Identifying Fuel Leaks Using Machine Learning

Capstone Report

2022

AIRBUS

ie SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

Alexandra Mathay | Alice Seynaeve | Diego Garate | Pierre Belthon | Timothé Rigaudeau

Table of Contents

Executive Summary

Code:
https://github.com/Timorig/Airbus_fuel_leak_detection

Maintenance is key for the aircraft industry. The high standards of safety and high costs of unavailability push companies to run their maintenance process with extended use of technology and data solutions to become safer and more efficient. Predictive maintenance has become an essential tool for anticipating possible events that could impact business continuity. In this context, combining more sophisticated machine learning methods and big data technologies applied to predictive maintenance processes has become a successful formula to sustain and improve the business.

The Problem

The main problem we want to solve is to be able to propose another automated solution for fuel leak detection for aircrafts. Currently, the fuel leak detection process for aircrafts is not accurate and manual. Consequently, airlines could face increased aircraft unavailability or aircraft on ground (AOG), affecting the whole fleet planning process and the company's operations.

Proposed Solutions and Business Value

To address the current problem, we came up with two main outputs, (1) a baseline machine learning model, and (2) a dashboard mockup that would indicate possible leaks using these models.

We aim to provide a machine learning model to detect small leaks in short flight sequences and thus, reduce the probability of an aircraft on ground. This model, if further developed, could be the backend of a real-time fuel detection dashboard for predictive maintenance for Skywise. With an improved model in place and the Skywise platform could enable airline companies to save up to USD120,000 per hour by avoiding AOG.

Data Exploration, Cleaning, Feature Engineering, and Preprocessing

We were provided Airbus data from eight military aircrafts which have a similar fuel system to the A380 commercial aircraft. The data contained 24 million rows of time series data, each row corresponding to one second of data.

For data cleaning, we checked the distribution of data, handled null values, identified and replaced outliers and we performed data manipulation and transformation to isolate flight sequences with a stable aircraft attitude.

For feature engineering, we applied transformations and created new features that allowed us to filter and prepare the data we would use for our anomaly analysis. Main feature creation focused on aggregating information and transforming features to smooth time series trends. As a result of data cleaning and feature engineering, we obtained a clean data set of around 250 thousand records, or close to 1% of the original number of records, and we created 7 new features for a total of 24 features per aircraft in the final dataset.

Our Approaches

We performed and tested three different approaches, (1) a statistics-based technique, (2) an unsupervised approach (PCA), and (3) a semi-supervised approach (Autoencoders). To properly structure these approaches, we tested the models in increasing complexity, starting with a simple statistical approach, then concluding with a more complex deep learning approach, the autoencoders, to be able to achieve improved results.

Approach 1: Statistical and Unsupervised Techniques

The starting point for our anomaly detection was an unlabeled dataset gathering the data we received from Airbus. Only one aircraft received confirmation of a minor leak. In this context, there was no "ground truth" or leak confirmation as our data was unlabeled. Since we could not identify with certainty cases of leaks, we added simulated leaks of three different severities to witness a change in behavior between the aircrafts.

We began identifying statistical anomalies based on the behavior of the error between FOB (given by the datasets) and FOB expected or calculated from consumption. We focused on the identification of outliers and we found a possible leak in MSN37, flight 98. We then compared the original data distribution with the simulated leaks distribution and identified a trend proportional to the severity of the leaks.
For the second part, we applied standard scaling and ran PCA on the final dataset. To identify possible cases of anomalies, we ran a cluster analysis using K-Means and an outlier analysis through the Local Outlier Factor (LOF) method. After watching the composition of each cluster, we concluded these methods did not manage to capture clusters of anomalous data.

Approach 2: Semi-supervised Technique

Neural networks are particularly efficient for complex problem-solving such as anomaly detection. Thus, autoencoders can be useful for our problem through a semi-supervised learning approach. We performed an iterative approach to identify the best model. Our input dataset contained clean flight sequences (labeled as normality), and the simulated leaks (labeled as anomalies) were used to test the model.
We first built a baseline model which was trained on unsampled data, using a minimal network architecture of only 2 layers and 3405 trainable parameters. It gave us an F1-Score of 64% and classified 95% of leaks as normal data. Our final model was trained using a 5 seconds resampled version of our data. We performed a feature selection, modified the optimizer to 'Adam', tuned the learning rate and set the cost function to 'mse'. Additionally, we significantly deepened the structure of the network to have 7 layers, totaling 35,910 trainable parameters. Our last model had an F1-Score of 87% and only miss-classified 19% of the leaks.

Improvements & Next Steps

We listed recommendations for future improvements and to further develop the project. For cleaning and data preparation, the accuracy/quality of the simulated leaks could be improved, as well as to look deeper into the other flight phases. For modelling, exploration of the Gaussian Mixture Model (GMM) approach and Long-Short-Term Memory (LSTM) autoencoders could be explored, as our hypothesis is that these models could enhance performance. Lastly, the dashboard could be tested on a BI tool and the deployment of the solution could be implemented in a real-time environment using the ACARS system, or more recent solutions like FOMAX.

1. Context

1.a. Big Data in Aviation

The Aviation Industry is an industry filled with data, from retail flight data to airplane sensor data. One airplane for example can have more than 200 thousand sensors, generating 10Gb of data every flight hour. Not to mention the data of suppliers of parts, producers of the aircraft, and other stakeholders that go into the everyday decisions. However, a big problem that each of these stakeholders has been that they only see part of the information and data that is out there. Therefore, aviation companies are producing solutions that integrate and synergize data from the different stakeholders, like Skywise by Airbus and AnalyTX by Boeing.

1.a.i. Skywise

As of 2017, Airbus launched an aviation data analytics platform, named Skywise. The goal of this platform is to reach the full potential of its data. This service provides their clients, more specifically the airlines, with the data all centralized in one location in the shape of pre-built workflows such as dashboards which can contribute to their operational efficiency, prevent delays, and reduce their chance of getting AOG's (Aircrafts on the Ground) which implies increasing revenues. The platform could help airlines to shape and support their business models, by taking better decisions using the data and understanding certain behavior during flights. In this way, clients could gain insights via the platform and thus, create shared value for the industry. Each flight produces significant amounts of data captured by their sensors. On top of this, post-flight reports are made. Airbus 'clients can learn from this data and as well leverage the cumulative knowledge of tons of Airbus engineers which is collected on the platform.

Skywise consists of three important pillars: Predictive Maintenance, Health Monitoring and Reliability Premium, all having in common the same final objective which is reducing operational interruptions to the minimum.

Another big advantage of the service is that it can be used across the entire supply chain. Aircraft equipment suppliers could improve their products by monitoring the performance across the product's lifetime. Original Equipment Manufacturers (OEM) as well could deliver more qualitative designs using the historical data. Operators and owners of commercial airlines, helicopters, military aircrafts and equipment could improve their operational efficiency.

The tool has been founded in collaboration with Palantir technologies – a software company specialized in big data analytics. Their CEO assured that the power of the combination of Airbus, its airline customers and their own software could lead to a revolutionizing platform.

1.a.ii. Boeing Analytx

Similarly, to Airbus, Boeing implemented AnalytX offering 3 categories of analytics: Digital Solutions, Analytics Consulting Services and Self-Service Analytics. The first one includes software applications to meet the needs of crew and fleet scheduling, flight/mission planning and operations, maintenance planning and management, and inventory and logistics management. The second one incorporates business, aviation and analytics related professionals who are eager to serve customers to help them improve their efficiency, economy and operational performance. The third one involves dashboards and other digital solutions based on the company's data which allows customers to explore and discover insights and opportunities.

1.b. Types of Maintenance

There are three general types of maintenance, (1) **scheduled,** (2) **unscheduled**, and (3) **predictive** maintenance.

1.b.i. Scheduled Maintenance

Scheduled Maintenance is performed at defined intervals at a regular state, to retain the plane at a serviceable condition. The main issue with this approach is that there is an added risk of the equipment breaking down due to accidental damage, as well as the cost of constantly replacing or checking perfectly functioning equipment. There are three main types of scheduled maintenance, (1) **light or line maintenance**, which refer to preflight checks, daily checks, or weekly checks, (2) **base or heavy maintenance**, which refer to heavier equipment checks, and (3) **shop or component maintenance**, which refer to maintenance on components when removed from aircraft.

1.b.ii. Unscheduled Maintenance

Unscheduled maintenance is maintenance performed to an equipment or item to a satisfactory condition from a known or suspected malfunction or defect. There are two different kinds of unscheduled maintenance such as, (1) **planned** or (2) **unplanned**. Planned maintenance refers to maintenance for modification of originated work, and rectification of deferred defects, while unplanned are activities originating form the cabin logbook, technical logbook, and ground findings.

The main use of these Big Data Technologies and models is to be enable stakeholders to identify a problem before it happens or when it is still minor, this is called predictive maintenance. This could possibly improve both scheduled and unscheduled maintenances to be able to identify something that will happen or identify something that has already happened but at an earlier time.

1.b.iii. Predictive Maintenance

Predictive maintenance is the use of software and algorithms to analyze and find patterns to detect an issue before it occurs. This type of maintenance can be costly to set up due to the added implementation of software and tools, however, it could save the company huge maintenance and downtime costs in the long run.

1.c. Problem: Fuel Leaks

In the aerospace industry Fuel Leaks can be one of the costliest types of maintenance, as this can take days to detect and fix which means longer days for Aircraft on Ground (AOG). To add, the current aircraft leak auto detection system thresholds are too high for practically detecting typical leak rates occurring across the fleet, requiring regular visual inspections of aircraft that sometimes fail to detect the occurring leak.

A Fuel Leak in an aircraft is when there is a hole or a crack in any of the tanks or in the feed system of the aircraft fuel system where fuel can leak out causing the plane to lose more fuel than expected. The causes of these leaks are typically structural defects and degradation of the seals and sealants, or extreme hot weather conditions and continuous bad landing.

1.c.i. Detecting a Fuel Leak

Detecting a fuel leak is not an easy task as it can be very manual and based on visual inspection, tracer gases, or pressure measurement during scheduled checks once the plane has landed. There are two main ways that you can currently detect fuel leaks, by visual inspection and test during scheduled or reactive maintenance, and by fuel checks by the pilot computing it using sensor data.

Sensors on the aircraft can only detect fuel leak when the problem has already been exacerbated through time when there is a discrepancy limit of 2,500kg in 20secs. Detecting the source could also be a very tedious process as these leaks usually start very small and can often be more than one source.

Procedural fuel checks are done usually before during and after the flight to detect discrepancies in Fuel on Board, Fuel Used/Fuel Uplifted, and Initial Fuel on Board plus minus an acceptable tolerance.

Context/ Problem Summary:

- The Aviation industry has huge amounts of data that could be utilized through a platform that synergizes all available data that are currently in siloes. This is why Skywise was created.
- One of the major use-cases for Big Data in this industry is in predictive maintenance particularly for fuel leaks.
- The current leak autodetection system thresholds are too high (2,500kg) for practically detecting typical leak rates occurring across the fleet, requiring regular visual inspections of aircraft that sometimes fail to detect the occurring leak. This cases have resulted in Aircraft On Ground (AOG) situations.

2. Business Value

2.a. Our Project: Detecting Fuel Leaks

We want propose improvements to the current way of detecting Fuel Leaks using machine learning models as a baseline for identifying small leaks. The main output for us is to be able to recommend a potential model or models that could serve as a baseline for Skywise, and create a mockup dashboard for Fuel Leak predictive maintenance for users of the new Skywise data analytics platform.

2.b. Framework: Impact x Probability Matrix

The framework we will be using to quantify the value of our project is the Impact-Probability matrix where we would look at the probability of the event happening, in this case, a fuel leak, and the impact it has. With this, we could see where in the matrix airbus is without predictive maintenance using these models, and their positioning after. To do this we have to look at the impact and the probability of a fuel leak.

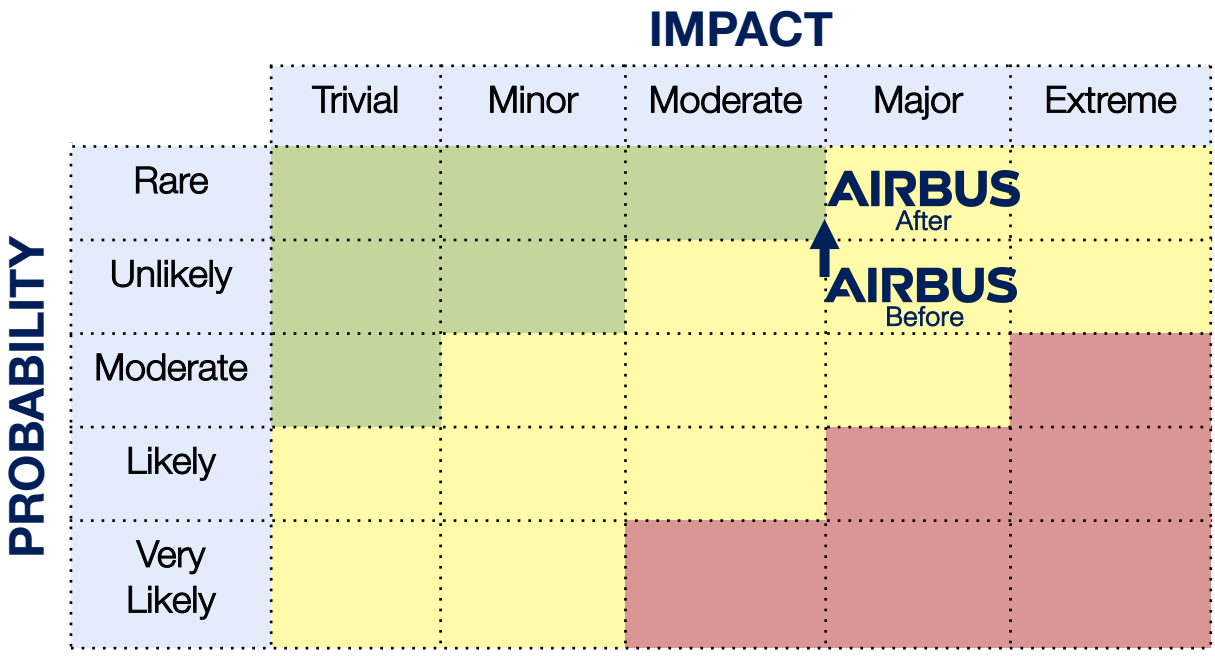


Figure 1: Impact Probability Matrix Before and After the ML Models

Figure 1 above shows that current positioning of Airbus on the probability impact matrix and the positioning of Airbus after utilizing machine learning models. The matrix shows that a fuel leak is unlikely, but has a major impact. Then through predictive maintenance, this could lessen the probability to 'Rare'.

In the next sections, we will be discussing both the impact and the probability of a fuel leak for aircrafts. We will be analyzing the impacts for commercial aircrafts similar to the model of the plane in the data provided.

2.c. Impact of Fuel Leaks

The three main types of impact are (1) **business costs**, (2) **reputational costs**, and (3) **human costs**. It is important to understand and look at all three types, for this case we have quantified business costs. The main type of business cost in case of a fuel leak is AOG costs.

2.c.i. Business costs: Aircraft on Ground (AOG)

If there is one situation an airline would like to avoid, it's to encounter an Aircraft On Ground (AOG). It occurs when the maintenance of the aircraft should be carried out because the airplane has been grounded due to a serious problem. When these circumstances take place, airlines try to fix the problem as quickly as possible. AOG can lead to delays causing inconvenience to passengers and a loss of potential revenue, which has a severe impact on the business.

In order to calculate the actual cost of an AOG we need to take several factors into consideration:

- Location of the aircraft on the ground
- Location and availability of the spare parts of the plane
- Price of the spare parts
- Logistic network to deliver the necessary parts
- Availability of engineers to work on the aircraft
- Skill level of the available maintenance technicians
- Time necessary to implement all the above

A maintenance check can take up to 120-150 hours to find the problem, locate and order the necessary parts and install them once they arrive. These numbers do not even consider the loss for the business while the aircraft isn't flying.

Cost of Aircraft on Ground from Fuel Leaks

In cases of fuel leaks, a typical fuel leak can ground an aircraft for days and sometimes weeks, making fuel leaks the most time-consuming and one of the costliest types of maintenance. To compute the impact of a fuel, leak **figure 2** below summarizes the estimated costs of an AOG from a fuel leak.

	2days	3days	4days	5days
Airport and Delay Costs (100,000 per hour)	$4,800,000	$7,200,000	$9,600,000	$12,000,000
Compensation in case of delay	$902,400	$902,400	$902,400	$902,400
Maintenance	$350,000	$350,000	$350,000	$350,000
Total Aircraft On Ground Cost	***$6,052,400***	***$8,452,400***	***$10,852,400***	***$13,252,400***

Figure 2: Cost of Aircraft on Ground for 2-5 Days

2.b.ii. Reputational Cost

Another cost to consider is reputational cost of both the airline, if it is a commercial flight, and reputational costs for Airbus. Specifically, these are opportunity costs when potential customers end up deciding not to ride with the airline or with airbus planes, due to the news of a fuel leak. However, this could be difficult to quantify and therefore we will consider it as a qualitative for this project.

2.b.iii. Human costs

Human cost is another cost to consider if any injuries or deaths are caused by failure of the plane due to a fuel leakage. This is highly improbable, due to the checks that happen before and during the plane. If by any chance an accident happens due to a fuel leak the estimated compensation per passenger is around USD100,000.

2.d. Probability of Fuel Leaks

Although fuel Leaks are very problematic, costly, it does not happen quite often. According to an Airbus Operations Report, “In more than 25 years of operation, there have been not more than a handful of events involving undetected fuel quantity issues.” because companies like Airbus ensure early detection and management. However with a new model, the probability of it happening can be further reduced and thus lessening the chance of having an AOG.

2.e. Estimated Savings

The business value of this project is to propose an alternative automated solution using ML models to improve the positioning of Airbus within the probability matrix. Impact of a fuel leak will improve slightly because the time for them to detect where the leak is would lessen and thus lessening the time it takes to do maintenance, however this will only lessen the business impact slightly. Probability of the fuel leak will improve using machine learning models that will enable maintenance team to identify fuel leaks earlier and thus be able to quickly repair the leak before it is aggravated by time. This would enable Airlines to save at least $126,092 per hour, based on AOG impact.

Business Value Summary:

- Through our project we want to do deliver two main outputs, (1) propose Airbus with another automated model that could potentially detect smaller leaks than the current threshold of 2,500kg in 20secs, and (2) with this model create a fuel leak detection dashboard mockup for Skywise.
- The using the Impact-Probability framework we’ve identified the impact and probability of fuel leaks
- By decreasing the probability of unscheduled maintenance and AOG, the total estimated savings could be $126,092 per hour.

3. Overall Approach

3.a. The Data

The data consists of 8 different datasets, see Figure 3, each in a CSV format corresponding to an aircraft. One of them included a flight-test aircraft (MSN02), the others were in-service aircraft. Also, one dataset (MSN37) was labeled as having a leak.

	Records	Features
msn02	623580	111
msn10	621610	19
msn11	4455992	19
msn12	3247664	19
msn14	4640993	19
msn29	4129447	19
msn37	3236645	19
msn53	3034227	19

Figure 3: Main Datasets – CSV Files Per Aircraft

The historical data itself contained time-series data coming from the sensors on the planes for every flight. There are 111 different features or columns some of the datasets, while MSN02 only contains 19 variables. There are 23,990,158 rows of time series data. The data consisted of the following:

- A/C and flight data:
  - Time, day, month, year and UTC date/time.
  - MSN and Flight number.
  - Flight phase.
  - Altitude, pitch and roll.
- Fuel/Engine system data:
  - Engine status (Running or not).
  - Fuel flow (to each engine)
  - Fuel used (by engines).
  - Fuel on board (“FOB”).
  - Fuel quantity per collector cell and surge tank volume.
  - Pump status (On/Off, normally/abnormally, immersed/not immersed).
  - Leak detection and leak flow.
  - Fuel transfer mode.

Some important variables that we looked at when filtering and cleaning the data were, UTC_TIME, FUEL_FLOW, FUEL_USED, VALUE_FUEL_QTY, and FLIGHT. It is important to note that we did not have data descriptions about all the features, except for general knowledge using the variable names themselves.
Most of the datasets have more than 90% null values for critical features such as fuel consumption.

3.b. Feature Creation

In a typical machine learning pipeline, the feature creation process is usually done after data cleaning, within feature engineering. However, in this paper and for explanatory purposes, we will first talk about the various created features that were necessary to clean the data.
In total, 7 new features were created in each dataset. Figure 4 below gives a short description of each one of them:

Features	Units	Description
DELTA_TOTAL_FUEL_USED	[kg/s]	Returns the fuel being consumed by the aircraft in real time
VALUE_FOB_FROM_CONSUMPTION	[kg]	Fuel on board calculated based on the real time fuel consumption of the aircraft DELTA_TOTAL_FUEL_USED
VALUE_FOB_vs_FROM_CONSUMPTION	[kg]	Difference between VALUE_FOB (given) and FOB calculated based on the real time fuel consumption VALUE_FOB_FROM_CONSUMPTION
DIFF_VALUE_FOB_vs_FROM_CONSUMPTION	[kg/s]	Error rate of VALUE_FOB_vs_FROM_CONSUMPTION
FOB_FITLINE	[kg]	Smoothened VALUE_FOB (given) Approximation of VALUE_FOB (given) by a polynomial function of degree 5
FOB_vs_FITLINE	[kg]	Difference between VALUE_FOB (given) and FOB calculated from the polynomial function of degree 5 FOB_FITLINE
FOB_vs_FITLINE_OVER_FOB	[%]	Normalized difference between VALUE_FOB (given) and FOB calculated from the polynomial function of degree 5 FOB_FITLINE

Figure 4: Summary of the Features Added

DELTA_TOTAL_FUEL_USED allows us to compute the real-time fuel consumption of the aircraft. Since the datasets only returned the accumulated fuel consumption for each engine, we had first to compute the total consumption of the four engines together, and then subtract this total consumption from one of the previous records. The final equation for DELTA_TOTAL_FUEL_USED is the following:

$$\text{DELTA\_TOTAL\_FUEL\_USED}\,(t) = \sum_{i=1}^{4} FUEL\_USED\_i\,(t) \; - \; \sum_{i=1}^{4} FUEL\_USED\_i\,(t-1)$$

VALUE_FOB_FROM_CONSUMPTION calculates the expected mass of fuel onboard based on the consumption of the four engines. Only the first value of fuel on board (at t = 0) is taken from the feature VALUE_FOB given by the datasets. The other values are calculated using the real time consumption of the aircraft DELTA_TOTAL_FUEL_USED:

$$\text{VALUE\_FOB\_FROM\_CONSUMPTION}\,(t) = \text{VALUE\_FOB}_{t=0} - \sum_{i=0}^{t} \text{DELTA\_TOTAL\_FUEL\_USED}\,(t=i)$$

VALUE_FOB_vs_FROM_CONSUMPTION computes the error between the theoretical or expected value of fuel on board calculated based on the aircraft's consumption VALUE_FOB_FROM_CONSUMPTION and the value of fuel onboard VALUE_FOB given by the dataset:

$$\text{VALUE\_FOB\_vs\_FROM\_CONSUMPTION}\,(t) = \text{VALUE\_FOB}\,(t) - \text{VALUE\_FOB\_FROM\_CONSUMPTION}\,(t)$$

VALUE_FOB_vs_FROM_CONSUMPTION computes the error between the theoretical or expected value of fuel on board calculated based on the aircraft's consumption VALUE_FOB_FROM_CONSUMPTION and the value of fuel onboard VALUE_FOB given by the dataset:

$$\text{VALUE\_FOB\_vs\_FROM\_CONSUMPTION}\,(t) = \text{VALUE\_FOB}\,(t) - \text{VALUE\_FOB\_FROM\_CONSUMPTION}\,(t)$$

DIFF_VALUE_FOB_vs_FROM_CONSUMPTION computes the error rate or error evolution between the expected value of fuel onboard VALUE_FOB_FROM-_CONSUMPTION and the one given by the dataset VALUE_FOB:

$$\text{DIFF\_VALUE\_FOB\_vs\_FROM\_CONSUMPTION}\,(t) = \text{VALUE\_FOB\_vs\_FROM\_CONSUMPTION}\,(t) - \text{VALUE\_FOB\_vs\_FROM\_CONSUMPTION}\,(t-1)$$

FOB_FITLINE is a feature that was created to smoothen the VALUE_FOB feature. This feature is needed to later filter out sequences of flights which have abnormal attitudes. A polynomial function of degree 5 was used as it managed to follow the fuel on board's general trend with good precision, but with enough approximation to not capture the irregularities from unstable attitudes.
FOB_vs_FITLINE is simply the difference in absolute value between the real value of fuel on board VALUE_FOB and the smoothened one from the polynomial function FOB_FITLINE. This difference will help measure how turbulent the attitude of the aircraft is compared to a theoretical smooth flight:

$$\text{FOB\_vs\_FITLINE}\,(t) = \left|\,\text{VALUE\_FOB}\,(t) - \text{FOB\_FITLINE}\,(t)\,\right|$$

FOB_vs_FITLINE_OVER_FOB takes the values of FOB_vs_FITLINE but normalizes them over VALUE_FOB. The idea is to weigh this difference using the total value of fuel onboard, since a difference of 10 kilograms is not as significant in an aircraft having 20 000kg of fuel as for an aircraft having 1000 kg of fuel remaining inside its tanks.

$$\text{FOB\_vs\_FITLINE\_OVER\_FOB}\,(t) = \frac{\text{FOB\_vs\_FITLINE}\,(t)}{\text{VALUE\_FOB}\,(t)}$$

After creating these 7 new features, we decided to create three new datasets for which we implemented artificially simulated leaks. After plotting various graphs, we realized how hard it was to visually detect a fuel leak on MSN37 (the aircraft with a confirmed minor leak). There were no significant changes in the behavior of MSN37's features compared to any other aircraft. To emphasize the differences among them and understand how a leak would behave and appear in the figures we plotted, we create three simulated leaks with different severities.

Now, one might ask: how can we simulate leaks without knowing how they behave in real life?

It is true that the team does not know how a leak behaves in real life, and therefore cannot create a model of a leak that reflects reality. If we knew exactly how it behaved, we would not need to design an anomaly detector, and our problem would become a regular classification problem. But we had to simulate clear anomalies because they could help us further understand the problem and guide our decisions, especially regarding which anomaly detection techniques work best with the given data. Therefore, we assumed a leak would be a constant loss of fuel on board, starting from the beginning of the flight phase studied. Below is the equation that replaced the VALUE_FOB features of the three datasets containing simulated leaks of different flows in kg/s and t in seconds:

$$\text{VALUE\_FOB}_{\text{leak}}(t) = \text{VALUE\_FOB (t) - LEAK\_FLOW} * t$$

Three simulated leaks were created in three different datasets, all derived from MSN37: a severe leak with a fuel loss of 5 kg/s, a medium leak with a fuel loss of 1 kg/s, and a minor leak with a fuel loss of 0.5 kg/s.
If we compare a flight from MSN37 and the same one with an added simulated leak, we can see how the behavior of fuel on board and various errors changes:

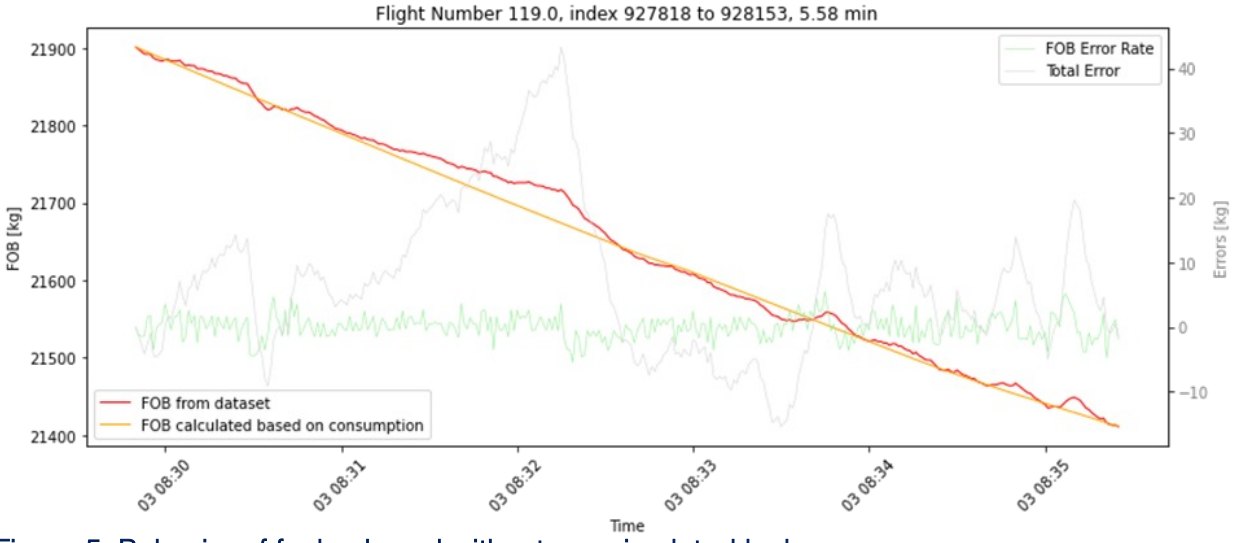


Figure 5: Behavior of fuel onboard without any simulated leaks

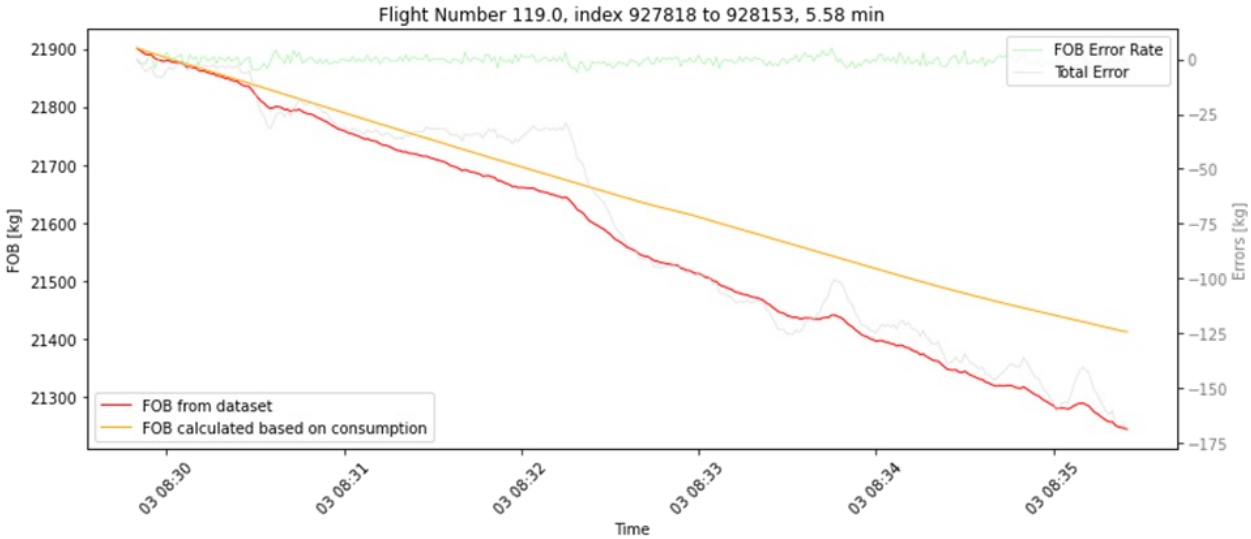


Figure 6: Behavior of fuel on board with a constant 0.5kg/s simulated leak

By adding a simulated leak, the total fuel on board error in grey increases over time (in absolute value), and its rate in green increases as well even though the scale of the graph does not allow proper visualization of this metric. We can now clearly distinguish an anomalous flight (Fig 5) from a normal (or very close to normal) flight (Fig 6).

3.c. Data Cleaning

Once new features were created, we could start using them to clean the 23 million records of data that were given by Airbus.

Choosing the most suited flight phase
Each dataset provided contains data for every flight phase of the aircraft, from preflight (phase 1) to postflight (phase 12). However, flight phases where the aircraft's engines are not running are less relevant for our study since the aircraft is not in typical operational conditions. For this reason, flight phases 1 and 12 were removed from the datasets. Taxi phases were also removed as the aircraft is rolling on the ground, turning and braking, creating a lot of turbulence inside the fuel tanks. Additionally, during taxi, the aircraft's fuel consumption (DELTA_TOTAL_FUEL_USED) is also constantly changing as pilots are manually controlling the thrust of the engines. This makes information extraction from the features more laborious for taxi-in and taxi-out.

The most suited flight phase for our study is a cruise. During the cruise, we can expect the attitude of the airplane to remain relatively stable, with small variations in pitch, yaw, and roll angles. Consumption is also constant as the aircraft is flying on autopilot and usually not maneuvering under high load factors. These conditions are unlikely to be met during takeoff, climb, approach, and landing phases, which is the reason why the only flight phase that was kept is flight phase 8: cruise.

Handling null values
After exploring the data and noticing the important number of records containing null values for features such as FUEL_USED_X (fuel consumed by the aircraft), it was decided to simply drop these records and not try to replace them. The reason behind this decision is that the information given by the fuel consumed is of critical importance when it comes to finding fuel leaks. Replacing and therefore assuming values of consumption would go against the philosophy of anomaly detection as these null values would have been replaced assuming normal consumption, which might not be the case, hence the presence of these null values.

The only null values that were replaced by 0 are APU_FUEL_FLOW_REQUEST_SIGNAL_1 values from MSN02. In this case, null values can be associated with the APU (Auxiliary Power Unit) being shut down. Dropping these records would be problematic as (in most cases) the APU is not in use during flight and relevant data regarding in-flight consumption or fuel on board would have been discarded.

Replacing outliers
Another concern that arose when exploring the data was sudden shifts in values of FUEL_USED_X as illustrated by the yellow line on Figure 7:

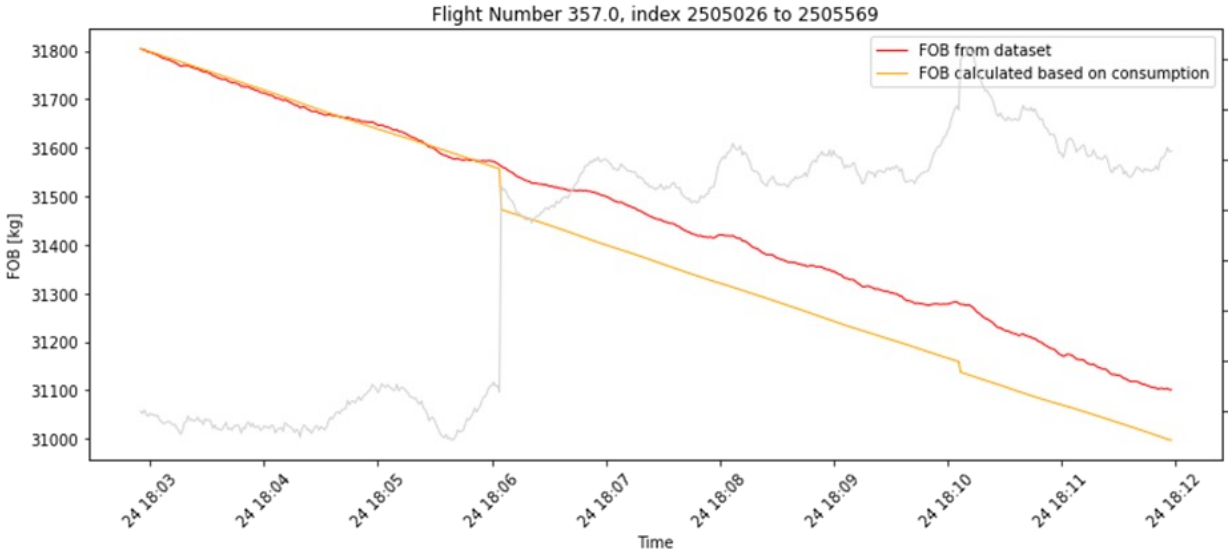


Figure 7: Behavior of fuel on board with a constant 0.5kg/s simulated leak

This yellow line represents VALUE_FOB_FROM_CONSUMPTION which is a function of FUEL_USED_X. Since these drops happen within one second of time and remain constant, it was assumed that they were due to sensor errors and were shifted back to normal values, still considering the real fuel consumption of the aircraft. To cancel this sensor error, the first value of the "fallen" segment was replaced using a backfill method, allowing the following values from that segment to be corrected without explicitly replacing them since they are calculated based on the previous (and therefore already corrected) value. Below is the representation of VALUE_FOB-_FROM_CONSUMPTION for the same flight sequence as shown in Fig. 7, but after the correction was applied:

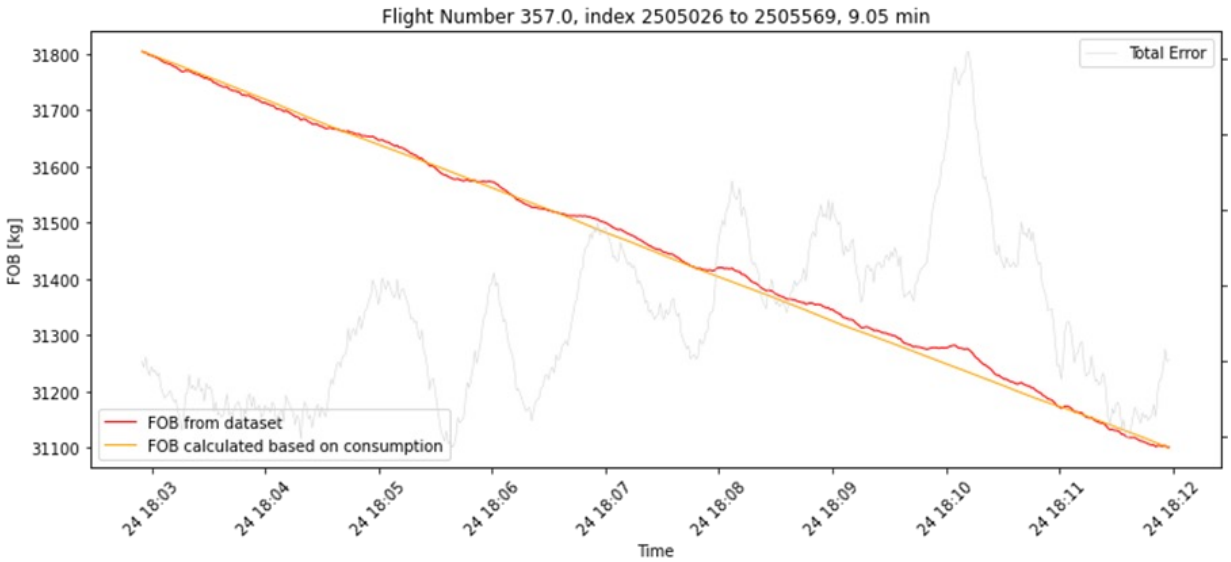


Figure 8: Sensor error on fuel consumption (FUEL_USED_X) after correction

Once these sensor outliers were removed, we tried to eliminate other types of sensor errors that come from the fuel flow sensors that calculate FUEL_USED_X, and from the sensors calculating the total quantity of fuel inside the tanks, or VALUE_FOB. Since every sensor has a slightly different behavior even if it comes from the same manufacturer, we wanted to get rid of their errors as we thought they could affect our anomaly detector's performance by adding a certain bias.

The sum of these two errors coming from the fuel flow sensor and the sensor measuring the quantity of fuel onboard should be captured inside VALUE_FOB_vs_FROM_CONSUMPTION (also known as the error between the value of fuel onboard given and the one calculated from consumption). In order to model that sum of errors and eliminate it, we tried to see if there was a correlation between VALUE_FOB_vs_FROM_CONSUMPTION and any other feature of the datasets. Unfortunately, it appeared that this feature was not correlated to any other feature. Modeling these sensor errors would therefore be more complicated than expected. Due to timing constraints and considering the minor impact they should have on our anomaly detector's performance; it was decided to abandon their modeling and removal.

Removing sequences of flights with unstable attitudes

One of the biggest challenges related to this predictive maintenance project is the fact that we are trying to build an anomaly detector for a machine that is freely moving in a three-dimensional space. Studying the behavior of the fuel on board to detect anomalies can be misleading as the latter is constantly moving inside the tanks, creating high volatility in the measures which could be wrongly interpreted as an anomalous behavior by the anomaly detection model. In order to avoid this kind of misinterpretation of the data, it is necessary to filter out agitated sequences of flights with unstable attitudes. The most optimal way of filtering these sequences would be to use the pitch and roll angles derivatives. Above a certain threshold in degrees per second, we could consider the flight sequence as abnormal in terms of attitude and eliminate the corresponding records. This is the approach Pizarroso Gonzalo took by filtering out records that had fuel pitch and fuel roll derivatives above |0.28| °/s and |0.14| °/s, respectively. However, unlike Pizarroso Gonzalo, we only have information about pitch and roll for aircraft MSN02. The attitude of the aircraft is not given in the other datasets, meaning we had to find another way of removing sequences with unstable attitudes.

To tackle this issue, we computed a smoothened version of the fuel on board as explained in the part above: FOB_FITLINE. This feature is represented by the grey dotted line on the graph below:

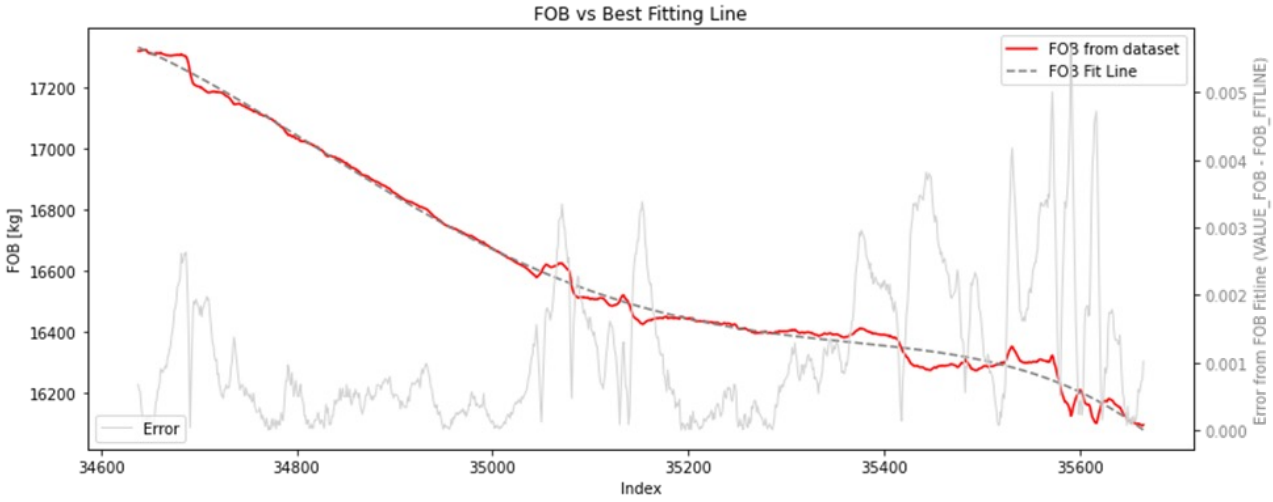


Figure 9: The real value of fuel on board (VALUE_FOB in red) with its smoothened version (FOB_FITLINE in dotted grey)

We then computed the normalized error between the real value of fuel on board and its smoothened version which we called FOB_vs_FITLINE_OVER_FOB (continuous grey line) to find indexes in the dataset where the fuel is being too turbulent inside the tanks. When the difference FOB_vs_FITLINE_OVER_FOB is high, it means that the value of fuel on board (in red) is far from its smoothened value FOB_FITLINE (in dotted grey). We can therefore consider their respective indexes as abnormal flight conditions and remove them. The threshold that was used is 0.0017 (in blue) as it seemed reasonable to achieve a decent smoothening. Any error above this threshold would eliminate the corresponding record from the dataset as shown below:

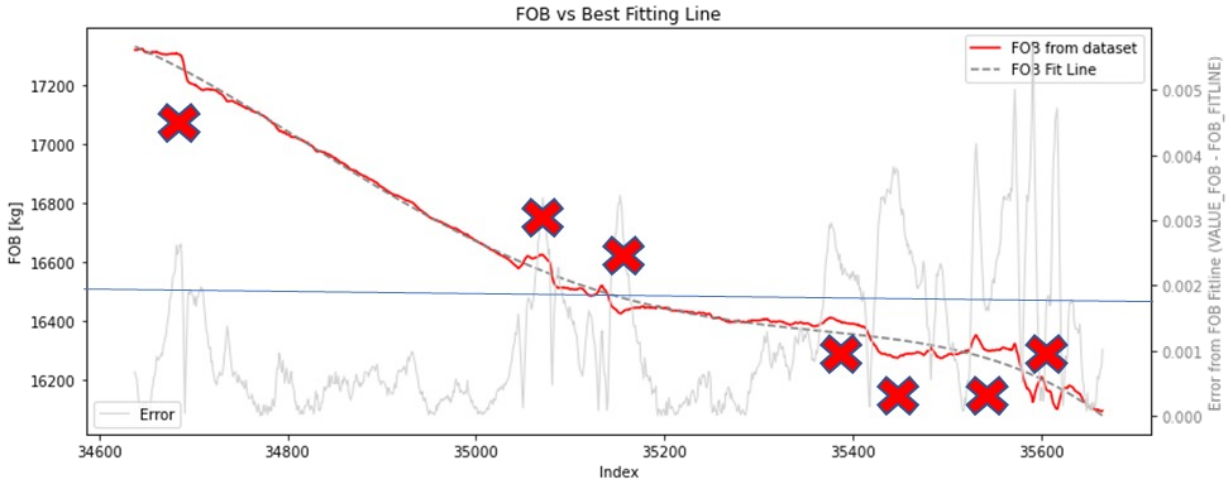


Figure 10: Elimination of flight sequences with unstable attitudes based on the threshold (in blue)

This filtering process allows for retaining smooth flight sequences, a crucial part of the elaboration of our anomaly detector. To give some perspective regarding the performance of this smoothening, if we look at aircraft MSN37, we were able to reduce the standard deviation of the derivative of its VALUE_FOB by 30%, from $\sigma = 4.4$ kg/s to $\sigma = 3.1$ kg/s.

Removing short sequences of flights

Due to this particular data cleaning and smoothening process, the original time series data that was provided to the team now contained many gaps in time, either due unstable attitudes of the aircraft that had to be removed or the elimination of null values. This resulted in the datasets being composed of small fragments of time series, many of them belonging to a same flight. Since we want to study behavioral trends in time, sequences that were below 180 seconds were removed in each dataset as they were considered too short. This allowed us to create a final dataset concatenating every aircraft's dataset with only long sequences of flights that were relevant to build our predictive maintenance model.

3.d. Overall Feature Engineering

Dropping features

Once the added features were created and used to clean the data, the idea was now to gather every aircraft under a common final dataset that would be used for our analysis and construction of our anomaly detector. Since MSN02 now contains 118 features (111 original and 7 added) and the other aircrafts 26 (19 original and 7,

added), it was necessary to drop every feature from MSN02 that did not exist in the other datasets and vice versa to only keep common features and concatenate every record to generate the final dataset. From the original 23,990,158 records, only 158,548 remain, meaning that less than 0.7% of the data from Airbus was used. An additional 9,709,935 records containing simulated leaks were added, bringing the total number of records to 33,700,093 before data cleaning. The final dataset which contains these simulated leaks has 24 features and gathers 251,823 records or 0.8% of the total number of records.

Of these 24 features, most of them were kept at this stage since we did not want to lose any relevant data. Depending on the method used to detect anomalies, some additional features were dropped, and others were gathered under a new one. These specific cases will be covered in the following parts of the report.

Scaling
Two different types of scaling were achieved depending on the method used for our anomaly detector. For the statistical approach and Principal Component Analysis (PCA), we used a standard scaler to standardize the distribution of our features. Standardization is needed when using PCA since this method creates a projection of the principal components on a new axis that is based on the standard deviation of the features.

Regarding the autoencoder approach, we used a min-max scaler as it showed better performance than the standard scaler.

3.e. Anomaly Detection

In the image below (Figure 11) you can see an overview of the possible algorithms and techniques to execute an anomaly detection. We will cover some of them in what follows.

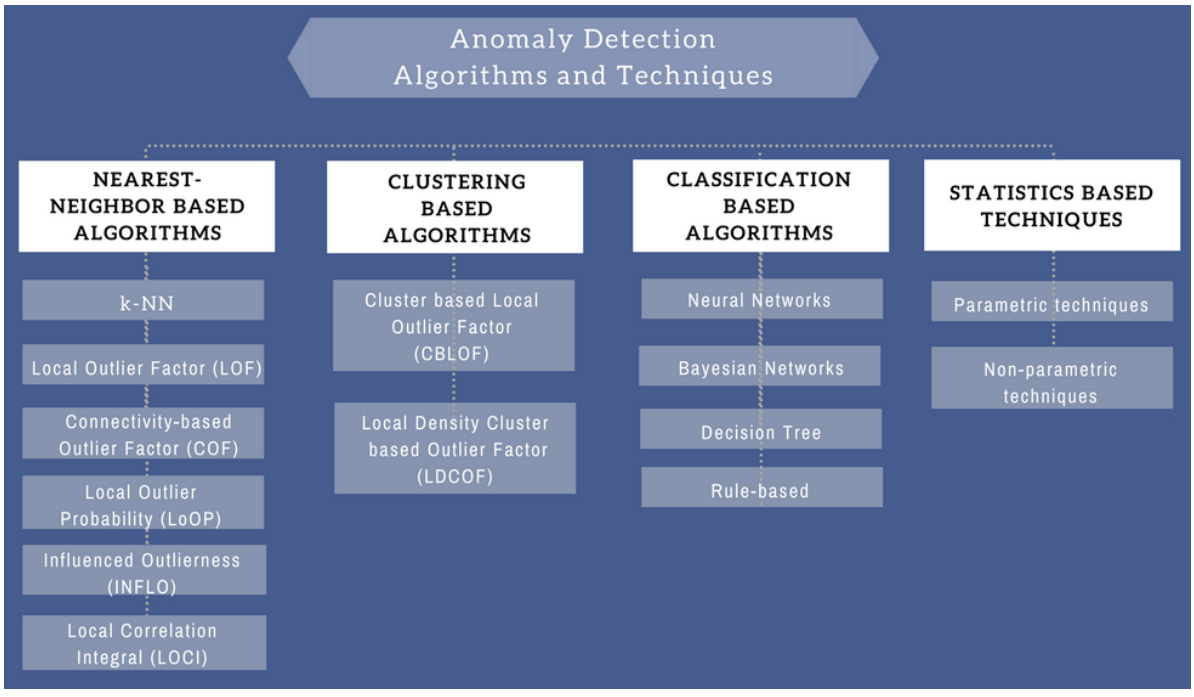


Figure 11: Anomaly Detection Algorithms and Techniques

3.e.i. Stat approach

Context and methodology

This simple but effective approach narrows the leak detection problem by looking at the error rate of the difference between FOB (given) and expected. The statistical method allows comparing with the same metric (%) the flights from all aircraft; in this context, we plotted in a histogram and focused on the values on the left side of the curve, filtering the flights contained in the percentile 10. From the filtered cases, we visualized each one of the flights individually. We graphed them, checking if the trend of fuel consumption features behaves according to a leak. Finally, we added simulated leaks to the data set to compare the results.

It is essential to establish that there is no "ground truth," which means that possible cases of leaks are not confirmed ex-post. The only fact that is known, based on the information provided by the company at the beginning of the project, is that msn37 is the only one that had leaks from the total sample of aircraft.

Finally, we added simulated flights with cases of different cases of leaks and checked the Error rate in the histogram, and compared them with real flights.

Results

The following figure shows the results after processing the data; First, we graphed the histogram with real data, and it was possible to infer that the anomalies are for the flights with negative average values of fuel consumption variation. After deep dive into the average variation, it is possible to see that aircraft 37 contains the most extreme cases of variation, which is consistent with the baseline information provided at the beginning of the project.

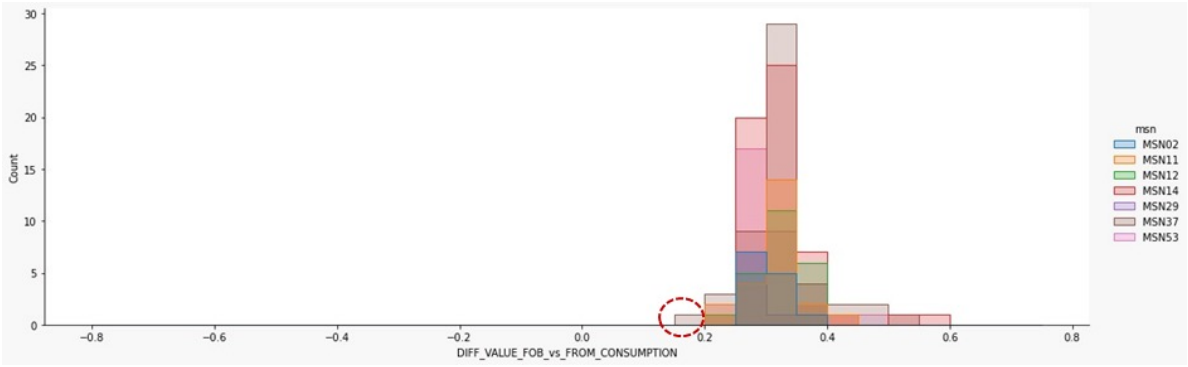


Figure 12: Histogram of Error rate in real data

Going deeper with the approach of looking for visual consistency in the consumption curves with the behavior of a leak, as shown in the following figure 12 Flight 98, the most extreme case in the histogram, shows a consistent behavior of fuel leak. Based on that assumption, Flight 98, from aircraft 37, was highly likable of a leak case from the total dataset.

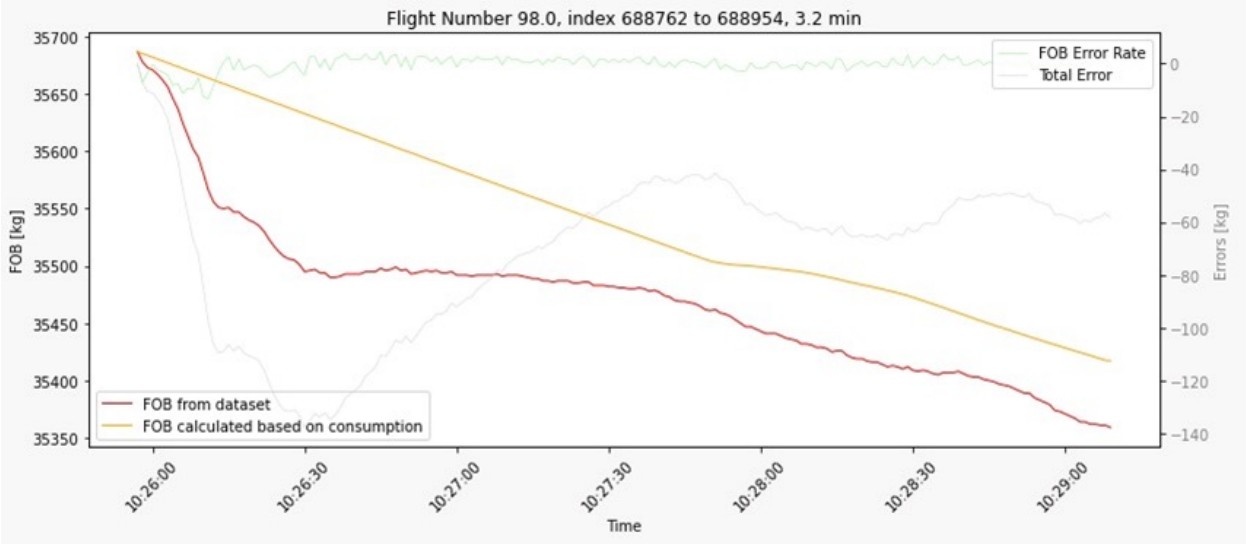


Figure 13: Flight 98 fuel consumption variation

After running a statistical test with a p-value of 5%, we could conclude that the Error Rate of flight 98 is statistically different from the rest of the flight. The null hypothesis is that the error rate for flight 98 is equal to the rest of the flights; the alternative hypothesis is that the value is lower than the mean of the rest. The p-value is < 5%, which means we could not confirm the null hypothesis and accept the alternative hypothesis. The results are presented in Figure 14 below:

	Flight 98	*Other Flights*
Mean	-0,302867482	0,055502854
Variance	7,953307866	4,095385962
Observations	193	158355
Hypothesized Mean Difference	0	
df	192	
t Stat	-1,764820649	
P(T<=t) one-tail	0,039591903	
t Critical one-tail	1,652828589	

Figure 14: Two-sample test for Error Rate

We finally compared the results with simulated cases of leaks. Figure 15 shows the histogram of the Error rate for real and simulated flights. The Error Rate moves towards the left of the X-axis when the leak increases, which is consistent with the Error Rate representation. The more negative Error Rate, the bigger the fuel leak.

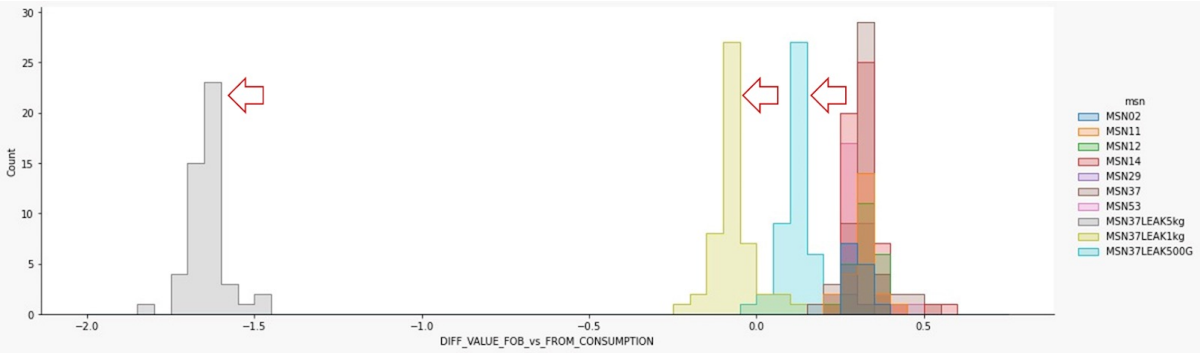


Figure 15: Histogram of Error rate in real and simulated data

3.e.ii) Unsupervised Learning

PCA

After exploring the statistical approach, we opted for an unsupervised learning approach and tried to see if there were any clusters being formed after performing a PCA. For this method, we took the standardized final dataset and tried to find the number of Principal Components (PC) needed to represent the data. To find an answer, we plotted the percentage of variance explained per PC shown in Figure 16 below.

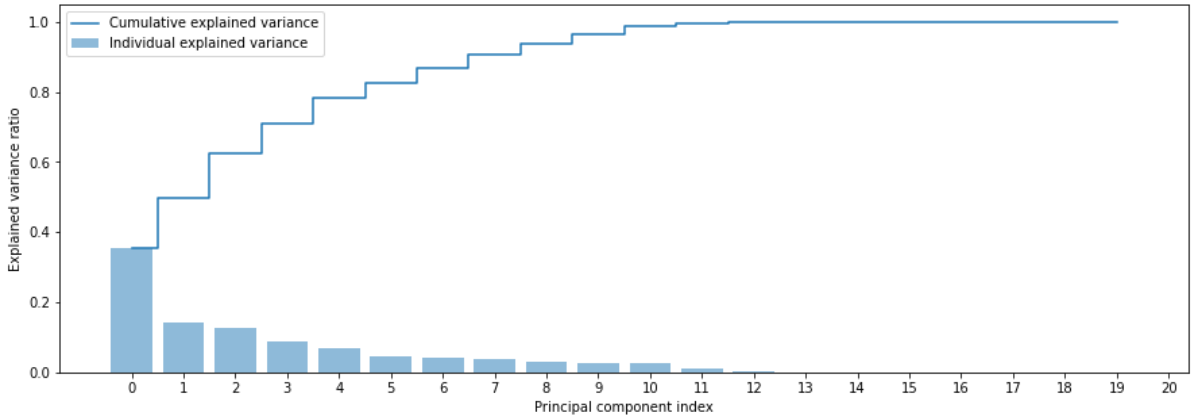


Figure 16: Cumulative explained variance of the PCs

If we take PC0, PC1, and PC2, we reach a cumulative variance of 62.4% which is sufficient for the analysis, and it allows us to plot them on a three-dimensional graph that is easily interpretable for the human brain.

We first plotted the graphical representation of the three PCs without simulated leaks to see if we could identify some clusters with the original data Airbus provided us. In red are data points from MSN37, the aircraft on which a minor leak was detected as seen in Figure 17.

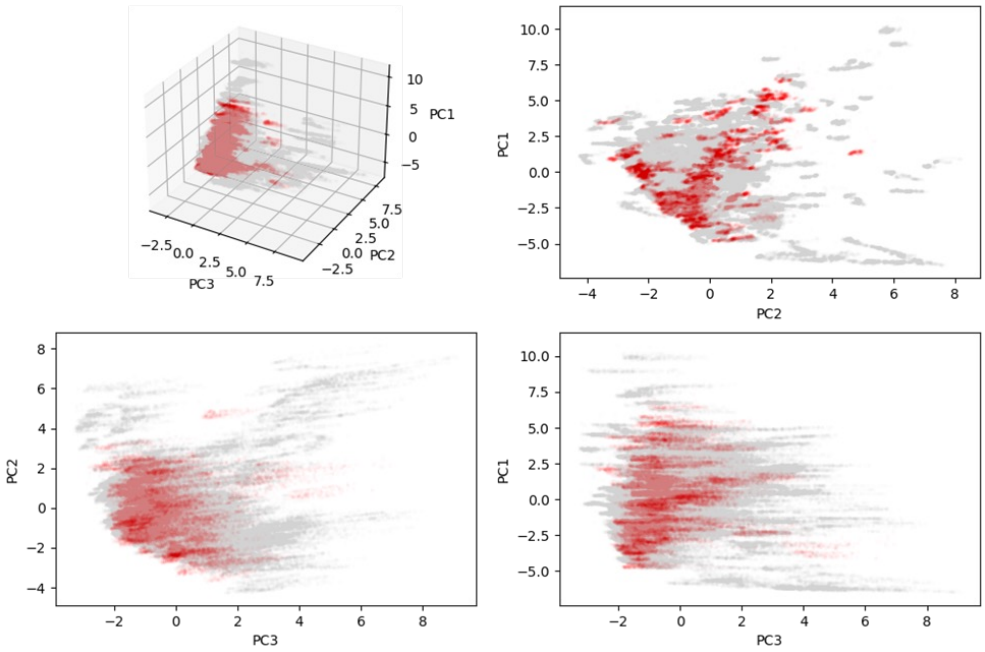


Figure 17: PCA without simulated leaks (3 views)

What we can see from Fig. 17 is that it is quite difficult to identify any clusters of data. MSN37, which is supposed to be an aircraft with anomalies, seems to behave just like any other aircraft from the dataset. Its leak might be so minor that it is hardly differentiable from other normal aircrafts.

The same exercise was done, but this time adding the simulated leaks of 5kg/s, 1kg/s and 0.5kg/s. The most severe leak (5kg/s) is colored in purple, the medium leak is in red, and the smallest one in yellow shown in Figure 18.

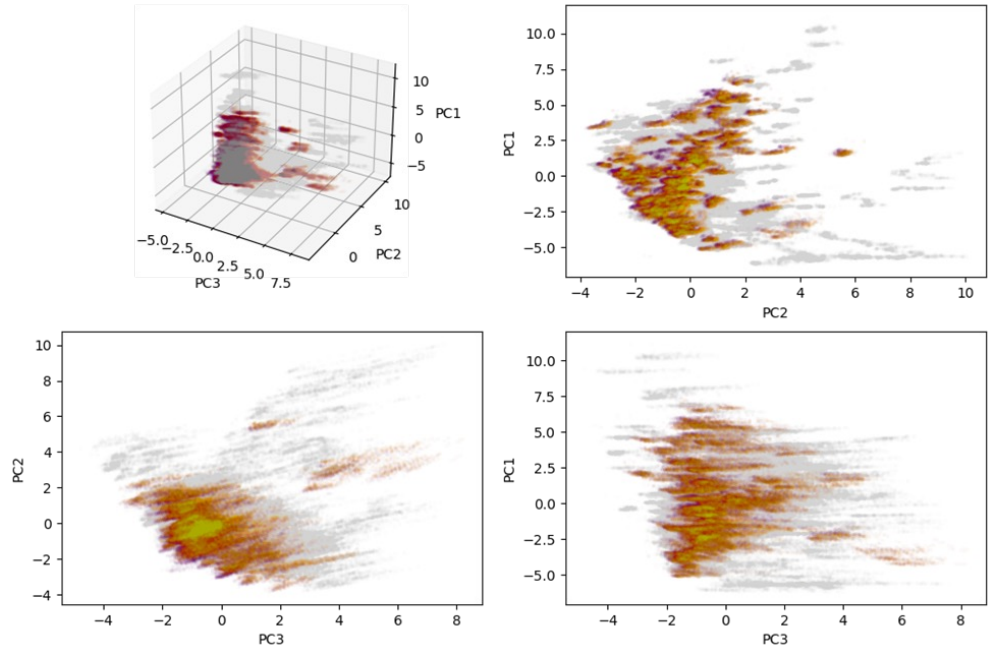


Figure 18: PCA with simulated leaks (3 views)

If there are still no real clusters of anomalies, we can however spot a trend. As the leak is getting bigger, it seems to shift to the left (negative values) of the PC2 and PC3 axis:

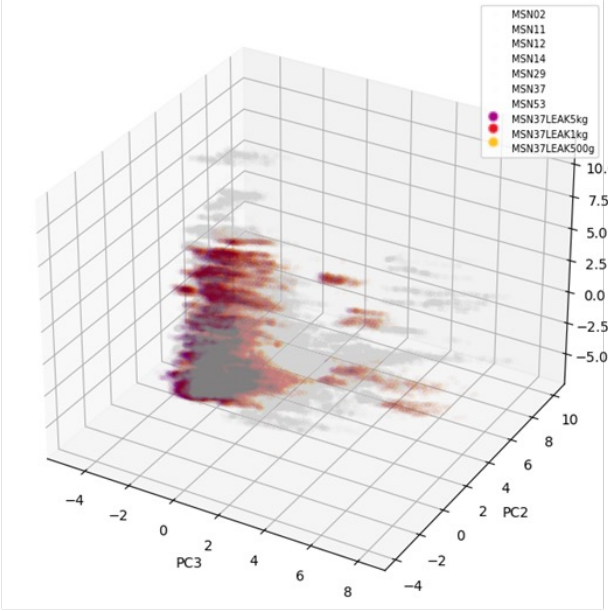


Figure 19: PCA with simulated leaks

If we had to conclude something about this PCA, we could agree that the results are below our expectations. Even after adding simulated leaks, there were no clusters of data but only vague trends as the severity of the anomaly increased.

PCA with K-Means

To further extend our clustering approach, we tried to create new clusters on top of PCA using K-Means. Our hope is that the algorithm will help us find different groups of flights, some with normal behavior and others with anomalies.

The first step is to find the appropriate number of clusters. For this, we followed the elbow principle shown in Figure 20.

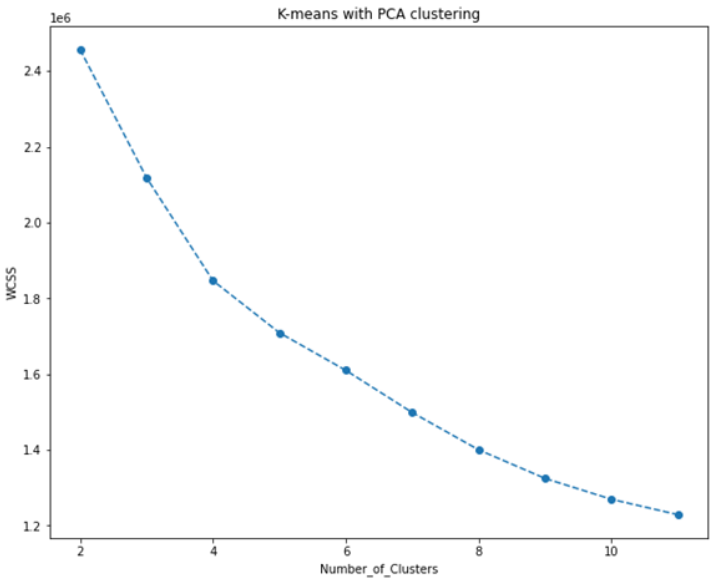


Figure 20: Inertia as a function of the number of clusters for K-Means

The optimal number of clusters corresponds to the value where the inertia starts decreasing. In our case, we can see on Fig.17 that the inertia starts decreasing for k = 4. Therefore, we will use 4 clusters for our analysis, and the 3 PCs from our PCA analysis.

We first ran K-Means without simulated leaks to see if by only using Airbus's data, the algorithm would be able to identify relevant clusters. Fig. 21 and Fig 22 are the results of this first analysis:

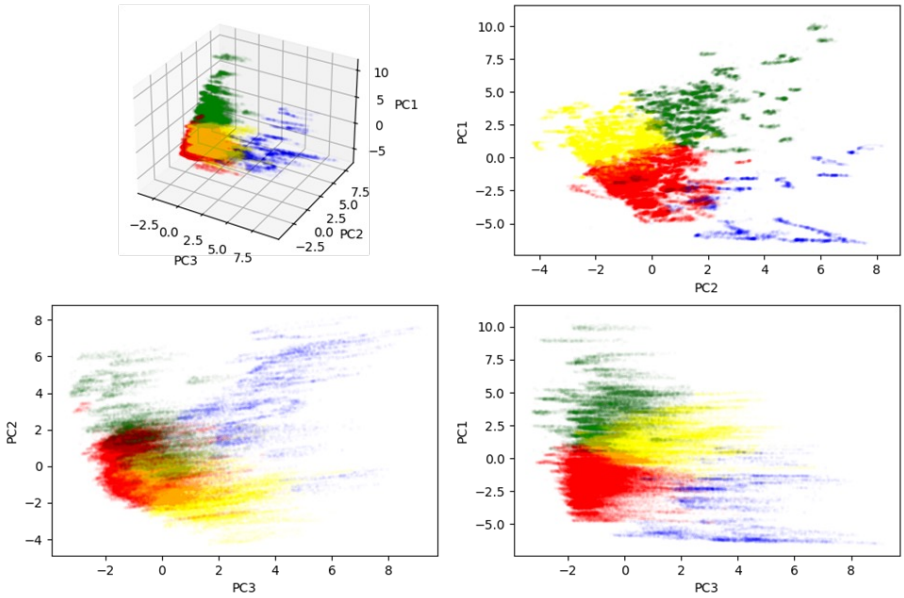


Figure 21: Clustering analysis using PCA with K-Means and without simulated leaks

When analyzing the composition of these clusters, nothing relevant was found. We hoped to find a cluster containing mainly sequences of flights from MSN37 which is the aircraft that has a confirmed minor leak, but K-Means could not capture our expectations. Every cluster is a mix of every aircraft, which does not allow us to draw any conclusion. We still did this same analysis, but this time adding the simulated leaks hoping the model would capture important anomalies such as the 5kg/s leak. Below are the results of this analysis:

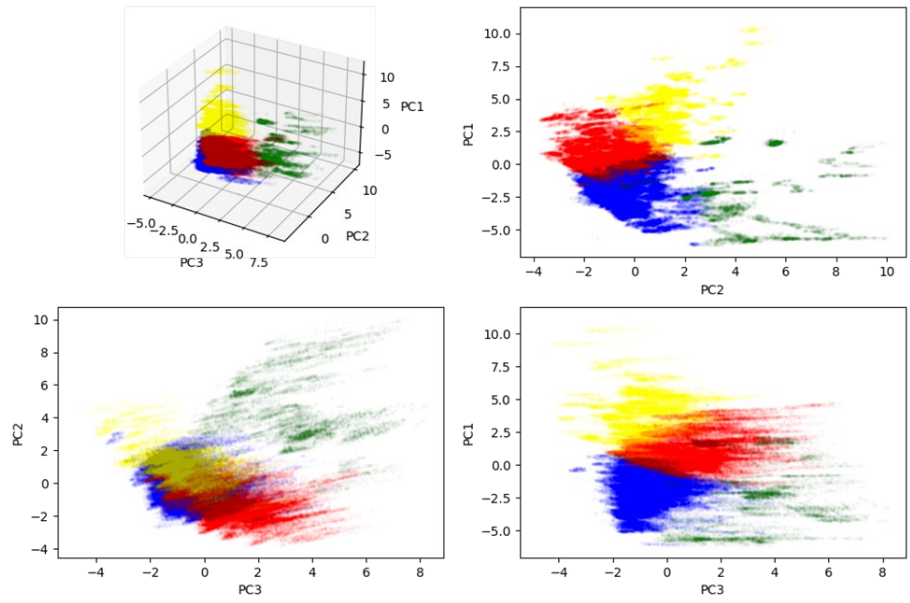


Figure 22: Clustering analysis using PCA with K-Means and simulated leaks

The same conclusion can be drawn for this analysis. The clusters are not capturing the anomalies we try to identify. K-Means does not seem to be an appropriate approach to finding leaks.

Local Outlier Factor (LOF)

After several trials with the statistical approach and experiments with K-Means and PCA or PCA as a stand-alone approach, the combination of this last one came to light together with the Local Outlier Factor (LOF) in order to look for the aircraft that could be considered as outliers. With this technique possible outliers can be located; in our case, those outliers could match with the anomalies that we are looking for.

When we classify the Machine Learning algorithms and techniques for anomaly detection, LOF is situated within the nearest neighbor-based algorithms because it detects outliers based on the local neighborhood. The data points are grouped based on their local densities. Items that have a significantly lower local density than their neighbors are classified as an outlier. Because of this, LOF can also be called a density-based outlier detection algorithm.

On top of that, LOF is calculated based on the average ratio of the local reachability density of an item and its k-nearest neighbors.

When we classify the Machine Learning algorithms and techniques for anomaly detection, LOF is situated within the nearest neighbor-based algorithms because it detects outliers based on the local neighborhood. The data points are grouped based on their local densities. Items that have a significantly lower local density than their neighbors are classified as an outlier. Because of this, LOF can also be called a density-based outlier detection algorithm.

On top of that, LOF is calculated based on the average ratio of the local reachability density of an item and its k-nearest neighbors.

In what follows, the steps we took to execute this algorithm will be explained. First, we performed a Principal Component Analysis (PCA). This dimensionality reduction technique creates new but fewer variables based on the features of the larger dataset but still tries to keep most of the information. After this step, we tried to locate the outliers. First, a data frame has been created with the three components as columns. Following this, we defined the fraction of in- and outliers. In our case, we chose to use 1500 samples to divide our 158.548 data points which accounts for approximately 100s or somewhat more than 3 minutes of data per sample and we fixed the outlier rate at 0.001%. Then, the parameter for the number of neighbors has been set to 50, a bit more than the default value because a higher number meant fewer potential outliers. Next, the algorithm has been fit and we made the predictions based on the classifier. Finally, the values have been plotted with the outliers in red in order to visualize them clearly.

This approach (PCA + LOF) has been repeated twice, excluding and including the simulated leaks.

A big advantage of this approach is the fact that it points at outliers that are local and not supposedly extreme values that would be detected with a global approach to finding outliers. On the other side, a typical drawback of this technique is that there is no predefined threshold that classifies outliers and normal points, but this is user-dependent as a consequence of the different parameter choices.

Unfortunately, when looking at the results of this approach, it doesn't seem promising. The technique detects some flights from all different aircraft without leaving one of them out. In reality, it does not make sense that in every dataset, which corresponds to every aircraft we analyzed, there would be a leak. Probably LOF is not precise enough or it is detecting another type of anomaly and thus not suitable for our problem. We can only conclude that this approach is invalid for detecting leaks and therefore, we didn't go further with it.

3.e.iii. Semi-Supervised Learning: Autoencoders

Why did we use autoencoders?

We explored the problem of leak detection in our dataset with a statistical approach and using unsupervised clustering methods but experienced very limited success, and were unsatisfied with our results. We decided to create artificial leaks to facilitate our understanding and visualization of the problem, enabling us to learn that if we have a leak in our dataset, it is very likely to be on average below 0.1kg/s in plane 37.

Machine learning has contributed significantly to predictive maintenance challenges since the models are based on pattern recognition and in particular "reconstruction" methods are theoretically very efficient for anomaly detection, contrasting with more traditional rule-based or statistical approaches. Therefore, we decided to add a new layer to our analysis using deep neural networks, with Autoencoders.

What are autoencoders, applications

Autoencoders are a deep learning model principally composed of two distinct parts: an encoder and a decoder. Very simply, the encoder tries to learn a general representation of our input data, reducing dimensions every time, and the decoder simply tries to reconstruct the original data based on the initial representation from the encoder.

There are several kinds of autoencoders, such as denoising autoencoders, variational auto-encoders, contractive autoencoders, and sparse autoencoders. Autoencoders can be applied for dimensionality reduction, image compression, image denoising, feature extraction, image generation, sequence-to-sequence prediction, and recommendation system. For our problem, we are mainly interested in deep autoencoders to facilitate anomaly detection.

Regarding their functioning, autoencoders are typically trained on normal data and the model learns how to reconstruct the normal form of the data, but for the test, abnormal data is introduced alongside normal data. When the decoder tries to reconstruct normal data, the reconstruction cost will be significantly low since the network learned what normal data looks like. However, whenever the decoder sees abnormal data which has never been seen before, the reconstruction error to come back to the "shape" of normal data will be higher. Of course, the greater the differences between abnormal, and normal data, the simpler it will be for the autoencoder to differentiate an anomaly.

Implementation of autoencoders

We are aware that autoencoders, although mainly known for being an unsupervised

method, can be used as a semi-supervised technique with labeled data. Given that the statistical and clustering techniques were not very successful to identify possible leaks with certainty, we decided to label all the flight data points without simulated leaks as normal with 1 and then labelled the data points of our flights with simulated leaks with 0. Our goal was to build a model that could differentiate between our normal data from the flights and the simulated leaks that could happen in a flight we injected to our dataset.

Baseline autoencoder model

We built a baseline autoencoder model that we would iteratively tweak to obtain better results. The baseline autoencoder dataset is our final dataset but with unsampled data containing simulated leaks of 5kg/s, 1kg/s and 0.5g/s. that did not contain any feature selection, contained 2 hidden layers (with 25 and 30 neurons each), with stochastic gradient descent as optimizer and mean average error as loss function. The training was performed on 25 EPOCHS and had a batch size of 100.

Figures 23 and 24 below show a snapshot of the encoder and decoder models.

```
autoencoder.encoder.summary()

Model: "sequential_24"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_77 (Dense)            (None, 30)                630

 dense_78 (Dense)            (None, 25)                775

=================================================================
Total params: 1,405
Trainable params: 1,405
Non-trainable params: 0
_________________________________________________________________
```

Figure 23: Baseline encoder model

```
autoencoder.decoder.summary()

Model: "sequential_25"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_79 (Dense)            (None, 25)                650

 dense_80 (Dense)            (None, 30)                780

 dense_81 (Dense)            (None, 20)                620

=================================================================
Total params: 2,050
Trainable params: 2,050
Non-trainable params: 0
_________________________________________________________________
```

Figure 24: Baseline decoder model

From Figure 25, the confusion matrix shows that our Autoencoder model is not effectively detecting leaks as it classifies 95% of them as normal data. Now our mission was to iteratively find ways to improve the performance of our model and the metric we would focus on is our f1 score since it enables us to balance our precision and recall.

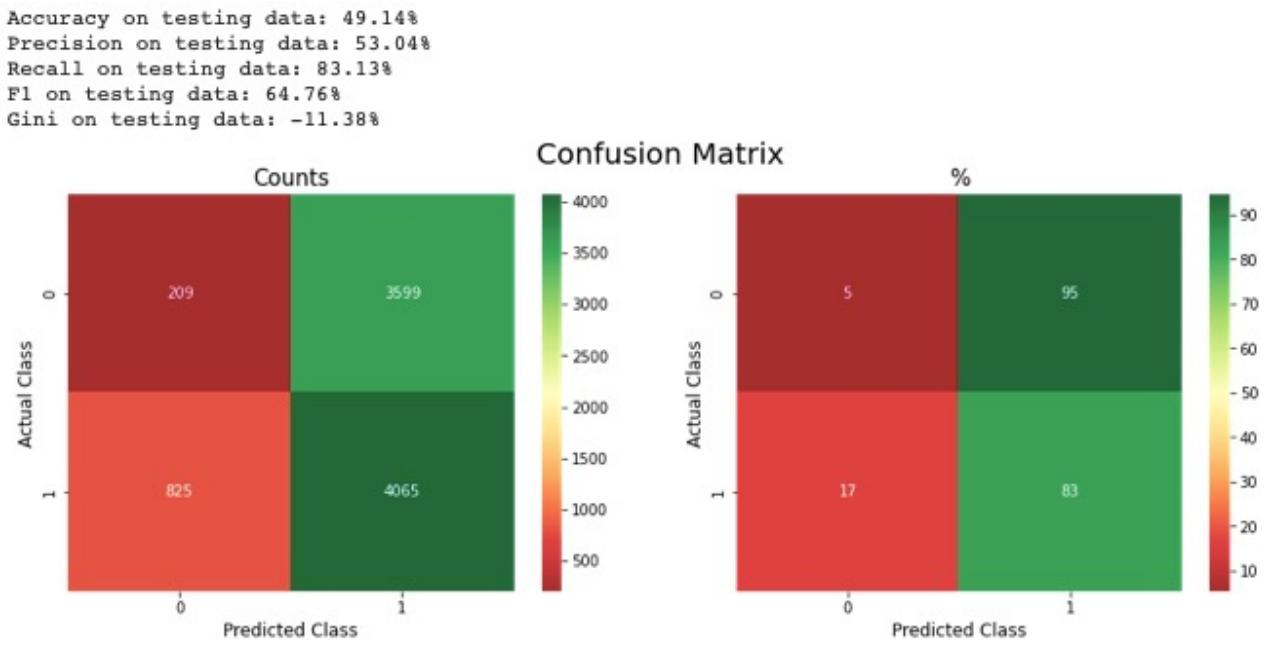


Figure 25: Baseline autoencoder confusion matrices

Consequently we decided to iteratively attempt to optimize the performance of our model by varying different parameters such as (a) the shape of our input data, (b) the features selection, (c) the model architecture, (d) the model optimizer and loss functions and (e) the leak sensitivity. Below we explain our findings that later led to our final model.

The input shape of our data

It made much more sense for us that whenever we perform fuel leak detection we look at sequences of seconds instead of data points for each second. Therefore, we proceeded to resample for every flight the data points of 1s to sequences of 5s of data and average the values we would have inside. In our case the results have been more or less similar when it comes to the model performance, however, when it comes to speed, the resampled dataset clearly was faster to train, and thus, we decided only to use datasets that were resampled from there on.

The features we select to input for the model

In order to simplify the complexity of the model, we created two new features. Firstly, TOTAL_FUEL_USED which is the sum of 4 fuel engine consumptions, and secondly, TOTAL_FUEL_QTY which is the sum of the 7 fuel tank quantities. Additionally, we also decided to remove FW_GEO_ALTITUDE because we knew in our model that it would not have any impact on leaks. We thus go from 20 features to only 10. Naturally, the performance of our model increased slightly, but if we decided to privilege datasets with our feature selection, it is in large part because the training speed increased as the number of trainable parameters was divided by 2.

The model architecture

Given that identifying fuel leaks is a complex problem our goal was to increase the number of trainable parameters. When the problem is complex, it is best to use deep neural networks, with multiple layers that would maximize the performance, however, the downside is the increase in training time. Thus we decided that for every new layer that we were adding, we would double the number of neurons each time. We trained the model adding 1 extra layer every time until we had a network of 9 layers and this

experimentation revealed that whenever the neural net has too many layers and trainable parameters, its performance also decreases. Thus, we learned that having 7 layers with 17,750 trainable params for the encoder and 18,160 for the decoder was the network architecture providing the best performance.

Activation and loss function change

Our baseline optimizer was stochastic gradient descent, which we switched for 'adagrad' and provided better results, we later switched for Adam which provided even better results. Additionally, we decided to try several learning rates, and the learning rate that provided the best performance was the learning rate of 1e-3. With respect to the loss function, we tried the mean average error, binary cross-entropy, and hinge, although these last 2 cost functions were not suitable for our autoencoder and we finally used the mean squared error as it was providing the best results.

Leak sensitivity

Lastly, we wanted to understand how sensitive our Autoencoder model was to the severity of the leak therefore we sequentially tried our model for either leak of 5kg/s, then of 1kg/s, and with 0.5kg/s. As expected the best results were given whenever the leaks were larger since whenever the anomalous sequence goes through the neural network, the reconstruction error will be greater as the data is more dissimilar than the normal. However, we are totally aware that when facing reality, an aircraft could be facing different types of leaks, thus, we will prefer to train our final models with several severities of simulated leaks in our dataset and see whether the model is able to label all these different leaks as anomalous.

Best model

After performing our iterative approach, we found out that the model with the greatest performance was achieved by training on the normal data from our resampled dataset of 5s per sequence. A train/test split of 80/20% and a feature selection were performed. We then decided to add 7 more layers doubling each time the number of neurons, and to switch our optimizer to Adam with a learning rate of 1e-3 and our cost function to mean squared error. We then proceeded to train with 200 EPOCHS and a batch size of 100.

```
autoencoder.encoder.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_67 (Dense)            (None, 80)                880

 dense_68 (Dense)            (None, 70)                5670

 dense_69 (Dense)            (None, 60)                4260

 dense_70 (Dense)            (None, 50)                3050

 dense_71 (Dense)            (None, 40)                2040

 dense_72 (Dense)            (None, 30)                1230

 dense_73 (Dense)            (None, 20)                620

=================================================================
Total params: 17,750
Trainable params: 17,750
Non-trainable params: 0
_________________________________________________________________
```

```
autoencoder.decoder.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_74 (Dense)            (None, 20)                420

 dense_75 (Dense)            (None, 30)                630

 dense_76 (Dense)            (None, 40)                1240

 dense_77 (Dense)            (None, 50)                2050

 dense_78 (Dense)            (None, 60)                3060

 dense_79 (Dense)            (None, 70)                4270

 dense_80 (Dense)            (None, 80)                5680

 dense_81 (Dense)            (None, 10)                810

=================================================================
Total params: 18,160
Trainable params: 18,160
Non-trainable params: 0
_________________________________________________________________
```

Figure 26: Final Encoder and Decoder Model

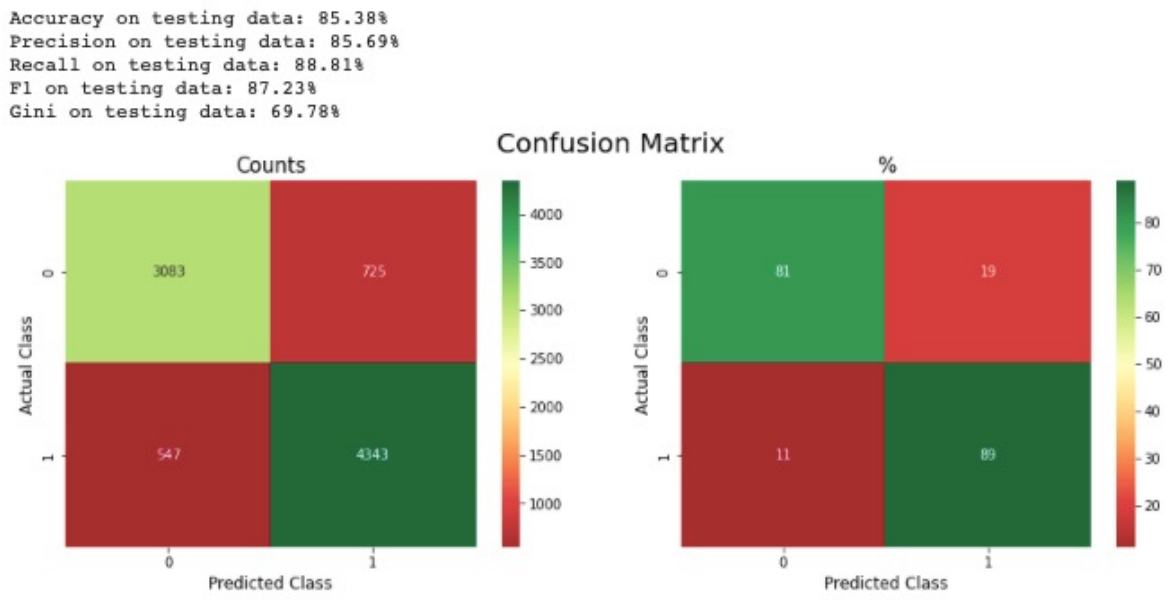


Figure 27: Confusion matrices of final model

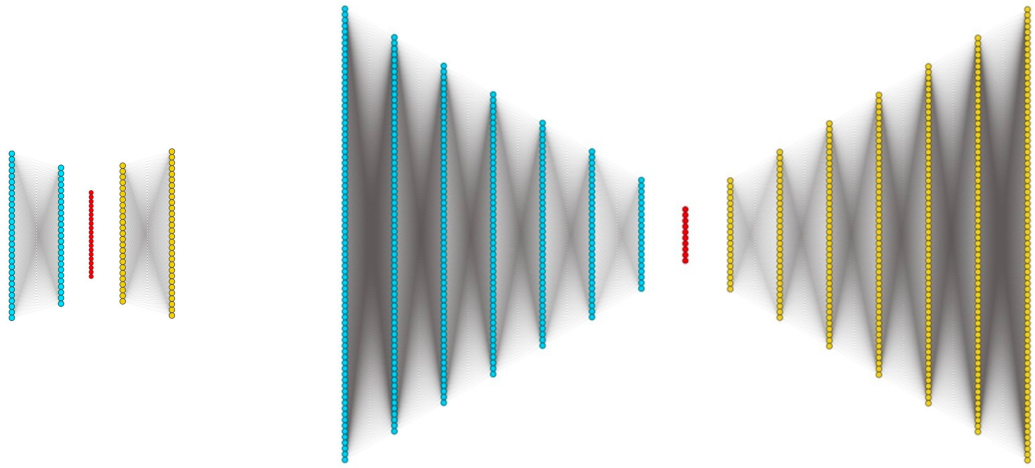


Figure 28: Baseline and Final Model Architecture

We should highlight that we achieve above 85% for both our recall and precision and our final F1-Score is 88%. Our data was slightly imbalanced therefore we should not pay too much attention to our accuracy, but rather that there are 19% of leaks that we miss-classify and based on our previous understanding of data, these are likely to be the simulated leaks of 0.5kg/s which have a shape very similar to the normal data.

To conclude we would like to highlight four things, (1) that the autoencoders were trained on simulated leaks that may not be exactly the way fuel leaks behave, (2) that the model may be more sensitive to larger leaks, in our case the 5kg/s simulated leaks provided the best results, (3) having more layers and more trainable parameters, to a certain extent empowered the model to better capture the simulated leaks properties and (4) switching the optimizer, including a learning rate, and using a different cost function also generated a model with a greater capacity to correctly classify normal data and leaks.

Potentially save the model and apply to new test sequences that contain a leak within 41 min from the leak start and thus confirm that our model could detect leaks faster.

4. Final Outputs

4.a. Results of the Models

After trying different approaches, namely, a statistical approach, PCA for an unsupervised approach, and Autoencoders for a semi supervised approach, the best performing model turned out to be the Autoencoders with a final F1 score of 88%. Figure 28 summarizes the results for the models.

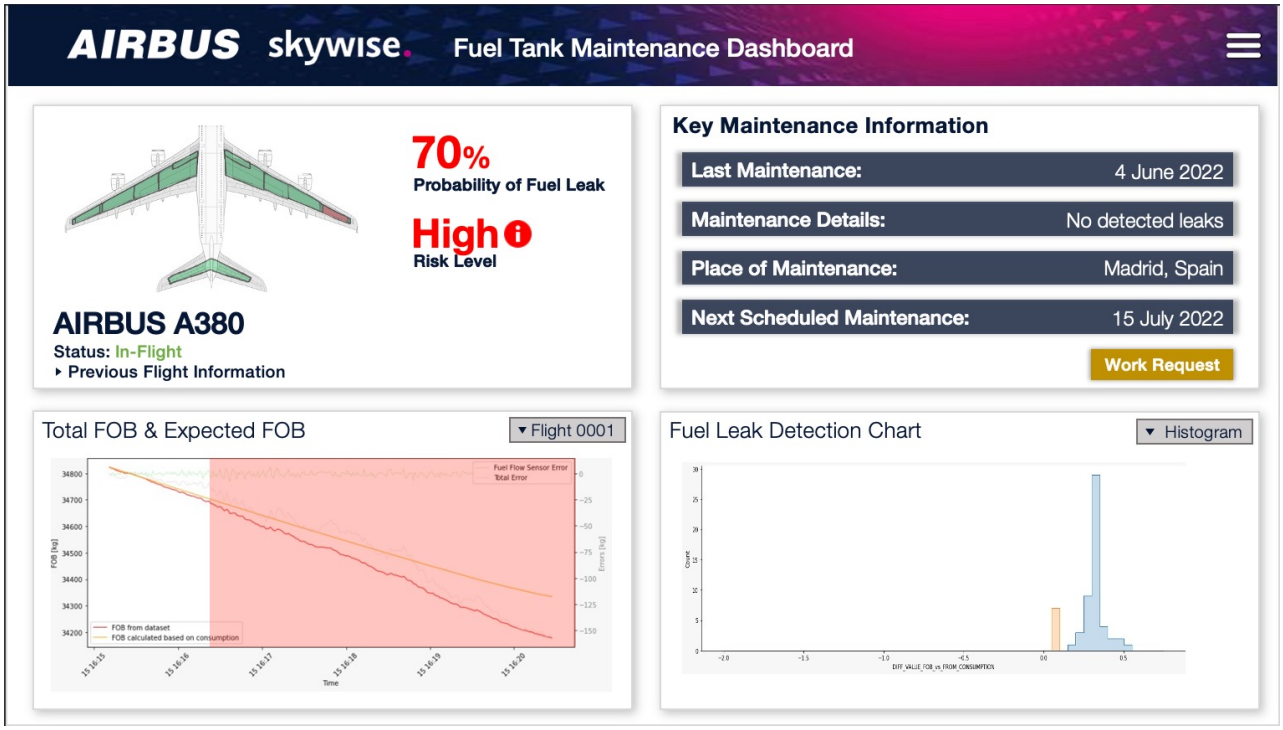
Model/Approach	Results
Statistical Approach	The Error Rate of flight 98 is statistically different from the rest of the flights. The null hypothesis is that the error rate for flight 98 is equal to the rest of the flights; the alternative hypothesis is that the value is lower than the mean of the rest. The p-value is < 5%, which means we could not confirm the null hypothesis and accept the alternative hypothesis.
PCA	With no simulated leaks: The data shows no identifiable clusters for anomalies but there is a trend. With simulated leaks: The data shows still no identifiable clusters for anomalies but there is a trend that As the leak is getting bigger, it seems to shift to the left (negative values) of the PC2 and PC3 axis
PCA with K-means	Every cluster is a mix of every aircraft, which does not allow us to draw any conclusion. K-Means did not prove to be an appropriate approach to finding leaks.
Autoencoders	The model with the greatest performance was achieved by training on the normal data from our resampled dataset of 5s per sequence obtaining a F1-Score of 88%

Figure 29: Summary Results

To summarize, we have tested different approaches to solving the problem, of finding an improved approach to detect fuel leaks. The most promising of these models was autoencoders as it provides a better performance when it comes to more complex problems such as detecting small leaks. However, it is important to note that these are only potential models and can be used as a baseline to be further developed. We have included a list of proposed improvements in the next section.

4.b. Fuel Leak Detection Dashboard

The second output of the project is to recommend a possible mockup for the data analytics platform Skywise. Below is a image of the mockup we have created that includes four main parts (1) Important and General Information of the Aircraft, (2) Key Maintenance Information, (3) Total FOB and Expected FOB Graph, and (4) a Fuel Leak Detection Chart.



Important Information

On the upper left portion you would find the important information about the aircraft such as the model, the status of the aircraft, a dropdown for previous flight information, the probability of a fuel leak based on the ML model, and the risk level. In addition there are also colors for each tank to be able to easily identify which tank has a leak.

Key Maintenance Information

On the upper right gives additional information on the last maintenance performed, its details, like if there was a leak detected, the place of maintenance, and finally the next scheduled maintenance

Total FOB & Expected FOB Graph

On the lower left there is a graph of the expected fuel onboard and the total fuel of board. This is to visually see the difference between the expected and actual FOB if in case of a fuel leak

Fuel Leak Detection Graph

On the lower right there is a graph to visualize in another way the fuel leak. In this mockup we have place a histogram of the potential fuel leak in yellow that is shifting away (to the left) from the normal histogram in blue.

5. Learnings and Improvements

This part gathers all our improvement ideas that should be considered to further investigate as they should improve the overall quality of our approach. These improvements have been divided into 3 parts, (1) Data cleaning and preparation, (2) Modelling and (3) final dashboards and output.

5.a. Data cleaning and preparation

5.a.i Intelligent sequence matching

With our current data cleaning, we are losing the opportunity to capture longer sequences of data as our filtering aims to retain uninterrupted sequences of at least 3 minutes. Beyond that, we are aware that uninterrupted sequences near three minutes, could be disconnected to other longer sequences because between them there are a few seconds containing null values, which could come from a minor sensor malfunctioning. An intelligent sequence matching could detect these null values and interpolate them so that we could connect our sequences together and thus avoid losing quality data that could be helpful for our analysis and model building.
Improved simulated leaks

To improve our simulated leaks we could use SIMULINK because currently our leaks are too perfect and nice. First of all we should make our leaks start at the middle of the flight and not since the beginning, secondly, our leaks are constant and in theory leaks are expected to grow as time passes because the pressure may increase the damage and thus the leak may grow, third, we need to add some noise and randomness to our leaks because we built them in a very homogeneous fashion (5kg/s, 1kg/s, and 0.5kg/s) which is unrealistic.

5.a.ii. Increased study period

During our cleaning we decided to focus for flight phase 8 since it was cruise phase and the plane is horizontally flying, it has reduced pitch and noise and therefore, fuel in tanks measured is most likely going to be accurately measured. It is the most logical approach especially since our predictive maintenance and leak detection missions become more relevant when performed in flight to quickly take action if a leak occurs. However, we think that it could have been interesting to also analyze pre-flight and once the plane landed as well since the plane is also horizontal and meanwhile the plane operates, it is possible that we could notice a mismatch between the actual FOB and the FOB we were expecting to have.

5.a.iii. Plotting aesthetics

Instead of a stats histogram we believed that it could be relevant to plot a continuous bell curves to have more precise and interpretable trends.

5.b. Modelling

5.b.i. Implement Gaussian Mixture Model

We could have tried to implement a Gaussian Mixture Model since it is an algorithm largely used for signal processing or anomaly detection. The Gaussian Mixture Model is not so much a model but rather a probability distribution which is part of unsupervised generative learning. The goal would be to identify normally distributed sub-populations within a larger population. We unfortunately learned about this very recently and never had any course about the method, but our hypothesis is that this approach could provide better results than our K-means approach and complement efficiently our statistical approach.

5.b.ii. Implement LSTM network

We could have tried to implement a Long Short Term Memory neural network which is also an unsupervised or semi-supervised learning approach for our leak detection and predictive maintenance approach. An LSTM is a special type of autoencoder that is privileged for time series data, however, the model is very complex and it was challenging for us to implement it given that we learned very recently about its existence and the time to preprocess our data for such sophisticated models can be significant. Nevertheless, our hypothesis is that this model could outperform our autoencoders with quality data, in terms of how sensitive it is to leaks classification.

5.c. Dashboard and final output

5.c.i. Improved aesthetics and visualizations

We believe that with a little more time we could successfully create our dashboard into a BI tool, that we could share and make interactive for an audience to try out. Additionally, with this tool, we could potentially show how the dashboard would behave with time, making it more realistic.

5.c.ii. Real-Time implementation

Once the model would have been perfected, the last step should be to investigate how to possibly deploy our model to operate in real-time with the data sensors from the plane and materialize our conceptual dashboard into a cost-saving tool. FOMAX and ACARS could be our solutions but we would have to consider the financial cost and temporal implications to implement this project.

6. Bibliography

Airbus. (2022). Skywise. Retrieved from Airbus: https://aircraft.airbus.com/en/services/enhance/skywise

Airlines for America. (2020, May 8). U.S. Passenger Carrier Delay Costs. Retrieved from Airlines for America: https://www.airlines.org/dataset/u-s-passenger-carrier-delay-costs/#:~:text=In%202019%2C%20the%20average%20cost,percent%20to%20%2424.55%20per%20minute

Boeing.(2022). Boeing AnalytX. Retrieved from Boeing: https://www.boeing.com/company/key-orgs/analytx/index.page#/overview

Carter Capner Law. (n.d.). The Right to Compensation for Plane Crash Victims. Retrieved from HG Org: https://www.hg.org/legal-articles/the-right-to-compensation-for-plane-crash-victims-42464

CMS admin. (2010, July 21). An easier fix for fuel leaks. Retrieved from Airforce Technology: https://www.airforce-technology.com/analysis/feature90366/

Cummins, N. (2020, June 9). How Much Does It Cost To Have An Aircraft On Ground (AOG)? Retrieved from Eways Aviation: https://www.eways-aviation.com/blog/how-much-does-it-cost-to-have-an-aircraft-on-ground-aog

IAG Cargo. (2018, July 3). Aircraft On Ground: How a technical fault can cost millions. Retrieved from IAG Cargo: https://iagcargomagazine.com/2018/07/03/aircraft-on-ground-how-a-technical-fault-can-cost-millions/#:~:text=When%20an%20aircraft%20is%20grounded,Boeing%20aircraft%20in%20operation%20today

Jayaswal, V. (2020, August 31). Local Outlier Factor (LOF) — Algorithm for outlier identification. Retrieved from Towards Data Science: https://towardsdatascience.com/local-outlier-factor-lof-algorithm-for-outlier-identification-8efb887d9843

Mitty, W. (2020, April 18). Airbus bet on Big Data. Retrieved from Digital Innovation and Transformation: https://digital.hbs.edu/platform-digit/submission/skywise-airbus-bet-on-big-data/

Proponent. (2022). What does AOG mean - the causes and costs of a grounded aircraft. Retrieved from Proponent: https://www.proponent.com/causes-costs-behind-grounded-aircraft/

Valcheva, S. (n.d.). 5 Anomaly Detection Algorithms in Data Mining (With Comparison). Retrieved from Intellspot: https://www.intellspot.com/anomaly-detection-algorithms/