Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

Machine Learning Software for Automated Satellite Telemetry Monitoring

Relator:
Chiar.mo Prof.
ANDREA ASPERTI

Presented by:
LUCIANO MASSACCESI

Co-Relator:

Propulsion System Engineer
INGOLF FISCHER

III Session
Accedemic Year 2018-2019

Abstract

During the lifetime of a satellite malfunctions may occur. Unexpected behaviour are monitored using sensors all over the satellite. The telemetry values are then sent to Earth and analysed seeking for anomalies. These anomalies could be detected by humans, but this is considerably expensive. To lower the costs, machine learning techniques can be applied.

In this research many different machine learning techniques are tested and compared using satellite telemetry data provided by OHB System AG. The fact that the anomalies are collective, together with some data properties, is exploited to improve the performances of the machine learning algorithms. Since the data comes from a real spacecraft, it presents some defects. The data covers in fact a small time-lapse and does not present critical anomalies due to the spacecraft healthiness. Some steps are then taken to improve the evaluation of the algorithms.

Contents

Chapter 1

Introduction

This thesis has been developed at OHB System AG, a space system integrator. This offered the possibility to test machine learning techniques on a real problem, using real data coming from a satellite.

Satellites play an important role in modern technologies. The number of active satellites is continuously increasing as well as their complexity. During their lifetime malfunctions may occur. To detect unforeseen behaviour of satellites, many sensors are employed to monitor telemetry values (such as temperatures, currents, thrusters status, and so on). Such data is then sent back to Earth for analysis.

One way to detect anomalies on telemetry data is to set a range of normal behaviour for each telemetry value. If the measured value is outside that range, an anomaly is detected. This simple technique, though, is prone to give false positives, so employed ranges are not really restrictive in order to detect only obvious anomalies.

Another way is to have a human expert with knowledge on all the measured values to judge the behaviour of the parameters. This solution is obviously expensive in terms of human resources.

A better solution would be training a machine with past telemetry data to make it recognise anomalies in a real time stream of data. In this work different machine learning techniques are described and applied to the problem. The techniques are then compared

and the best one is selected.

In Chapter 2 the problem is explained more in details. In Chapter 3 all techniques taken from literature are explained. Chapter 4 explains how the techniques are adapted to the problem and proposes new techniques formulated especially for this problem. In Chapter 5 all results are shown and the different techniques are compared. In Chapter 6 conclusions are drawn.

Chapter 2

Set-up

2.1 Problem Characterization

To test different machine learning techniques, the data from a real satellite have been used. This satellite is one of the SmallGeo satellites developed by OHB System AG. SmallGeo is a family of versatile geostationary satellite platforms.
The data received from the satellite present the following characteristics:

- Dense data: over 200000 data points per year.
- Lack of data: only one year of data is available, while expected phenomena in the data have yearly periods. This makes it difficult to use the data to both train and test machine learning techniques
- Multidimensional data: each measured state of the system can be described by many dimensions, one for each sensor measurement. For this research a subset of 30 dimensions is used. This subset is associated with the power subsystem of the satellite, which includes temperatures, battery voltages and currents.
- Lack of anomalies: no serious anomalies are present in the telemetry data.

2.2 Starting Point

A set of tools has already been developed by OHB in prevision of this work.
That set of tools can be used to:

- Pre-process data
- Split data between train and test sets
- Generate realistic anomalies
- Other minor features

Raw data received from the satellite is not readable by machine learning algorithms. The data presents gaps with no values and invalid values. Furthermore, different sensors in the satellite are measured with different frequencies so they are not synchronized. This data needs to be prepared in order to be handled by machine learning techniques during the pre-processing phase. Part of this work has been already done in a previous activity. Nevertheless further pre-processing is necessary to improve machine learning results.

To test machine learning techniques, the data is split between train set and test set. To help splitting the data, the existing tool offers different algorithms to split data in different possible ways.

Existing anomalies in the considered data are non-relevant anomalies. Because of the lack of real relevant anomalies needed to test the algorithms, a tool generates realistic anomalies which simulate few cases of studied anomalies.

Chapter 3

Theoretical Background

This chapter introduces the theory behind the techniques that have been used in this research.

3.1 Types of anomalies

Anomalies could be classified in three categories: point (or global) anomalies, contextual anomalies and collective anomalies.

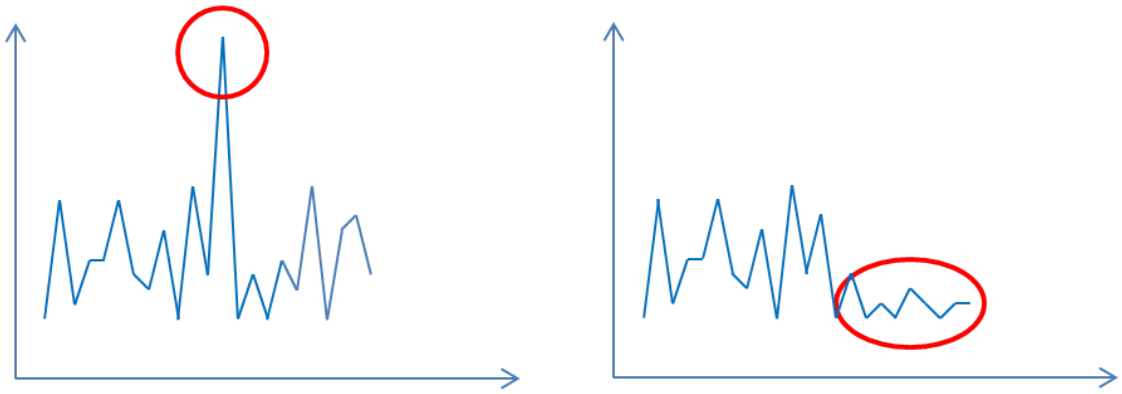
Point anomalies are data points which are considered anomalies independently from the context. For example, in satellite telemetries, a temperature value out of bounds could be seen as anomalous because in any possible case, the temperature should not be higher than a certain level.

Contextual anomalies are data points which are considered anomalies because of the context where they are. The context is often temporal such as the period of the year but there are also other kind of contextual anomalies. In satellite telemetries, for example, the current of the solar panel during the eclipse should be about zero (when the Sun is overshadowed by the Earth, the solar panel does not generate current). If the current is high during the eclipse, this could be an example of contextual anomaly.

Collective anomalies are groups of data points which are considered anomalous only because they are together (while if taken one by one are not considered so). In satellite

telemetry, the battery is continuously charged and used. The voltage of the battery should go up and down within a range. If the voltage stays at a constant value for a certain time frame, it could be seen as an anomaly, even if the value itself is not anomalous.

The graphs in Figure 3.1 show the difference between a global anomaly (Figure 3.1a) and a collective anomaly (Figure 3.1b). In Figure 3.1b the values of data points are low for a long period of time. Even if the values are not that low to be considered global anomalies, the fact that for a continued period of time all the values are low could be considered as an anomaly.



(a) Global anomaly (b) Collective anomaly

Figure 3.1: An example of two different types of anomalies

Understanding which kind of anomaly is likely to be present in the data is important to apply the right techniques. To detect contextual anomalies it is sufficient (in the case examinated in this research) to add the time as an additional data point dimension. On the other hand, global anomalies are less critical compared to the contextual anomalies. If a malfunction occurs in a satellite, it is likely to have a persistent problem such as something degraded, detached or broken. This problem should then be visible for a long period of time, not only for an instant. For this work only continued anomalies have been considered even though global anomalies could also be present in the data.

Even if some collective anomalies could be seen as a series of global anomalies and found as such, better techniques consider such property of the anomalies and use it to

increase the performances.

3.2 Machine Learning Techniques

Before explaining the different machine learning techniques, it is important to understand how machine learning techniques for anomaly detection work in general [6].

Machine learning techniques are algorithms that perform a task based on the experience instead of explicit instructions. Two set of data are needed: the train set and the test set. Machine learning techniques build a mathematical model based on the train set and use that knowledge to make predictions in the test set. The predictions could be of different types. In anomaly detection, a prediction is usually a score which stands for how much the data point is likely to be an anomaly.

There are two main classes of machine learning algorithms for anomaly detection: supervised and unsupervised. Supervised algorithms need to have train data with labels. These labels describe the anomalous status of the data. Unsupervised algorithms do not require any labelled data and assume that the train data is mainly non anomalous data.

3.2.1 Gaussian Mixture Model (GMM)

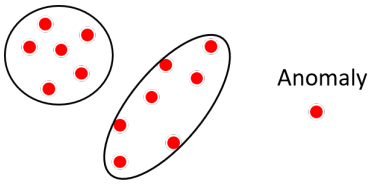


Figure 3.2: Gaussian mixture model

Gaussian mixture model is an algorithm based on clusters (Figure 3.2). In machine learning a cluster is a group of data, which is grouped following a rule (usually based on the distance from the centre of the cluster). The train set is fit into a number of clusters which represent the train data. Then the clusters are used to detect anomalies in the

test data: the more a point fits the clusters, the less it is likely to be an anomaly [8]. To fit the data inside the clusters, the expectation maximization algorithm is usually used. A number of elliptical clusters is randomly initialized. Then the shape, orientation and position of clusters is iteratively changed to fit the data more and more. When it is not able to fit the data any better, the algorithm stops. However, it can only converge to a local best fit of the clusters, which highly depends on how the clusters are initialized. The number of clusters is another sensible parameter. Depending on how many clusters there are, different distribution of data can be represented.

3.2.2 Local Outlier Factor (LOF)

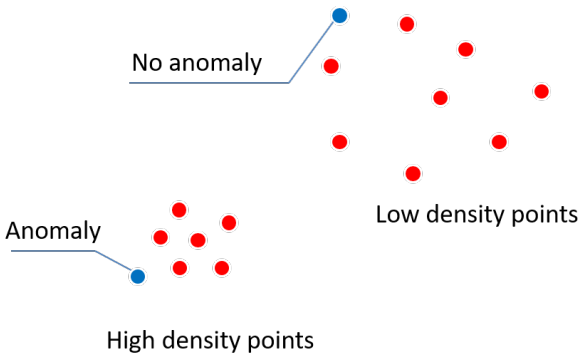


Figure 3.3: Local outlier factor

One simple way to detect if a data point is an anomaly is to check how close the K nearest points are. The mean of the distances of a point from its neighbours can be defined as the density around the point. The higher is the density, the more likely it is for the data points to be an anomaly. However, non-anomalous data points do not have the same density everywhere.

Local outlier factor compares the density measured in every test point with the density in the K neighbours. This gives a measure on how the point is close to its neighbours compared to how it should be (Figure 3.3)[1].

3.2.3 Angle Based Outlier Detector (ABOD)

Due to the high number of dimensions of the data, the concept of distance loses its meaning. This algorithm uses the concept of angles instead of distances. If a point is not an anomaly, it will probably have more points around it than an anomaly.

To determine if other points are around the considered one, an angle is formed with the test point and all other possible pair of points. A mean of the angles is then computed. The lower the result, the more likely it is for the data point to be an anomaly (Figure 3.4).

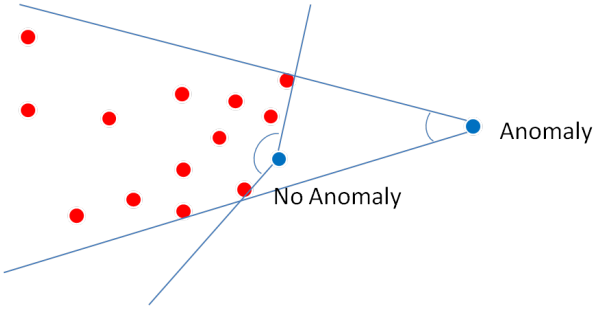


Figure 3.4: Angle based outlier detection

Since calculating the angle with all possible pairs is computationally hard, as an approximation, only the nearest points could be considered [5].

3.2.4 Neural Networks

A neural network is composed of many neurons, usually split in layers of parallel non-connected neurons as showed in Figure 3.5 [3]. Each layer can be fully connected with the next one or not (depending on the network design). The first layer is usually the input, the last layer is the output and all the other layers are called “hidden layers”. The value of the input passes through and gets modified by all the other neurons until the output layer.

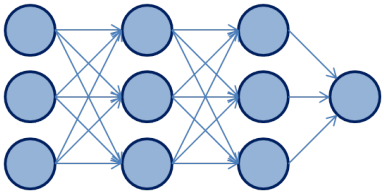


Figure 3.5: Neural network schema

A neuron (the scheme of which could be seen in the Figure 3.6) is the basic component of a neural network. It gets many values as input (one from each other neuron connected to it) and it has one output, which could be sent to different neurons.

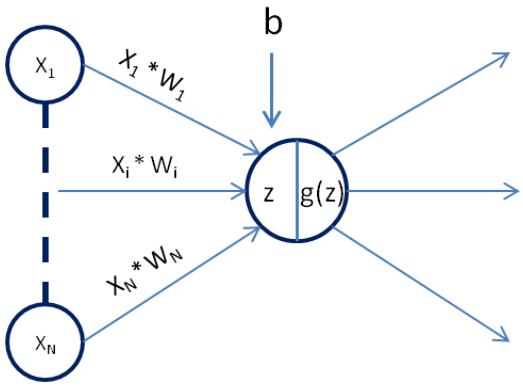


Figure 3.6: Neuron schema

The value of the neuron can be calculated as:

$$z = g\left(\left(\Sigma_{i=0}^{N} X_i * W_i\right) + b\right)$$

Here $X_1..X_N$ is the input from the precedent layer, $W_1..W_N$ are the weights of each connection and b is the bias. The function g(x) is the activation function, which could change from one layer of neurons to another. While training, the neuron can change the weights $W_1..W_N$ and the bias. The value of the neuron is then passed through the activation function before being transmitted to the next layer.

Training a network consists in changing weights and biases for each neuron in each layer to have the best output possible. The first step is then to define when an output is better than another one. A supervised neural network can easily compare its output with the expected values to get trained. When the network is unsupervised, there is

no such value to compare the result with. The core of unsupervised anomaly detection neural networks is to find a way to use the train set to train the network.

The loss function is responsible for telling the neurons how good the result is (the lower the loss function, the better the result). Once the loss function is defined, the neurons must be trained by changing the weights and the biases. To do that a back-propagation algorithm is usually used. This algorithm starts from the output neurons and back propagates until the beginning of the network. The previous weights are recursively changed to obtain the expected result. For the last layer the expected result is the expected output. The weights and biases of the last layer are changed to try to obtain the expected values. The values of each layer are then recursively changed to obtain the expected values on the next layer as for the last one. For each iteration the algorithm tries to minimize the loss function. However, finding the minimum in a function is not always easy: a local minimum could be found instead. Moreover, each step of the minimization function is computationally expensive. The input must in fact pass through all the neurons and all the weights must be adjusted for each step. To reduce the number of iterations, a function that gives slightly worse results but can find an acceptable minimum in just few tries is usually used.

3.2.5 Generative Adversarial Network (GAN)

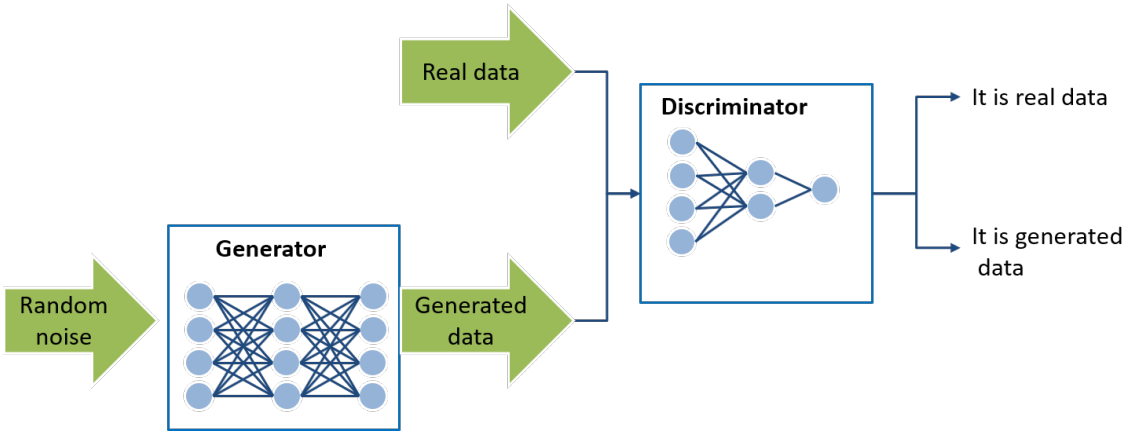


Figure 3.7: Generative adversarial network schema

As described in the scheme above, two different neural networks are connected together to make generative adversarial network working [2]. The two networks are the generator and the discriminator.

The generator takes a random input and outputs something which should be as similar as possible to the real train data. It gets rewarded if the discriminator fails.

The discriminator can randomly receive as input either real train data or the data generated from the generator. This neural network gets rewarded if it manages to distinguish the train set data from the artificial one.

Generative adversarial networks can be used for many different applications. When it is used for anomaly detection, only the discriminator is used while the generator is discarded once the train has been completed. The discriminator learns to discriminate if the data is similar to the train set. If it is not, it is more likely to be an anomaly.

GAN Improvements

Generative adversarial networks are not easy to train. The two networks that compose the GAN algorithm are in a competition one against the other. While training, both the discriminator and the generator get trained and try to outperform the other. This competition makes the system more complex because the improvements on the loss function of one network worsens the loss function of the other network. For this reason, if one network trains too quickly, the training process of the other network stops.

Another danger using this network is to over-fit the training data. The network could adapt more to the single data points rather than capturing the general behaviour of the data. This could lead to a failure when the test data differs significantly from the train data.

Two main techniques to solve these problems are introduced.

Label Smoothing

To avoid one of the two networks to over-perform the other, label smoothing could be used. Usually, the labels 1 and 0 are used to mark the data as generated or real. Ideally, 1 means that the data is for sure fake data, while 0 means that it is for sure real data. Using label smoothing, the values of truth are changed from an absolute value to a probabilistic value. The labels associated are no longer 0 and 1 but instead a value slightly more than 0 and a value slightly less than 1 (for example, 0.1 and 0.9). This change of labels slows down training but avoids one network to over-perform the other [9].

Noise Addition

When a machine learning algorithm trains too well, it starts to learn the details and the noise of the training data rather than the general trend. If this happens, the algorithm starts failing. To reduce this phenomenon, one technique is to add random noise to both generated data and real train data. The noise could be generated by changing some values by a small amount. When the noise continuously changes, the network is not able to learn the noise and starts focussing on the trend.

3.3 Splitting Data Methods

To test and validate machine learning algorithms, the data must be split in three sets: train set, test set and validation set. The train set is used to train the algorithm while the algorithm is being developed. The test set is used to test the algorithm during development phase. The validation set is used to test the algorithm during validation phase. This last set is used to check that the algorithms did not adapt strictly to the test set.

When training machine learning algorithms, some information outside of the train set (for example from the test set) could be used to train the algorithm by mistake. This could lead to unrealistically good performances. This phenomena is known as data leakage. A good splitting of data limits or avoids that.

When the dataset is time-dependant, temporally close points are more likely to be

similar. When temporally close points are both in train and test set, some information from one set is indirectly passed to the other, which could cause data leakage.

With enough data, it is possible to just use the first part of the data for training, the second one for testing and the last one for validation. When data is not enough, this could cause some phenomenon to only be present in the train set, or in the validation set. This can cause the failure of the algorithm. Ideally, each phenomenon should be split between all different sets to have a bit of each phenomenon in each data set.

Some possible techniques for splitting the data are:

- Random sampling
- Cross validation
- Split in intervals

3.3.1 Random Sampling

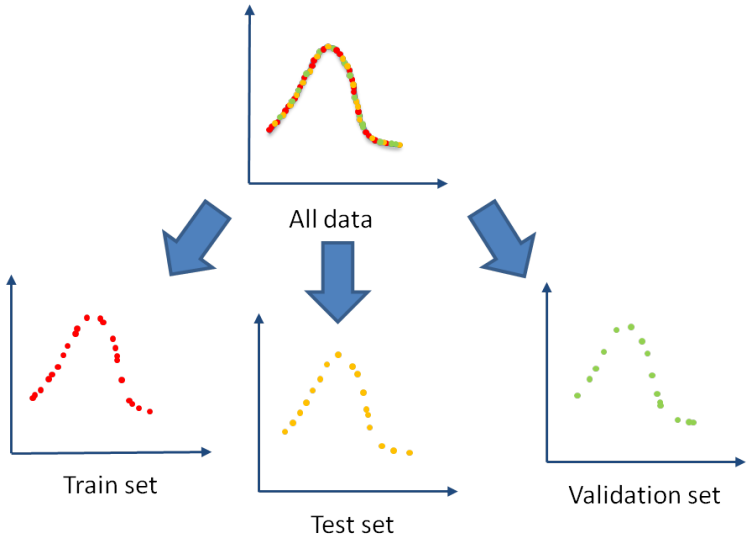


Figure 3.8: Random sampling

One way to split the data is to split each data point randomly between the three sets (like in Figure 3.8), with a different probability of being in each set (to have different

size of the sets) [11]. If points are split in that way, there is a high possibility that for each data point in the train set there is a temporally close data point in each other set. As explained before, this can cause data leakage in time-dependant dataset.

3.3.2 Cross Validation

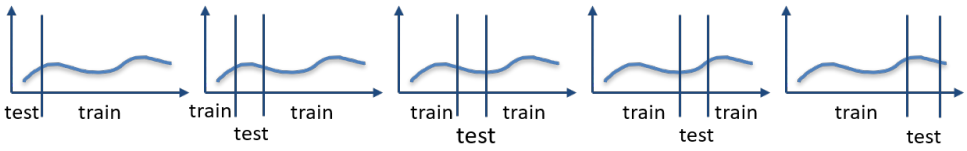


Figure 3.9: Cross validation

One of the most used techniques is cross validation (Figure 3.9). The data set is split in intervals. For each interval one test is done using that interval for testing, while the rest of the data for training. At the end all scores of different tests are combined together (for example making an average of singular results) [4]. This works well with only 2 sets: the train set and the test set. If the validation set must be included, one of the tests is just kept for validation and used only to validate the algorithm. The main advantage of this technique is that all the data is tested, and all data is used for training (in the development phase). The unlucky case where for a bad coincidence the test set is difficult to be predicted and the algorithm fails is mitigated by other tests. On the other hand, the validation test is done only once, and the advantages given by cross validation algorithm do not affect the validation set. The main disadvantage of this technique is that if the data is not enough, the validation set could be not predictable and the test could fail for every machine learning technique used.

3.3.3 Split in Intervals

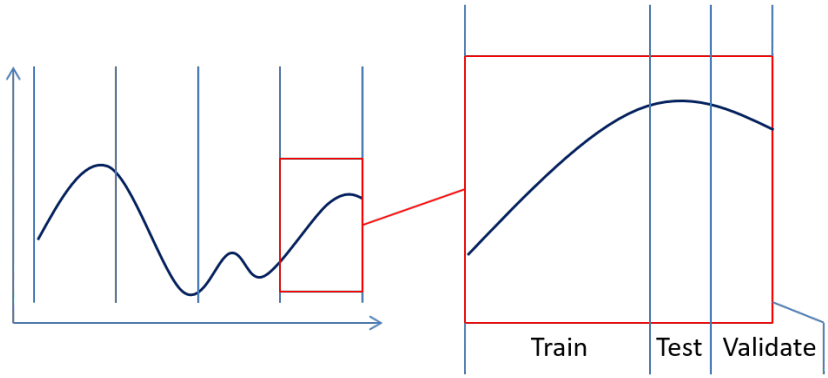


Figure 3.10: Split in intervals

In this technique, data is first split in intervals. For each of them, the first part is used as train set, the second is used as test set and the last one as validation set (Figure 3.10). The train set from every interval is then grouped together to create only one train set. The same happens to test and validation sets. This algorithm could cause data leakage if the intervals are too many and by consequence too small. This algorithm fails to correctly split short lasting phenomena if the intervals are too long. A good compromise must then be carefully chosen while setting the number of intervals.

3.4 Precision and Recall

When different techniques are compared together, metrics must be defined to grade each algorithm. The most used scores are precision and recall [7].

Precision indicates how much the algorithm is precise when detects an anomaly. Recall indicates how many anomalies are found compared with the total amount of anomalies. The two can be calculated with:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

Where:

- TP are the true positives which are the true anomalies that the algorithm detects correctly as anomalies.
- FP are the false positives which are the data points wrongly detected as anomalies by the algorithm, but that are not anomalies.
- FN are the false negatives which are the anomalies not found by the algorithm.

This definition is useful for global anomalies, when each point could be either an anomaly or not. When collective anomalies are considered, a group of data points is considered an anomaly only if considered all together. As a consequence, detecting only a part of an anomaly is a case that is not considered in the common definition of precision and recall.

When a collective anomaly is present, it is important to measure if the anomaly is found by the algorithm (the whole or a part), how much of the anomaly is detected, how many anomalous set of points are detected within the real anomaly and where they are placed inside the anomaly. To add this information, an extended definition of precision and recall must be used as explained in Section 4.6 [10].

Chapter 4

Phases of the Research

The research is structured in several phases. First data is prepared to be used by machine learning algorithms. In this phase data is split using the most suitable technique (Section 4.1), it is pre-processed (Section 4.2) and anomalies are artificially generated in test and validation sets (Section 4.3). Then different machine learning algorithms are employed (Section 4.4). Machine learning results are then post-processed (Section 4.5). Finally the algorithms are scored and compared (Section 4.6).

4.1 Data Splitting

For this research, a mixed splitting strategy between split in intervals and cross validation is proposed (Figure 4.1).

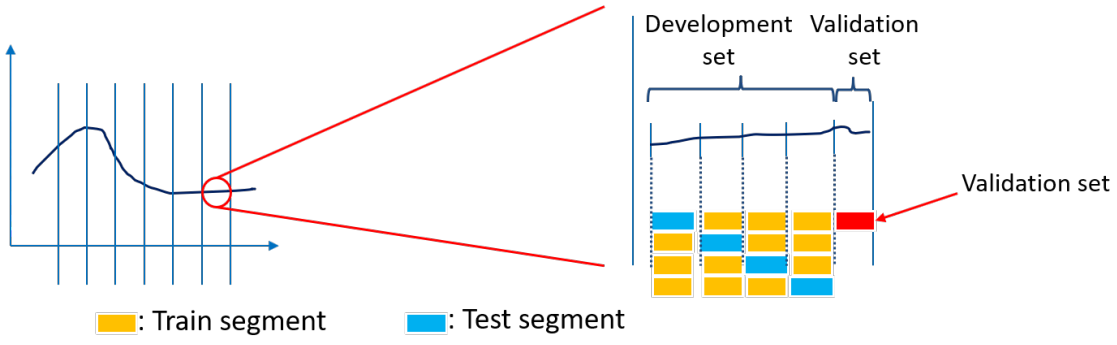


Figure 4.1: Mixed strategy: cross validation on each interval

Each interval is split into sub-intervals. In the beginning, the first sub-intervals of

each interval are combined and used for testing, while the rest are used for training. Then the second sub-intervals of each interval are combined and used for testing, and so on. This strategy has the same advantage of the cross validation but splits better the phenomena contained in the data. The number of intervals combined with the number of sub-intervals can give a size of each test sub-interval. By knowing this size, it is then possible to derive how temporally close the points in the test set from the ones in the train set are. As explained in Section 3.3, if data points in test set are temporally too close to data points in the train set, there is data leakage.

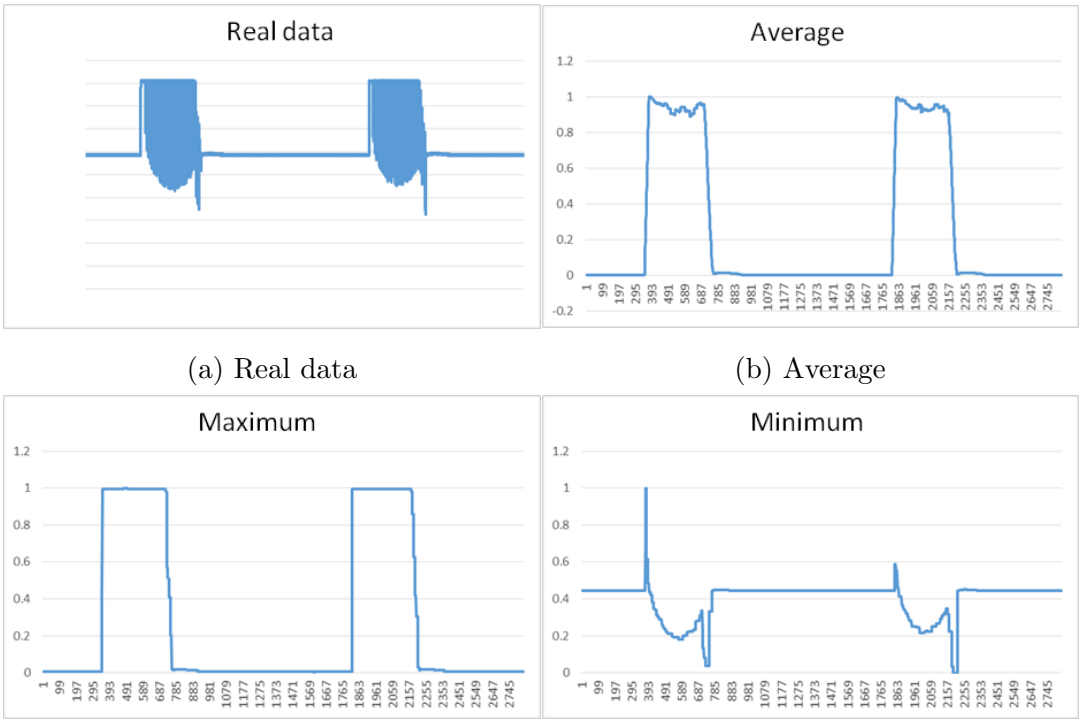
Since a validation set is needed, one test is used for a final validation while other tests are used during development phase. To avoid data leakage, the set of data used for validation is not used for training machine learning algorithms while other tests are done. Nevertheless, all other segments are used to train the algorithms while the validation is done. As a result, during validation the algorithms will be trained by one more segment compared with the tests. Having more data for training permits better results, so the validation test is expected to have better results than other tests.

4.2 Pre-Processing

4.2.1 Focusing on the Trend

Considering continue anomalies (collective anomalies), the singular data point loses its importance while data trend is more significant. After an examination of the considered data, it is evident that the data oscillates rather quickly from lower values to higher values (see Figure 4.2a), while the maximum and minimum values are quite regular.

Minimum and maximum give the range where the values oscillate but do not give any information about how the data is distributed between the two values. This information could be added by the average. The following graphs (Figure 4.2) show how maximum, minimum and average could represent the trend of the data.



(a) Real data (b) Average

(c) Maximum (d) Minimum

Figure 4.2: Data approximated using max, min and average

For every data point, a moving maximum, minimum and average with a defined window are calculated. The original data points are then removed and only the maximum, minimum and average are used to both train and test machine learning techniques.

Moving maximum, minimum and average consists in calculating minimum, maximum and average over a window of a certain size (in number of data points or time-wise). As showed in Figure 4.3, for every iteration the window moves and the maximum, minimum or average in the window is calculated.

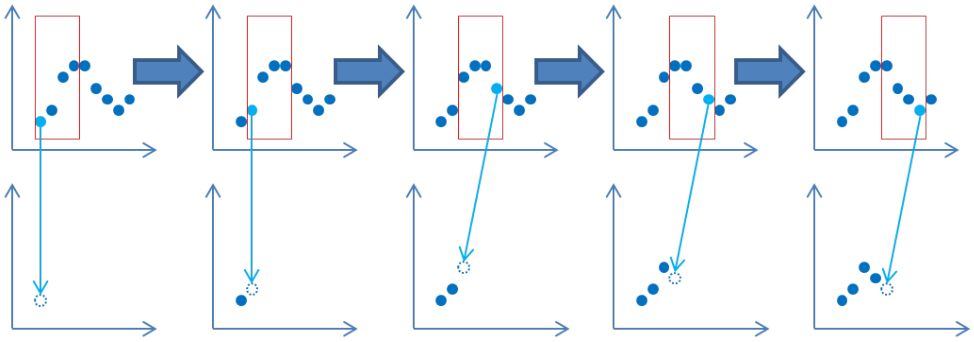
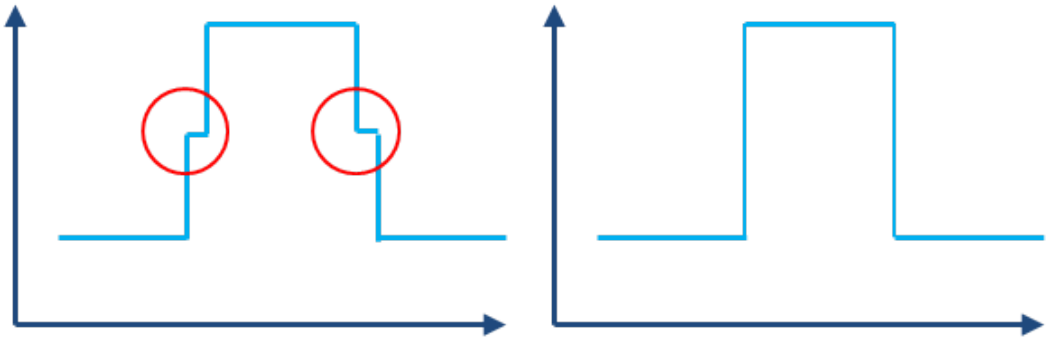


Figure 4.3: Moving minimum: for each iteration the red corner is moved and the lower value is taken

4.2.2 Further Approximation

Approximating with a moving window does not reduce the size of the data. Every data point is in fact just replaced with maximum, minimum and average, which triples the size of the data instead of decreasing it. Reducing the data used while executing machine learning techniques would be a better approach. If it could be possible to keep the same algorithm performances while lowering the size of the data this would lower the train and test time for the machine learning techniques.

After an investigation on the data, it is evident that the data obtained is quite redundant as the maximum and the minimum do not change that often. Moreover, when the data changes really quickly from a value to another like in Figure 4.4a, there could be some really rare data points between the two values. Those really rare values could be not enough to train machine learning algorithms during training while they could be seen as an anomaly (false positive) while testing the algorithms. When the values change a lot in a really small period of time it is really important to know which are the values before and after the change, but the transitory values add no important information to the data (like in Figure 4.4b).



(a) Before approximating (b) After approximating

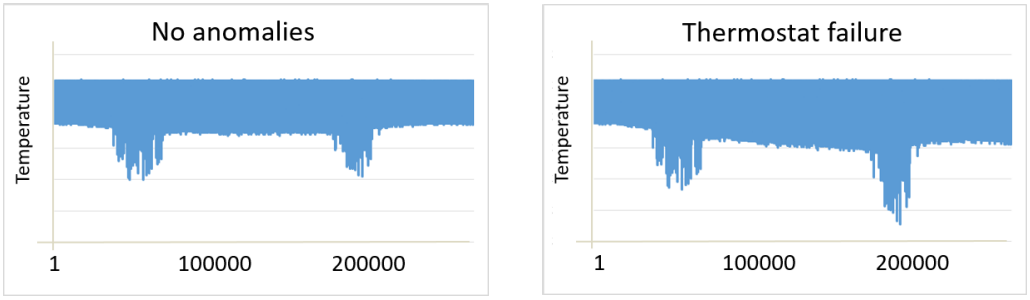
Figure 4.4: Approximation to avoid intermediate values

For these two reasons, an approximation technique is proposed. Before calculating maximum, minimum and average, the data is grouped into equally spaced blocks according to the data time. Each group is then substituted with its maximum, minimum and average. These new values are then used to calculate the moving maximum, minimum and average with the same moving window as proposed before. Maximum, minimum and average should not change from the previous approach, but the final number of points are arbitrary decreased depending on the size of the groups.

After this approximation is done, the data passes from a size of over 200000 data points and 30 dimensions to a size of about 3000 data points and 90 dimensions.

4.3 Anomaly Generator

When machine learning algorithms are tested for anomaly detection, it is necessary to have some anomalies to score the algorithms. The considered satellite is young and did not have any critical failure yet. However, satellites orbit around the Earth since decades and some possible cases of failure have already been studied. In this case the anomalies are simulated based on how the anomalies on other satellites usually are (as shown in Figure 4.5).



(a) Real temperature values (b) Simulated thermostat failure anomaly

Figure 4.5: Original temperature values compared with a simulated anomaly. In this case the anomaly is generated by decreasing lower values over time to simulate a progressive deterioration

4.4 Testing Machine Learning Algorithms

The following machine learning techniques are tested:

- Gaussian mixture model
- Local outlier factor
- Angle based outlier detection
- Generative adversarial neural network using label smoothing and noise addition

The hyper parameters are optimized to fit the problem as much as possible. This tuning has been done manually, but with the help of an optimization algorithm. Hyper parameters have been optimized to the following values:

Technique	Parameter name	Parameter value
GMM	number of clusters	30
LOF	number of neighbours	30
ABOD	number of neighbours	20

Table 4.1: Best hyper-parameters found for the problem

4.5 Post-Processing

If an anomaly on a satellite persists for a long time, a strange behaviour in the telemetry is likely to be present for the same amount of time. This strange behaviour could be not particularly evident for the whole period, but should be present for a considerable time lapse. This property could be used to make a filter on machine learning algorithms output and improve their performances.

The machine learning score is a set of values which gives for each data point on the test set a score of how likely the data point is an anomaly. This value must be transformed into a Boolean value representing whether the data point in an anomaly or not. Usually, a threshold is set. If the score for a data point is higher than the threshold, then it is considered an anomaly. The threshold is usually set in order to lower the false positives while increasing true positives. Intuitively, if the data point is more anomalous than a threshold, then it is likely to be an anomaly.

Another approach could be considered for continued anomalies. If there is a group of continued data points all with a score higher than a threshold, then this group is probably an anomaly. Here the anomalies are not considered anymore as global anomalies but as collective anomalies. In Figure 4.6 there is an example of a machine learning score filtered using this approach.

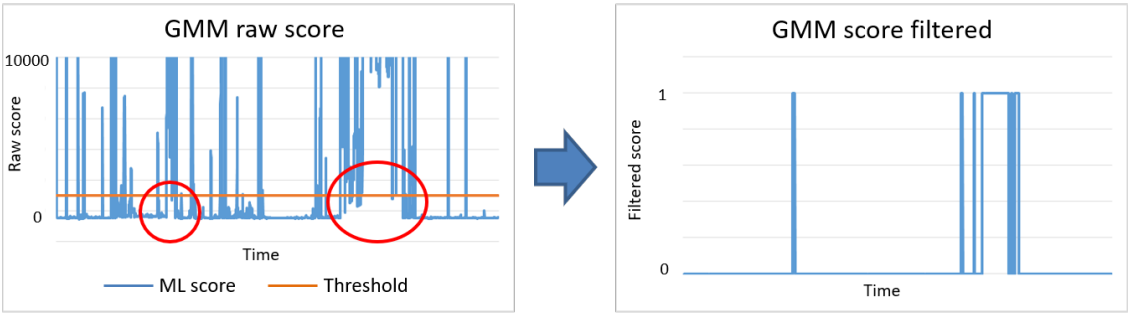


Figure 4.6: Machine learning score before and after being filtered

4.6 Validation

First of all, the algorithms must fit a series of basic requirements:

- Efficiency with lots of data dimensions

- RAM and time constraints to train and test the algorithms

To grade the algorithms and find the best one, many different metrics could be used. In this case precision and recall are considered. For this work the definition of recall will be extended as explained in [10]:

$$recall = existence + cardinality + size + position$$

Where existence, cardinality, size and position are:

- Existence: basic score given if the algorithm manages to detect at least a part of the anomaly

- Cardinality: decreases if more anomalies are detected during the same real anomaly

- Size: fraction of the anomaly that is found

- Position: how quickly the anomaly is detected. It will have the maximum value if the anomaly is found at the beginning of the real anomaly, the minimum value if the anomaly is found at the end.

Figure 4.7 visually shows the quantities defined above. In this work the original definition of precision will be used ($precision = \frac{TP}{TP+FP}$). This does not create an incoherence with the recall because the original definition of precision is a particular case of the extended definition. However, the original definition of precision can give a good quantification of how many mistakes machine learning algorithms actually do.

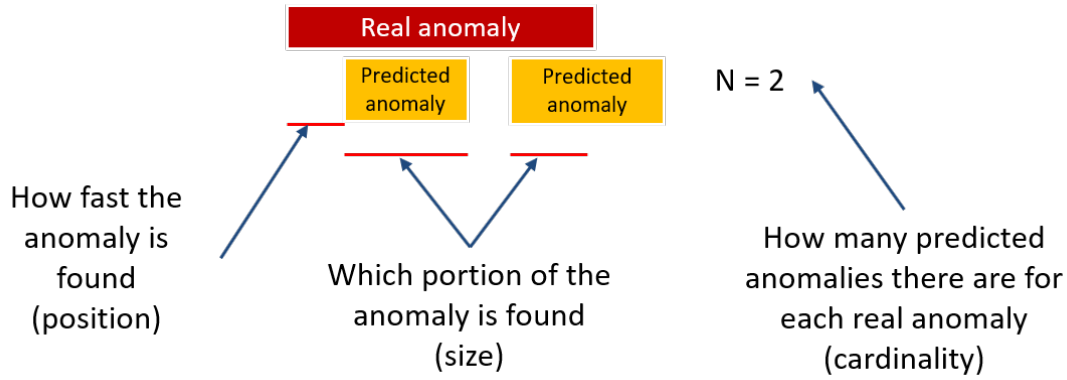


Figure 4.7: Extended definition of recall

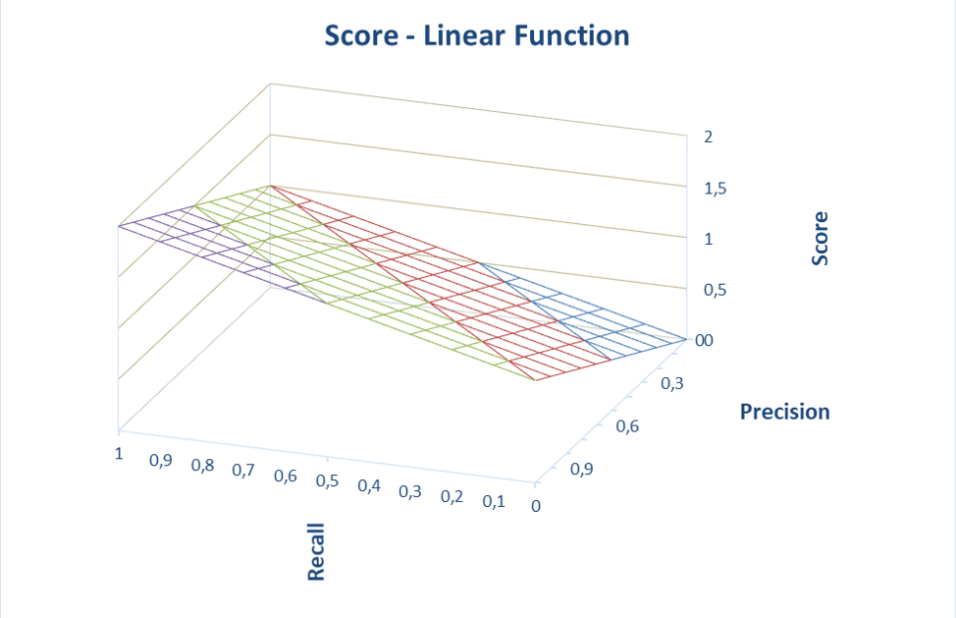
Precision and recall are usually summed together in a formula like:

$$score = precision * A + recall * B$$

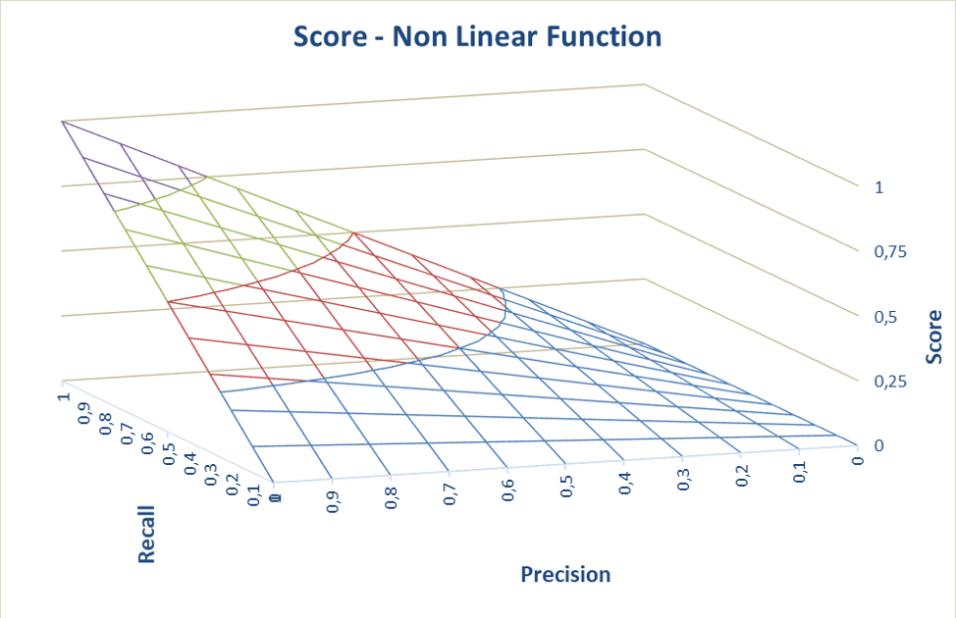
Where A and B are weights. Using this score formula a problem could arise. If the precision or the recall is 0, the score could still reach relative high values. However, a recall of 0 implies that no anomalies are found, while a precision of 0 implies that all guessed anomalies are wrong. Those two cases are the worst cases possible so they should have a score of 0.
To solve this problem, another way to calculate the score is proposed:

$$score = precision^A * recall^B$$

As well as for the previous definition of the score, A and B can be used to differently weight the precision and the recall. This formula will highly penalize the case where the precision or the recall are particularly low.
Figure 4.8a shows the score calculated as $score = precison + recall$ for every value of precision and recall. Figure 4.8b shows the proposed way to calculate the score ($score = precision * recall$).



(a) $score = precision * A + recall * B$



(b) $score = precision^A * recall^B$

Figure 4.8: Sum and multiplication comparison

Chapter 5

Results

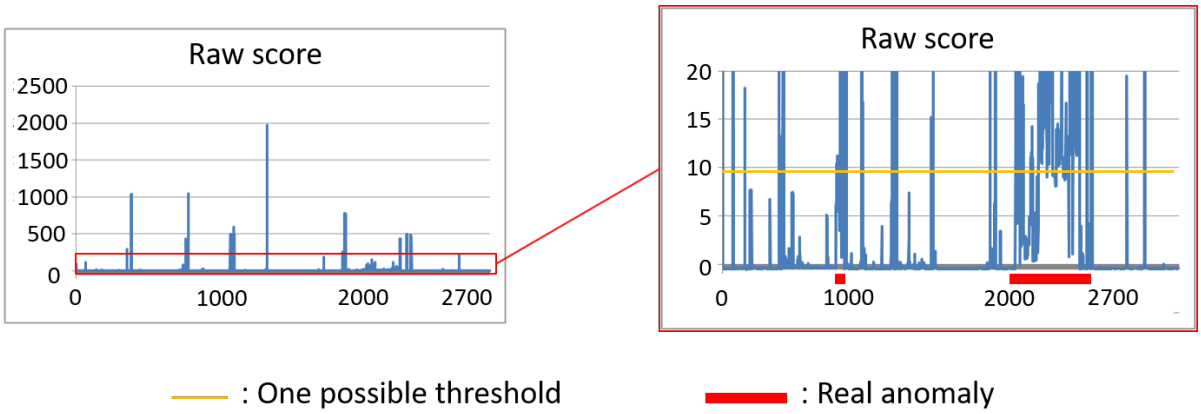
In this chapter the results obtained using different techniques are shown. First the post-processing usage is discussed (Section 5.1). Then pre-processing and post-processing impact on the machine learning algorithms are investigated (Section 5.2). Finally the different machine learning algorithms performances are compared (Section 5.3 and 5.4).

5.1 Post-processing

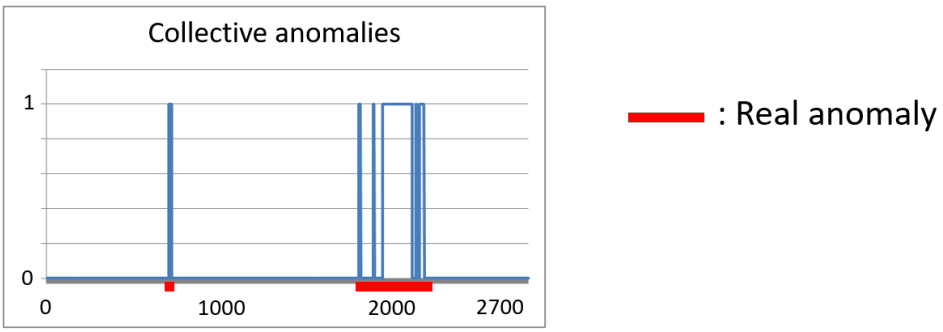
Both results obtained with raw data and approximated data (as showed in Section 4.2) are studied. For both cases tests have been conducted to assess whether machine learning techniques performances increase by applying the post-processing (as explained in Section 4.5).

When raw data is used, only some discontinuous points have an anomalous behaviour while an anomaly is present. As a consequence, it is not possible to increase machine learning algorithm performances by post-processing the results as described in Section 4.5.

However, when approximated data is used, the benefit obtained by considering only collective anomalies is evident. When a threshold is set, a compromise must be reached between finding more anomalies (lower threshold) and decreasing the mistakes (higher threshold).



(a) Raw machine learning score, before applying the threshold



(b) Filtered machine learning score, where 1 is assigned to detect the anomalies when the raw score is higher than a threshold for three days

Figure 5.1: Machine learning score before and after applying the post-processing

The phenomenon just mentioned can be observed in Figure 5.1. In Figure 5.1a a machine learning score is shown before applying the threshold. Here, it can be observed that if only the peaks are considered (high threshold), the detected anomalies do not match the real anomalies. By focussing on lower score values, it is possible to see that when there is an anomaly, the lowest computed score is higher than a certain value for a long period of time. Considering this value as a threshold (threshold in Figure 5.1a), the anomalies would be detected but there still would be many false positives. By post-processing the results, it is possible to get the score showed in Figure 5.1b, where a score of 1 signifies that the data point is an anomaly and 0 that it is not. The results reached with post-processing could not be reached only by adjusting the threshold.

5.2 Approximation with Minimum, Maximum and Average

In the previous section it is shown that when results are post-processed, it is better to also pre-process the data. However it must be proven that the combination of those two techniques can improve the results with respect to using none of them. Both cases have been tested with non-neural network machine learning techniques (GMM, LOF and ABOD).

In this thesis only the results of Gaussian mixture model will be shown, but other techniques reach similar results. In the following table precision and recall will be shown, using the traditional definition ($precision = \frac{TP}{TP+FP}$, $recall = \frac{TP}{TP+FN}$) before and after using the described techniques.

Technique	Precision	Recall
none	0,994	0,0022
pre-processing and post-processing	1	0,12

Table 5.1: GMM results before and after approximating using minimum, maximum and average and post-processed

By using the traditional definition, the recall is quite low in both cases, but this is caused by how data is labelled. Indeed, when a malfunction of the satellite is simulated, it could be detectable only during a part of it, even if the malfunction is fully labelled as an anomaly. However this causes the same drop of the recall for both techniques, so the two recalls can still be compared together.

As always, precision and recall depend on the threshold. In this case the thresholds have been set to get a high precision. The thresholds could be changed to increase the recall. However by using the first strategy (without using pre-processing and post-processing), to have a recall of $0,0022$, the precision would be about 0.5.

5.3 Machine Learning Algorithms Comparison

Using the described techniques (Section 3.2) the following results could be reached with the classical machine learning algorithms (non-neural networks).

Machine learning results						
	Development tests			Validation		
Machine learning technique	Precision	Recall	Score	Precision	Recall	Score
GMM	0.94	0.63	0.68	0.95	0.7	0.74
LOF	0.93	0.59	0.64	0.95	0.66	0.71
ABOD	0.79	0.7	0.72	0.65	0.73	0.71

Table 5.2: Machine learning results

In Table 5.2 for each algorithm the scores are shown for both cross validation tests done during development phase and final tests done during validation phase. Here the score values are calculated as $score = precision^{0.2} * recall^{0.8}$ to give more importance to detecting the anomalies rather than decreasing false positives.

All precision scores are high in these results, but it must be considered that for this test half of the data was anomalous (even if it is an unrealistic situation) so it can only be used to compare different algorithms and not to have an idea on how many false positives there would be with real data. A precision score of 0.65 in the validation test for the ABOD algorithm is then a relatively low value, even if the score is high. Also, both precision and recall should increase from development phase to the final test because of an increase of the amount of training data used (as explained in Section 3.3). The fact that for the ABOD algorithm the precision decreases means that the model created have over fitted the training set. This makes it failing with different data like the validation set. The result of the ABOD algorithm must be then considered as invalid.

5.4 GAN results

Tuning GAN hyper parameters is a longer and more difficult process compared with tuning other considered algorithms. As well as for other considered algorithms, a manual tuning with the support of an optimizer has been done. Even if the tuning process

and the application of label smoothing and noise addition increases the performances, no sufficient results have been reached. Possible reasons which could explain it include:

- The tuning process could be not adequate to the high number of hyper parameters to tune
- GAN algorithm could need more data to work efficiently
- GAN algorithm could be not ideal to find anomalies in this particular data set

Chapter 6

Conclusions

Considering the results shown in the previous chapter, it is possible to conclude that for this particular problem and using this data, Gaussian mixture model is the best considered algorithm. The approximation of the data using maximum, minimum and average shows the importance of adapting the algorithms to the right type of data and anomalies.

The considered problem is a non ideal one due to the lack of data and anomalies. Anyway, using the right strategies it is possible to achieve satisfactory results. None the less, results have lower reliability compared to an ideal problem where data is enough and real anomalies can be used to test the algorithm.

Bibliography

[1] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[2] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[3] A. K. Jain, J. Mao, and K. M. Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[4] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[5] H.-P. Kriegel, M. Schubert, and A. Zimek. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 444–452, 2008.

[6] S. Marsland. *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC, 2014.

[7] I. D. Melamed, R. Green, and J. Turian. Precision and recall of machine translation. In *Companion Volume of the Proceedings of HLT-NAACL 2003-Short Papers*, pages 61–63, 2003.

[8] D. A. Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741, 2009.

[9] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[10] N. Tatbul, T. J. Lee, S. Zdonik, M. Alam, and J. Gottschlich. Precision and recall for time series. In *Advances in Neural Information Processing Systems*, pages 1920–1930, 2018.

[11] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.