AI Project Summary

Emma Grandgirard - David Jeauneau - Christelle Latorre - Coraline Le Brun
Université Paul Sabatier - M2 SID - March 2020

1 Introduction

The aim of this project was to detect anomalies in accelerometer measurements sequences performed on Airbus' helicopters at different locations, angles and flights. Each sequence consisted of 61440 values, reflecting 1 minute of recording.

For the first phase of the project, we were provided with two datasets: a training set that only contained data without anomalies, and a validation set that contained data with and without anomalies. The goal was to learn the characteristics of normal sequences in order to be able to detect unusual patterns in new datasets, making our problem a semi-supervised one. During the first phase, we could make predictions on the validation set and get our score on the competition website immediately. During the second phase, we were provided with a new dataset, called test set, to make two final predictions without getting our scores.

To solve this problem, we set up two distinct methods. In a machine learning approach (described in Part 2.1), we created features to reduce the dimensionality of the data. Some of them consisted of descriptive statistics like mean, median, or variance of all the signals or rolling windows. Others were aimed at measuring the complexity of data sequences, meaning its entropy. We also used an Autoregressive Integrated Moving Average (ARIMA) model to measure the degree of predictability of each sequence. After having computed these features, we applied several models to assess the degree of normality of each signal. To do so, the models needed a threshold value, that is a separator between anomalies and normal sequences. We therefore implemented a method to find automatically the best threshold, using cross-validation.

Our second approach (Part 2.2) consisted in creating a deep learning model called autoencoder, aimed at compressing normal data in a new smaller space. This compression method sought high and low levels characteristics in normal data to reduce their encoding. Then, the model found the characteristics of normal data that allowed them to be compressed, and then decompressed easily to reconstruct the original signal. An anomalous signal would be characterised by an inaccurate reconstruction, resulting in a high error rate between the reconstruction and the original signal. Here again, we implemented a method to find the optimal threshold.

When making predictions, we found out that combining predictions from several models gave us better results. These so-called "ensembling" methods (Part 2.3) allowed us to reduce the variance of our predictions, and thus get better generalisation.

We used Python to implement our solutions, mainly with pandas, numpy, scipy, scikit-learn, and keras libraries.

2 Methods

2.1 Machine learning approach

Feature engineering

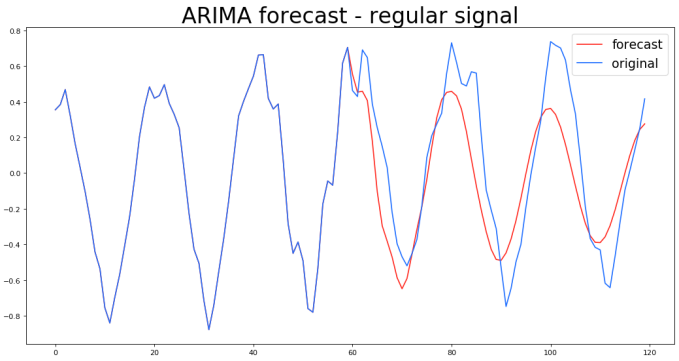
Throughout the project we used several feature engineering techniques.

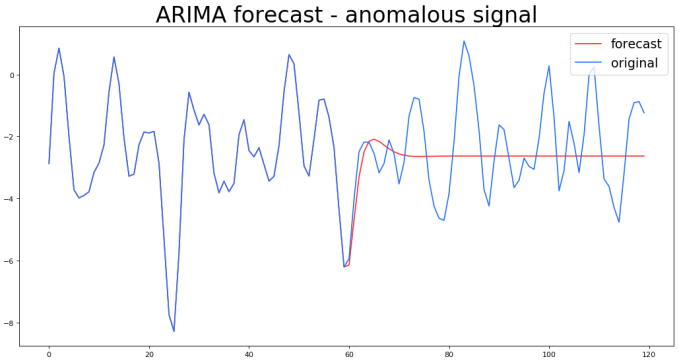
First we computed standardised features, which enabled us to have result values on the same scale that we could directly use as metrics. Indeed, some of the anomaly detection models we were going to use needed standardised data to work. Moreover, we were informed that the train and validation data had been multiplied by a random number, thus making irrelevant the use of their absolute values. Engineering standardised features was therefore a solution to this problem, allowing us to be free of the scaling problem.

An example of this is the use of rolling windows: we divided each signal into chunks, computed statistics on each one and compared the variations between them. We used these relative variations as features in our models.

We also used basic descriptive statistics like the mean, the standard deviation, or the quantiles for each signal.

Furthermore, we applied an ARIMA (Autoregressive Integrated Moving Average) model to our data. ARIMA is a statistical method used for time series forecasting. We performed this on sequences of time series and then compared the forecasts with the actual sequences: a higher difference meant that the series was more likely to contain anomalies.





On the first graph, the forecast is rather close to the original signal, whilst on the second graph the model fails to make an accurate forecast, resulting in a high error rate and therefore a high probability of classifying the signal as an anomaly.

These features allowed us to reduce the dimensionality of our datasets, from 61440 values to twenty for each signal. We also implemented a parallelizer for computational efficiency and a "feature bank" to store all our features.

Furthermore, visualising our features was of the utmost importance, hence we created several tools to visualise them either individually or in interaction, such as the distribution of pairs of features between the train, validation and test datasets.

Models

To find anomalies in the validation and test datasets, we performed several models on the previously computed features.

- Distance to barycentre

For each signal, we created graphs in which each signal was represented by its mean in abscissa and its standard deviation in ordinate.
We ranked the signals of the validation set and then test set based on their distance to the barycentre of the training set. We presumed that abnormal signals would have a greater distance to the barycentre.
Then, we created a feature using the absolute difference between the number of values above and under the mean. We considered that the signals with the greatest distances were the ones which were more likely to be anomalies.
Finally, we created a rule combining these two features and the anomaly rate in the validation and test datasets to create a third feature indicating whether a signal was an anomaly or not.

- Density-based methods

- **Gaussian Mixture Modelling**

Gaussian Mixture Modelling is a global approach of density-based methods. It consists in fitting k Gaussians to the dataset, then finding the parameters of the Gaussian distribution (mean and standard deviation) for each cluster. Given a new point, computing its distance to each Gaussian allowed us to deduce its probability of belonging to the corresponding clusters. A point with low probabilities was likely to be an anomaly.

- **Local Outlier Factor**

The Local Outlier Factor (LOF) is a local approach of density-based methods. It consists of a score that compares the density of a given point with the density of its neighbours. If the LOF was much greater than 1, that meant that the density of the point was much smaller than the densities of its neighbours. Subsequently, such a point would be classified as an anomaly.

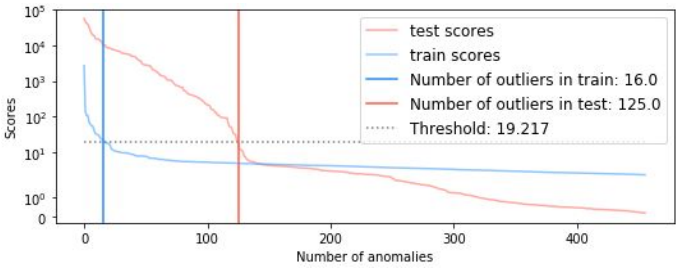
- Isolation Forest

Another model that we used is the Isolation Forest. The goal of this algorithm is to isolate a point from the rest of the dataset by creating decision trees over random attributes. If the point was easy to isolate, then it was probably an anomaly.

Automatic threshold

For each model, finding the threshold value which set the limit between normal and anomalous signals was crucial. In order to do this automatically, we used a cross-validation technique on both test sets and train set.

By only learning on a partial training set, we could evaluate scores on the remaining data of the train set and on the test set. That way, we could see which threshold was too aggressive by looking at the anomaly rate in the train set, or too lenient by looking at the anomaly rate on the validation or test set. Combining these two pieces of information, optimums could be found. We used matplotlib library to observe the slope of the anomaly rate in the validation or test set as a function of the threshold, and then visualise the anomaly rate in each dataset for the chosen threshold.
On the graph (see opposite), a sharp drop in the test scores curve can be observed: the model therefore set the threshold to the value that corresponds to this drop. The number of anomalies in each dataset can be read on the abscissa axis.



2.2 Deep learning approach

The second method used consisted of an autoencoder: an algorithm that compresses data by capturing automatically its essential features. In order to do so, a deep neural network is required. This neural network design consists in creating a "bottleneck" that forces the neural network to find a way to process it without losing information, before trying to restore the data as close as possible to the original data.

By training an autoencoder on normal signals, we hoped that it would find the best way to catch their features. By doing so, when confronted with an anomaly, our model would not be able to compress it efficiently. By comparing the error rates between original signals and the compressed and decompressed ones, we set up a method to detect anomalies.

2.3 Models combination

In order to improve our results, we performed what is called "ensembling" in machine learning: using predictions obtained with different models to capture the best characteristics of each model. For each signal, we looked at how many of our models predicted it as an anomaly. Since our models were designed to capture quite distinct types of anomalies, we chose to classify as an anomaly any signal classified as one by half of our models.

3 Results

3.1 Metrics description

The metric used to evaluate our predictions was the F1 score, which is a combination of two statistical metrics: recall and precision. Recall is the number of anomalies correctly identified (true positives) divided by the total number of anomalies. Precision is the number anomalies correctly identified (true positives) divided by the number of predicted anomalies. The F1 score is given by the formula:

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

3.2 First phase scores

During the first phase, we had to predict anomalies on the validation dataset. We already knew that the dataset included 50% of signals with anomalies.

The best scores obtained during the first phase were:

Recall: 0.9798 - Precision: 0.97
Resulting in F1 score: 0.97487

The recall and the precision were pretty close even though the recall was a little higher.

These scores were obtained by computing the ensembling model described in part 2.3.

3.3 Final phase scores

During the second and final phase we had to predict the anomalies on a new dataset, the so-called test dataset. This time we didn't know the proportion of anomalies in the dataset.

Our final scores are:

Recall: 0.46190 - Precision: 0.29985
F1 score: 0.3664

These results are lower than the ones we got during the first phase. However we can be pleased with the fact that the recall is still better than the precision. Indeed, considering we are dealing with potential helicopter crashes, it is better to predict a risk of crash even if it is not going to occur, than to fail to predict an anomaly which is going to make a crash occur.

4 Discussion

The scores obtained in the final phase were lower than the ones obtained in the first phase. The anomalies in the first test set were very different from the anomalies in the second test dataset and our models performed badly to recognise these new anomalies. We think that is the reason why we had a lower score in phase 2 than in phase 1. Nonetheless, we managed to rank in the top 30% of the teams in both phases.

We believe that we could have got better results if we had been provided with more insight about helicopter flights and sensors functioning. This is a key element we, and all teams, lacked in this project: proper knowledge of helicopter flight issues. This could be improved for further editions of the challenge by either giving room for more communication between students and experts of the sector of study, or by providing more information at the beginning of the project: explicit datasets, or documentation.

Even though we were disappointed by our results, we learnt a lot from this challenge. First, this was an opportunity to conduct a scientific research. We learnt to work as a team in a research project.
Moreover, we discovered new machine learning and deep learning algorithms to detect anomalies, like Local Outlier Factor and Isolation Forest. We also learnt to choose and compute our own features adapted to the problem. We developed skills in the fields of anomaly detection and time series processing, and we made our code reusable in case we need it for further machine learning projects.