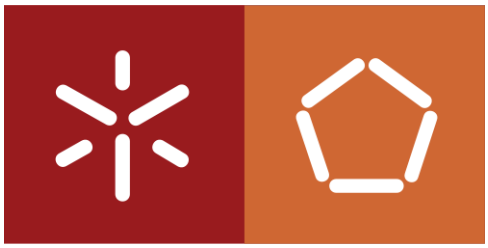

Universidade do Minho
Escola de Engenharia

Projeto em Engenharia Informática

Technical Report - AlertAI

PEILoad

RideCare

Master's Degree in Informatics Engineering
2020/2021

BOSCH

February 1, 2021

TECHNICAL REPORT - ALERTAI

RideCare

Universidade do Minho
Escola de Engenharia

ID Doc	RT-20201218-PEI2020
Version	1.0
Access	Restrito
Issue Date	February 1, 2021
Authors	André Coutinho (pg39284) Gabriela Martins (a81987) João Costa (a81822) José Pinto (a81317) Leandro Costa (pg41083) Luís Correia (a81141) Paulo Martins (pg17918) Pedro Barbosa (a82068) Rafaela Silva (a79034) Rui Santos (pg41850) Tiago Fontes (a80987)
Course	PEI2020

Contents

1 Summary

RideCare takes advantage of real-time classification algorithms, based on classic Machine Learning methods to detect anomalous situations inside shared vehicles. From labelled and unlabelled data, several algorithms from Supervised and Unsupervised Learning paradigms were tested and insights were taken during that extensive exploratory work. Additionally, other supervised learning algorithms, which performed well during our research process, are hosted by a cloud application -AlertAI Cloud, allowing alternative predictions for the raw data collections.

During the development workflow of the current project, we started for modelling and requirements identification, in order to infer the business problem and develop knowledge about all the concerns we should take into account during its development process. After identifying main features and requirements, the whole architecture was designed, aiming representing in a realistic way packages, services and communications protocols. Alongside these process, discussions about technologies and components were taken place.

Defining the architecture of the system is a leypoint, in order to give a visual representation of the conceptualised components, was the kickoff of the development stage. Subteams were created for the implementation of all components - **Frontend**, **Website Backend**, **Machine Learning**, **Datalake** and **FieldCrawler**.

This report is meant to detail all the technical issues and development decisions which were made. The document is organised as follows: section 2 details four main activities: Data Collection [2.1], Data Visualisation 2.2, Data Preprocessing 2.3 and last but not least in section 2.4 all the algorithms considered and tested, alongside their results.

2 AlertAI

The AlertAI system is responsible for **classification** and **detection** of **anomalies** that occur inside the vehicle's cabin. However, in order for this classification process to take place, which will be explained in detail in 2.4, it is necessary to go through a development pipeline that will be fundamental to enhance and maximise the classification and detection capabilities of our algorithms.

As we will see in 2.1, the various data collection scenarios are clearly defined. After these data are collected, it is necessary to have a 2.2 framework in terms of the records that were collected in order to understand trends, the importance of values and their distribution according to the different collection scenarios. After this analysis of the data, it is necessary that they undergo transformations and checks 2.3, which are extremely important to clean the data and ensure its consistency. This is crucial since in the context portrayed, Artificial Intelligence, the **quality** of the data is fundamental for the **obtaining** of **good results**.

2.1 Data Collection

In view of the data collection that needs to be carried out, the group devised and defined some **plans** of **harvest** of **data** that will allow us to understand the type of data, reference values and above all, produce large amounts of data that are indispensable for training and validating the models that we will see later.

Thus, and since the group is considering articulating several algorithms, from the beginning with **supervised** or **unsupervised**, several scenarios were identified for each of these approaches.

In terms of estimation, the group intends to obtain between 50-55% of normal situations, as well as 45-50% of abnormal situations, since this balance is also very important for consistent performance of the algorithms.

2.1.1 Data Collection - Supervised Approach

In this approach, it is important that the collected data characterise in a faithful and consistent manner, since the classifications collected here will be very important, so that the different models perceived the anomalous events as accurately as possible.

Thus, three collection groups were created: normal situations, with **existence of smoke** and with **existence of bad odour**.

Dividing the collection scenarios into these three groups, we ended up with:

- Normal -> No smoke, windows closed, AC off
- Normal 2 -> No smoke, windows open, AC off
- Normal 3 -> No smoke, windows closed, AC on
- Normal 4 -> Existence of outside smoke, closed windows

With this planning, it was possible to acquire **variability** in the collected data, since this is a very important characteristic for the models to be able to extrapolate knowledge. In this sense, the various scenarios, when contemplating the use of open and closed windows, as well as the Air Conditioning of the car on and off, allow us to diversify the collected data and to understand the influence of these external agents in the performed collection.

On the other hand, the abnormal scenarios contemplated are:

- Anomaly 1 -> Existence of indoor smoke, closed/open windows
- Anomaly 2 -> Existence of bad smell, closed/open windows

As you can see, the glasses were kept here, since it was not possible to identify a relationship of influence of the Air Conditioning in the collections made.

2.1.2 Data Collection - Unsupervised Approach

In this approach, where models will have to autonomously perceive and infer which records belong to each situation (normal or anomalous), a substantially higher amount of data is needed. In this sense, and since it is anomaly detection, it will be important for the models to clearly understand which are the most common values for anomalous situations. With this, a quantity of data in the order of thousands should be collected in the normal scenarios, in order to expose the normal scenarios in large quantities, and in this way the model can accurately infer which are the normal records.

In this way, and only for validation scenarios, some records related to anomalies will be provided, to ascertain that, in fact, the models perceive anomalous scenarios relatively easily.

In this sense, the group considered it reasonable to have between 90-95% of catches in a normal driving situation, in order to clearly identify what an anomalous situation is. On the other hand, about 5-10% of catches from anomalous situations as a means of verification.

2.2 Data Visualisation

In this section, some conclusions will be drawn from the data that were collected on the Data Collection plan defined earlier. In order to ensure that the conclusions are accurate as possible, the data that we will deal are the Raw data, which are the direct output from the sensor.
Thus, initially a context will be given about the dataset under study. Then, we proceed to statistical analysis, visual inference and correlation study.

2.2.1 Initial Contextualisation

Before addressing the analysis, it is important that an initial context be provided,in order to sustain the conclusions and thoughts drawn from the data that will be described in the following sections.
The dataset that will be handled counts with 2062 elements, divided between normal and anomalous situation. Of these, approximately 75% are related to normal situations.
The distribution of the elements is, concretely, as follows.

```
Total Number of elements: 2062
--> Normal situation elements: 1530
--> Anomalous situation elements: 532
```

Figure 2.1: Number of elements in the dataset.

The anomalous situation that is recorded in the dataset is the presence of smoke inside the vehicle. The data concerning the presence of smoke inside the vehicle were captured with resort to a real cigarette.

Second stage of Development

After the first phase of development was completed with the detection of smoke inside the vehicle, we move to a phase in which the study of the effect of intense smells on the sensors is started. Thus, at this stage, the dataset has about 10 000 records distributed across various situations that differ from the normal situation.
In concrete numbers, the dataset is as follows.

```
Total Number of elements: 9918
--> Normal situation elements: 8777
--> Anomalous situation elements: 1141
```

Figure 2.2: Number of elements in second stage dataset.

These numbers include situations in which there is smoke, cigarettes and a simulation can, intense perfume smell and intense garbage smell. The presence of perfume (which does not mean an anomaly) is intended to allow research into the possibility of distinguishing between a pleasant and unpleasant smell. If the ability to distinguish is verified, we will add the unpleasant odour anomaly to the system. Therefore, we proceed to the analysis.

2.2.2 Statistical Analysis

The first analysis is related to the statistical values of the data and which conclusions can be drawn from them. In order to ease the understanding , we present the data separately
In the following images, it is possible to see the statistical values of the data.

	sensors.pm25	sensors.pm10	sensors.temperature	sensors.gas	sensors.humidity	sensors.pressure	sensors.altitude
count	1530.000000	1530.000000	1530.000000	1530.000000	1530.000000	1530.000000	1530.000000
mean	6.887843	12.433399	20.431573	50655.024837	49.010644	988.548380	208.730837
std	7.093866	9.012871	4.246811	12496.094799	10.501339	17.085545	144.932120
min	0.000000	0.000000	10.512617	9779.000000	25.600421	955.262052	-59.780313
25%	2.200000	7.025000	18.035029	40524.750000	42.553079	974.112714	118.990220
50%	4.500000	10.300000	20.824141	55955.500000	48.610579	990.046460	195.001528
75%	9.200000	14.275000	24.026924	60075.250000	56.501340	999.039430	331.062110
max	41.600000	52.700000	27.949531	68118.000000	72.686949	1020.450858	494.377666

Figure 2.3: Normal situation statistics.

	sensors.pm25	sensors.pm10	sensors.temperature	sensors.gas	sensors.humidity	sensors.pressure	sensors.altitude
count	532.000000	532.000000	532.000000	532.000000	532.000000	532.000000	532.000000
mean	435.720865	796.131015	18.204949	26721.716165	60.358144	1008.564616	39.377654
std	242.600065	511.758778	2.049690	12093.884448	5.422557	9.288305	78.796258
min	0.000000	0.000000	11.200313	3841.000000	49.369326	967.816309	5.684227
25%	213.100000	383.200000	17.177119	18521.750000	56.223787	1009.039051	10.977083
50%	440.500000	670.700000	18.362617	26354.500000	61.051800	1011.106052	17.865142
75%	609.200000	1136.300000	19.554170	32558.000000	64.455652	1011.932234	35.118169
max	999.900000	1999.900000	22.019648	69055.000000	71.766466	1012.567452	385.325117

Figure 2.4: Anomalous situation statistics.

Trough combined analysis of the figures above (Fig 2.3 and Fig 2.4), the average values of the columns *pm25* and *pm10* rise significantly in abnormal situations compared to their reference value. Still in the *pm's* columns, is perceptible that in anomalous situations the standard deviation is much higher than in normal situations, this peculiarity indicates points to dispersed data, which may not be advantageous for Machine Learning models. A less expressive but noteworthy variation occurs in the gas values that in an abnormal situation the mean value is about half the value of reference.
The remaining columns do not present, at first look, significant variations.

Second stage of Development

In order to understand the influence of the new data, as in the previous phase, the first thing to do is analyse the statistical values like the mean and standard deviation of the columns that compose the dataset. Since the statistical values of normal situation remained very close to the previous phase, we will only compare the anomalous situations.
The result is shown in the following figure

	sensors.pm25	sensors.pm10	sensors.temperature	sensors.gas	sensors.humidity	sensors.pressure	sensors.altitude	sensors.classification
count	1141.000000	1141.000000	1141.000000	1141.000000	1141.000000	1141.000000	1141.000000	1141.000000
mean	309.261700	626.679842	18.748447	42651.442594	60.381668	999.629140	114.554295	2.713409
std	293.245202	645.917091	3.315124	21984.508079	7.756193	12.453751	104.938890	1.873472
min	2.700000	8.700000	7.844306	3841.000000	38.051415	967.816309	-45.444814	1.000000
25%	35.600000	56.100000	16.932344	24673.000000	55.181451	987.973221	16.188072	1.000000
50%	213.100000	457.500000	18.610859	37799.000000	59.235854	992.727919	172.278727	1.000000
75%	510.400000	1083.800000	20.132344	65632.000000	63.503448	1011.307156	212.604422	5.000000
max	999.900000	1999.900000	32.225238	112731.000000	88.338576	1018.720306	385.325117	5.000000

Figure 2.5: Statistical values of anomolous situation, on 2nd stage

It is important to note that the values that was previously referenced, such as *pm25* and *pm10*, which showed a very significant increase and were higher in abnormal situation than in normal situation, decreased on this phase. Although they decreased a bit, they remain quite high in relation to the 'normal' situation.The value variation in anomalous situations between the first and second stage shows some variability in this field between different anomalies.
The gas values have risen considerably in relation to smoking anomalies, the value of the mean increased to almost twice the value stated in the first phase . The high standard deviation shows that the data diverse within the metric.
The remaining values are equivalent, so the group considers that these, possibly, will not be important for the classification.

2.2.3 Visual Analysis

Even though this is not the final stage of development, it is possible, through the analysis of a portion of the data, extrapolate the relation between the different columns.
The following images show all column combination graphs, and through these, it is possible to extract (visually) which columns are likely to have more influence on the final result, that is, detection of anomalies.

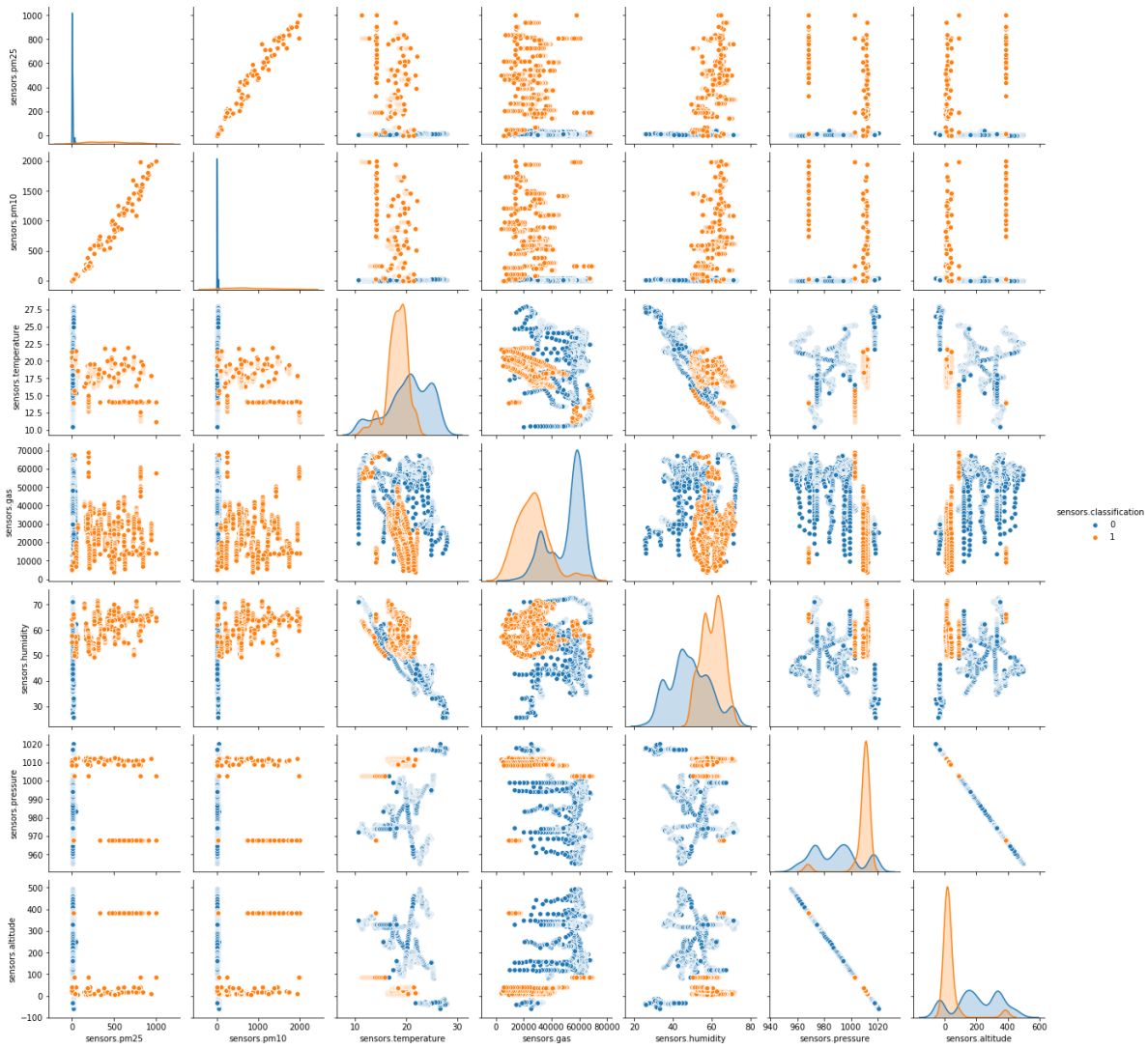

Figure 2.6: *Pairplot* columns combination

As the legend of the graphs shows, in blue the normal situations (represented by the value '0' in the dataset) and orange the abnormal situations (represented value 1 in the dataset).
At the first sight, it is immediately noticeable that, in the first two rows of the 'graphical matrix' (for pm25 and pm10, respectively), anomalous and normal situations are easily distinguished, this means that it is quite likely that these attributes are considerable for detecting an anomaly.
On the opposite, it is possible to state that the column referring to altitude, should not be considered once, that by the physical characteristics it represents, its variation does not depend on any act performed vehicle user.
From a more closer look, the group realised that there are some interesting gas patterns for detecting anomalies, namely, a decrease in values in anomaly cases (as seen in the values of the statistical tables previously presented) is an interesting behaviour.
On the two remaining lines (humidity and pressure), there are no visible patterns like those on

the previous ones. However, there are a few graphs where the humidity values appear to show identifiable patterns, making this the next dataset column to consider.

Second stage of Development

We knew that in this stage it would be more difficult to find and verify patterns and differences between data through general pairplots, due to the increase in the variety of records that generates some confusion in the graphics. Then we decided, for research reasons, to explore the visual approach.
The following figure shows the intersection of each column with each other with the second phase dataset.

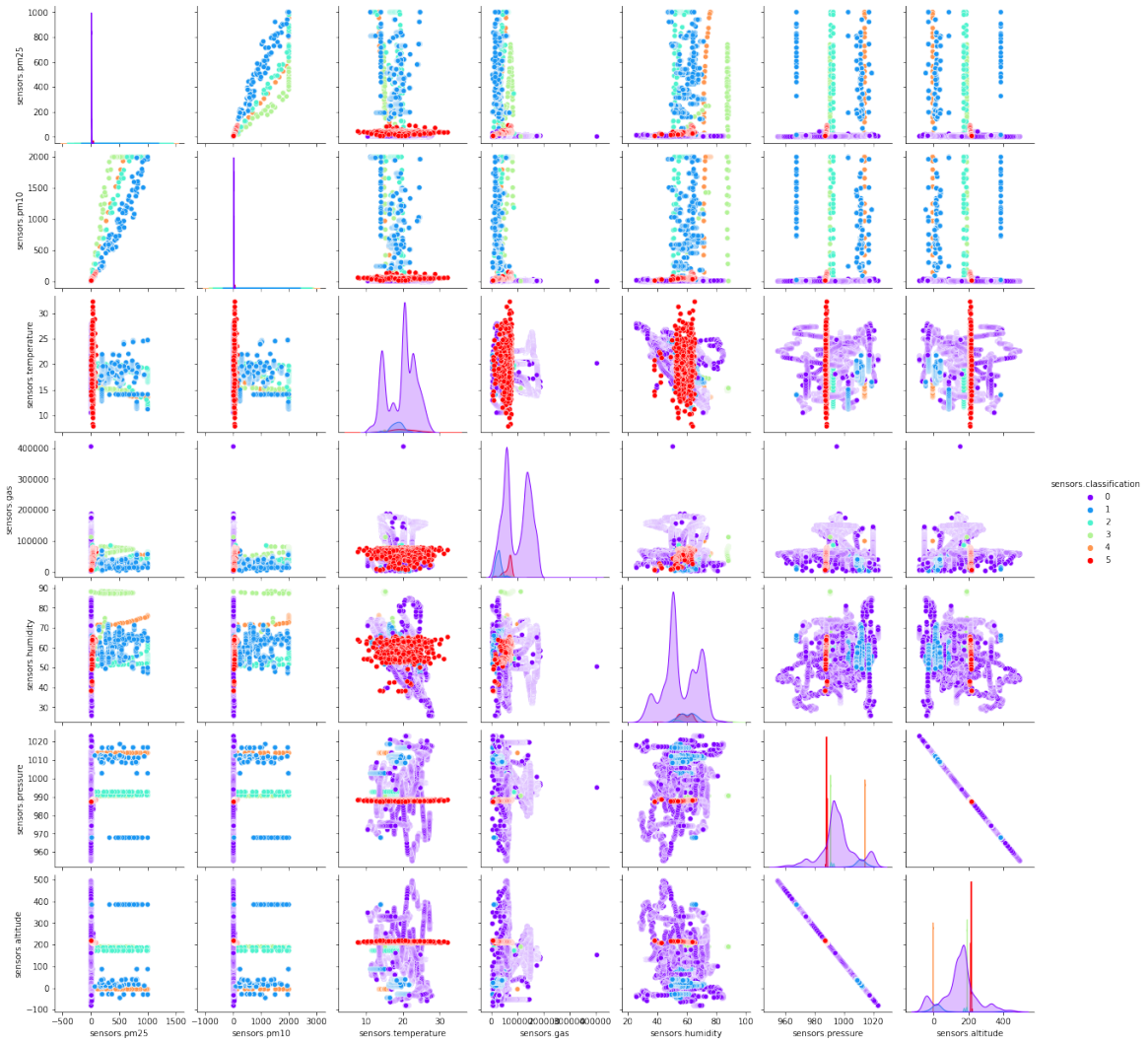

Figure 2.7: Second stage *Pairplot*.

The analysis of these graphs aims to improve our perception of a possible ability to distinguish scent (perfume) records from 'normal' records (purple) and records where exist 'smoke' (blue). Regarding the objective, it is notable that in some of the graphics (such as *pm's* and humidity) it is possible to possible to differentiate between the orange dots from the blue dots.
This conclusion led the group wondering about the ability to differentiate a pleasant smell from an unpleasant smell, hence the introduction of registers with an intense smell of garbage (red). Visual analysis shows that there is an opportunity, since the red records show dispersion patterns different from the orange registers, corresponding to the perfume.

2.2.4 Correlation Analysis

The visual analysis of the graphs that cross the different columns with each other allows a view on the variation of the data, and a better understanding about them as well. In most cases the conclusions we drawn are trustworthy. However, in order to confirm that we are seeing it the right way we will look at the coefficient matrix and then, using the selectKBest method, check what are the three top contributors to the anomaly identification.

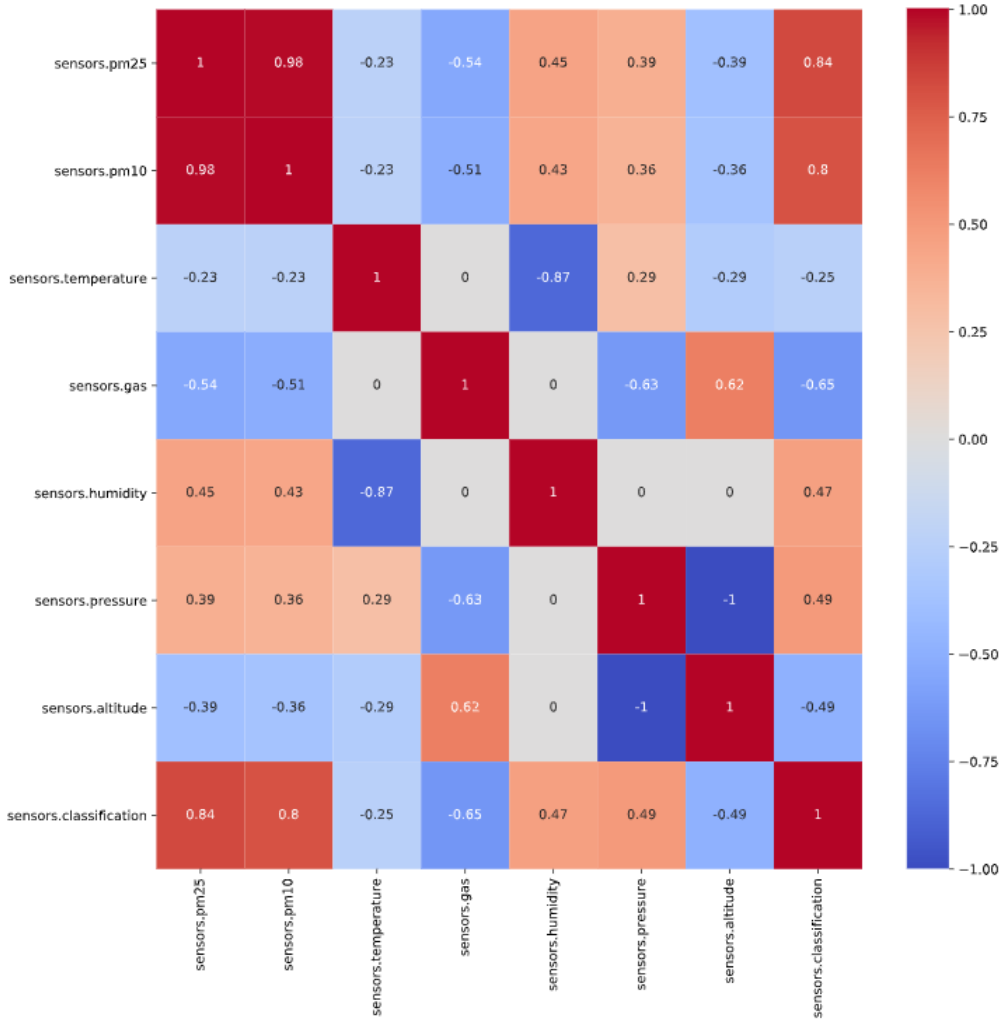

Figure 2.8: *Heatmap* of correlations.

To analyse the heatmap represented above, we need to consider the last column, which is the classification column. Thus, the correlation matrix shows that the conclusions taken through the previous graphs is correct because the values in the columns *pm25* and *pm10* 0.84 and 0.8, respectively. High values means that these are the characteristics that most influence the detection of the anomaly, in this case smoke. And the increase is proportional, that is, the greater the amount of 'pm' particles, the more likely it is to be anomaly.

The third feature that most contributes to the detection of anomalies is the amount of gas with a value of -0.65, the negative value means that it contributes in inverse proportionality, that is, the smaller the amount of gas, the more likely it is to be an anomaly. Another column that calls attention, in the previous section, is humidity, however the matrix shows that its influence on smoke detection is not so remarkable, having only a correlation of 0.47.

It is important to note that, although the altitude has a value of -0.49, it does not has any relationship with the detection of the anomaly, as stated above, due to its physical definition.

Another approach used by the group, as told earlier, was the use of the *selectKBest*, which shows the best columns to take into account for an objective, and the result was as follows.

```
The best three features are:
    - pm25
    - pm10
    - gas
```

Once again, the previous visual analysis is confirmed, with the selection of columns *pm25*, *pm10* and *gas* as the most important. In order to visually show this result, the graphs of these three selected columns will be displayed.

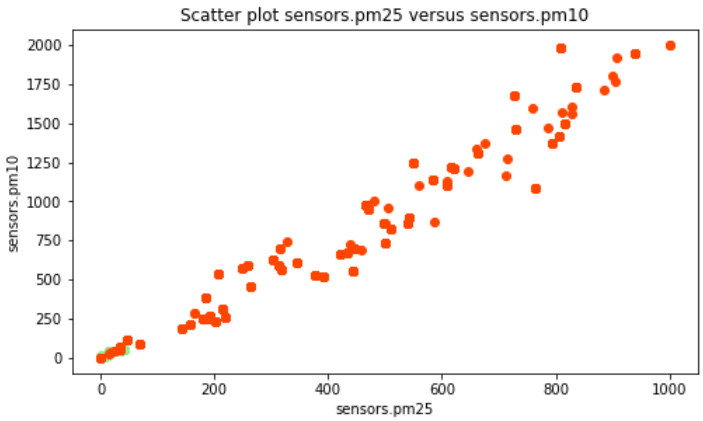

Figure 2.9: Graph **pm25** x **pm10**.

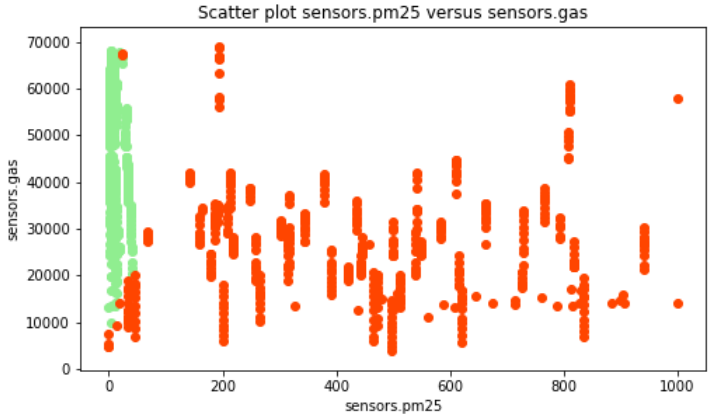

Figure 2.10: Graph **pm25** x **gas**.

As we can see from the graphs above, it is quite intuitive to notice the difference in values between those that are classified as anomaly and those that are classified as normal behaviour. These three columns are the culmination of a process of learning and understanding the data

resulting from different approaches, statistical, visual and correlation.
An important note is that in figure 2.9 it is possible to see data related to anomalous situations very close to 0. All this analysis shows that values of these columns so low and representing anomalies are most likely the result of hardware or reading problems, so this type of data must be dealt with in the data pre-processing phase, described later.

Second stage of Development

In order to maintain the same process as in the first phase, we carried out the correlation analysis between the columns in an attempt to determine which are the most influential columns. Thus, in this section, the correlation heatmap will be analysed. Then we will see the result of the KBest algorithm, from Sci-Kit Learn, and as a way to verification the graphics of the features considered most influential will be analysed.
The correlation matrix of the dataset in this phase is shown in the following figure.

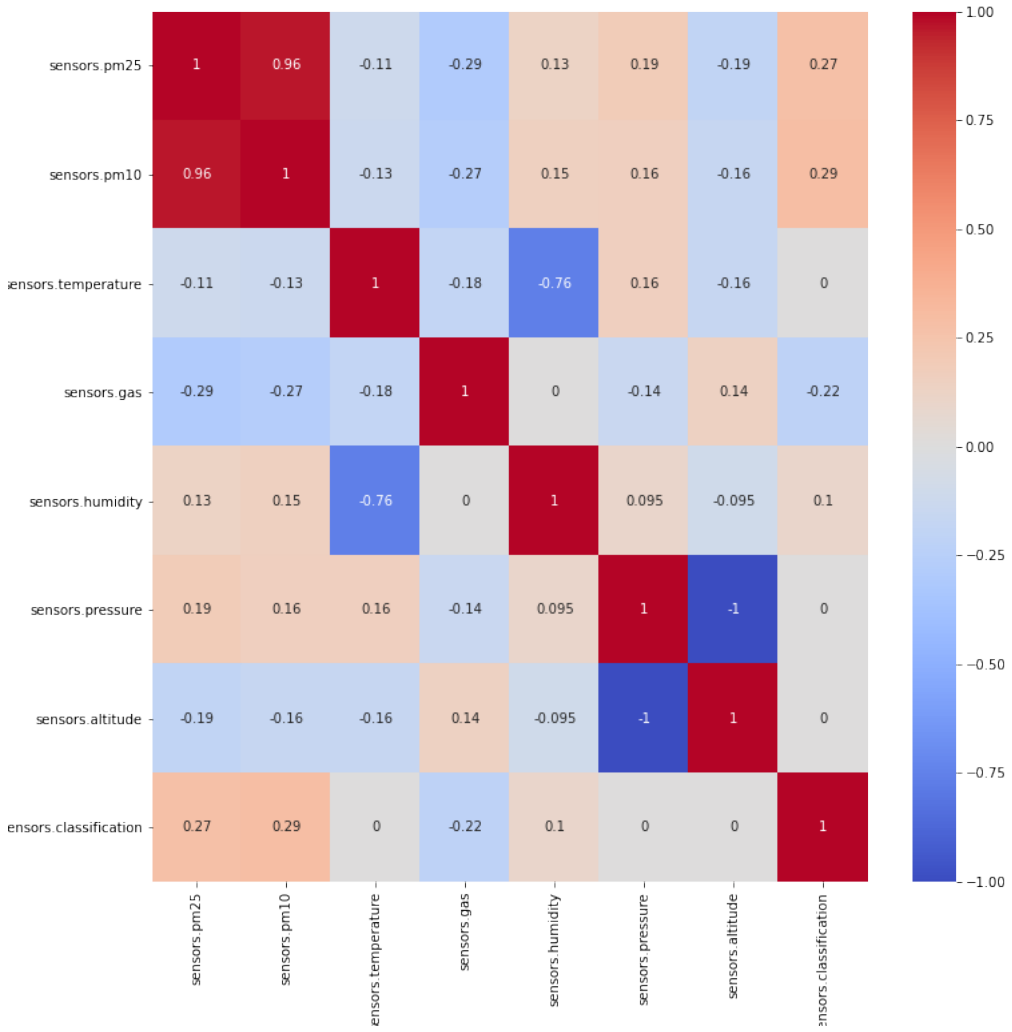

Figure 2.11: Correlation matrix.

In comparative terms with the heatmap, illustrated in Fig 2.8, referring to the previous phase, is clearly visible that the correlation is much lower at this stage, this decrease is due to the the fact that the values are more dispersed and distributed among more types of anomaly.
The dispersion of values causes a decrease on the influence of a single feature on the final result, thus requiring combining them to get a prediction. Even so, it is certain that the most influential columns of the previous phase, remain the most influential.
Now, we applied the KBest method over the data. The expected result is the same as in the previous phase, since in the correlation matrix also indicates that. So the result of the KBest is as follows.

```
Version 2:
```

```
The best 3 features are:
    - pm25
    - pm10
    - gas
```

The result is as expected. Despite the differences in values and the increase in situations to be detected these features continue to contribute the most to the prediction. Then the graphs of these 3 columns were examined, in order to understand why these columns are the chosen ones, not only through visual analysis but also through the correlation one

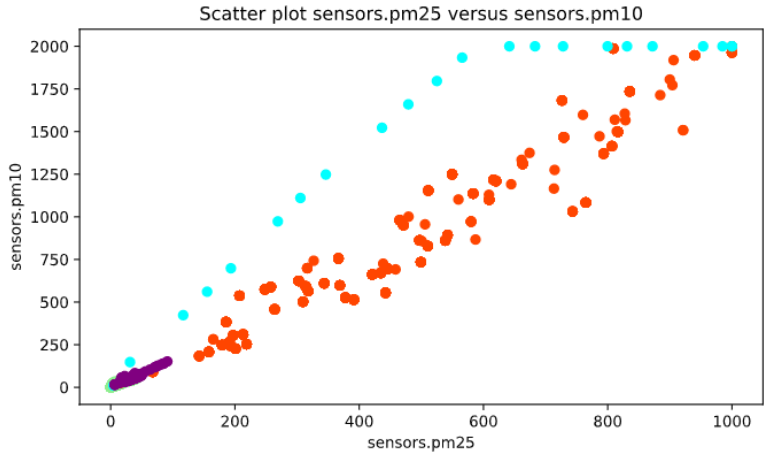

Figure 2.12: Graph *pm25* x *pm10*

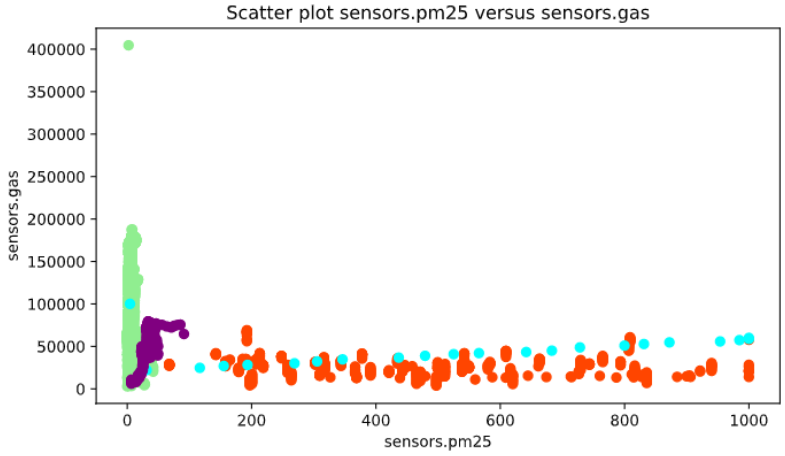

Figure 2.13: Graph *pm25* x *gas*

The distinction between the smoke (orange) and perfume (cyan) is easier to verify in this moment as it is possible to distinguish between the corresponding points on the graph in Fig 2.12. A greater challenge will be to distinguish perfume (cyan) from bad odour (purple). Visually in

the pairplot, because it has a more intense color, lead us to think that it would be possible to make this distinction. It cannot be said that it will be impossible, since by the values in both the graphs are concentrated, which shows that the values are within a certain range. However, it is also unwise to say that it will be possible. It is an interesting research that might be reward if it turns out that it is effectively possible to distinguish between odours.
In the following sections, we will talk about the pre-processing of the data, which is very important to ML models. Then their transition into optimised predictive models are covered by the group.

2.3 Data Preprocessing

As previously mentioned, the preprocessing stage is very important in the context Machine Learning models applications, and also in many Data Science projects. Thus, in this stage, were are going to apply on data small transformations and cleaning some erratic values, in order to keep consistency in the dataset. Additionally, some of the collected metrics are plot on BoxPlots, to understand the distribution of their values, spotting outliers and if necessary, apply some type of treatment to them. Thus, a data transformation technique that will be used for the analysis of future results. Moreover, error correction mechanisms will be applied on our datasets, particularly if any of the sensors cannot read the actual data (miscaptures due to physical failure).

2.3.1 Data Handling

As a way of identifying each capture/registration, the microcomputers combined with the sensors have very specific information about the records that are important for their characterisation, but they will have to be removed for the training and validation process of our algorithms. Therefore, some of the columns that are collected and filled in will be deleted for the the training process, these being:

- Id - capture identifier
- CarId - licensePlate from the vehicle that capture was taken
- CarLocation - GPS coordinates, identifying the position of the vehicle during the anomaly
- Timevalue - date and time corresponding to the capture made

In this sense, is expected that only the metrics corresponding to the values sensors, since only these data is useful and important for the process of learning, since identifiers of vehicle,etc should not be part of the process.The previous format of the records, as well as the current format after removing the accessible columns, can be seen below.

	sensors.id	sensors.carId	sensors.carLocation	sensors.timeValue	sensors.pm25	sensors.pm10	sensors.temperature	sensors.gas
0	553	66-ZZ-66	41.5608 -8.3968	2020-12-02 23:54:53	9.5	13.0	21.639570	21515
1	552	66-ZZ-66	41.5608 -8.3968	2020-12-02 22:24:04	16.5	24.7	20.131758	34165
2	551	66-ZZ-66	41.5608 -8.3968	2020-12-02 22:23:53	16.1	25.5	20.107344	33695
3	550	66-ZZ-66	41.5608 -8.3968	2020-12-02 22:23:43	16.0	26.4	20.089375	33113
4	549	66-ZZ-66	41.5608 -8.3968	2020-12-02 22:23:32	15.9	24.4	20.071992	32696

sensors.pm25	sensors.pm10	sensors.temperature	sensors.gas	sensors.humidity	sensors.pressure
9.5	13.0	21.639570	21515	43.570138	982.296484
16.5	24.7	20.131758	34165	46.910041	982.307568
16.1	25.5	20.107344	33695	46.782896	982.325957
16.0	26.4	20.089375	33113	46.737320	982.321309
15.9	24.4	20.071992	32696	46.847113	982.322707

Figure 2.14: Before and after unnecessary columns removal

In this way, only the columns that are strictly necessary and relevant are maintained, these being pm25, pm10, temperature, gas, humidity, pressure. Altitude must be ignored as we demonstrated on 2.2. Continuing the process of cleaning and processing data, it is important to proceed with the verification of inadmissible values in our data set, such as missing values, NaN, among others. In this way, the group proceeded to verify these same values, indicators that really attest to the absence of unwanted values.

```
Number of null values:
    sensors.id                0
    sensors.carId             0
    sensors.carLocation       0
    sensors.timeValue         0
    sensors.pm25              0
    sensors.pm10              0
    sensors.temperature       0
    sensors.gas               0
    sensors.humidity          0
    sensors.pressure          0
    sensors.altitude          0
    dtype: int64

Number of NaN:
    Total: 0
```

2.3.2 Presenting and removing outliers

One of the very important steps in the Machine Learning workflow is, without a doubt, the visualisation and removal of outliers. First, the group presented some BoxPlots for each of the features, to visually the possible existence of some value outside the normal range. Thus, some examples of these are presented below.

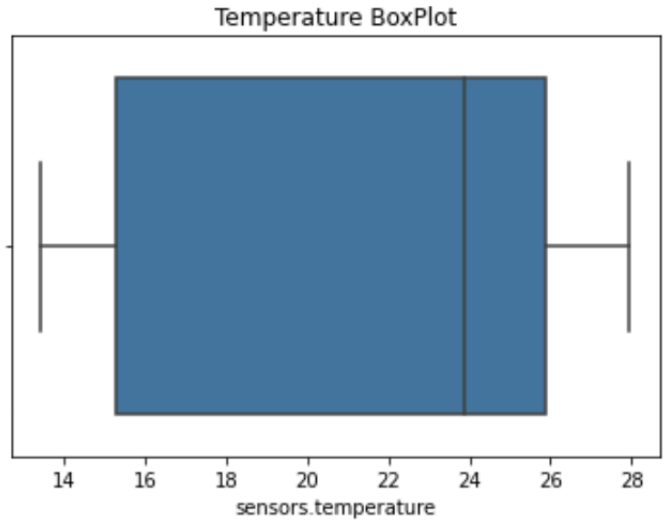

Figure 2.15: Temperature boxplot.

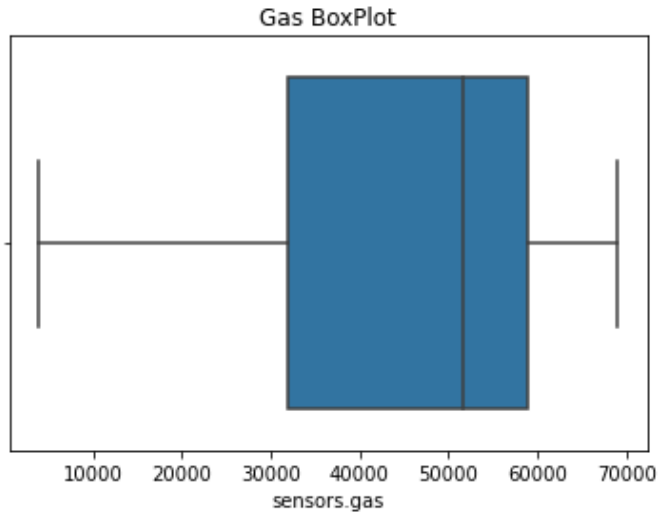

Figure 2.16: Gas boxplot.

As you can see by looking at the graphs shown, there is any values outside normal or spectrum that has already been inferred to be permissible. Still, the group decided to implement two of the most used techniques to identify and remove outliers: Z-score method and also IQR. The

application of these two techniques identified some values as being outliers, and a threshold of 3 was used for the Z-score method (value of reference) and knowing that the IQR method will remove what is beyond Q25 and Q75, this that is, quartile 25 and quartile 75, respectively. However, and considering the entire data collection process, knowing in advance that the **values** were **recorded** in **controlled environments**, with adequate conditions for recording and that each catch was correctly identified according to the situation it characterises, the group decided **not removing** the captures identified as outliers since they represent an **important source of variability** for generating knowledge in the process of **classification** and **detection** of **anomalies**.

2.3.3 Handling Miscorrecting captures

Often, hardware presents itself as a limitation in several ways. In this concrete project, hardware plays a key role and it will be important that the our system is resilient in terms of physical device failures. Therefore, before training the models and drawing the first conclusions, the group realised that the sensors had erroneous readings, mainly the SDS011 (particle) sensor. Some values were, in fact, strange since they had zero values for pm25 and pm10. Now, these captures, inserted between readings with substantially higher values, they told us that there were some bad readings. In that follow-up, those erroneous readings were eliminated from our data set. The results obtained are as follows.

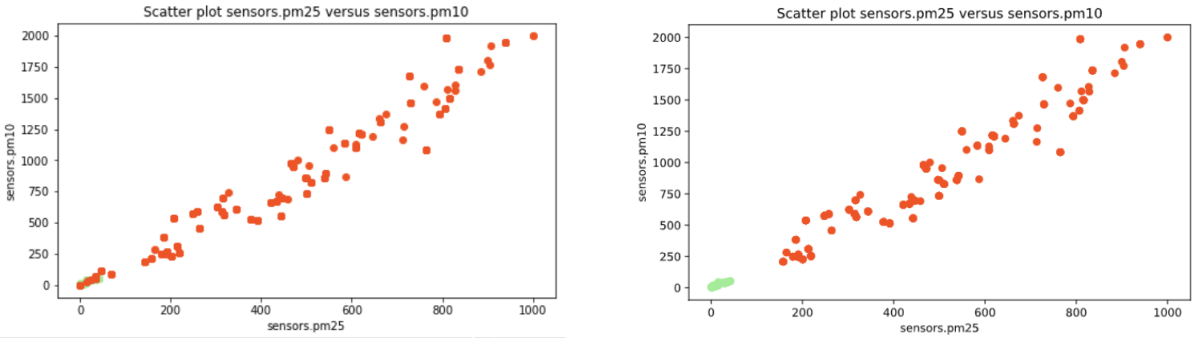

Figure 2.17: Before and after of data correction

2.4 Algorithms

After analysing and pre-processing data, it was decided to resort to the implementation of two types of algorithms that could be relevant in the context, Supervised and Unsupervised Machine Learning Algorithms.

Supervised algorithms are those that use an algorithm to learn the function mapping between

input variables (X) and output variables (y) [1]. During the learning phase, the algorithm has access to the correct “answers”, that is, the classification real. In this way, it is possible to make predictions iteratively on the training data and correct them, until the algorithm performs well. Considering that the data collection processes were carried out by the group in controlled scenarios, it was possible to obtain the correspondence of the data classification values and to allow the use of supervised algorithms.

In contrast, the unsupervised algorithms have only input data (X), not holding the corresponding output variables (y) [1]. Thus, this type of algorithms focuses on modeling the structure and distribution of data to learn more about them.

In this situation, as already mentioned, it became possible to implement supervised and unsupervised algorithms. Of each type, some were selected, according to their characteristics, which will be described below.

It is important to note that all columns resulting from 2.3 - [’pm25’, ‘pm10’, ’temperature’, ’gas’, ‘humidity’, ’pressure’] were used.

Next, the different implemented algorithms are presented, as well as the results when applied to the dataset, where only normal and smoke data (anomalies) are included.

2.4.1 Supervised Learning Techniques

- SVM - Support Vector Machine

  For data classification, this algorithm is based on the concept of maximising the minimum distance between a hyperplane and the nearest sampling point. This hyperplane, in a dimension greater than space, represents a decision limit in the input space, resulting from the mapping of the input data set to a space with multidimensional characteristics, using a determinant kernel function [2].

  The SciKit Learn library was used to implement this algorithm. Tests were made to the kernel functions "Polynomial", "RBF", "Sigmoid" and "Linear". Bearing in mind that, at this moment, binary classification is being made - normal or anomaly, the "Linear" model obtained the best results, as expected by the analysis of the data distribution.

  Also using SciKit Learn, the Grid Search tuning technique was used to optimise the values of the hyperparameters, in this case for “C” (regulation parameter) and “gamma” (the coefficient for some kernel parameters).

```
=============Results=============
Accuracy =>  1.0
Precision =>  1.0
Mean Absolute Error => 0.0
Mean Squared Error => 0.0
Root Mean Squared Error => 0.0
Classification Matrix :
[[451   0]
 [  0 141]]
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       451
           1       1.00      1.00      1.00       141

    accuracy                           1.00       592
   macro avg       1.00      1.00      1.00       592
weighted avg       1.00      1.00      1.00       592
```

Figure 2.18: SVM algorithm results

```
Best parameters =>  {'C': 0.1, 'gamma': 1, 'kernel': 'linear'}
```

Figure 2.19: Result of using Grid Search

- NN - Neural Networks

  Neural Networks consist of the distribution of processing by a dense interconnection of small units, called neurons [2]. There are multiple variants of NN, however, in the simplest cases, there is a set of input neurons that are activated by input data. Here, the data will undergo processing defined in the model and its transformation will be passed on to the next layer. After performing the defined iterations, the output neuron is activated and the process ends.

  Regarding the implementation, two variants were used. Firstly, Multi-layer Perceptron Classifier from the SciKit Learn library was implemented. Three hidden layers with 6 neurons each and a “ReLU” activation function were used, obtaining excellent results.

```
Algorithm: SK-Learn Neural Network
=============Results=============
Accuracy =>  1.0
Precision =>  1.0
Mean Absolute Error => 0.0
Mean Squared Error => 0.0
Root Mean Squared Error => 0.0
Classification Matrix :
[[451   0]
 [  0 141]]
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       451
           1       1.00      1.00      1.00       141

    accuracy                           1.00       592
   macro avg       1.00      1.00      1.00       592
weighted avg       1.00      1.00      1.00       592
```

Figure 2.20: Result of the MLP algorithm through SciKit Learn

Subsequently, the group opted to implement a sequential model, through the Keras library. Then, the Dense layers used are presented, the first and second with activation function "ReLU" and the last "sigmoid". It should also be noted that the loss “binary_crossentropy” was used in the compilation of the model, at this stage we are only doing a binary classification - normal and anomaly.

With only 20 epochs, the model achieved excellent results.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 128)               896
_________________________________________________________________
dense_5 (Dense)              (None, 16)                2064
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 17
=================================================================
Total params: 2,977
Trainable params: 2,977
Non-trainable params: 0
```

Figure 2.21: Layers of the NN model through the Keras library

```
Algorithm: Keras Neural Network
=============Results=============
Accuracy =>  0.9915540540540541
Precision =>  1.0
Mean Absolute Error => 0.008445945945945946
Mean Squared Error => 0.008445945945945946
Root Mean Squared Error => 0.09190182776172597
Classification Matrix :
[[451   0]
 [  5 136]]
Classification Report :
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       451
           1       1.00      0.96      0.98       141

    accuracy                           0.99       592
   macro avg       0.99      0.98      0.99       592
weighted avg       0.99      0.99      0.99       592
```

Figure 2.22: Result of the NN algorithm through the Keras library

- Decision Trees - Random Forest

  Decision Trees consist of making decisions according to established rules. These rules are associated with branches that lead to the next nodes until the end of the tree [2]. To avoid the common overfitting caused by decision trees, the Random Forest ensemble learning method was used, which consists of training a series of decision trees, returning the class with the majority overall existing ones in the set.

  For this algorithm and in this intermediate phase, the Random Forest Regressor implementation of the SciKit Learn library was used, with “n_estimators” (number of trees in the forest) equal to 50.

```
Algorithm: Random Forest
=============Results=============
Accuracy =>  1.0
Precision =>  1.0
Mean Absolute Error => 0.0
Mean Squared Error => 0.0
Root Mean Squared Error => 0.0
Classification Matrix :
[[451   0]
 [  0 141]]
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       451
           1       1.00      1.00      1.00       141

    accuracy                           1.00       592
   macro avg       1.00      1.00      1.00       592
weighted avg       1.00      1.00      1.00       592
```

Figure 2.23: Result of the Random Forest algorithm

- Naive Bayes

  This algorithm is a probabilistic classifier and is characterised by the strong assumption of independence between the child nodes of the same parent node, that is, it disregards the correlation between the features [2]. (Expansion of Bayes' Theorem)

  Regarding its implementation, the simplest version of the SciKit Learn library was used.

```
Algorithm: Naive Bayes
=============Results=============
Accuracy =>  0.9780405405405406
Precision =>  0.9155844155844156
Mean Absolute Error => 0.02195945945945946
Mean Squared Error => 0.02195945945945946
Root Mean Squared Error => 0.14818724459095478
Classification Matrix :
[[438  13]
 [  0 141]]
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.97      0.99       451
           1       0.92      1.00      0.96       141

    accuracy                           0.98       592
   macro avg       0.96      0.99      0.97       592
weighted avg       0.98      0.98      0.98       592
```

Figure 2.24: Result of the Naive Bayes algorithm

2.4.2 Unsupervised Learning Techniques

- K-means Clustering

  This is one of the most popular clustering algorithms, being the classification made from the analysis and comparisons (through distances) between the input data (X), since it doesn't have access to the labels (y). In this way, clusters and respective centroids are formed, recalculated at each iteration of the algorithm until converging [3].

As already mentioned, in this phase of the project, the classification covers only two situations - normal and anomalies. Thus, the implementation of this algorithm was performed using the SciKit Learn library, with the number of clusters provided equal to 2. However, regardless of the maximum number of iterations, the results are quite variable and inconsistent.

```
Algorithm: K-Means Clustering
Training data Accuracy :
============Results============
Accuracy =>  0.7653874004344677
Precision =>  0.5146103896103896
Mean Absolute Error => 0.2346125995655322
Mean Squared Error => 0.2346125995655322
Root Mean Squared Error => 0.48436824789155225
Classification Matrix :
[[740 299]
 [ 25 317]]
Classification Report :
              precision    recall  f1-score   support

           0       0.97      0.71      0.82      1039
           1       0.51      0.93      0.66       342

    accuracy                           0.77      1381
   macro avg       0.74      0.82      0.74      1381
weighted avg       0.86      0.77      0.78      1381
```

Figure 2.25: Result of the K-means algorithm

The figure below is an example of the results obtained by this model, applied to the 'pm25' and 'pm10' columns. Through it, it is possible to verify that, although the centroids of each cluster are visibly separable, there is a great dispersion of values, which is reflected in the results of the model.

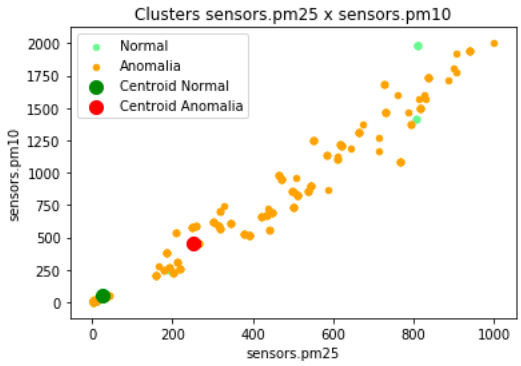

Figure 2.26: Clusters formed by the K-means algorithm - 'pm25' and 'pm10'

- Local Outlier Factor (LOF)

  LOF detects anomalies due to the local density deviation of a data point compared to its neighbours [3]. Through this calculation, it becomes possible to ascertain the samples whose density is lower, which will be considered outliers.

Regarding the implementation of this unsupervised algorithm, the SciKit Learn library was once again used. For the number of neighbours to consider, after several tests, it was defined that 10 would be a good value and the metric used was "manhattan".

As in the previous algorithm, the results were quite inconsistent, with zero precision in relation to the anomaly values.

```
=============Results=============
Accuracy =>  0.9738562091503268
Precision =>  0.0
Mean Absolute Error => 0.026143790849673203
Mean Squared Error => 0.026143790849673203
Root Mean Squared Error => 0.16169041669088866
Classification Matrix :
[[1490    0]
 [  40    0]]
Classification Report :
              precision    recall  f1-score   support

           0       0.97      1.00      0.99      1490
           1       0.00      0.00      0.00        40

    accuracy                           0.97      1530
   macro avg       0.49      0.50      0.49      1530
weighted avg       0.95      0.97      0.96      1530
```

Figure 2.27: Result of the algorithm *LOF*

When looking at the figure 2.28, which shows the values 'pm25' and 'pm10' of the data considered outliers and, therefore, anomalies, a dispersion of data is also visible, as in the figure 2.26.

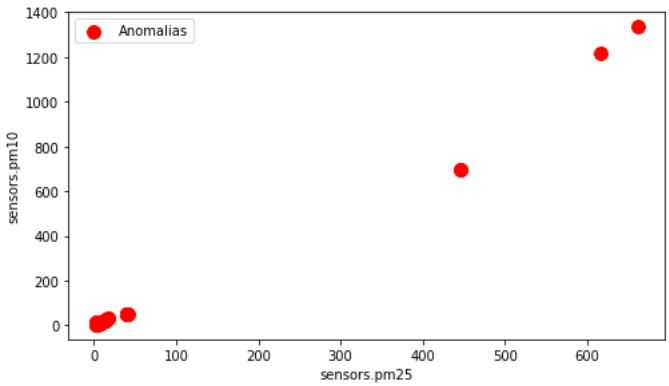

Figure 2.28: Outliers detected in the LOF algorithm - 'pm25' and 'pm10'

- Isolation Forest

  As the name implies, this algorithm consists of creating decision trees on random attributes, in an attempt to isolate anomalies through the partitions [3]. In the end, each value is assigned a score, associated with the possibility of being an anomaly or not, depending on the level of isolation of that point.

Using the SciKit Learn library, this algorithm was implemented with the number of estimators equal to 64 and the maximum number of features equal to 3, which means that this will be the number of features to use from the dataset to train each estimator base.

The results of this model were similar to previous unsupervised algorithms.

```
=============Results=============
Accuracy =>  0.7518987341772152
Precision =>  0.4948453608247423
Mean Absolute Error => 0.2481012658227848
Mean Squared Error => 0.2481012658227848
Root Mean Squared Error => 0.498097646875374
Classification Matrix :
[[201  98]
 [  0  96]]
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.67      0.80       299
           1       0.49      1.00      0.66        96

    accuracy                           0.75       395
   macro avg       0.75      0.84      0.73       395
weighted avg       0.88      0.75      0.77       395
```

Figure 2.29: Result of the Isolation Forest algorithm

In the figure below are the values 'pm25' and 'pm10' of the data considered anomalies, depending on the score assigned by the model. The same pattern of unsupervised algorithms is repeated.

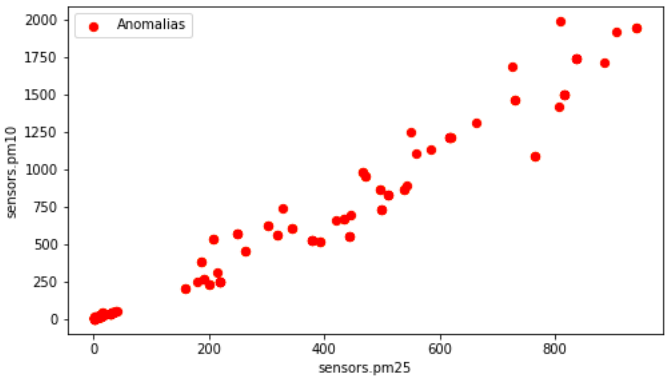

Figure 2.30: Anomalies detected in the IF algorithm - 'pm25' and 'pm10'

- One-Class SVM

  This algorithm consists of training the model only with the values corresponding to normal data, in order to know the limits of the same [3]. In this way, the model becomes able to classify, later, values that do not belong to the set of these data.

  For the implementation of this algorithm, the SciKit Learn library was also used, with the kernel function "rbf". The results were also similar to previous unsupervised algorithms.

```
=============Results=============
Accuracy =>  0.6911392405063291
Precision =>  0.4577777777777778
Mean Absolute Error => 0.30886075949367087
Mean Squared Error => 0.30886075949367087
Root Mean Squared Error => 0.555752426439751
Classification Matrix :
[[170 122]
 [  0 103]]
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.58      0.74       292
           1       0.46      1.00      0.63       103

    accuracy                           0.69       395
   macro avg       0.73      0.79      0.68       395
weighted avg       0.86      0.69      0.71       395
```

Figure 2.31: One-Class SVM algorithm result

The 'pm25' and 'pm10' values of the data considered normal (green) and anomalies (orange) are shown in the graph below. It appears that the model shows good results for detecting normal values (0), but the same does not happen with anomalies (1), with dispersion of values once again.

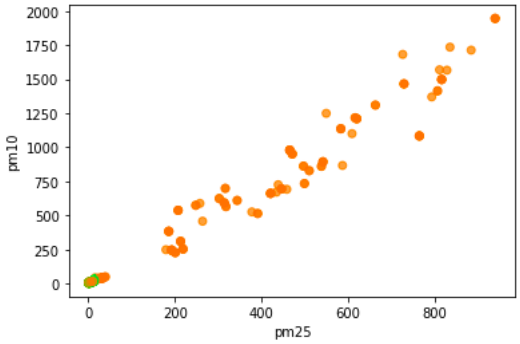

Figure 2.32: One-Class SVM algorithm result

2.4.3 Intermediate Observations (Smoking)

In short, there is a huge difference in the evaluation and results of supervised models compared to unsupervised ones.

This discrepancy is due to the dataset to which the models were applied. As already mentioned in the data analysis, there is great variability and dispersion of anomalous data (smoke). Obviously, unsupervised models, having no access to the classification value (y), have much more difficulty in classifying the data. Supervised models, having labels available, end up being successful in getting around this problem.

2.4.4 Application of Algorithms to the Final Dataset

As already mentioned, initially the algorithms were collected, analysed and tested with data only referring to normal situations and anomalous situations, corresponding to smoke.

In a more advanced stage, it was decided to introduce anomalies associated with bad odours, and data were collected for them. These values were classified as "2".

In this case, taking into account the unsatisfactory results of the unsupervised models, the decision was made to use only the classification resulting from supervised learning techniques. The data remains highly variable and, consequently, the unsupervised models would continue to maintain worse results.

Thus, the results obtained from the dataset resulting from the collection of normal data (0), smoke (1 - anomaly) and bad odour (2 - anomaly) are followed.

- SVM - Support Vector Machine

  This model maintained the structure already presented in 2.4.1, using the Grid Search tuning technique, to optimise the values corresponding to the hyperparameters.

```
=============Results=============
Accuracy =>  0.9948320413436692
Precision =>  0.9949188977916748
Mean Absolute Error => 0.0103359173126615
Mean Squared Error => 0.020671834625323
Root Mean Squared Error => 0.14377703093791788
Classification Matrix :
[[138   0   2]
 [  0 130   0]
 [  0   0 117]]
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       140
           1       1.00      1.00      1.00       130
           2       0.98      1.00      0.99       117

    accuracy                           0.99       387
   macro avg       0.99      1.00      0.99       387
weighted avg       0.99      0.99      0.99       387
```

Figure 2.33: SVM algorithm results

```
Best parameters =>  {'C': 0.1, 'gamma': 1, 'kernel': 'linear'}
```

Figure 2.34: Result of using Grid Search

- NN - Neural Networks

  As in 2.4.1, two variants were used for this technique.

  First, the Keras library was used. In the figure 2.35 it's possible to observe the layers used in the model, the first and second with "ReLU" activation function and the last "softmax".

In this case, because it is possible to obtain three different classifications, the adaptation was made with the loss passing to "categorical_crossentropy", instead of "binary". The "adam" optimiser was also used.

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_19 (Dense)             (None, 64)                448
_________________________________________________________________
dense_20 (Dense)             (None, 16)                1040
_________________________________________________________________
dense_21 (Dense)             (None, 3)                 51
=================================================================
Total params: 1,539
Trainable params: 1,539
Non-trainable params: 0
```

Figure 2.35: Layers of the NN model through the Keras library

```
=============Results=============
Accuracy =>  0.6795865633074936
Precision =>  0.8157186010117553
Mean Absolute Error => 0.599483204134367
Mean Squared Error => 1.1576227390180878
Root Mean Squared Error => 1.0759287797145718
Classification Matrix :
[[ 25   7 108]
 [  0 128   2]
 [  0   7 110]]
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.18      0.30       140
           1       0.90      0.98      0.94       130
           2       0.50      0.94      0.65       117

    accuracy                           0.68       387
   macro avg       0.80      0.70      0.63       387
weighted avg       0.82      0.68      0.62       387
```

Figure 2.36: Result of the NN algorithm through the Keras library

In addition, a version of the network was also implemented through the SciKit Learn library. However, this time there was also an optimisation of the hyperparameters corresponding to the size of the layers, the activation function, the solver, the alpha, and the learning rate, using the Grid Search technique.

```
=============Results=============
Accuracy =>  0.7777777777777778
Precision =>  0.8579546237430313
Mean Absolute Error => 0.3772609819121447
Mean Squared Error => 0.6873385012919897
Root Mean Squared Error => 0.8290588044837288
Classification Matrix :
[[ 77   3  60]
 [  0 110  20]
 [  0   3 114]]
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.55      0.71       140
           1       0.95      0.85      0.89       130
           2       0.59      0.97      0.73       117

    accuracy                           0.78       387
   macro avg       0.85      0.79      0.78       387
weighted avg       0.86      0.78      0.78       387
```

Figure 2.37: Result of the MLP algorithm with SciKit Learn

```
{'activation': 'tanh',
 'alpha': 0.05,
 'hidden_layer_sizes': (128, 64, 16, 3),
 'learning_rate': 'constant',
 'solver': 'adam'}
```

Figure 2.38: Result of using Grid Search

- Decision Trees - Random Forest

  Regarding this model, the use of the SciKit Learn library was maintained.

  However, as in the previous cases, Grid Search was implemented to optimise several hyperparameters, such as the number of trees in the forest.

```
=============Results=============
Accuracy =>  0.9948320413436692
Precision =>  0.9949188977916748
Mean Absolute Error => 0.0103359173126615
Mean Squared Error => 0.020671834625323
Root Mean Squared Error => 0.14377703093791788
Classification Matrix :
[[138   0   2]
 [  0 130   0]
 [  0   0 117]]
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       140
           1       1.00      1.00      1.00       130
           2       0.98      1.00      0.99       117

    accuracy                           0.99       387
   macro avg       0.99      1.00      0.99       387
weighted avg       0.99      0.99      0.99       387
```

Figure 2.39: Result of the Random Forest algorithm

```
{'bootstrap': True,
 'max_depth': 10,
 'max_features': 'auto',
 'n_estimators': 40}
```

Figure 2.40: Result of using Grid Search

- Naive Bayes

  Finally, this algorithm was also optimised using the Grid Search technique. "StratifiedK-Fold" was used for cross-validation, with three splits.

```
=============Results=============
Accuracy =>  0.979328165374677
Precision =>  0.9794329595673794
Mean Absolute Error => 0.041343669250646
Mean Squared Error => 0.082687338501292
Root Mean Squared Error => 0.28755406187583576
Classification Matrix :
[[135   0   5]
 [  0 130   0]
 [  3   0 114]]
Classification Report :
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       140
           1       1.00      1.00      1.00       130
           2       0.96      0.97      0.97       117

    accuracy                           0.98       387
   macro avg       0.98      0.98      0.98       387
weighted avg       0.98      0.98      0.98       387
```

Figure 2.41: Result of the Naive Bayes algorithm

```
{'var_smoothing': 1e-07}
```

Figure 2.42: Result of using Grid Search

2.4.5 Concluding Remarks

As can be seen, only the Neural Network algorithm shows less satisfactory results, showing difficulty in detecting anomalies for these data. This model would need more data to obtain better results.

However, with the excellent results obtained in the other supervised models, it is possible to conclude that, in fact, it is possible to distinguish and detect anomalies from smoke and/or bad odour.

3 AlertAI Cloud

The creation of the AlertAI Cloud component, as the name implies, requires an extension of the component previously presented, AlertAI, with particular extension in its domain and Cloud hosting. In this sense, this component will aim to constitute a source of investigation in terms of the anomaly detection process, allowing comparisons with the algorithm chosen for AlertAI. In this way, this component will house several algorithms for classifying and detecting anomalities so that, in terms of research and exploration of the system, it is possible to perceive the behaviour of several algorithms, equally trained and operating under the same conditions may possibly differ. Objectively, an application was built using the Python language and the framework Flask to produce a simple REST API that will serve to fulfil the various requests that will be received. In a very synthetic way, the main features of this component are:

1. Receive, process, classify and store classifications
2. Return all the alternative classifications for a given vehicle (licensePlate) and timeValue (timestamp)

Next, we will detail each of these features in some detail.

3.1 Classification with Alternative Algorithms

As a way to persist the various classifications that the alternative algorithms will return, it was necessary to develop a logic similar to that described in the AlertAI component. In this context,

the alternative algorithms are the algorithms that, among the presented in 2.4, have similarities in terms of performance to the chosen algorithm chosen for AlertAI, leaving the one that best features and capabilities presented for local application in systems that will accompany users inside vehicles. Thus, using an SQLite database, data as soon as it is received, in raw format, are sent to the set of alternative algorithms previously trained so that they can classify that same data. In the end, for each of the algorithms, the respective classification and the raw data that originated that decision are kept.

3.2 Return Algorithm Classifications

As a way of making available to the user the alternative classifications that the models AlertAI Cloud have ruled, it is necessary to make these available so that the user can perceive, for example, in visual format the differences that the various algorithms suggest and in what occasions. With this, an endpoint was developed where it will be possible to access all classifications of algorithms present for a given classification.

3.3 Deploy

As a way to deploy this system, the group decided to dedicate a viral machine independent for this purpose, since the number of operations that will eventually be required will be high for each order, so if the number of orders is considerable, it was important to make enough resources available for the normal functioning of the component. In order to accommodate all the operating logic, a Shell script is provided that will react use the connection to the Cloud bucket created to store all configuration files needed. After the script is initialised, you will be responsible for:

- Copy necessary files
- Install docker on Virtual Machine
- Create exception on firewall for specific port (5000)
- Build docker image
- Start container

Thus, the Docker image will contain the necessary Python version to allow the execution of the component, as well as installing all the necessary requirements in terms of packages. after the process finished, port 5000 will be listening for requests that are destined for AlertAI Cloud.

4 Conclusion

The AlertAI component required extensive work in terms of statistical analysis of the data collected. It was important to create a context for each of the scenarios - normal and anomaly due to the presence of smoke or a bad odour, accompanied by a strong study of the values, to understand the influence of factors such as intensity, distance and sensitivity of the sensors, among others.

After that, it was necessary to build a complete pipeline for implementing Machine Learning algorithms, in order to classify and detect anomalies. This process included performing data processing and building the various algorithms, as well as optimising them. Knowing in advance that this AlertAI component is central, the group devoted much of the time to improve its functioning.

AlertAI Cloud was also used as a component extension, to store other developed algorithms.

Thus, it is concluded that the system is perfectly capable of detecting anomalies, whether they come from smoke or a bad smell.

Bibliography

[1] Sathya, R. & Abraham, Annamma. (2013). Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. International Journal of Advanced Research in Artificial Intelligence. 2. 10.14569/IJARAI.2013.020206.

[2] A. Singh, N. Thakur and A. Sharma, "A review of supervised machine learning algorithms," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2016, pp. 1310-1315.

[3] Khanam, Memoona & Mahboob, Tahira & Imtiaz, Warda & Ghafoor, Humaraia & Sehar, Rabeea. (2015). A Survey on Unsupervised Machine Learning Algorithms for Automation, Classification and Maintenance. International Journal of Computer Applications. 119. 34-39. 10.5120/21131-4058.