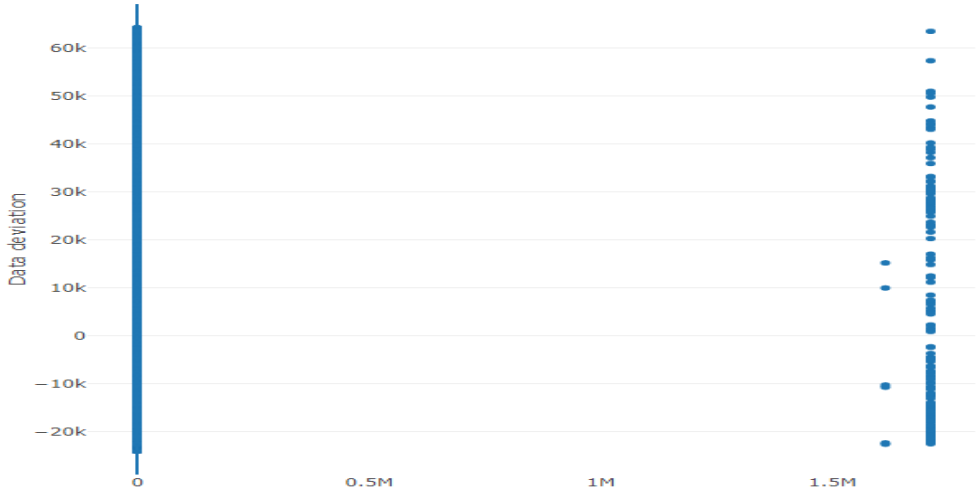



*Figure 28– 2D plotting output of Standard Deviation in year 2019 dataset*

Although **plotly** does not provide 3D plots, it does provide better 2D visualization in comparison with our two previously discussed models. Since, the grid does not spread as much we can observe more clearly each atypical point, even those that are closer to each other. Results remain unchanged for all three models, due to the ease of classifying the produced abnormalities.

| Metrics calculated | Column ‘A’ value (float) | Column ‘B’ value (float) |
|---|---|---|
| **Mean** | 6.806494e-14 | -2.098631e-12 |
| **Standard deviation** | 47898.545820 | 21081.661653 |
| **Median** | -1314.552317630138 | -1314.552317630138 |
| **Maximum** | 1.712933e+06 | 6.430870e+04 |
| **Minimum** | -1475.512299 | -24013.973323 |

*Table 10 – Standard Deviation model metric results for year 2019*

Here, the minimum values have decreased in comparison with previous year, but the maximum values have skyrocketed. That separation is also shown in the standard deviation which has been more than doubled. These maximum values of course are where the anomalies reside. By using this technique we can easily classify the points with such values and separate them. For example, a charging that has much higher energy consumption than usual will have its Volume feature increased and correspondingly, its standard deviation between the other points will also increase. Thus, providing to our model a clear distinction and easier identification. With that information, we should be able to terminate the charging before it causes wasteful energy consumption, or damage in the meter connected.

### *Year 2020*

Resulting 2D plot created by our algorithm in the year 2020 with the insertion of 250000 charging records.


*Figure 29– 2D plotting output of Standard Deviation in year 2020 dataset*

Finally, here we can see even clearer the gap between the outliers as discussed in the Kmeans section for this year. The few points at the top are even more apart from the rest than we originally thought.

| Metrics calculated | Column 'A' value (float) | Column 'B' value (float) |
|---|---|---|
| **Mean** | -7.905764e-13 | -2.843768e-12 |
| **Standard deviation** | 31788.141445 | 21447.692428 |
| **Median** | -636.8824693186098 | -636.8824693186098 |
| **Maximum** | 1.713890e+06 | 6.340964e+04 |
| **Minimum** | -855.615030 | -31748.790528 |

*Table 11– Standard Deviation model metric results for year 2020*

Again, the minimum values have decreased in comparison with previous year, but the maximum values have remained in the same heights. But because they are less than last year the standard deviation is smaller.

This concludes the section of Standard Deviation and the section of unsupervised algorithms. Next we will dive in the implementation of the supervised ones.

### 6.2.4 Supervised Models Implementation

The implementation of the supervised models is very simple. They require very few parameters and they take as inputs the two trained data sets produced by the **train_test_split** tool.

We will use the function **anomalies_check**, which will create an array of ones and zeros. Zero values being the normal points and one values the anomalies.

```
def anomalies_check (row):
    if row < 110000 : # normal point
      return 0
    if row >= 110000 : # anomaly point
      return 1
```

*Code snippet 22- Anomalies classification function*

After it is done categorizing, we will inserted this new array as a column in our files, so each record will labeled accordingly.

```
pca_2_columns_dataframe['Anomalies'] = pca_2_columns_dataframe['A'].apply
(lambda row: anomalies_check(row))
data['Anomaly'] = pca_2_columns_dataframe['Anomalies']
data.to_csv('dataset'+year+'_Output.csv', index=False)
```

*Code snippet 23- Defining anomalies as labels and adding them to our file as additional column*

To create these two trained data sets, as we have mentioned above, we will follow the procedure below. In addition to the eleven features that we have used, we will add to our stack the anomalies array, which was created beforehand.

```
# Begin of supervised algorithms
#============================================================
============
anomalies_array = np.array(pca_2_columns_dataframe['Anomalies']).reshape(-
1,1).astype(np.float)
dataset_12_features = np.hstack((duration_array,
                                 volume_array,
                                 authentication_id_array,
                                 charge_point_id_array,
                                 calculated_cost_array,
                                 connector_id_array,
                                 value_array,
                                 value_type_array,
                                 reading_context_array,
                                 transaction_id_array,
                                 connection_id_array,
                                 anomalies_array
                                ))
```

*Code snippet 24 – Adding anomalies in the data mix*

We then use in our newly formed dataset the function **train_test_split**, which as the name suggests, splits arrays/matrices into random train and test subsets of data. The test_size parameter with value 0.3 means that the data will be split 70% train and 30% test. The random_state parameter controls the shuffling applied to the data before applying the split. Function nan_to_num converts all null values if they exist in our set to zeros.

```
from sklearn.model_selection import train_test_split

train, test = train_test_split(dataset_12_features, test_size=0.3, random_state=42)

X_train = np.nan_to_num(np.array(train[:,0:10]), nan=0.0)

Y_train = np.nan_to_num(np.array(train[:,11]), nan=0.0)

X_test = np.nan_to_num(np.array(test[:,0:10]), nan=0.0)

Y_test = np.nan_to_num(np.array(test[:,11]), nan=0.0)
```

*Code snippet 25- Creation of supervised algorithms training sets*

It should be mentioned that the supervised algorithms cannot function without the presence of anomalies. So in the instance of no outliers being discovered, we interrupt the flow of execution as a measure of precaution and terminate the program at that point.

```
if (len(np.unique(pca_2_columns_dataframe['Anomalies'])) == 1):
    sys.exit("No anomalies where found by the unsupervised algorithm")
```

*Code snippet 26- Exit in case of no anomalies*

### 6.2.5 Metrics

The **accuracy_score** function will be given a value of True in its "**normalize**" parameter to return as a result the fraction of correctly classified samples.

For the **precision_recall_fscore_support** function, we will calculate the average values of precision, recall and f-measure with three distinct ways by altering the "**average**" parameter. Multiple options to utilize will give us more variation in our results and better understanding of the predictions. In the case of support, since a value was provided for average, None is returned as a result.

- **micro**: This option will calculate the above metrics globally by counting the total true positives, false negatives and false positives.
- **macro**: This option will calculate metrics for each label and find their unweight mean.
- **weighted**: This option will calculate metrics for each label and find their average weighted mean (the number of true instances for each label).

Essential libraries

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import confusion_matrix
```

*Code snippet 27 – Libraries import for scoring metrics*

The **y_true** variable represents the predictions of each supervised algorithm in the form of an array and the **y_pred** variable an array of estimated targets to use. The metrics were outputted up to two decimal points.

```
    y_true = y_pred_svc
    y_pred = Y_test

    print('\n')
    print('Metrics for Support Vector Machine supervised algorithm:')
    print("The accuracy score is " + str(float(accuracy_score(y_true, y_pred,
normalize=True)))[0:4])
    print("The presicion / recall / f1 score / support scores with parameter macro are ")
    macro_metrics_svc = precision_recall_fscore_support(y_true, y_pred,
average='macro')
    print(str(float(macro_metrics_svc[0]))[0:4] + ' / ' +
str(float(macro_metrics_svc[1]))[0:4] + ' / ' +
        str(float(macro_metrics_svc[2]))[0:4] + ' / ' + str(macro_metrics_svc[3]))
    print("The presicion / recall / f1 score / support scores with parameter micro are ")
    micro_metrics_svc = precision_recall_fscore_support(y_true, y_pred,
average='micro')
    print(str(float(micro_metrics_svc[0]))[0:4] + ' / ' +
str(float(micro_metrics_svc[1]))[0:4] + ' / ' +
        str(float(micro_metrics_svc[2]))[0:4] + ' / ' + str(micro_metrics_svc[3]))
    print("The presicion / recall / f1 score / support scores with parameter weighted are ")
    weighted_metrics_svc = precision_recall_fscore_support(y_true, y_pred,
average='weighted')
    print(str(float(weighted_metrics_svc[0]))[0:4] + ' / ' +
str(float(weighted_metrics_svc[1]))[0:4] + ' / ' +
        str(float(weighted_metrics_svc[2]))[0:4] + ' / ' + str(weighted_metrics_svc[3]))
    print("The confusion matrix is ")
    print(confusion_matrix(y_true, y_pred))
    print('\n')

    y_true = y_pred_nb

    print('Metrics for Gaussian Naive Bayes supervised algorithm:')
    print("The accuracy score is " + str(float(accuracy_score(y_true, y_pred,
normalize=True)))[0:4])
    print("The presicion / recall / f1 score / support scores with parameter macro are ")
    macro_metrics_nb = precision_recall_fscore_support(y_true, y_pred,
average='macro')
    print(str(float(macro_metrics_nb[0]))[0:4] + ' / ' +
str(float(macro_metrics_nb[1]))[0:4] + ' / ' +
```

```
            str(float(macro_metrics_nb[2]))[0:4] + ' / ' + str(macro_metrics_nb[3]))
        print("The presicion / recall / f1 score / support scores with parameter micro are ")
        micro_metrics_nb = precision_recall_fscore_support(y_true, y_pred, average='micro')
        print(str(float(micro_metrics_nb[0]))[0:4] + ' / ' +
str(float(micro_metrics_nb[1]))[0:4] + ' / ' +
            str(float(micro_metrics_nb[2]))[0:4] + ' / ' + str(micro_metrics_nb[3]))
        print("The presicion / recall / f1 score / support scores with parameter weighted are ")
        weighted_metrics_nb = precision_recall_fscore_support(y_true, y_pred,
average='weighted')
        print(str(float(weighted_metrics_nb[0]))[0:4] + ' / ' +
str(float(weighted_metrics_nb[1]))[0:4] + ' / ' +
            str(float(weighted_metrics_nb[2]))[0:4] + ' / ' + str(weighted_metrics_nb[3]))
        print("The confusion matrix is ")
        print(confusion_matrix(y_true, y_pred))
        print('\n')

        y_true = y_pred_dtc

        print('Metrics for Decision Trees Classification supervised algorithm:')
        print("The accuracy score is " + str(float(accuracy_score(y_true, y_pred,
normalize=True)))[0:4])
        print("The presicion / recall / f1 score / support scores with parameter macro are ")
        macro_metrics_dtc = precision_recall_fscore_support(y_true, y_pred,
average='macro')
        print(str(float(macro_metrics_dtc[0]))[0:4] + ' / ' +
str(float(macro_metrics_dtc[1]))[0:4] + ' / ' +
            str(float(macro_metrics_dtc[2]))[0:4] + ' / ' + str(macro_metrics_dtc[3]))
        print("The presicion / recall / f1 score / support scores with parameter micro are ")
        micro_metrics_dtc = precision_recall_fscore_support(y_true, y_pred,
average='micro')
        print(str(float(micro_metrics_dtc[0]))[0:4] + ' / ' +
str(float(micro_metrics_dtc[1]))[0:4] + ' / ' +
            str(float(micro_metrics_dtc[2]))[0:4] + ' / ' + str(micro_metrics_dtc[3]))
        print("The presicion / recall / f1 score / support scores with parameter weighted are ")
        weighted_metrics_dtc = precision_recall_fscore_support(y_true, y_pred,
average='weighted')
        print(str(float(weighted_metrics_dtc[0]))[0:4] + ' / ' +
str(float(weighted_metrics_dtc[1]))[0:4] + ' / ' +
            str(float(weighted_metrics_dtc[2]))[0:4] + ' / ' + str(weighted_metrics_dtc[3]))
        print("The confusion matrix is ")
        print(confusion_matrix(y_true, y_pred))
        print('\n')
```

*Code snippet 28 – Calculation of metrics for supervised algorithms*

### 6.2.6 Support Vector Machine

Introduction of the parameters used by the Support Vector Machine and code explanation.

| Parameters | Data type (Default value) | Short Description |
|---|---|---|
| **kernel** | **String (“rbf”)** | Specifies the kernel type. Possible values include ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ or a callable. |
| **C** | **Float(1.0)** | Penalty parameter of the error term. |
| **random_state** | **Integer/ RandomState (None)** | Calculates the pseudo-randomness of the selection and split values for each branching step and each tree. |

*Table 12 - Support Vector Machine parameters*

For our supervised algorithms, the Support Vector Machine presented the most challenge. Almost all types of kernels had to be tested (‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’) to have a more complete view on the results of the algorithm. The “precomputed” kernel could not be used in our type of problem.

```
from sklearn.svm import SVC

svc = SVC(kernel='linear', C = 1.0, random_state=42)
svc.fit(X_train, Y_train)
```

*Code snippet 29 – Support Vector Machine code*

### *Metric Scores Results*

The following tables present the metric scores of the various functions discussed previously in the predictions of the Support Vector Machine algorithm in the datasets of the years 2019 and 2020. No tables will be shown for the year 2018 due to lack of anomalies found. The results will show how well our algorithms managed to accomplish the given task of prediction and how valid these predictions where.

### *Year 2019*

| **Metrics calculated** | **Linear kernel** | **Sigmoid kernel** |
|---|---|---|
| **Accuracy (%)** | 1.0 | 0.99 |
| **Confusion Matrix** | [ [74948 0] [ 0 51] ] | [ [74948 3] [ 0 48] ] |

| **Metrics calculated Sigmoid kernel** | **Macro average** | **Micro average** | **Weighted average** |
|---|---|---|---|
| **Precision (%)** | 0.97 | 0.99 | 0.98 |
| **Recall (%)** | 0.99 | 0.99 | 0.99 |
| **F-measure score (%)** | 0.99 | 0.99 | 0.99 |

| **Metrics calculated Linear kernel** | **Macro average** | **Micro average** | **Weighted average** |
|---|---|---|---|
| **Precision (%)** | 1.0 | 1.0 | 1.0 |
| **Recall (%)** | 1.0 | 1.0 | 1.0 |
| **F-measure score (%)** | 1.0 | 1.0 | 1.0 |

*Table 13 – Support Vector Machine metric results for year 2019*

The closer these scores are to the value of 1.0 the better and more conclusive the results of the program in its totality. In our case all of these metrics are very close or even precisely 1.0. This can be attributed due to the linear nature of the problem, which is given to the algorithms. Even more, which can be classified as the main point, this is because of the clear separation between the data points, normal and anomalous. This gives our models a significant edge in their respective logics and in the score of the predictions.

### *Year 2020*

In this model, separated results are shown for kernels linear and sigmoid due to separate given metrics based on the same data inputs.

| **Metrics calculated** | **Linear kernel** | **Sigmoid kernel** |
|---|---|---|
| **Accuracy (%)** | 1.0 | 0.99 |
| **Confusion Matrix** | [ [74977 0] [ 0 23] ] | [ [74977 5] [ 0 18] ] |

| **Metrics calculated Sigmoid kernel** | **Macro average** | **Micro average** | **Weighted average** |
|---|---|---|---|
| **Precision (%)** | 0.89 | 0.99 | 0.93 |
| **Recall (%)** | 0.99 | 0.99 | 0.99 |
| **F-measure score (%)** | 0.99 | 0.99 | 0.99 |

| **Metrics calculated Linear kernel** | **Macro average** | **Micro average** | **Weighted average** |
|---|---|---|---|
| **Precision (%)** | 1.0 | 1.0 | 1.0 |
| **Recall (%)** | 1.0 | 1.0 | 1.0 |
| **F-measure score (%)** | 1.0 | 1.0 | 1.0 |

*Table 14 – Support Vector Machine metric results for year 2020*

It should be mentioned that the lower scores were from the Support Vector Machine algorithm, while being tested with the “sigmoid” type of kernel.

The sigmoid function is a logistic function which gives an ‘S’ shaped curve. It takes any real number and maps it into a value between 0 and 1. If the curve goes to positive infinity, the outcome will close to 1, and if the curve goes to negative infinity, the outcome will accordingly close to 0. If the output is more than 0.5, we can classify the outcome as a normal point, and if it is less than 0.5, we can classify it as an anomaly. [36]

### 6.2.7 Gaussian Naive Bayes

In the case of the Naive Bayes algorithm, we have used the Gaussian variation, instead of the Multinomial. The reason is because the Multinomial variation cannot handle the negative values that are produced during the processing of the columns values by the tools we have experimented with in the thesis. The Gaussian Naive Bayes model does not require any parameters.

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train, Y_train)
y_pred_nb = nb.predict(X_test)
```

*Code snippet 30- Gaussian Naive Bayes code*

#### *Metric Scores Results*

The following tables present the metric scores of the various functions in the predictions of the Gaussian Naive Bayes algorithm in the datasets of the years 2019 and 2020.

#### *Year 2019*

| Metrics calculated | Linear kernel |
|---|---|
| **Accuracy (%)** | 1.0 |
| **Confusion Matrix** | [ [74948 0] [ 0 51] ] |

| Metrics calculated | Macro average | Micro average | Weighted average |
|---|---|---|---|
| **Precision (%)** | 1.0 | 1.0 | 1.0 |
| **Recall (%)** | 1.0 | 1.0 | 1.0 |
| **F-measure score (%)** | 1.0 | 1.0 | 1.0 |

*Table 15– Gaussian Naive Bayes metric results for year 2019*

### *Year 2020*

| Metrics calculated | Linear kernel |
|---|---|
| **Accuracy (%)** | 1.0 |
| **Confusion Matrix** | [ [74977 0] [ 0 23] ] |

| Metrics calculated | Macro average | Micro average | Weighted average |
|---|---|---|---|
| **Precision (%)** | 1.0 | 1.0 | 1.0 |
| **Recall (%)** | 1.0 | 1.0 | 1.0 |
| **F-measure score (%)** | 1.0 | 1.0 | 1.0 |

*Table 16– Gaussian Naive Bayes metric results for year 2019*

As it was the case with the previous supervised algorithm the resulted values show a perfect score in their attempt to classify their given test labels as anomalies. It should be noted, that these kinds of metrics are very difficult to acquire in any given scenario such high scores. That is why many different supervised algorithms are employed to help choose, the best strategy needed for the separations. This proves empirically that the proper infrastructure of data with the right method of handling can have such positively drastic effects in the detection of anomalies. Cyber-attacks that must inadvertently change some feature in order to function in the context of a charging thus can definitely be alienated from the bulk of normal charging and be instantly stopped. Of course in the case of more elaborate types of attacks, more specialized approaches must be taken.

### 6.2.8 Decision Trees

In Decision Trees instance only one parameter was tampered.

| Parameters | Data type (Default value) | Short Description |
|---|---|---|
| **splitter** | **String ("best")** | The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split. |

*Table 17 - Decision Trees parameters*

For the Decision Trees Classification **random_state** was provided with a number and we decided to work with the "**spliter**" parameter of "random" value instead of "best", so we can see more changes in its output.

```
from sklearn import tree

dtc = tree.DecisionTreeClassifier(splitter="random")
dtc = dtc.fit(X_train, Y_train)
```

*Code snippet 31- Decision Trees code*

#### *Metric Scores Results*

The following tables present the metric scores of the various functions in the predictions of the Decision Trees Classification algorithm in the datasets of the years 2019 and 2020.

#### *Year 2019*

| Metrics calculated | Linear kernel |
|---|---|
| **Accuracy (%)** | 1.0 |
| **Confusion Matrix** | [ [74948 0] [ 0 51] ] |

| Metrics calculated | Macro average | Micro average | Weighted average |
|---|---|---|---|
| **Precision (%)** | 1.0 | 1.0 | 1.0 |
| **Recall (%)** | 1.0 | 1.0 | 1.0 |
| **F-measure score (%)** | 1.0 | 1.0 | 1.0 |

*Table 18 – Decision Trees Classification metric results for year 2019*

### *Year 2020*

| Metrics calculated | Linear kernel |
|---|---|
| **Accuracy (%)** | 1.0 |
| **Confusion Matrix** | [ [74977 0] [ 0 23] ] |

| Metrics calculated | Macro average | Micro average | Weighted average |
|---|---|---|---|
| **Precision (%)** | 1.0 | 1.0 | 1.0 |
| **Recall (%)** | 1.0 | 1.0 | 1.0 |
| **F-measure score (%)** | 1.0 | 1.0 | 1.0 |

*Table 19– Decision Trees Classification metric results for year 2019*

Finally, as it was expected given the results of the two previous models our results show how easily we can classify our anomalous labels. Having such encouraging results, we can easily determine, given the nature of our data, how out of the ordinary a charging process can be if a tampering occurs in any of its predetermined functionality.

### 6.2.9 Execution Time

In the context of this thesis many changes were performed to form the best code for our scenario. From deducting the number clusters necessary to discovering which kernel gave us the best accuracy, tasks such as these require a significant amount of time. So in parallel we needed to make the program take the least amount of time to run. When we overcompensated some of the parameters, i.e. the amount of estimators for Isolation Forest and maximum iterations for KMeans, time was raised exponentially. Such increase is not recommended as it is unnecessary and did not produce any better results.

Fastest process was recorded by the KMeans algorithm with visible difference (excluding the part of finding the optimal number of clusters, which takes several minutes), taking less than a minute to finish. Next is the Standard Deviation technique many seconds behind, around one and a half minute. Finally is the Isolation Forest algorithm, very close the previous one. The two key variables to consider for the time of execution for each algorithm is the number of records used and the number of anomalies presented in each dataset.

# 7. Summary

## 7.1 Data manipulation

During this thesis, the datasets that we have acquired by the standard EV charging enterprise, spanning 3 years of collected records, were put through testing by the three unsupervised algorithms that we have previously presented in detail in the chapters above. After the anomalies were predicted and plotted by them, three supervised algorithms were also used for further testing. This was done to check the validity of these results in more depth. An unsupervised learning model provides unlabeled data that the algorithm tries to categorize by extracting features and patterns on its own volition. In a supervised learning model, the algorithm learns on a labeled dataset, providing an answer key that the algorithm can use to evaluate its accuracy score on training data.

Because of the many features that exist in our datasets, the data records had to go through a lengthy transformation and categorization process, so they can be refined and used be our final data frame shown in the plotting. First of all figuring, which features we would manipulate in our project, was the most crucial task. Some of the columns had to be excluded due to lack of values or repetitive context. When the selection was completed, managing the values so we could have a usable input to work with was in order.

Faulty or null values had to be handled accordingly for the program to function. Columns of the datasets with not the usual numeric type values could not be used directly. They had to undergo changes with the use of functions or tools like the **sklearn CountVectorizer**, so proper inputs could be inserted. After acquiring all these values in the form of one dimensional array of float numbers, we stack them together, to create a uniformity of a final array to be successfully passed into the data frame. As a footnote, the more features added in the data frame will probably help produce an even clearer prediction results, but will take a toll in the time of execution of the program.

One other problem that occurred was in the dataset of the year 2019 one of the records had been inserted with one additional column and as a result stopping the flow of execution. It was resolved with the addition of **error_bad_lines=False**, which skips the faulty lines of the given file. Nevertheless it is an issue that should be addressed and be dealt in the future, otherwise the data could end up misleading or even worse corrupted.

## 7.2 Anomalies

During this testing of thousands of recordings we can observe that samples over 110000 in value in the axis of P1 show abnormal activity compared to rest of the data, so we can assume them as outliers. This value is derived by the manipulation done to the data by the PCA tool and the number of features employed. They are created further away from the rest of the samples group, making their own cluster unit in the visualization. This can be seen in the casing of all 3 algorithms in the plotting of both the 2D and 3D outputs, which makes the results conclusive as to their validity. We can furthermore conclude by the visualization, that as the years progress the normal samples tend to be closer together and the outliers even further.

With the deployment of the unsupervised machine learning techniques finished, the outliers are found and transformed in an array of their own. We now can use them as labels for our supervised machine learning part of the thesis. They are added in our previous created uniformity and will train our 3 supervised algorithms in random sets of inputs created by the sklearn tool **train_test_split**. Possible null values are again converted to zeros. They then will classify our established anomalies accordingly by their own way of learning. Optimal parameters are also set here, as is the case of the unsupervised algorithms for better results. The predictions that are formed consist of two parts, one of the data and one of the anomalies labels. Finally, the data part of the predictions and the Y_test array of random labels are used in the calculation of several metric scores (accuracy score, precision, recall, f1 score and confusion matrix) in order to

discover the percentage of how close they were to locate the anomalies in relevance with the unsupervised algorithms.

In the case of the scoring metrics all are very close or even precisely 1.0. This can be attributed due to the linear nature of the problem, which is given to the algorithms. Even more, which is the can be classified as the main point, this is because of the clear separation between the data points, normal and anomalous. This gives our models a significant edge in their respective logics and in the score of the predictions.

## 8. Observations and Interest Points

As we can observe from these results, in the amount of records that we tested, the datasets are fairly low on the presence of anomalies that can be considered harmful to the grid, which is something very promising for the electric vehicle industry.

From our datasets we tested each year multiple times by increasing the number of records to be able to see differences in the outputs. After testing 100000 lines of records the one with most number of anomalous spikes found was the year's 2019, with a number of 75, second being the year's 2020 with 36 and final the year's 2018, with no anomalies.

After testing 150000 lines of records the one with most number of anomalous spikes found was the year's 2019, with a number of 119, second being the year's 2020 with 49 and final the year's 2018, surprisingly with no anomalies. Increase in the amount of outliers in the first two datasets is what was expected as a result and is something complete normal, when analyzing such a large volume of records.

Finally, with 250000 lines of records the one with the most number of anomalous spikes found was the year's 2019, with a number of 196, second being the year's 2020 with 86 and final the year's 2018, still surprisingly with no anomalies. Again, the results were almost as anticipated.

The absence of anomalies in the year 2018 can have either of two explanations. Either the selected records happened to be of normal nature, meaning the selection of another list of records from the dataset would potentially yield some outliers. Either, which would be something very unlikely, although not completely stretched, the whole time frame had all its charging values within the same limits. As a consequence of that the testing of supervised algorithms on this particular frame was not conducted. The supervised execution part simply cannot operate with all the produced labels being zero.

The sudden spike in anomalies in the year 2019 can be possibly attributed to some major change(s) in the standard charging procedure of the EV's, where their owners maybe made some errors and took some time to adjust accordingly. The drop of anomalous spots in the following year proves that the adjustment was progressively stabilized.

As technologies, materials and practices improve a clearer and more refined pattern is created in the charging processes, which means that they are being optimized in correlation of all the variables that were present in the datasets. Chargings processes progressively take less time to charge an EV, even though the volume input is increasing. As the CP network expands, the charging schedule of the consumers, who own EVs is becoming more stable. This is very important because as we have mentioned before, the where the charging was done, in which CP and in connection with what ConnectorId and ChargePoint_ID it was completed, affects our data analysis and will show the possible cyber-attacks, that we seek easier.

A few final remarks, the data from all years presented a big fall in the number of spikes, when columns were gradually removed for testing. Starting by the basic eleven columns we each time took a column from the collective array. We did show so we can see the replications on the program and have a better understanding of the importance of each individual element. When analyzing such problems, you must be certain which inputs will deliver the best information. As it turns out our algorithms are unable to find anomalies when the number of columns drops below 5. That shows that having some basic columns like Volume, Duration and Calculated Costs is not enough, for our models to determine a solid answer, but with the help of more specialized and unique components like AuthedicationID and ConnectorID it can make a massive difference, as it sets each record apart from the rest and makes it very difficult for anomalies, such as cyber-attacks to hide in our plotting.

Furthermore, our results' scoring is linked directly not only to distance between normal and abnormal points, but to the ratio between them. The percentage of anomalies in our data is small compared to the other percentage and that plays a significant role in the calculations. By implementing the technique of oversampling the plots start to change. Oversampling is the practice of adding false labels in the data, so the ratio amongst them alters to a testing preferable degree, to observe how they react in a scenario. After adding these false data in the mix, the points tend to divide even more in their respective clusters. By increasing even further the algorithms become disorganized and make small mistakes, which are latter shown in the metric values, designed to keep them in check.

One major issue that arises as time goes by in our data analysis is what happens, when more consumers are added to the existing flow of charging processes. The bigger of the bulk of information, the more anomalies will be made visible in our plotting, many of which will be inconsequential, i.e. charging processes that were terminated earlier by the user in their usual schedule. The solution is adding more regular data plotting in the records deriving from the CPOs.

# 9. Conclusions & Future Work

This thesis examined threats and cyber-attacks in the charging process and the targeting BMS and other parts of the car's electronics' system. The biggest advantage of the usage of electric vehicles lies in its contribution to the reduction of pollution. Electric vehicles have almost zero pollution trails, causing minimal air pollution and zero pollution of the moving space. In a recent study it has been estimated that electric vehicles are 98% cleaner than the conventional fossil fueled type. Another great advantage is that electric vehicles are much more silent compared to vehicles with internal combustion engines, which reduces sound pollution, which is ideal for city usage. Also, they are much easier to construct because the electric motors are very simple in their structure. Since it is powered by electronic power converters, which are regulated electronically, water is usually not required for cooling and does not use filters or oil, so it does not present problems caused by low ambient temperature. Finally, an electric car consumes energy only when it is moving. When not moving e.g., stopping at a traffic light or during heavy traffic jams, its energy consumption drops completely.

A PHEV communicates with and is controlled by a charging station. Charging stations play an important role and they multiply other functions such as providing and controlling the energy to the EV using the EVSE component, collecting the measurements from the meter for each charge, identifying and authorizing EV owner by an user specific authentication factor. Moreover, charging stations also present additional remote capabilities to the Charge Point (e.g., adjustment of the maximum current allowed) via the Local Controller component over WAN. The main contribution of this thesis is to give new knowledge of examining possible threats and abuse of smart charging in Electrical Vehicles. In other words, applying specific algorithms in real collected database of a standard EV charging enterprise, to test possible abnormal activity on the charging station can trigger designers and programmers of the networks of smart charging Station, to reconsider the security issues. Moreover, if algorithms can be applied

in real time, then detection of abnormalities during smart charging could trigger an alarm that some abnormal activity is taking place.

Many smart charging cyber-attacks on a given smart grid environment have already been identified. EV charging being still new at its designing process is susceptible to masquerading, tampering, eavesdropping, and denial of service attacks. It has also many privacy concerns that must be addressed along the way. Charging can be attacked by methods like eavesdropping, man-in-the-middle and various tampering attacks which are connected to the payment process and the amount of energy delivered amongst the meter and the EV. Finally, there exists the possibility for malicious software installed within the vehicle to affect a CP, or vice versa. [37]

This means that if an attacker could intrude the software of the charging station, it might be possible to influence the charging behavior of the vehicle. This Thesis examined whether the charging station hardware can be hacked in order to send these erroneous signals (either locally or remotely) and how the charging stations can be made tamper-proof and how cyber-attacks can be detected. Using Isolation Forest Algorithm and python language, we presented a fully functioned program, giving as input charging processes, obtained by a standard EV charging enterprise's database and as output we isolated, abnormal charging processes. In this case an abnormal charging process is the situation, where an electric vehicle requests high volume of power.

To conclude, there are some possible ways for charge stations, on order to detect abnormal situations in charging processes of Electrical Vehicles. In this thesis, isolation forest seems to be useful in order to test the charging processes that requests high voltage of power. Therefore, charging stations network could trigger an alarm in order to investigate an abnormal situation, possibly a cyber-attack.

# References


1. Yosra, Fraiji & Azzouz, Lamia & Trojet, Wassim & Saidane, Leila. (2018). Cyber security issues of Internet of electric vehicles. 1-6. 10.1109/WCNC.2018.8377181.

2. Li, Yanmei & Ha, Ningning & Li, Tingting. (2019). Research on Carbon Emissions of Electric Vehicles throughout the Life Cycle Assessment Taking into Vehicle Weight and Grid Mix Composition. Energies. 12. 3612. 10.3390/en12193612.

3. Acharya, Shree & Choi, Kyung-Ho & Wi, Young-Min & ee, Jaehee. (2018). Smart Charging for Grid-Connected Electric Vehicles to Provide Regulation Service. Journal of the Korean Institute of Illuminating and Electrical Installation Engineers. 32. 32-39. 10.5207/JIEIE.2018.32.1.032.

4. Fauzan, Ts Dr Mohd Faizal & Feng, S. & Zureel, M. & Sinidol, B. & Wong, D. & Jian, G.. (2019). A REVIEW ON CHALLENGES AND OPPORTUNITIES OF ELECTRIC VEHICLES (EVS). Journal of Mechanical Engineering Research & Developments. 42. 130-137. 10.26480/jmerd.04.2019.130.137.

5. Hajebrahimi, Ali & Kamwa, Innocent. (2018). A Novel Approach for Plug-in Electric Vehicle Planning and Electricity Load Management in Presence of a Clean Disruptive Technology. Energy. 158. 10.1016/j.energy.2018.06.085.

6. Xiang, Yue & Hu, Shuai & Youbo, Liu & Zhang, Xin & Liu, Ji. (2018). Electric Vehicles in Smart Grid: A Survey on Charging Load Modelling. IET Smart Grid. 2. 10.1049/iet-stg.2018.0053.

7. Igbinovia, Famous & Fandi, Ghaeth & Mahmoud, Rateb & Tlustý, Josef. (2016). A Review of Electric Vehicles Emissions and its Smart Charging Techniques Influence on Power Distribution Grid. Journal of Engineering Science and Technology Review. 9. 80-85. 10.25103/jestr.093.12.

8. Martinenas, Sergejus & Pedersen, Anders Bro & Marinelli, Mattia & Andersen, Peter & Træholt, Chresten. (2015). Electric vehicle smart charging using dynamic price signal. 2014 IEEE International Electric Vehicle Conference, IEVC 2014. 10.1109/IEVC.2014.7056150.

9. Parkinson, Simon & Ward, Paul & Wilson, Kyle & Miller, Jonathan. (2017). Cyber Threats Facing Autonomous and Connected Vehicles: Future Challenges. IEEE Transactions on Intelligent Transportation Systems. 1-18. 10.1109/TITS.2017.2665968.

10. https://www.vreg.be/sites/default/files/uploads/siemens.pdf

11. Hanauer, Dieter. (2018). Mode 2 Charging—Testing and Certification for International Market Access. World Electric Vehicle Journal. 9. 26. 10.3390/wevj9020026.

12. Borges, J. & Ioakimidis, Christos & Ferrão, Paulo. (2010). Fast charging stations for electric vehicles infrastructure. WIT Transactions on Ecology and the Environment. 130. 275-284. 10.2495/ISLANDS100241.

13. Lopes, Joao Abel & Soares, Filipe & Almeida, Pedro & Moreira da Silva, M.. (2009). Smart Charging Strategies for Electric Vehicles: Enhancing Grid Performance and Maximizing the Use of Variable Renewable Energy Resources.

14. https://www.smartgrid-forums.com/wp-content/uploads/2018/06/11.-SmartSec-Europe-2016-ENCS-Michael-John.pdf

15. Ferreira, João & Monteiro, Vitor & Afonso, J.L. & Silva, Antonio. (2011). Smart electric vehicle charging system. 758 - 763. 10.1109/IVS.2011.5940579.

16. Afonso, J.L. & Ferreira, João & Monteiro, Vitor & Silva, Antonio. (2013). Smart Electric Vehicle Charging System.

17. Wen, Zhenyu & Yang, Renyu & Garraghan, Peter & Lin, Tao & xu, Jiudong & Rovatsos, Michael. (2017). Fog Orchestration for Internet of Things Services. IEEE Internet Computing. 21. 16-24. 10.1109/MIC.2017.36.

18. M.A. Mustafa, N. Zhang, G. Kalogridis, Z. Fan Smart electric vehicle charging: security analysis IEEE PES Innovative Smart Grid Technologies Conference (2013)

19. S. Fries, R. Falk Electric vehicle charging infrastructure-security considerations and approaches INTERNET 2012: the Fourth International Conference on Evolving Internet (2012), pp. 58-64

20. Clairand, Jean-Michel & Rodríguez-García, Javier & Alvarez, C.. (2018). Smart Charging for Electric Vehicle Aggregators considering Users' Preferences. IEEE Access. PP. 1-1. 10.1109/ACCESS.2018.2872725.

21. Twentyman J. (2018). Smart energy: Why vehicle-to-grid technology is on the move, Internet of Business, April 2018, https://internetofbusiness.com/smart-energy-why-vehicle-to-grid-technology-is-on-the-move/

22. Antoun, Joseph & Kabir, Mohammad & Moussa, Bassam & Atallah, Ribal & Assi, Chadi. (2020). A Detailed Security Assessment of the EV Charging Ecosystem. IEEE Network. PP. 1-8. 10.1109/MNET.001.1900348.

23. Schmutzler, Jens & Andersen, Claus & Wietfeld, Christian. (2013). Evaluation of OCPP and IEC 61850 for smart charging electric vehicles. 1-12. 10.1109/EVS.2013.6914751.

24. https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e

25. K. Wang et al., "A Survey on Energy Internet: Architecture, Approach, and Emerging Technologies," in IEEE Systems Journal, vol. 12, no. 3, pp. 2403-2416, Sept. 2018.

26. Zamini, Mohamad & Hasheminejad, Seyed Mohammad Hossein. (2019). A comprehensive survey of anomaly detection in banking, wireless sensor networks, social networks, and healthcare. Intelligent Decision Technologies.

27. Jing, Wentao & Yan, Yadan & Kim, Inhi & Sarvi, Majid. (2016). Electric vehicles: A review of network modelling and future research needs. Advances in Mechanical Engineering. 8. 10.1177/1687814015627981.

28. Chandola, Varun & Banerjee, Arindam & Kumar, Vipin. (2009). Anomaly Detection: A Survey. ACM Comput. Surv. 41. 10.1145/1541880.1541882.

29. Kong, Peng-Yong & Karagiannidis, George. (2016). Charging Schemes for Plug-In Hybrid Electric Vehicles in Smart Grid: A Survey. IEEE Access. 4. 6846-6875. 10.1109/ACCESS.2016.2614689.

30. Singh, Ran & Bhatia, Mohinder :Pal Singh. (2011). Data clustering with modified K-means algorithm. 717-721. 10.1109/ICRTIT.2011.5972376.

31. Hassani, Hossein & Ghodsi, Mansi & Howell, G.. (2010). A note on standard deviation and standard error. Teaching Mathematics and Its Applications. 29. 108-112. 10.1093/teamat/hrq003.

32. Ben-Hur, Asa & Weston, Jason. (2010). A User's Guide to Support Vector Machines. Methods in molecular biology (Clifton, N.J.). 609. 223-39. 10.1007/978-1-60327-241-4_13.

33. Martin, Daniel Jurafsky & James H. 2016. «Naive Bayes and Sentiment Classification. » Speech and Language Processing, Daniel Jurafsky & James H. Martin. Stanford University.

34. Liu, Fei Tony & Ting, Kai & Zhou, Zhi-Hua. (2012). Isolation-Based Anomaly Detection. ACM Transactions on Knowledge Discovery From Data - TKDD. 6. 1-39. 10.1145/2133360.2133363.

35. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

36. https://medium.com/python-in-plain-english/understanding-logistic-regression-and-building-model-in-python-1752a7e562a8

37. Zeinab El-Rewini, Karthikeyan Sadatsharan, Daisy Flora Selvaraj, Siby Jose Plathottam, Prakash Ranganathan. "Cybersecurity challenges in vehicular communications", Vehicular Communications, 2020