UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

School of Industrial, Aerospace and Audiovisual Engineering
of Terrassa (ESEIAAT)

Artificial Intelligence For Aircraft Predictive Maintenance

Final Master's Degree Project

Author: Bernat Garreta Piñol

Advisors: David Duran Pérez, Dr. Xavier Franch & Dr. Manel Soria Guerrero

Master's Degree in Aerospace Engineering

June 2022 - Summer 2022 Call

Terrassa, Barcelona

This page intentionally left blank

Infinites gràcies a totes les persones que m'han ajudat al llarg de tants anys.
Un record per aquelles persones que ens han deixat però que seguim recordant.

Contents

List of Tables

List of Figures

Abstract

At first, the goal of this project was threefold: create the first version of a working interface, obtain more data to train the algorithm, and improve the performance of the deep learning classifier. Basically, to build upon what was implemented in the previous project.

The first task was successfully completed, and a usable interface was created. Obtaining new data from real world aircraft proved to be impossible to achieve before the delivery of the project. Although an agreement was reached with a manufacturer that could provide the required data for this project, the delivery dates did not line up correctly.

The results shown in the previous project were outstanding. However they were not reproducible in any kind of way. Moreover, many issues were found with the work presented on the previous project. An investigation was performed to obtain a deeper understanding of this matter. It was found that the previous project had a wonderful facade, but the foundations were not well established. It was necessary to start the project from the ground up if it was to be build up.

A new database was created by transforming the old database. After thoroughly analyzing the previous dataset, a second dataset was created by removing certain variables, transforming certain fields and creating new variables. Alongside this new database, a new ANN code was also developed, with the intention of improving the overall performance.

The results obtained with the new dataset and the new ANN were considered acceptable.The accuracy of the model was not high, but this was to be expected. Given the kind of data that was available for this project, only a small fraction of the total defects could ever be detected.

All in all, although at first the goal was to build upon what was already done, this project has been successful in creating a solid base for the future work on the predictive maintenance tool. Without this, any attempt at building on top of the flimsy foundations would have ended in total failure.

Glossary

This list presents the definition of the most important terms used during this report.

Availability Ability to keep a functioning state in the given environment. 2

Deep learning An artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making 3

Failure rate The frequency with which a system or item fails, expressed in failures per unit of time 8

Maintainability Ability to be timely and easily maintained (including servicing, inspection and check, repair and/or modification) 2

Predictive Maintenance A condition-based maintenance carried out as suggested by estimations of the degradation state of an item 3

Reliability Ability of a product or service to perform as expected under deviant conditions. 2

Safety Ability not to harm people, the environment, or any assets during a whole life cycle 1

Acronyms

ADASYN Adaptive Synthetic 26

AI Artificial Intelligence 18, 20

AMR Aircraft Malfunction Report 12, 13, 14

ANN Artificial Neural Network 21, 28, 40, 42, 88

ATP Acceptance Test Procedure 13

CSV Comma Separated Values 23

DIR Defect Investigation Report 12, 13, 14

DSR Discard 13

DV Defect Verified 13, 15

FRACAS Failure Reporting, Analysis And Corrective Action System 12, 14, 15, 16, 22

ML Machine Learning 51, 52

MLP Multi-Layered Perceptron 20, 26

MTBF Mean Time Between Failures 8, 15, 16

NAN Not A Number 24

ND No Defect 14

NFF No Fault Found 13

PDM Predictive Maintenance 3, 6, 7, 16, 17, 18, 50, 53

RAMS Reliability, Availability, Maintainability And Safety 2, 13

RPA Reliability Prediction Analysis 8, 11, 15, 16

SBS Scheduled Service Bulletin 14

SCM Scheduled Maintenance 14

SMOTE Synthetic Minority Over-sampling Technique 26

Chapter 0: General introduction

From its humble beginnings and throughout its history, the **safety** of the aircraft and their cargo has been of huge importance for the aviation industry. The first airplane in history was flown by the Wright brothers in 1903. Only 5 years later, the first airplane passenger death occurred [7]. On September 17, 1908 Thomas Selfridge was a passenger in a demonstration flight when the right-hand propeller of the aircraft broke, hitting a guy-wire bracing the rear vertical rudder. The rudder swiveled to the horizontal and sent the Flyer into a nose dive, followed by a sudden crash. Selfridge died after fracturing the base his skull. The figure 1 shows the aftermath of the crash.

Figure 1: First fatal crash in aviation, 5 years after its beginning

This tragedy was not caused due to a human error of the pilot or due to adverse flying conditions. It was caused by a mechanical failure. This could have been avoided if the safety of the airplane had been subjected to higher standards. Since then, it is estimated that over $100,000$ people have died as a result of an airplane crash. However, aircraft safety is not something that can be easily solved. Airplanes are enormous and extremely complicated machines. They are composed of hundreds of thousands of different parts crammed in relatively tight spaces, which makes interactions very probable and potentially harmful. A somewhat minor safety hazard that is dismissed for being considered irrelevant can end up causing a catas-

trophic disaster.

Aviation safety is of much importance mainly for three reasons: human lives, public trust, and economy. Firstly, catastrophic safety failures in aviation usually end up with a huge death toll. Avoiding these situations justifies enormous investments of resources. Secondly, a high degree of safety is to be expected for the general public to trust aviation. Even today, with very high safety standards, fear of flying is considerably widespread and as such, many potential clients are lost. Trust in aviation is crucial for its stakeholders. And thirdly, from an economic point of view safety is very important. The loss of an aircraft or some of its equipment has a huge monetary cost for the affected company. Not only that, but the high safety standards of the aviation industry make considerably more expensive every part of the life cycle of an aircraft. For these reasons, improvements in the safety of aviation result in less loss of life, more trust in the system, and a better economy.

Closely related to safety, **reliability** consists of the probability of an aircraft, equipment or system to perform its intended function under working conditions without any failure given a certain time frame. A high degree of reliability is required for critical equipment whose failure could cause a catastrophic effect, and this has a high cost. However, not all failures lead to such grave consequences. In these cases, **maintainability** plays a greater role. Maintainability refers to the ease to perform the maintenance activities on the aircraft equipment. With greater equipment reliability, the maintenance costs are reduced. Ans with greater maintainability of the equipment, the costs associated with aircraft safety are reduced. Lastly, **availability** is defined as the ability to have equipment at hand for a given amount of time. The different equipment and systems are of no use if they are unavailable at the time and place where they are required.

The conjunction of these four terms (reliability, availability, maintenance, and safety) form the acronym **Reliability, Availability, Maintainability and Safety (RAMS)**. Since these four concepts are closely related and should not be treated independently from one and other, the umbrella term of RAMS provides the tools to improve performance and reduce costs.

Circling back to the focus of this project, it is estimated that the cost of maintenance in aviation is approximately 50 to 60% of the total operation cost of an aircraft [8]. Since aircraft equipment costs millions of euros, aviation maintenance is a huge expense and of high importance to all stakeholders involved. Three different approaches to maintenance have been developed: reactive, scheduled, and predictive. Reactive maintenance consists of simply applying a maintenance action after the failure has occurred. Scheduled maintenance applies the maintenance action after a set amount of operational or total time as given per the reliability of

the equipment.

Finally, Predictive Maintenance, which can be abbreviated to Predictive Maintenance (PdM), consists of applying the maintenance action before the predicted failure of the equipment. Each step of the maintenance ladder lowers the overall cost of maintenance at the expense of adding extra complexity. As such, the development of a successful PdM tool using Big Data analytic and Deep learning presents a huge opportunity in the field of aviation maintenance.

This study is the second step of a **long-term project**: the goal of this project is to further build on the work previously carried out by improving the algorithm performance, obtaining real aircraft data, and implementing a client interface. The target of this project is ambitious, but there is a lot of potential and it presents a big opportunity for the aviation industry.

Accurately predicting the instant an aircraft will fail is extremely complicated and difficult. Airplanes are very complex machines composed of hundreds of thousands of different parts. All these parts are crammed in tight spaces and interact between each other, sometimes unintentionally. Each part, component and system have multiple failure modes, and each failure mode can have different effects. Small local failures can have huge ramifications for the entire aircraft. For this reason, 22% mechanical errors

Chapter 1: Introduction

In this chapter, the targets that have to be achieved in this project is presented by defining its objectives and justification. The scope and specifications of the thesis are also detailed.

1.1 Objectives

This study is the second step of a long-term project on predictive maintenance. As such, the general goal of this thesis is to build on the work that has been done to this point. The firs step of the project was tasked with creating a prove of concept that big data analytics can be used as reliable predictors using historical data. The aim was to analyze the possible correlations of the failures of an aircraft with its flight data and to predict the failures occur ed in an aircraft due to external conditions. This was achieved by developing an algorithm and testing it. The results were promising and supported the premise of the project. Three objectives have been set for this project based on these results.

The first objective is to improve the working database by obtaining more and better data from first hand measurements. The database used in the previous part of the project was created by combining the failures of the aircraft FRACAS with the flight data obtained using Open-Sky and Meteosat. The database would be considerably improved if on-board measurements recorded during aircraft flights could be obtained. To do this, an agreement would have to be reached with an aviation company that keeps a record of on-board measurements. Afterwards, this data should be processed and curated in order to be added or replace the existing database. If this is achieved, it is expected that the algorithm will produce significantly better results.

The second objective is to improve the algorithm that has already been developed. The algorithm that was developed in the previous part of this long-term project has much room for improvement. It was somewhat generic and not tuned for its intended role. As such, a new approach will be used to create a new algorithm version. Better results should be obtained by better adapting the algorithm to the specifications of predictive maintenance. This would also make it easier to negotiate with third parties.

The third objective is a first implementation of the algorithm on Robin RAMS software. With considerable effort, this implementation would become the first step

in the commercialization of the product. Moreover, giving the algorithm a decent interface would immensely improve the process of data training and model creation. Besides, implementing the algorithm on Robin RAMS would give more weight to the project and improve the standing with potential clients.

1.2 Scope

The activities developed to achieve the main aim of this project are:

- Brief literature review on reliability and different systems to compute it.
- Literature review on deep learning and different classifiers.
- Describe and present the work that has already been done.
- Contact and negotiate with potential clients in order to obtain real monitoring data of aircraft.
- Improve, enlarge or replace the working database.
- Create a new and improved version of the working algorithm.
- Compare the results obtained with the two algorithm versions.
- Implement the algorithm on Robin RAMS as a new module.
- Analysis of the results and main conclusions.

1.3 Requirements and specifications

The basic specifications and hypotheses of this study are summarized below:

- The flight database previously created will form the base for future databases.
- All code files already created for this project might be used at some point with changes.
- The new algorithm version will be based on the first algorithm created.
- The algorithm will be implemented on Robin RAMS as a new separate module.

Related programming requirements of this project are summarized below:

- Back-end: Python, Django, Pandas and SQL (Postgresql)
- Front-end: JavaScript, HTML and CSS
- Artificial Intelligence fundamentals: Keras and TensorFlow

1.4 Justification of the project

Every failure in aviation entails a certain cost. The failed equipment has to be removed from the aircraft, replaced with a working part and repaired in order to be returned to service. However, the expenses associated to maintenance can be severely reduced by improving the approach. As seen in Figure 1.1, currently there are three types of maintenance:

- Reactive maintenance is based on replacing failed component. It is the most simple but also most expensive.
- Preventive maintenance is based on replacing components after scheduled operational hours. In terms of complexity and cost, this maintenance type represents a middle ground.
- Predictive maintenance is based on replacing components based on a developed model. It has the most effective cost, but it requires a lot of work to obtain a good and reliable model to predict failures.

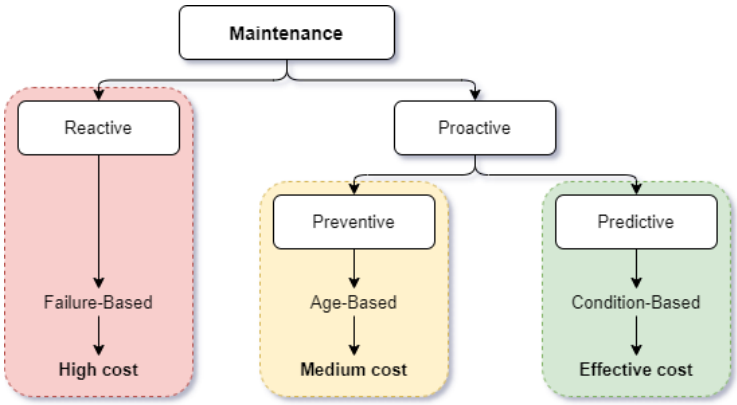


Figure 1.1: Maintenance classification

By developing a predictive maintenance tool for aircraft, a huge advancement in the field of aviation would be possible. If this PdM tool is successfully implemented, the maintenance costs in aviation could be significantly reduced. Moreover, the company in charge of the project should enjoy fruitful rewards and recognition. But most importantly, aviation safety could be significantly improved. If this is achieved, many lives could be saved. If the aviation industry is provided with a tool that is able to accurately predict failures before they happen, mechanical failures that posed a safety hazard would be eliminated to a great extent: no more engine failures, no more losses of control, no more landing gear blockades, no more cabin fires.

If one looks at aviation history, it is not hard to find several instances where a PdM tool could have avoided a terrible catastrophe. The following list contains some of these cases. The complete list is much larger. These people could have been saved. And many people that would have been involved in similar crashes will be saved if this project is successful.

- **Southwest Airlines Flight 812**: Depressurization caused by the structural failure of the fuselage skin. **2** injuries.
- **BKS Air Transport Flight C.6845**: The left flap operating rod failed due to metal fatigue. **6** fatalities.
- **Formosa Airlines Flight 7623**: System failures caused by the failed RH Main Bus. **13** fatalities.
- **Aeroflot Flight 3932**: failure of the electrical supply, causing incorrect indications by the main artificial horizon and the compass system. **108** fatalities.
- **Aeroflot Flight 964**: Multiple navigation instrument failures, including the compass and artificial horizon. **122** fatalities.

However, it must be made clear that the development of a predictive maintenance tool would not entirely replace reactive maintenance. Even if the best model to predict failures is created, there is always a slime probability of a random failure of a component. Moreover, some failures are produced rare occurrences, such as accidents or bird strikes.

Chapter 2: Literature Review

With the goal of framing the project and explaining some essential concepts that will be used during the rest of the report, this chapter presents a general literature review of reliability, predictive maintenance and machine learning.

2.1 Reliability

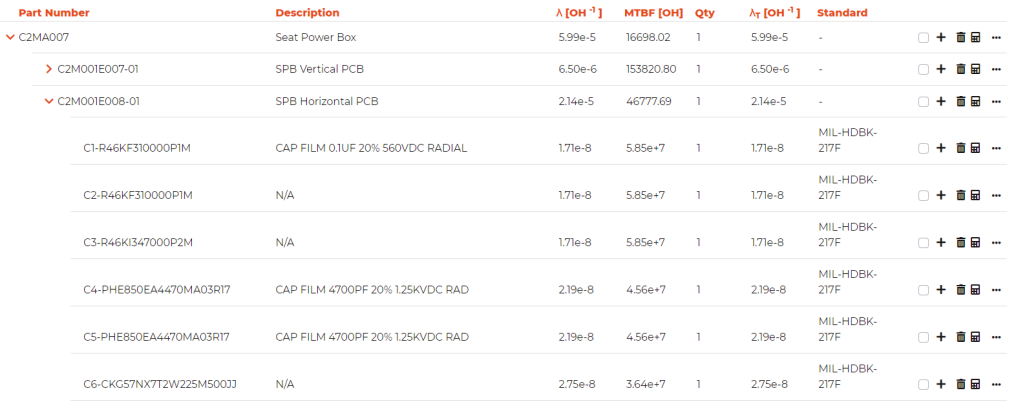
In engineering, reliability is defined as the probability of an item to perform its required function under given conditions for a stated time interval [9]. A failure occurs when an item stops performing its function. As such, the reliability of an item or system is usually expressed in two different ways, with the failure rate, denoted by the Greek letter lambda (λ), or as the Mean Time Between Failures (MTBF). The failure rate is usually expressed in failures per million hours, while the MTBF is expressed in hours. To go from failure rate to MTBF and vice-versa, it's as simple as doing the multiplicative inverse.

2.1.1 Reliability prediction with RPA

The manufacturers and designers of aircraft components and systems are required to predict the reliability of their product. This is mainly because the reliability of the system is to be estimated for the product certification, and because customers and clients usually demand to want to know the reliability before closing a deal. As such, the reliability of a system or item is predicted during the design phase, before being used in its intended use in the real world.

The reliability of a product is predicted using the Reliability Prediction Analysis (RPA). This analysis consists of breaking down the product into all of its basic components, calculating the failure rate of each component according to its operational conditions, and adding all of the failure rates to obtain the failure rate of the whole system or item. The figure 2.1 shows the general view of an RPA from Robin RAMS.

To obtain the failure rates of the different components, a priority order is often followed. This is based on the level of reliability and accuracy of the calculated failure rate values. This is important because failure rates that are obtained using a low accuracy calculation have a high degree of tolerance. Therefore, the failure rate value is increased in order to take into account this high degree of deviation



Part Number	Description	λ [OH⁻¹]	MTBF [OH]	Qty	λ_T [OH⁻¹]	Standard
C2MA007	Seat Power Box	5.99e-5	16698.02	1	5.99e-5	-
C2M001E007-01	SPB Vertical PCB	6.50e-6	153820.80	1	6.50e-6	-
C2M001E008-01	SPB Horizontal PCB	2.14e-5	46777.69	1	2.14e-5	-
C1-R46KF310000P1M	CAP FILM 0.1UF 20% 560VDC RADIAL	1.71e-8	5.85e+7	1	1.71e-8	MIL-HDBK-217F
C2-R46KF310000P1M	N/A	1.71e-8	5.85e+7	1	1.71e-8	MIL-HDBK-217F
C3-R46KI347000P2M	N/A	1.71e-8	5.85e+7	1	1.71e-8	MIL-HDBK-217F
C4-PHE850EA4470MA03R17	CAP FILM 4700PF 20% 1.25KVDC RAD	2.19e-8	4.56e+7	1	2.19e-8	MIL-HDBK-217F
C5-PHE850EA4470MA03R17	CAP FILM 4700PF 20% 1.25KVDC RAD	2.19e-8	4.56e+7	1	2.19e-8	MIL-HDBK-217F
C6-CKG57NX7T2W225M500JJ	N/A	2.75e-8	3.64e+7	1	2.75e-8	MIL-HDBK-217F

Figure 2.1: General view of an RPA with different levels from Robin RAMS [2]

between the estimation and the performance of the product in the real world. This means that a lower accuracy entails a higher degree of conservativeness. It follows that most of the times, the failure rate values calculated with higher accuracy are considerably lower than those with less accuracy.

First of all, the preferred method to calculate the failure rate of a component is to obtain said value from the supplier. If the component supplier has done the required research and analysis on the reliability of their product, the resulting failure rate value is very close to what should be expected on operational conditions. However, if the suppliers properly do their job, they update the failure rate of their product with more accurate values obtained from the analysis of the performance of the product in real world operating conditions. This is explained in more detail in the following subsection 2.1.2.

The second best option is to estimate the failure rate of the components is to use a reliability prediction standard. Some examples of these standards that are used in the aviation industry are the MIL-HDBK-217F, the FIDES guide 2009 and the RiAC-HDBK-217Plus. Basically, these standards offer different formulas to calculate the failure rate of a component based on two factors: the nature of the component, such as it's type, construction, year of manufacture, number of pins, and the like; and the operating conditions, such as ambient temperature, applied voltage, cycling rate, and others. However, the various standards have distinct approaches to the failure rate calculation and different levels of complexity, accuracy and conservativeness. All the various reliability prediction standards produce reasonable results, but the choice of standard to be used should be made with solid reasons and be consistent throughout the analysis.

As an example, the figure 2.2 displays the required inputs for the failure rate calculation of a capacitor using the military standard MIL-HDBK-217F on Robin RAMS. It can be seen that the user is required to introduce capacitor style, the

capacitance, the stress ratio, the circuit resistance and the quality of the capacitor. Afterwards, the calculation is performed and the figure 2.3 displays the outputs of this prediction. The most important value from the output is the failure rate, expressed in failures per operating hours. This is displayed at the top.

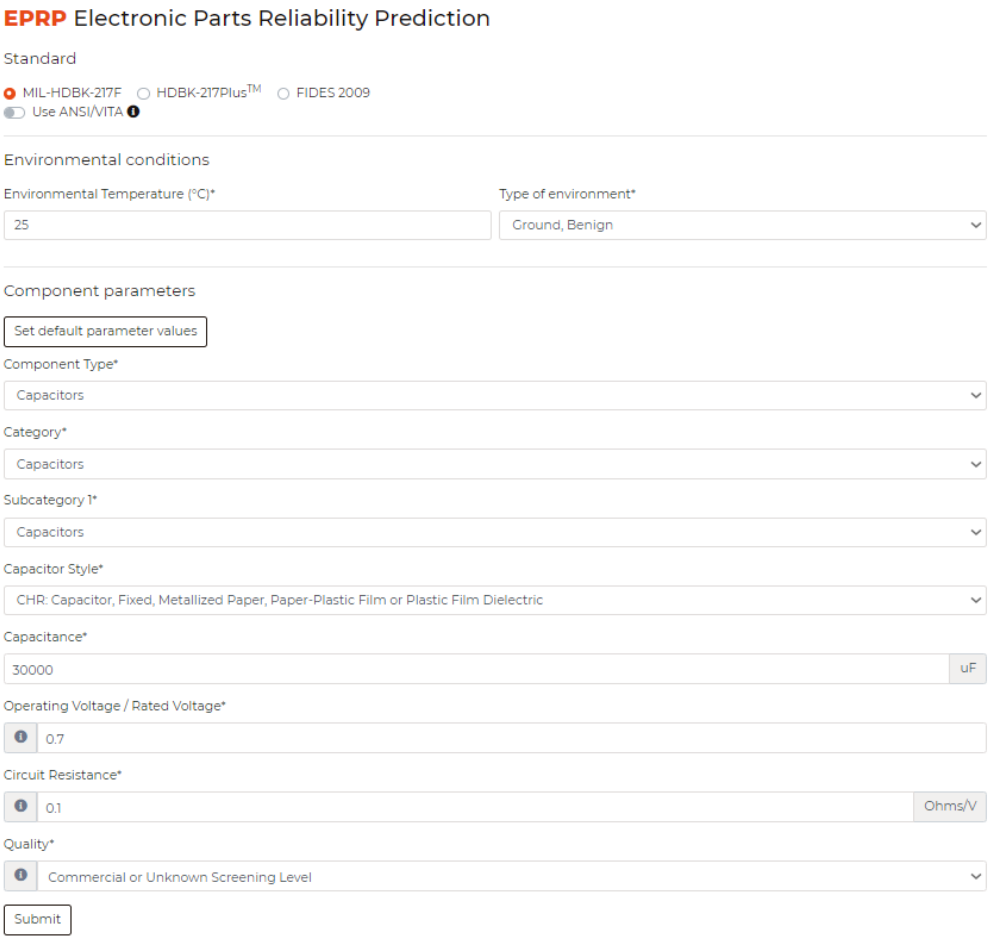


Figure 2.2: Inputs required for the reliability calculation of a capacitor using MIL-HDBK-217F on Robin [2]

The third option is to estimate the failure rate using historical data of similar components. There are several databases, such as the EPRD 2014 or the NPRD-2016, that provide failure rate for all kinds of components based on historical performance. Compared to reliability prediction standards, historical databases offer very little customization, often just the type of component, the quality level and the operating environment. As such, their results tend to be less accurate and more conservative than those calculated with a standard. As an example, the figure 2.4 shows the different options available in NPRD-2016 to obtain the failure rate for a piston. Note that the only inputs here are quality level, environment and confidence level.

The final option and last resort is to estimate the failure of the component with

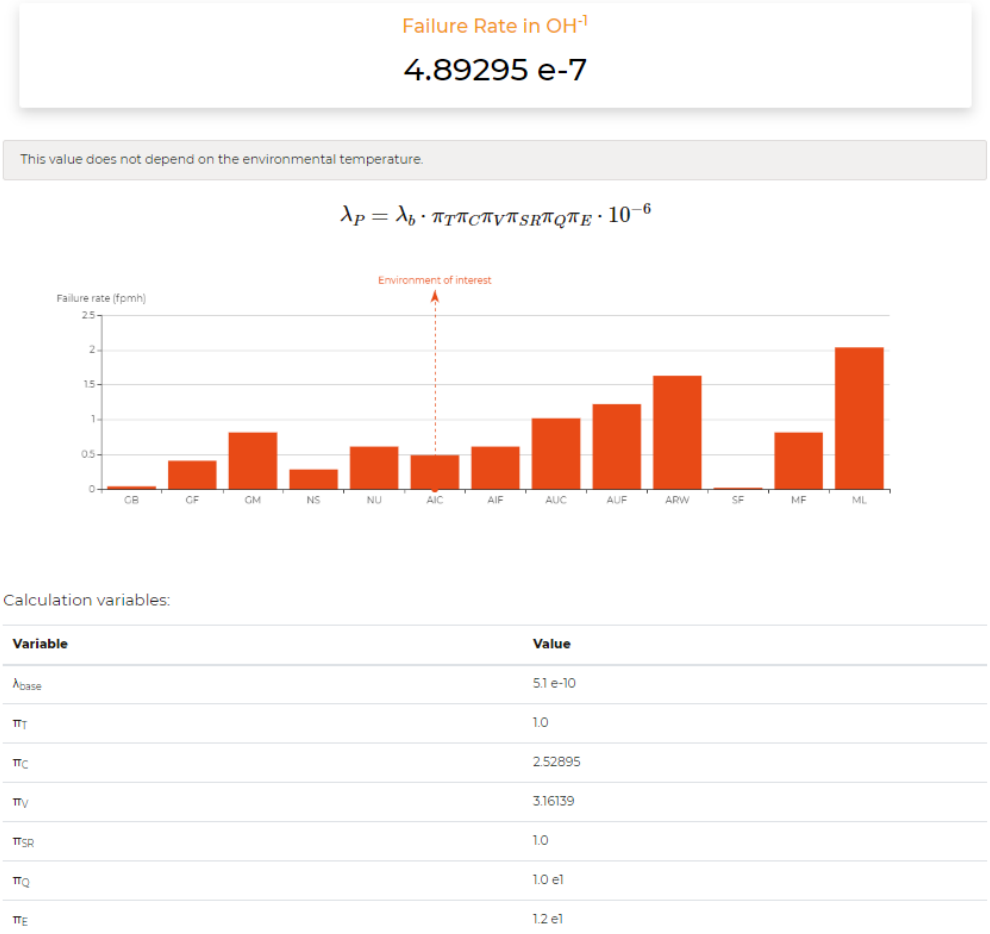


Figure 2.3: Outputs obtained from the reliability calculation of a capacitor using MIL-HDBK-217F on Robin [2]

a reasonable informed guess. This only happens when the supplier does not provide information on the reliability of the product and the component very new or specific that it is not present on the reliability prediction standards nor the historical databases. In such cases, the only option left is to estimate a conservative value based on the general conditions of the component.

When all the different components of the system have been analyzed and their failure rate has been estimated, the reliability of the system can be calculated by adding all the different values. Then, the MTBF can then be easily calculated from the obtained failure rate. It has to be noted that the RPA only estimates the reliability of the product. As such, it provides no information on the effects of the failures, only on their probability during operational conditions.

Part Description	Quality Level	Environment	Failure rate fails per E6 units	Failure rate units	Confidence Level
Bracket,Strut To Piston	Military	AC	0.08707	Hours	
Block,Piston	Military	N	0.845637	Hours	
Connecting Rod,Piston	Military	N	2.764138	Hours	
Casting,Fork,Piston	Military	N	0.422819	Hours	
Cam,Centering,Piston	Military	AC	0.17414	Hours	
Cover Plate,Piston	Military	AC	0.348281	Hours	
Crank,Piston	Military	ARW	4.289581	Hours	
Cylinder And Piston	-	-	0.000011	Cycles	
Cylinder And Piston	Military	-	0.000011	Cycles	
Cylinder And Piston	Military	AUA	0.000011	Cycles	
Cylinder And Piston Assembly,Landing Gear	Military	AUA	0.000005	Cycles	

Figure 2.4: NPRD-2016 options for a piston on Robin [2]

2.1.2 Reliability calculation with FRACAS

After the system or item has entered operational use, the reliability of the product under real world conditions can be calculated. The resulting failure rate should be lower or equal than that which was estimated during the design phase. The calculation is usually done with the Failure Reporting, Analysis and Corrective Action System (FRACAS). This system is used to collect analyze and correct all failures and defects occurring during the production and testing of a product [9]. The figure 2.5 illustrates the general functioning of FRACAS.

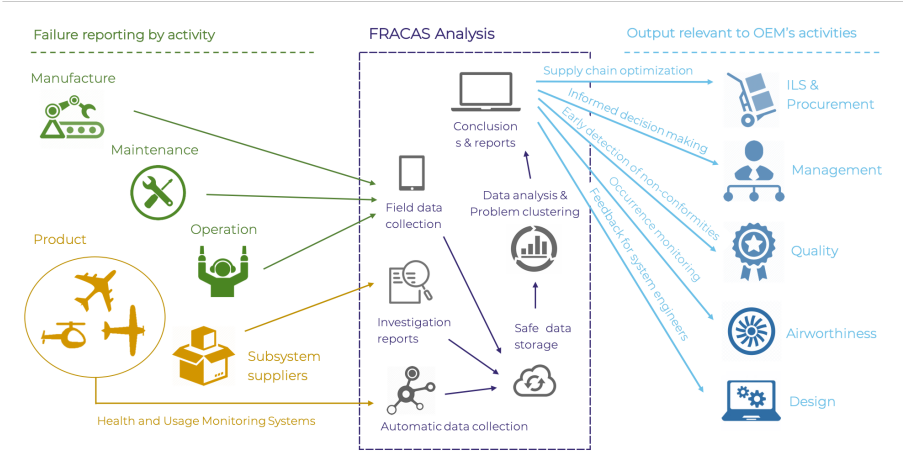


Figure 2.5: FRACAS process by DMD Solutions

To calculate the real reliability of the product, the FRACAS is used to collect all the reported failures and defects that occur during the use of the system. In the field of aviation, two reports are of interest for the FRACAS: the Aircraft Malfunction Report (AMR) and the Defect Investigation Report (DIR).

After each landing, the pilot or crew reports to the maintenance team any defects or failures that the equipment has sustained during the flight. Most of the

times, the maintenance crew present at the airport simply replace the faulty part with a spare part available in the facilities. Sometimes, some testing is done on site to try to replicate the reported issue on ground, however this is not required every time. The different equipment and systems of the aircraft are designed to be easily and quickly replaceable. As such, the aircraft can be put back into action in any kind of facility that has spare parts and in a reasonable amount of time. A description of the failure tat occurred during flight, all the actions that have been done to fix the reported issue and the on-ground testing results of the equipment have to be written on the AMR, alongside all the necessary aircraft and equipment information. It is required that after the maintenance of the aircraft is completed, the AMR is sent to the aircraft company.

The part or system replaced in the aircraft is sent back to its manufacturer for closer inspection and repair. All the following process is done at customer charge if there is no valid warranty. The manufacturer is then tasked with testing the faulty equipment, and most of the times the reported defect can be replicated on the test bench. The source of the issue has to be found with thorough inspection and testing. Then, the part is usually repaired, although some times the item is scrapped due to being beyond economical repair. Following that, the part is submitted to the testing described in the Acceptance Test Procedure (ATP) to ensure that the item is in working condition again. After the ATP is passed, the component is then returned to the airport facilities to be used as a spare part or any other purpose. All the work that has been done on the part and the findings that have emerged during the investigation, have to be written on the DIR. The manufacturer is required to send the DIR to the aircraft company after all the work on the part is completed.

After the DIR is received, a RAMS engineer has to analyze and categorize the failure. Not all failures are the same, since only the failures that have been verified count towards the reliability calculation. The following list displays different main categories of failures. It has to be noted that the notation used may vary from system to system and that there are also other categories that are rarely used.

- Defect Verified (DV): Assigned when the failure has been confirmed.
- Open: Defect not investigated yet. Generally, there is a policy fixed by the company which computes these defects as DV. (e.g. 90% of Open are DV)
- No Fault Found (NFF): Assigned when once the part has been tested, the failure is not detected and the part is fully functioning to specifications
- Discard (DSR): Assigned when the malfunctioned part cannot be repaired and it is directly removed and scrapped without any further analysis.

- Scheduled Maintenance (SCM): Assigned when the part is removed for a programmed maintenance.
- Scheduled Service Bulletin (SBS): Assigned when the part is removed to carry out an modification called Service Bulletin.
- No Defect (ND): Assigned when the defect is not addressable to the unit itself, but to an an external factor which can be environmental (e.g. lighting, bird strike), a man-induced malfunction (e.g. mishandling) or wrong-performed maintenance. Removals due to usage wear of consumables such as tyres are also categorized as ND.

After categorizing the failure, the information of the aircraft, the part and defect subject to investigation are introduced to the FRACAS database. The table 2.1 displays all the required FRACAS input data. If both the AMR and DIR have been received and are correctly completed, there should never be any kind of problem to introduce the failure data into the FRACAS.

Aircraft information	AC type	Aircraft type (Manufacturer, model, etc)
	AC SN	Aircraft Serial Number (Given when manufactured)
	AC Flying Hrs	Current flying hours accumulated by the aircraft when the defect occurred.
Part information	Part Number	Part number that identifies it.
	Part SN	Part serial number given when manufactured.
	ATA Chapter	Identifies where the part is located.
	Part Flying Hrs	Accumulated flying hours of the part when the defect occurred.
Defect information	Defect date	Date of the defect
	Description	Description of how the defect was detected and its consequences
	Defect type	The defects can be categorized. It depends on the company's policy.
	Finding	After the removal for investigation, the DIR is received with the finding. The finding explains the cause of the defect and the repair action applied to the part.

Table 2.1: FRACAS input data

At any point, the aircraft manufacturer can use FRACAS to calculate the reliability of their product. The Mean Time Between Failures indicates the average time between the occurrence of two failures in the product with operating conditions. The figure 2.6 illustrates the time between failures of an aircraft. As explained before, the MTBF is the inverse of the failure rate.

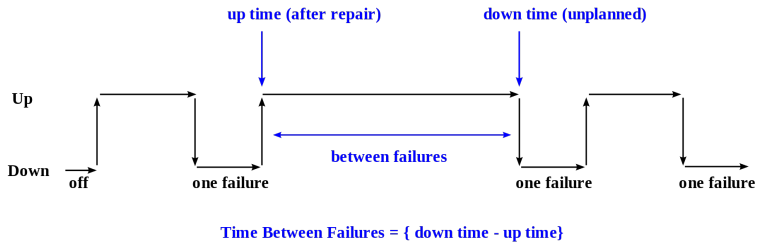


Figure 2.6: Time between failures (Source: [3])

To calculate the MTBF using FRACAS, the equation 2.1.1 is used. Basically, all the operating hours of the aircraft are added and then divided by the total number of failures that have been verified (in the case of the FRACAS database used in this project, these are the categories DV, acrDSR and acrRDV). Also, it is estimated that about 90% of the failures that are currently still open, due to missing documentation or awaiting a response, will be verified. As such, they are also taken into account in the calculation of the MTBF.

$$MTBF = \frac{\mathscr{P}i_{aircraft} FH_{i_{aircraft}}}{\#DV + \#DSR + \#RDV + f \cdot \#Open} \tag{2.1.1}$$

The failure rate of the product can be easily obtained from the MTBF using equation 2.1.2.

$$\lambda = \frac{1}{MTBF} \tag{2.1.2}$$

The resulting MTBF and failure rate calculated using FRACAS is considered to be the real reliability of the component. This value is very important because customers tend to be very interested in this parameter. If a product is more reliable than the competition, the way to prove it is by comparing the failure rate values. Also, by comparing the predicted reliability obtained with the RPA and the calculated reliability obtained with FRACAS, it can be discerned if the component is underperforming or over-performing in the field of reliability.

2.2 Predictive maintenance

Up to this point, it has been shown how to estimate and calculate the reliability of a product using RPA and FRACAS respectively. The resulting MTBF is used in some systems to perform preventive maintenance. For example, if a component is known to regularly fail after a certain time of operation with a high degree of accuracy, a preventive maintenance task can be scheduled at intervals equal to the calculated MTBF. However, for most components, the use of the MTBF to schedule maintenance tends to result in either unnecessary repairs if the value is too conservative or in catastrophic failures if the value is not conservative enough.

Predictive maintenance (PdM) offers a better alternative to preventive maintenance. PdM is a condition-driven preventive maintenance program. Its objective is to obtain better results than preventive maintenance by lowering the amount of unnecessary repairs and the number of catastrophic failures. To do this, instead of using the MTBF to schedule maintenance activities, PdM uses direct monitoring of the mechanical condition, system efficiency, and other indicators to determine the actual mean time to failure [4]. A correctly established PdM program will minimize unscheduled breakdowns and detect failures before they become serious. Since the cost of repair is usually tied to the seriousness of the failure, detecting the failures before becoming serious will reduce maintenance costs. The figure 2.7 illustrates the basics of the predictive maintenance process.

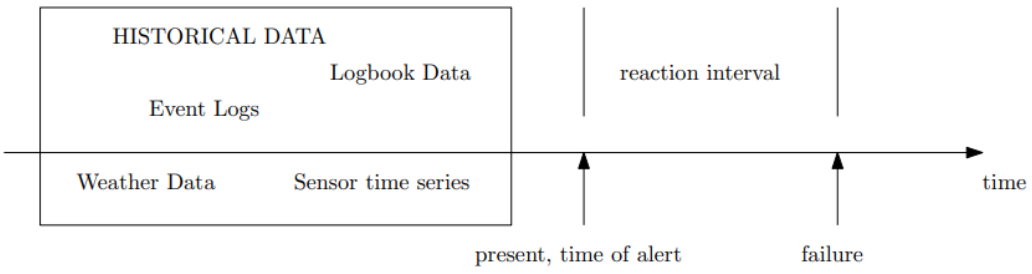


Figure 2.7: Predictive Maintenance process [4]

There are many benefits associated to implementing a PdM system. Overall safety of the product is improved due to components being replaced or repaired before failing. Downtime, that is, time that the system is out of action, is reduced, as well as the size and scale of repairs. The potential exposure to liability is decreased, the same way as the cost associated to needing spare and stand-by units. A PdM system reduces the overall maintenance costs through better use of labor and materials [10].

To successfully create a predictive maintenance system, two things are needed:

an algorithm to train the artificial intelligence using machine learning and a large amount of data to feed the algorithm. The different approaches and issues of machine learning and artificial intelligence are discussed in the section 2.3. Nonetheless, it can be argued that the problems and issues associated with the data needed to train the algorithm have a higher influence on the PdM results than the algorithm used. A model trained with a bad algorithm may take longer time and training data to reach acceptable results, or the results may be poorer than those obtained with an optimized algorithm. However, a model trained with bad data may not produce any kind of passable results at all, no matter how good or optimized is the algorithm.

In the current state of affairs, the following list presents the most common issues related to training data that a project like this could face during its development:

- **Data acquisition**: In most industries, and in the aviation industry in particular, most companies that do extensive monitoring are reluctant to share their product data. This is mainly for two reasons. On the one hand, the data protection policy of each company may make it impossible to share the product data generated by users to third parties. And on the other hand, companies tend to be protective of the data they collected, since they expect an important return for sharing their data. As such, they may be difficult to convince.

- **Data size**: All around the world there are tens of thousands of aircraft that take part in millions of flights each year. Each aircraft may have thousands of sensors recording data each instant. As such, it is obvious to see that the amount of data generated by a fleet of aircrafts quickly becomes very cumbersome and very difficult to work with. When the data is enormous, some filtering and size reduction has to be performed before training the algorithm.

- **Data noise**: When it comes to artificial intelligence, most algorithms are sensitive to noise, that is, uncompleted data. Holes in the database have to be fixed in some way, so as not to interfere with the training of the algorithm. This may prove difficult as data is often gathered using different methodologies, which may lead to some noise.

- **Ineffective data**: Among all the collected data, irrelevant or ineffective data should be filtered out to avoid confusing the algorithm. The algorithm should only be trained with data that may be relevant to predict maintenance in order to diminish the necessary computational workload and, most importantly, avoid unacceptable results caused by feeding the training algorithm with irrelevant or ineffective data.

Although the development of a predictive maintenance system may pose some

difficult obstacles, the potential benefits of its implementation clearly outweigh their potential drawbacks. This is starting to be seen on the aviation industry. For example, SAS (Scandinavian Airlines System), a European airline mainly based on the Nordic countries, reached an agreement with Airbus to become the first customer for the Skywise Predictive Maintenance Solution. Around 70 aircrafts of the A320 family will be subjective to proactive PdM with the goal of preempting operational disruptions and accelerate maintenance decisions [11]. This is a clear example of the future developments that will experience the aviation industry.

2.3 Machine learning

Since the beginning of computer science, the subject of Artificial Intelligence (AI) has gathered much interest and resources. From the early days, problems that can be described using a series of logical and mathematical rules have been easy to solve for computers. However, in order to solve complex problems with no clear logical rules, the computer has to learn from experience in order to understand complicated concepts by building them out of simple ones. The subject of this section is to explain this process and introduce the basic concepts required to follow the rest of the thesis. The source for most of this section is a book considered to be the "bible" of deep learning, a comprehensive guide to deep learning featuring mathematical and conceptual background and techniques used in industry [5].

In order to define the concept of deep learning, first several concepts have to be clarified. These are AI, machine learning, and representation learning. The figure 2.8 displays the relation between the different concepts and it is of great use to follow the next explanations.

The subject of **Artificial Intelligence** can be approached in many different ways. For example, there have been attempts to code knowledge in formal languages in what is know as knowledge bases. However, this approach has been proven to have a lot of limitations. This is because the world is too complex to be described with a set of formal rules. As an example, it is very easy for humans to identify what is and what is not a chair. However, to describe this identification process with logical rules would be very difficult.

A better approach to AI is **machine learning**. Instead of directly coding the world knowledge into the algorithm, the system is left to acquire the knowledge on its own by extracting patterns from data. Machine learning is not necessarily complex, since a simple logistic regression is considered to be a machine learning algorithm. Logistic regressions and similar kinds of simple machine learning present some limitations. These algorithms depend on the representation of the

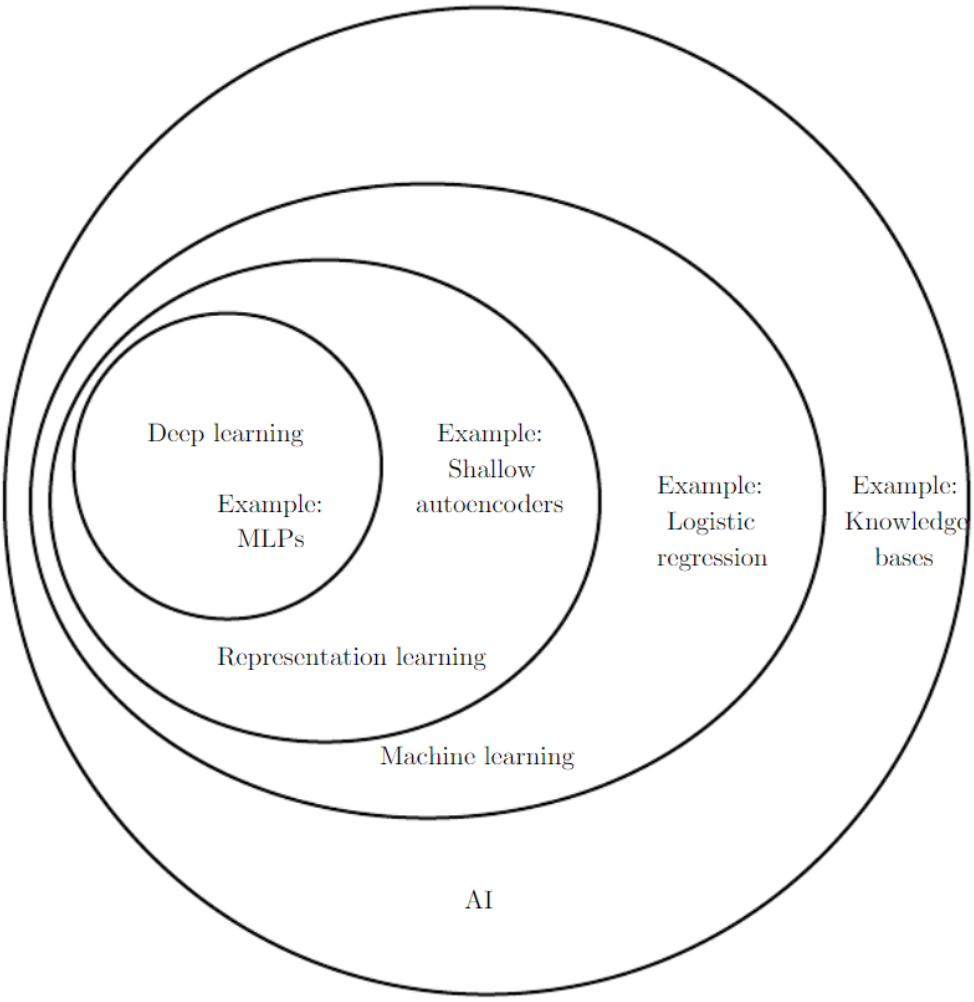


Figure 2.8: Venn diagram of deep learning [5]

input data. This means a lot of times human input is required to process the data. For example, a logistic regression algorithm can be used to identify the optimal velocity of a road given its features. However, it would be impossible for the algorithm to identify said properties on its own.

Representation learning is a more complex approach to machine learning that tries to solve these issues. Instead of only trying to understand the mapping from the representation to the output, representation learning also tries to discover the representation itself. This is done with an autoencoder, which codes the input data into a different representation and decodes the new representation back into the original format. The main problem with this approach lies with the factors of variation. In order to learn the different features of the data, representation learning requires to be fed the different factors that influence the observed data. For example, a representation learning algorithm that tried to identify cars in images would

need to be fed all the different factors that influence the appearance of a car in an image, such as time of day, light, position, and many others.

Finally, **deep learning** solves this problem of representation learning by introducing representations that are expressed in terms of simpler representations. This is done with a deep network or Multi-Layered Perceptron (MLP). Basically, deep learning is fed input data in what is known as the visible layer, which could be the different pixels of an image. A series of hidden layers extract increasingly abstract features from the previous layer, such as the edges, then the corners, and finally the objects. In the end, the last hidden layer is used to create the deep learning output, which in this case would identify the image elements.

In summary, deep learning is an approach to machine learning, which in turn is an approach to AI. Deep learning achieves great power and flexibility by representing abstract concepts using a series of layers of simple concepts.

Chapter 3: Previous Work Summary

As stated before, this thesis builds up and expands on the previous work done by Lovejinder Singh and Josep Lopez as part as a long-term project to create a working PdM tool. This work represents the second step of the creation of a predictive maintenance tool integrated in Robin RAMS. For this reason, this chapter is dedicated to explaining and summarizing the work that has already been accomplished to this point, with the goal of understanding the starting point of this thesis [1].

3.1 Database structure

At first, one of the goals of the project was to use real on-board data recorded during flights, which would have to be obtained from an aircraft manufacturer. However, this resulted impossible to accomplish, and was left as potential future development to be accomplished by the next steps of the long-term project.

With this setback, an alternative approach was to be implemented. Instead of gathering on-board data, the database used to train the algorithm would be constructed from three different databases. These were the following:

- **FRACAS** database, which contained all the defects and failures of the fleet of an aircraft manufacturer.
- **FLIGHT** database, which contained the routes followed by all the aircrafts.
- **METEO** database, which contained all the environmental and external flight conditions.

This three databases would then be used to create a unified database following the input requirements of the Artificial Neural Network (ANN) in order to train the model. The resulting database would then be subdivided into three different databases, each in charge of training the model, validating the progress, or testing the results of the algorithm. The figure 3.1 displays a diagram of the database structure.

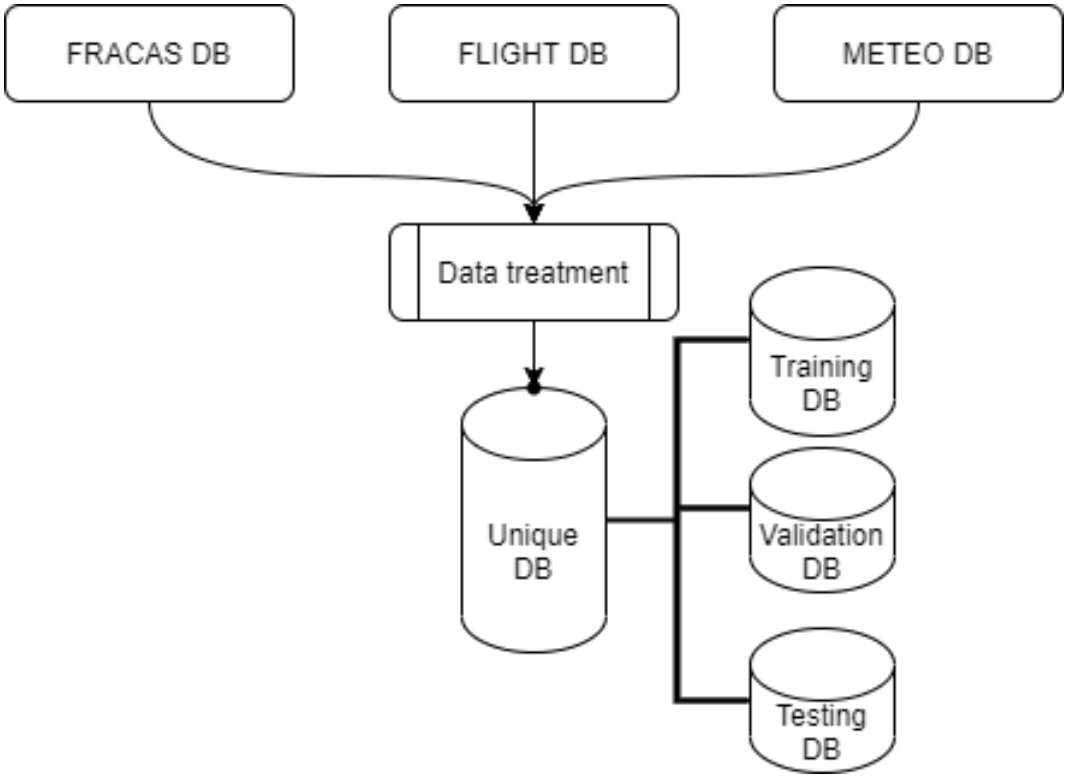


Figure 3.1: Database diagram

3.2 Data gathering

With the database structure explained on the previous section, the next step was to gather or collect all the relevant data. Thus, the goal of this section is to explain all the data gathering process and how that data was processed in order to build the different databases.

3.2.1 FRACAS database

The FRACAS database was created from the data of the FRACAS of an aircraft manufacturer. As explained on section 2.1.2, the FRACAS contains all the recorded defects and failures sustained by the entire fleet, as well as the monthly operational hours of each aircraft. After processing the data, three different tables were created for the FRACAS database, which were the following:

- **Aircraft table**: a list of all the aircraft of the manufacturer's fleet, with their type, serial number, registration born date and, most importantly, the ICAO 24 code. The total number of aircraft was 18.
- **Flight hours table**: for each month, this table contained all accumulated flight hours and landings of each aircraft of the aforementioned fleet. There was

some noise in the data, which was fixed with interpolation. This table had a length of $1,912$.

- **Defects table**: this table contained all the recorded defects that the fleet sustained. For each defect, the table stored the date of the defect and the aircraft involved in the failure. A total of 893 defects were recorded.

All the data used to create these three tables was extracted from a Microsoft Access application in a Comma Separated Values (CSV) format.

3.2.2 Flight database

At first, it was tried to obtain the flight data from FlighRadar24, a flight tracking service that provides real-time information about thousands of aircraft around the world. However, this proved unsuccessful. As such, the database was to be built with the second option, OpenSky Network, a non profit association that provides open access to flight control data. A library that provides a Python interface to historical flight data from OpenSky Network called pyOpenSky was used to collect the data. However, this library had a major inconvenience. It only provided historical data from 2016 onwards. For this reason, the database would have to be built with historical data from 2016 to 2021.

To build the database, OpenSky was used to obtain the flight data of all 18 aircraft previously mentioned. Using the aircraft table of the FRACAS database, a list of ICAO24 codes was obtained. These were the aircraft that were relevant for the creation of the flight database. As such, OpenSky was used to retrieve the aircraft (ICAO24 code), the coordinates(latitude and longitude), velocity (over ground), altitude (barometric and GNSS), heading (track angle), vertical speed and position (on-ground or on-air). This was done for a number of instants of all flights of the aircraft fleet.

All this data was downloaded over the course of roughly 20 days. Out of the studied 18 aircraft, this data gathering process generated a database containing $700,000$ entries, with each entry representing an instant of flight.

3.2.3 Meteorological database

As explained above, the flight database did not contain any data regarding external conditions of the aircraft during the time of flight. As such, the objective of the meteorological database was to obtain the external conditions associated with the flight paths of the aircraft fleet. The locations and dates stored on the flight database were to be cross referenced with historical meteorological data to obtain the external conditions that the flight endured.

To achieve this goal, the Meteostat Python library was used. Meteostat provides a simple API for accessing weather and climate data, where historical observation and statistics are collected from different public interfaces. By using the flight database, the exact location of the aircraft at any point of time during the flight could be obtained. Then, Meteostat would be used to retrieve the measurements of the three closest stations to the aircraft. Using their measurements, the values for the aircraft location would be estimated using the weighted sum of the triangulated stations.

The meteorological data obtained with Meteostat consists of the following fields: air temperature, dew point, relative humidity, one hour precipitation, snow depth, average wind direction and speed, peak wind gust, average sea-level air pressure, one hour sunshine, and weather condition code. This data was downloaded over the course of roughly 20 days and contained information about the studied 18 aircraft.

3.3 Data processing

After creating the three different databases, the resulting data was to be processed in order to create the definitive database. The figure 3.2 shows the relations between the different databases, as it has been explained in the previous sections. To reduce processing load, decimal values were rounded to the third digit. And to avoid issues, rows with Not a Number (NaN) in any field were deleted.

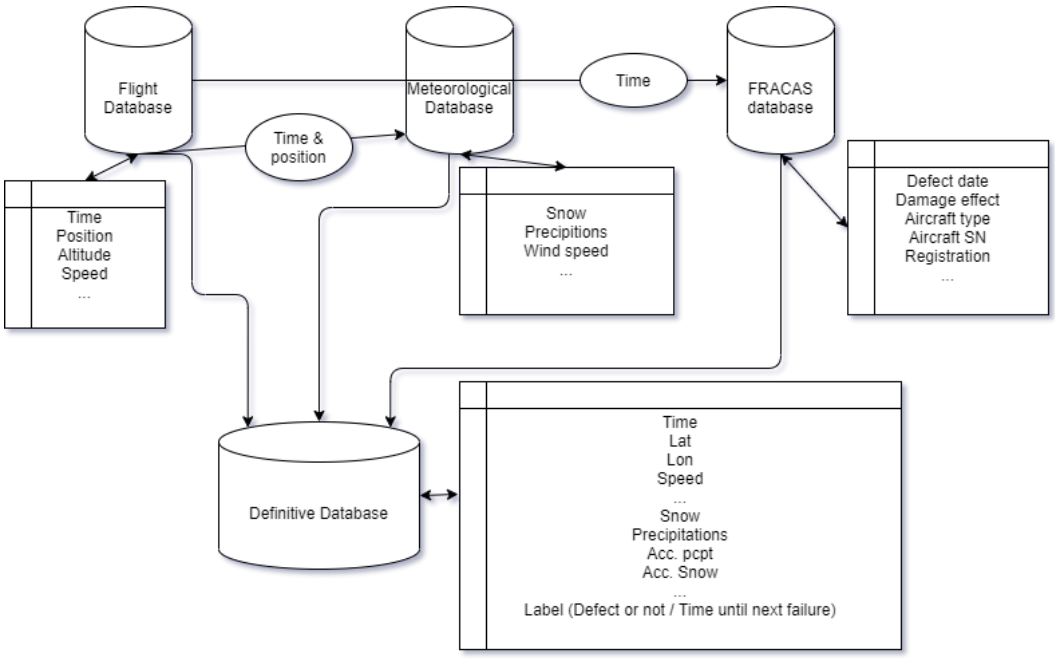


Figure 3.2: Dataflow between the three database sources [1]

The definitive database of the project contained 23 columns. These can be seen on table 3.1. The 23 columns can be divided in three types of data. Firstly, the column **icao** consisted of the aircraft identifier, which was transformed from an alphanumerical field to an integer ("aca1d6" to "2", "a42653" to "3", etc.). Secondly, there were the 21 different fields that contained the data needed to train the algorithm. And thirdly, the column **deffect** contained the defect label, that is, what the algorithm is tasked with predicting based on the input data.

time	Current time (of each state)
icao	Unique identification for each aircraft.
lat	Latitude coordinate of the aircraft
lon	Longitude coordinate of the aircraft
onground	Indicates if the aircraft is on ground (onground=1) or not (onground=0)
baroaltitude	Altitude measured by the barometer [m]
geoaltitude	Altitude provided by the GNSS (GPS receiver) [m]
velocity	Current speed of the aircraft [m/s]
heading	Direction of movement of the aircraft [º]
vertspeed	Vertical speed of the aircraft. Ascending (vertspeed>0) or Descending (vertspeed<0) [m/s]
temp	Current temperature [ºC]
dwpt	Dew point temperature [ºC]
rhum	Relative humidity (%)
prcp	One hours precipitation total [mm]
snow	Snow depth [mm]
wdir	Average wind direction [º]
wspd	Average wind speed [km/h]
wpgt	Peak wind gust [km/h]
pres	Average sea-level air pressure [hPa]
tsun	One hour sunshine total [min]
coco	Weather condition code
flyhours	Accumulated flying hours until the current time
deffect	Defect label

Table 3.1: Definitive database columns

Two kinds of defect labels were created, which were named **binary** and **interval**. Binary labels contained a single digit, either 0 or 1, indicating whether there

was a reported defect in FRACAS on the same day as the flight. On the other hand, interval labels contained an integer that could go from 0 to 10 indicating the number of fly hours to the next failure in the aircraft. A 0 indicated that a failure was imminent, while a 10 indicated that there were more than 900 hours until the next failure. Each integer in between represented a 100 hour interval.

3.4 Algorithm implementation

The implementation of the algorithm was performed using **Python** as the programming language. A part from the usual generic libraries, the code used the machine learning platform **TensorFlow**. Running on top of that library, **Keras** was used as a deep learning API. Keras was used for the implementation due to its simplicity and fast experimentation.

It was decided to use the **Binary Cross-entropy** loss function since it was the function with the better performance. Out of the different options, **ADAM** was found to be the best optimizer because of its speed and performance. The classifying algorithm consisted of a MLP (multi-layered perceptron), with added **1-D convolutional** and **LSTM** (Long Short-Term Memory) layers.

One problem with the data that was found was that the two categories were unbalanced, since the total number of entries of corresponding with defects was much smaller than those corresponding with no defects. To solve this issue, the methods of Adaptive Synthetic (ADASYN) and Synthetic Minority Over-sampling Technique (SMOTE) were used to over-sample the dataset, that is, to create more defect entries in order to have a better balance.

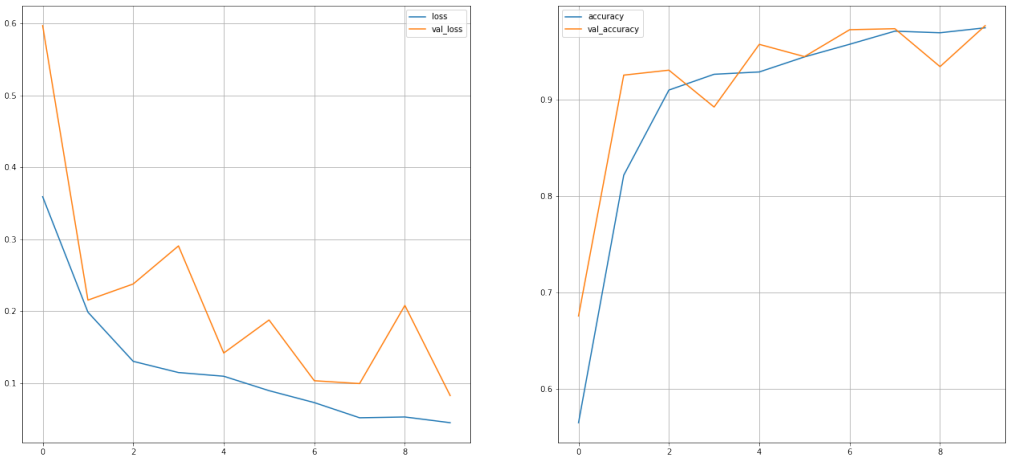
Another tool used in the implementation was **Scikit-learn**, a library centered on machine learning. This was used to randomly split the data into the training and testing datasets, and to perform a standardization of the datasets, that is to transform the variables to a Gaussian distribution with zero mean and unit variance.

On top of that, **Scikit-learn** also provided the functions needed to both train the algorithm and perform predictions using the trained model. And finally, this library was also used to obtain metrics on the overall performance of the algorithm, such as obtaining the performance indicators (precision, recall, accuracy) and plotting the confusion matrix.

3.5 Algorithm results

The first results obtained by the algorithm were the evolution of the loss and accuracy values over epochs (fig 3.3). These two figures show that the algorithm quickly

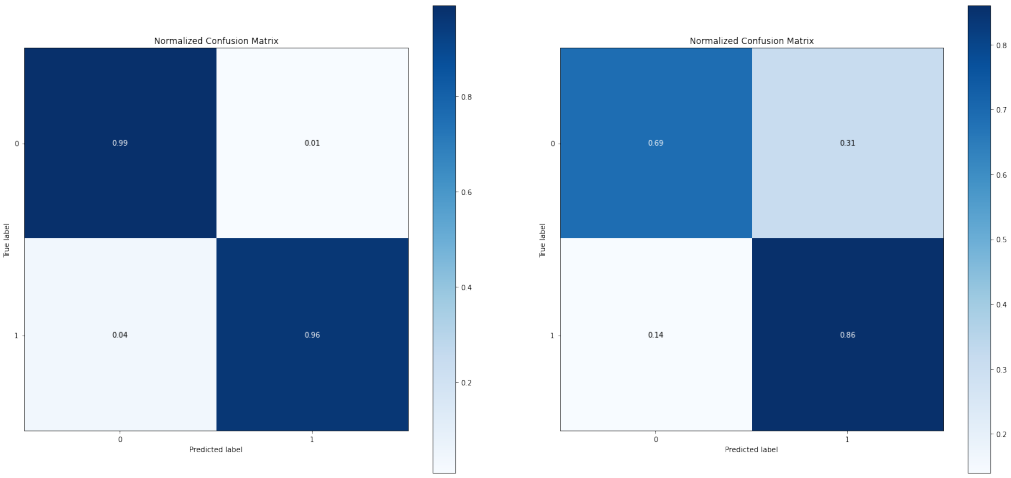
converges, since with very few epochs the algorithm is able to considerable reduce the loss function value and improve the accuracy of the model. According to the report of the previous project, these figures also show that "there are no signs of over-fitting or under-fitting".



(a) Training vs Validation Loss *(b) Training vs Validation Accuracy*

Figure 3.3: Training vs Validation evolution [1]

The figure 3.4 shows the normalized confusion matrix of the validation data (left) and the test data (right). On one hand, the validation confusion matrix shows that the algorithm managed to converge very effectively. On the other hand, the test confusion matrix shows that the algorithm was able to train a very accurate model, capable of predicting most failures with a high degree of precision.



(a) Validation Confusion Matrix *(b) Test Confusion Matrix*

Figure 3.4: Validation and Test confusion matrix [1]

The overall great algorithm performance is also corroborated by looking at the performance indicators shown in table 3.2. These indicators show an incredible

performance of the trained model, which is capable of predicting most defects according to the report of the previous project.

Performance Indicators	Test (%)
Accuracy	67.724
Precision	98.968
Recall	79.991
F-score	78.627

Table 3.2: ANN performance [1]

Finally, the previous project also performed a comparison between the different classifying methods. This can be seen on figure 3.5. Contrary to what could be expected, the best results where not obtained using with the ANN but with the **Random Forest** and the **AdaBoost Classifier** methods. No further explanation is given on the report.

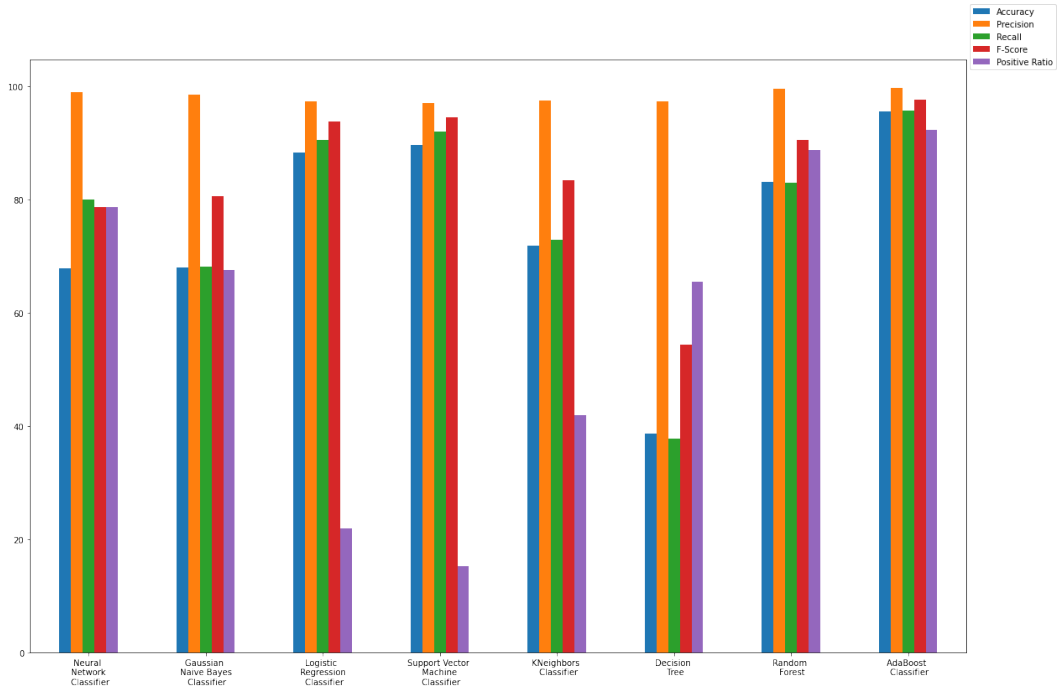


Figure 3.5: Result comparison between different methods [1]

In conclusion, from the results shown in the previous project report, it can be stated that the algorithm provides excellent results. A high degree of accuracy in predicting defects has been shown.

Chapter 4: Interface implementation

The previous project focused on creating an algorithm that was capable of predicting aircraft failures. Once this is successfully accomplished, the obvious next step is creating usable interface. This chapter explains the work on this matter and presents the first version of the Robin module for Predictive Maintenance.

4.1 Justification of interface implementation

At first, developing the interface of the predictive maintenance algorithm may seem to be of little importance. It could be said that the important and difficult part of the project is creating an algorithm capable of detecting upcoming defects in aircraft previous to their flight, and that the implementation of the interface is a secondary and very simple part of the project. However, this is far from the truth, and this section will put forward the reasons of why this part of the project is not as simple as it seems and is of considerable importance.

4.1.1 Interface implementation complexity

To any person uninitiated in programming in general and the programming of interfaces in particular, a decent interface is taken as something that is taken for granted. If the interface works perfectly the user does not pay much attention to it, in fact, it tends to go unnoticed. However, when the interface has some problems or issues, the user is quick to notice and complain. Moreover, instead of thinking that interface implementation is a complex matter that can be difficult to solve adequately, users tend to think of a malicious intentions or an inept implementation behind the failed interface.

The fact is that implementing a good user interface can be very complex and difficult. Many hours of programming and design have to be poured onto a project in order to obtain an acceptable result. The user can interact with the interface in an almost infinite number of ways, and the interface has to be robust enough to be able to deal with each and every one of the user actions. Also, there is always a fine balance on the amount of information provided to the user, since the design has to find away to relay all the relevant information to the user without overwhelming them with too much data.

4.1.2 Interface implementation importance

Not only is the implementation of the interface complex enough, but it is also quite important in the overall scope of the project. Obviously, the predictive maintenance tool will not be commercialized until the interface is finished. But even before that point in the development process, the interface implementation can be very useful to the project for the following reasons:

- **Data acquisition**: The main difficulty of any machine learning project is obtaining the required data. For this reason, the data acquisition process is of high relevance. The implementation of a good interface can considerably improve the prospects of this matter. Potential clients might be more willing to reach an agreement to share their data if they see a usable working interface rather than some unintelligible lines of code.
- **User perspective**: Implementing the first version of a user interface can provide some insight in the user perspective, that is, how could future users use the tool. For example, one could obtain info about how do users interact with data, how is the process of predicting defects from the user point of view, and how could this be improved.
- **Process improvement**: Last but not least, if the implementation is really successful, the interface could severely improve the process of data processing, data analysis, model training and model analysis. Before the implementation, this had to be done using raw code. A good interface can considerably reduce the amount of work needed to be inputted to obtain results.

4.2 General considerations

Before starting to explain the main part of this chapter, some considerations have to be put forward. A handful of things have to be put into context so that things are clear and well situated when they are shown. These are shown in the list below.

- First of all, the user interface is created as an integral part of the Robin RAMS tool. This project was born under the umbrella of DMD Solutions, and as such, the intention is to commercialize it as a new module in the Robin RAMS tool. The marketing part of this part of the tool is yet to be formulated.
- As part of Robin, the implementation of the interface has been completely developed using the Django environment. Plenty of open documentation about Django exists for those unfamiliarized with the environment [12].

- The interface shown in this chapter is intended to be a first version of the final product. This means that this will form the basis for the end product but that more iterations and polish are required to reach that point.
- The algorithm used to train the defect models is that which was obtained in the previous project. The implementation of the interface and the development of a new better algorithm were parallel processes, and as such, it was unwise to use the latest algorithm version.
- As of present time, no effort has gone yet into the server deployment of the user interface. All work has been done locally, and it is left for the future to upload the new Robin module to the server and solve any issues that may arise.
- Finally, it has to be understood that there are some limitations due to code confidentiality. Robin RAMS is the flagship product of DMD Solutions, and as such, the company is interested in limiting the amount of code that may end up in the public domain.

4.3 Preliminary design

Before one starts programming the user interface, a considerable amount of work has to go into creating a preliminary design. Little coding can be performed without a clear reference of what is to be achieved. Obviously, the final design will more than probably be different from the preliminary design, since many obstacles and opportunities can be found during the programming process. But in the end, the preliminary design will provide the basic foundations of the final product.

4.3.1 Intended approach

The interface framework can be understood as the ideas that tie everything together. During the design and implementation of the user interface many little and seemingly independent decisions are made. All these decisions are made with a same set of ideas in mind. Each decision has different context and outcome, but they all share a basic set of principles or approach:

- User focus
- Simplicity
- Clarity
- Consistency

- Streamline actions

4.3.2 Target capabilities

Beforehand, the set of target capabilities were set, in other words, what would the user be able to do with the interface. These are shown below. It has to noted that each capability tends to correspond with a distinct page.

- Organize work in projects
- Upload and manage aircraft datasets
- Analyze each dataset
- Train and manage defect models
- Analyze each model
- Make defect predictions

4.3.3 Page distribution

After outlying the different capabilities that the interface should have, these have to be distributed in the different pages. This was done by basically assigning each capability to a different page. This way, the user would have the interface very structured and clear, without much interference between the different capabilities.

Once this is done, each the distribution of each page has to be somewhat thought and sketched. By drawing each page in a very simple manner, one can obtain a very clear idea in their head of how each feature will be implemented and where each part of the page should go. This step is by no means necessary, but if the interface implementation is to be successful it is a must.

4.4 Database structure

In the Django environment, the project data is organized using a set of models. It is unfortunate that the very same word is used to define so different concepts such as Django models and classification models. In order to avoid confusion, the prefix of "Django" is usually added to the word model where it's deemed necessary.

In the context of Django, a model refers to a Python class that contains the essential fields and behaviours of the data being stored. Each model is a subclass of a generic Django model, which is personalized for the data requirements of the project. Each attribute of the model represents a database field. All models of this project are saved on the project database, but this is not required by default. The

structure of the Django models, the relationships between them and how do they interact with one another represents the foundations of the Django environment.

The figure 4.1 displays a schematic showing the structure of the project database. This structure of the model relationships was entirely derived from the preliminary design that was made at the beginning of the interface implementation. Although it is possible that this could change in the future, the model relationships represents the most solid part of the implementation of the interface.

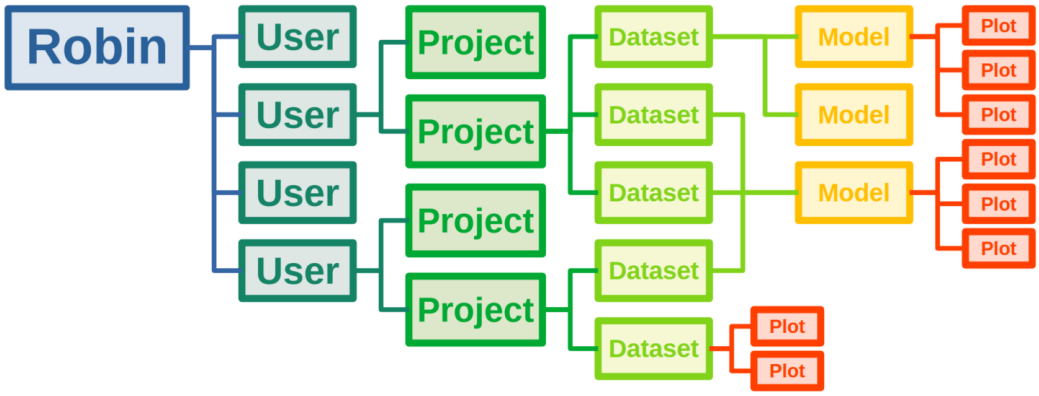


Figure 4.1: Database structure schematic

First of all, on the leftmost part of the schematic 4.1, Robin represents the umbrella of the project. There is no model named "Robin", instead, Robin is the project itself. Robin can be hosted locally or in the server, and each "hosting" of the project acts totally independently. Robin has multiple modules, and this interface implementation represents the newest added module.

Each project contains multiple users. The user model is a generic class that had already been developed for Robin when the interface implementation started. It is intended that each person that works with Robin RAMS has their own unique user. However, account sharing is not expressively forbidden and there it is something not uncommon in Robin.

Each user can create a number of projects. This is done to organize the work done in Robin in the way that the user wants. At the beginning, each project is only related to a single user. However, the capability of sharing projects between users is a feature that will not be very difficult since it has already been implemented in other modules.

Each project can contain multiple datasets. Each dataset contains a table with the data that would be used to train a model. The idea is that users upload new datasets to their projects as the data is being generated. This dataset model would serve as a way to both storage and organize all the different training data.

Each project also can contain multiple models, not to be confused with Django

models. Every model can be created using different datasets, and each dataset can be used to train different models. It is intended that each user can train models with their own data and that there exists a set of generic models trained by DMD Solutions available to all users.

Finally, each dataset and model have different associated plots. These plots are unique to each instance and provide an important insight on both data and models. They are intended to be used to take decisions such as what dataset to use or not use, and which model to choose over the others to make predictions.

4.5 Interface display

Although there are still some placeholders, the overall interface display is pretty much finished. All the different pages have been implemented following what was previously explained.

4.5.1 Home page

The first page of the module is the home page (fig 4.2), which contains the different projects used to organize the work and some various statistics and numbers.

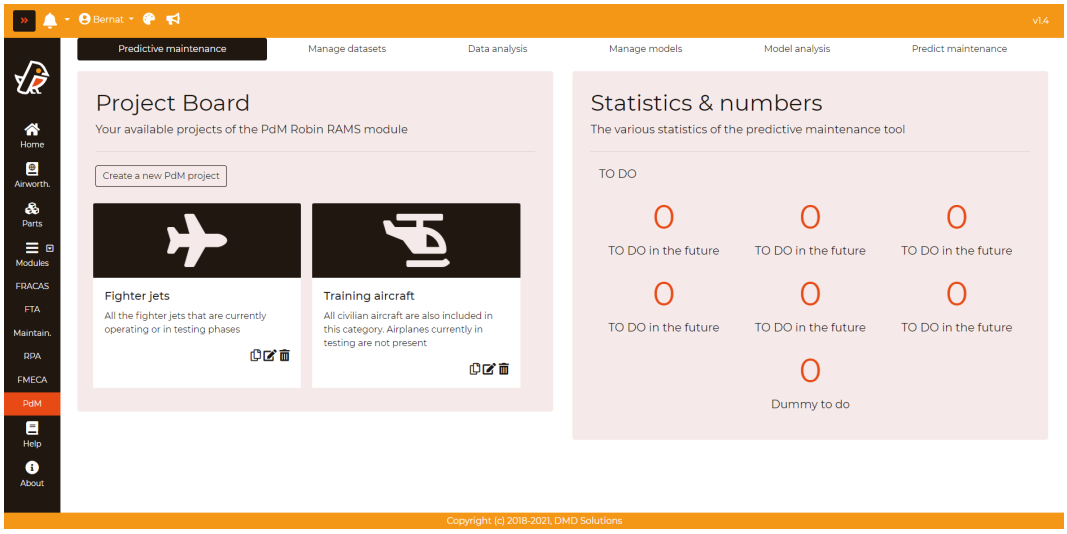


Figure 4.2: Home page of PdM module in Robin

4.5.2 Datasets page

When entering a project, the first page that pops up is the manage datasets page (fig 4.3). This page displays all the datasets that are present on the database and allows the user to manage them.

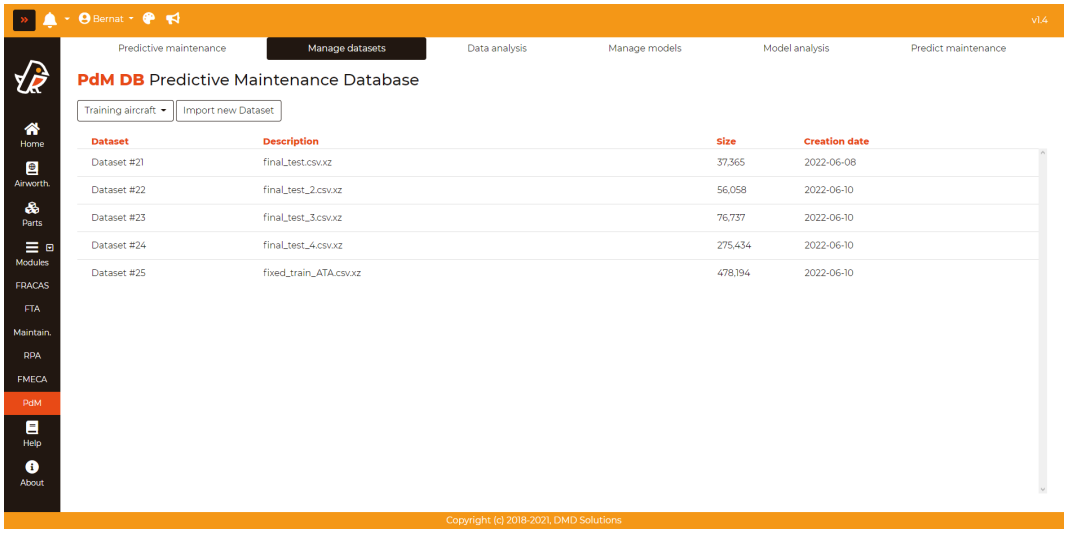


Figure 4.3: Manage datasets page of PdM module in Robin

From the previous page, the user can import new datasets. This is done using a special page (fig 4.4) that allows to easily upload CSV files through the interface. The input required by the user is heavily reduced with the use of the "autocomplete", which fills the columns with the most appropriate options without any need of indication. Obviously, this can also be manually be entered by the user.

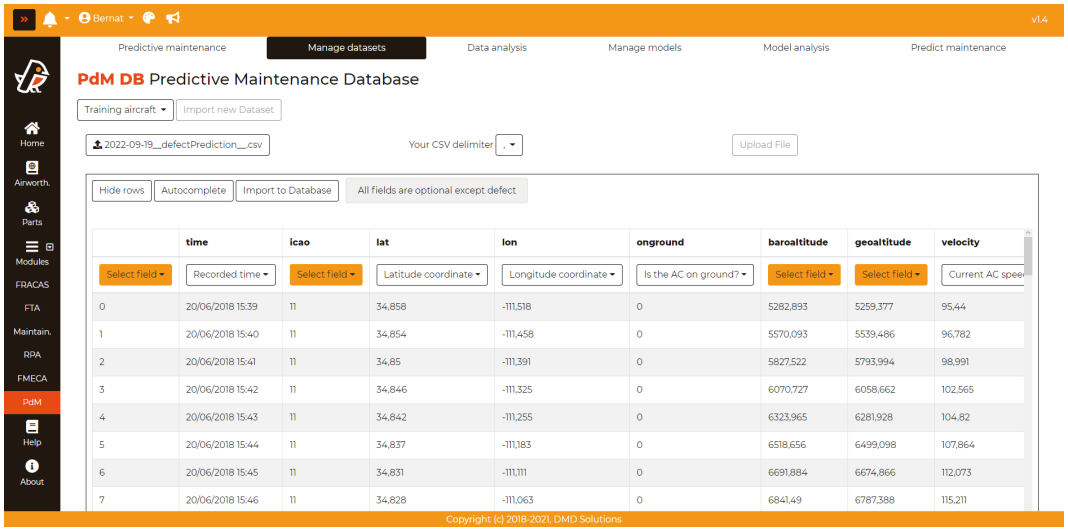


Figure 4.4: Import datasets page of PdM module in Robin

4.5.3 Data analysis page

With all the different datasets, the next page (fig 4.5) allows the user to perform a basic analysis of the data contained in those datasets present on the database. Basically, several graphs and statistics are displayed on the page, and the user is

left to interpret those results and act on those conclusions. This could be useful to learn about which datasets could be more useful to train the algorithm.

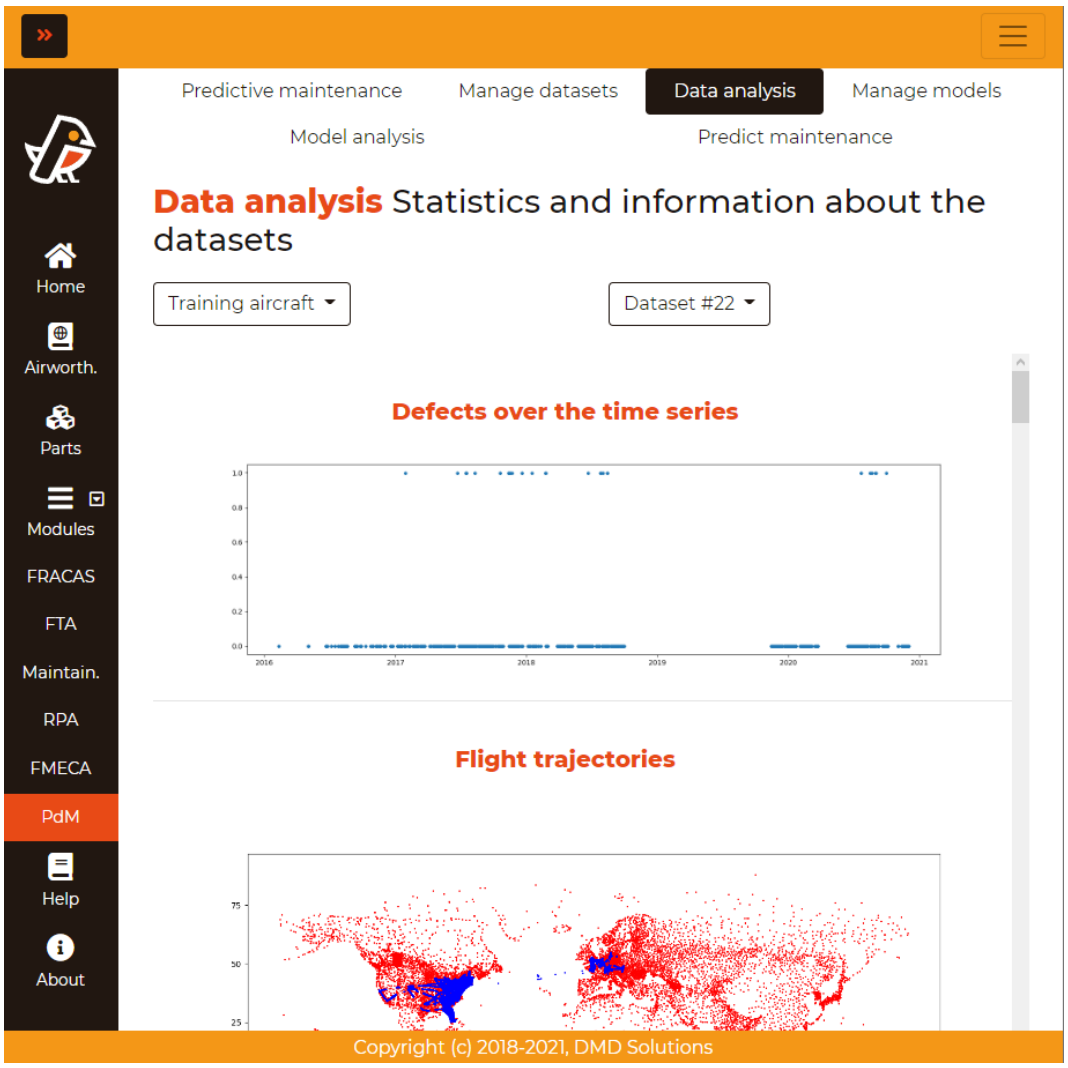


Figure 4.5: Data analysis page of PdM module in Robin

4.5.4 Models page

Similarly to the datasets page, the manage models page (fig 4.6) displays the different models and allows the user to interact with them. It is worth noting that there are a set of generic models that are provided by Robin to all users. These have been trained using different kinds of varied data and should be able to provide a decent performance for most cases. However, the user is also able to create and train new models. The intention is that these would be less generic, therefore they could provide a better performance when used in the specific kinds of aircraft they were trained in. These are not shared between users.

From the previous page, the user is able to create new models using the train

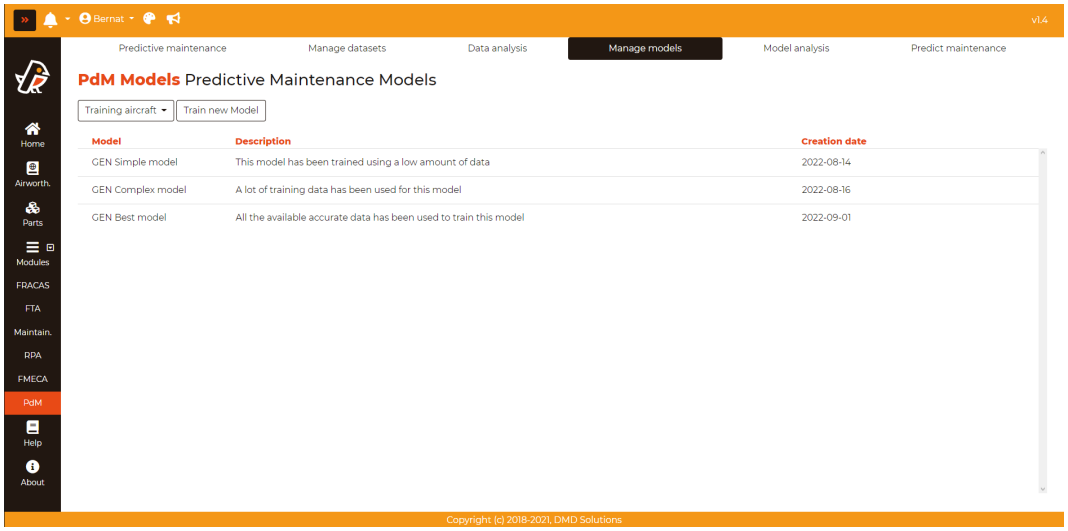


Figure 4.6: Manage models page of PdM module in Robin

models page (fig 4.7). The user is able to select the different datasets that they wish to use to train the algorithm from the list of datasets. Some settings will be available to the user in order to tune the training of the algorithm.

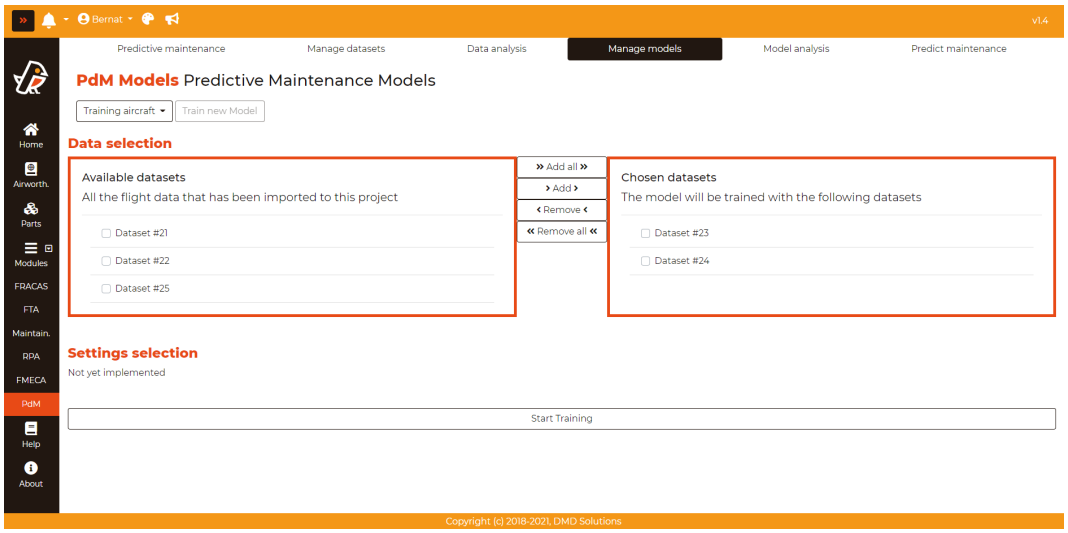


Figure 4.7: Train models page of PdM module in Robin

4.5.5 Model analysis page

In a similar way to the datasets, the different models created by the user and the generic provided by Robin can be analyzed in the model analysis page (fig 4.8). In this page different plots and figures are shown for each model, providing valuable information to the user about the performance of the models.

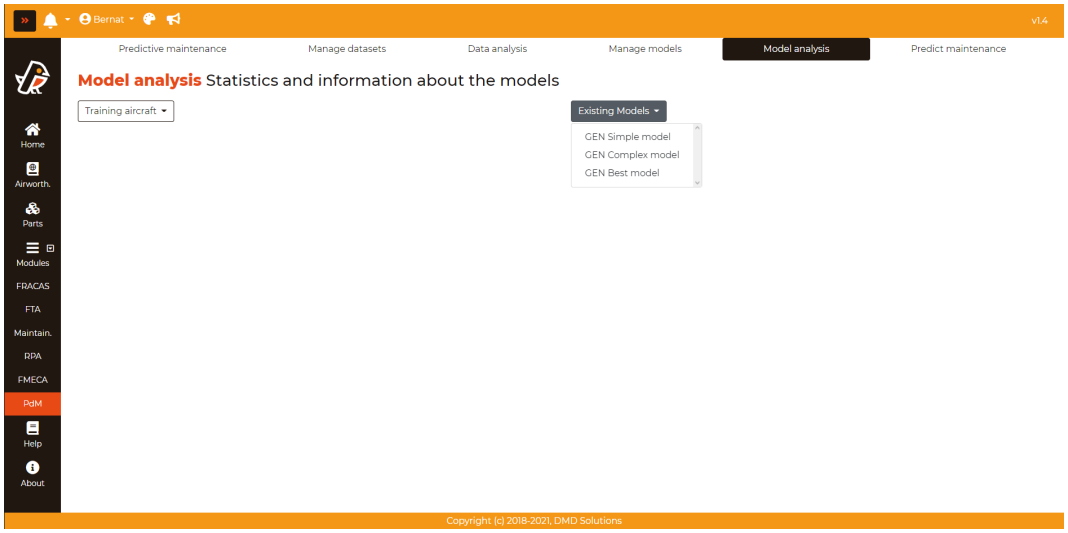


Figure 4.8: Model analysis page of PdM module in Robin

4.5.6 Failure prediction page

Finally, the last page (fig 4.9) is used to make failure predictions. Basically, the user uploads a dataset without labels, Robin uses the selected model to predict the labels of the dataset, and it is downloaded by the user. It has to be noted that the import page is similar to that previously shown since they perform a very similar function.

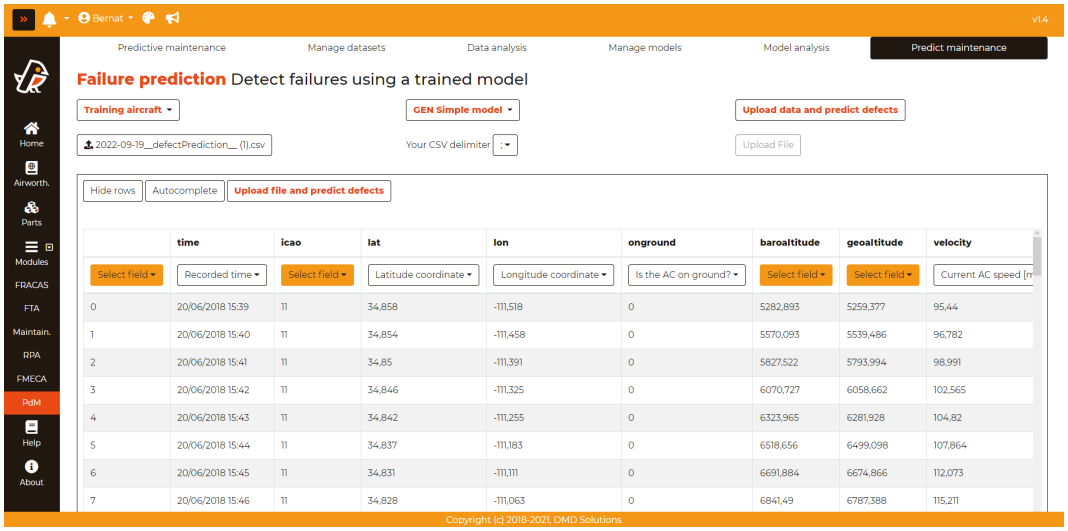


Figure 4.9: Predict maintenance page of PdM module in Robin

4.6 Summary of the Robin implementation

In conclusion, the interface of the prediction maintenance tool has been successfully implemented with a first version on Robin as a new separate module. The implementation was designed to be as robust as possible and to easily allow changes on the deep learning algorithm. It would not be difficult to change the predictive maintenance classifier part of the implementation. It has to be admitted that there is still more work needed and many improvements can still be implemented, however this first version of the interface represents a big step forward in the completion of the project.

Chapter 5: Analysis of previous work

From the beginning of this work, the goal of the project was set to build upon the previous work done, by expanding and improving the methods and results obtained at an earlier date. However, during the development of this project several issues with the approach and methods have been found. The goal of this chapter is to explain these issues and to analyze certain decisions taken in the previous project, always keeping in mind that a big portion of the previous work has been and will be very useful.

5.1 The problem with previous results

The results shown on the report of the previous project [1] are nothing short of impressive. These results alone should prove beyond any reasonable doubt that the project successfully achieved its goals. In some aspects, it could even be argued that the results obtained are beyond expectations.

5.1.1 Results as shown on the report

The results of the previous projects are shown and discussed in section 3.5. Here is presented a brief summary. As seen on table 5.1, the ANN achieved high values of accuracy, recall, and above all, precision. These outstanding results can also be visualized in the confusion matrix 5.1.

Performance Indicators	**Test (%)**
Accuracy	67.724
Precision	98.968
Recall	79.991
F-score	78.627

Table 5.1: Previous ANN performance [1]

5.1.2 Replication of the results

In order to replicate the results, the very same code used to train the previous algorithm was used without changing a single comma. The same can be said for the data, which was not changed and remained as it was. Also, the very same

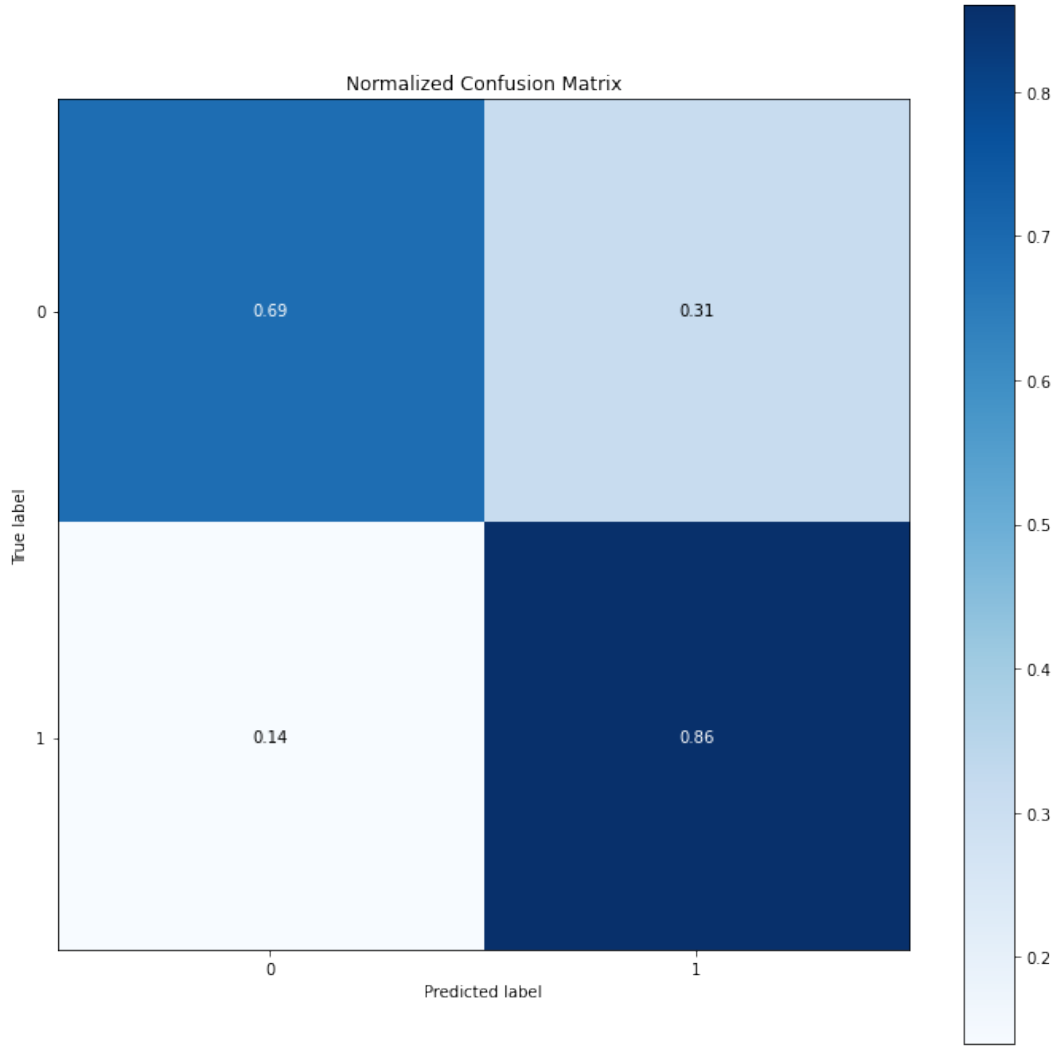


Figure 5.1: Previous confusion matrix [1]

software tool was used to run these tests to replicate the output shown in the report. The only noticeable difference is that the hardware was different, as the graphics card was upgraded. The hardware difference should not matter to a big extent. Its effects would be negligible at best, and barely noticeable at worst. There is also a randomized factor. Every time that the code is run, different results will be obtained based on luck. However, overall, these differences are very small. And by executing several times the same code a high degree of confidence can be achieved.

For these reasons, it would be expected that very similar results could be easily replicated. The code was executed several times in two different hardware systems in order to minimize the randomness factor. However, every time that the code was executed a very similar outcome was obtained: the results were completely off. Out of the many executions on either system, the results came nowhere close to those shown in the report of the previous work.

5.1.3 Replicated results

The replicated results of the previous projects are shown below. As seen on table 5.2, the ANN achieved nowhere near the high values of accuracy, recall, or precision of the previous project. These results can also be visualized in the confusion matrix 5.2.

Performance Indicators	Test (%)
Accuracy	30.450
Precision	50.214
Recall	51.482
F-score	50.840

Table 5.2: Replicated ANN performance

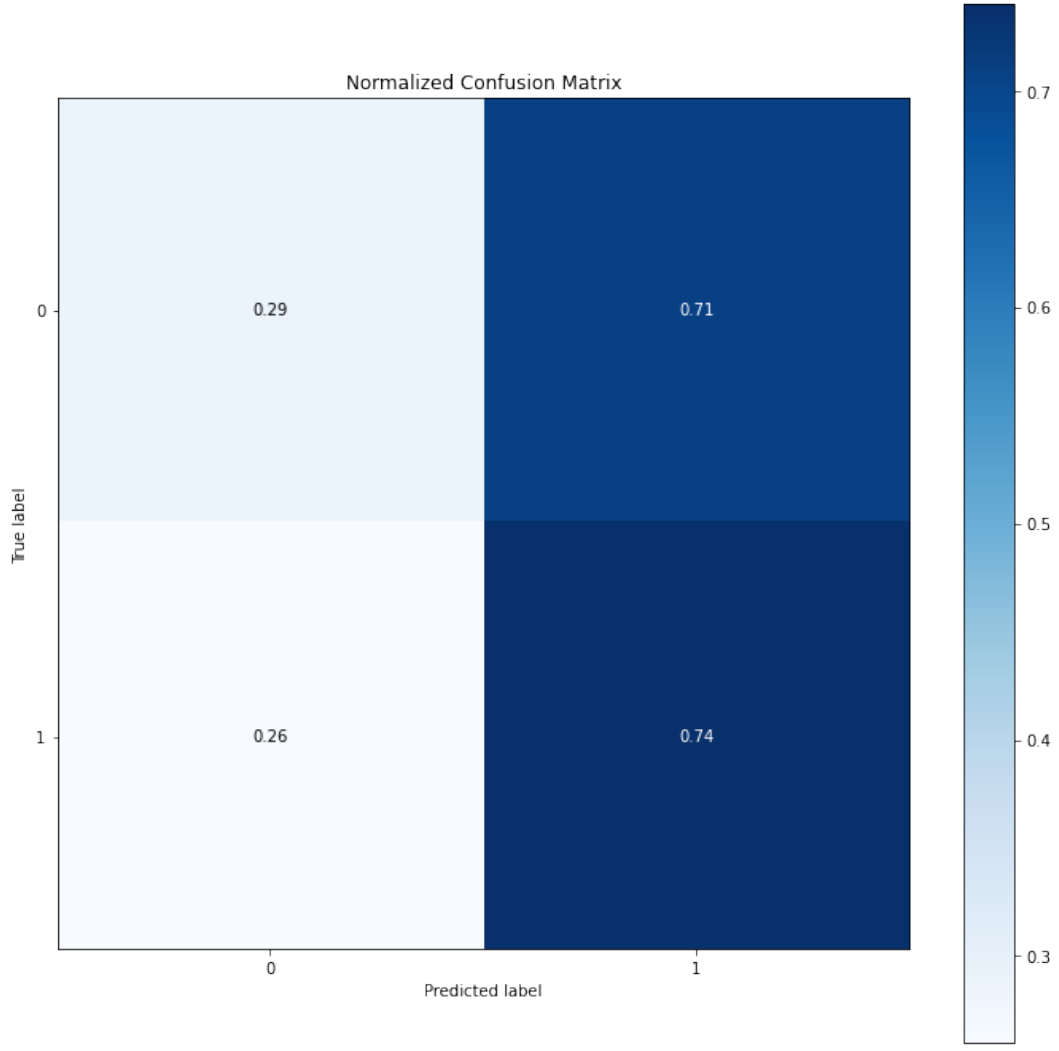


Figure 5.2: Replicated confusion matrix

5.1.4 Observations on the obtained results

After comparing the two results, the ones from the report of the previous project 5.1 and the ones obtained from replicating the results 5.2, only one conclusion can be reached. The performance of the deep learning algorithm is totally insufficient. All the performance indicators significantly deviate from the results shown in the report of the previous project. The absolute drop in performance ranges from 30% to 50% in all indicators. As explained above, some deviation is to be expected, but this is beyond any acceptance. The results of the previous ANN showed that it could very well classify the different data points. However, the replicated results show that the deep learning algorithm is not better than a coin flip.

To explain this further, in the classification problem of this project there are two classes. Either there is a defect or there isn't one. Therefore, one totally random method to classify any new data point would be to toss a coin. If it lands on tails, the data point is classified as a "defect", and if it lands on heads, it is classified as a "no defect". After a large enough number of data points, all the performance indicators (accuracy, precision, recall, F-score) would stabilize at around 50%. This is because this classification method has a fifty-fifty chance of guessing the correct class of any data point.

If one looks at the replicated ANN performance 5.1, the comparison with the coin flip is very grim. In precision, recall and F-score, the algorithm barely outperforms the coin flip. The total improvement is only around 1%. This is totally not significant and could even be explained by other factors. Moreover, when looking at accuracy, the ANN vastly under-performs. The algorithm only has a 30% chance of classifying correctly any given data point. Tossing a coin has a 50% chance. If a classifying method is outperformed by a totally random method, it can only be accepted as a total failure to achieve the intended result.

The causes of this can be partially uncovered by looking at the validation results obtained during the training of the algorithm. These are shown on the following table 5.3.

Performance Indicators	**Test (%)**
Accuracy	50.136
Precision	58.367
Recall	50.136
F-score	53.939

Table 5.3: Replicated ANN validation results

The validation results show that the algorithm barely improves their performance

during the training. The precision rises by only 8 points, while the accuracy stagnates at around 50%. Therefore, it can be concluded that the algorithm is unable to properly learn during the training. Moreover, the meagre increase in performance obtained from the validation data vanishes when trying to classify the test data. However, it is very difficult to obtain the reason behind this apparent lack of learning. Deep learning can sometimes be similar to a black box, meaning that the input and output can be thoroughly analyzed, while what happens inside is more difficult to understand.

5.1.5 F-score issue

Even though the raw results of the previous project can not be accessed, some basic scrutiny quickly casts a shadow of doubt over their validity. There is one performance indicator, F-score, that can be calculated by only using the other indicators. To test the validity of the results 5.1, the F-score is calculated from the precision and recall, and then it is compared to the value shown in the results. This is shown as follows in equation 5.1.1.

$$\text{F-score} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}} = \frac{2}{\frac{1}{0.7999} + \frac{1}{0.9897}} = 0.8847 = 88.47\% \neq 78.63\% \tag{5.1.1}$$

The results shown in equation 5.1.1 can not be explained. The F-score results of the calculation and those shown in the report should coincide, they should at least be very similar However there is an absolute difference of 10 points between the two values. To obtain these values, the function *precision_recall_fscore_support* of the *sklearn* package was used. This function and package have been widely used for many years. There is little space to doubt the output it provides. Therefore, it is inexplicable that the performance indicators shown in the report of the previous project are not mathematically consistent.

5.1.6 Results assessment

In conclusion, for the matters of this current project, the results presented in the report of the previous project can not be accepted as valid since they could not be replicated. If the results can not be replicated, they shall be ignored. However, it has to be noted that the rest of the work done in the previous project is still valuable.

5.2 The reasons for deep learning

At its core, this project tries to solve a classification problem: from a set of classified data points, the goal is to develop a classification model for future data points. The

previous project [1] decided to accomplish this using a deep learning algorithm. This decision is not acknowledged nor justified in any part of the report. It is only implied that deep learning is the correct method to solve the classification problem of this project. However, there are many methods or procedures that can be used to solve the classification problem. None of them are perfect, as they all have different advantages and disadvantages. Different situations require different methods. In any case, the choice of a method over the others should be justified because this is not trivial.

As a classification method, the main advantage of deep learning is that it is able to discover hidden patterns and to understand complex relationships between a large number of interdependent variables. The main drawback is that it requires a higher amount of resources, mainly input data and processing power, than other classification methods. This means that in complex problems with enough available resources, deep learning will outperform the rest of classification methods. However, in simple problems, and/or situations where the resources are insufficient, machine learning is vastly inefficient and it tends to under-perform. Another important drawback of deep learning that has to be considered is the lack of insight into the inside of the algorithm, since the decision-making process of the algorithm tends to be obscured and difficult to access.

In other words, the reasons for choosing deep learning over other classification methods are the following:

- The problem has a high degree of complexity.
- Enough reliable data can be provided.
- Enough processing power can be allocated.
- Algorithm insight is less important than result accuracy.

Taking this into consideration, let us analyze point by point if the problem of this project is suited to be solved using deep learning.

5.2.1 Problem complexity

The typical classification problems in deep learning consist of speech recognition, image categorization and natural language processing. These problems are well suited for deep learning because they are really complex, as can be seen below.

- **Speech recognition:** the recommended sample rate for audio data is of $16.000Hz$, which means in each second **16.000** different measures are taken. Since training requires a large enough audio data set, the total volume of

audio data quickly skyrockets. The complexity of the problem arises from trying to classify the audio data from the raw input data. The algorithm has to extract the pitch, identify the phonemes and understand the spoken words. Deep learning finds the patterns and relations between the different layers of the problem.

- **Image categorization:** the usual resolution of input images is of 256x256, however a deep learning algorithm can have good results with images with a minimum resolution of 96x96, which means that for each data point **9.216** different samples are taken, one for each pixel. Moreover, if the image is in colour, this number has to be approximately multiplied by 4. When trying to classify an image, many layers have to be discovered and relations between the different samples have to be understood.
- **Natural language processing:** there are approximately **500.000** different words and terms in the English language. It is not difficult to see that the amount of different combinations to form phrases is immense. The complexity of the problem arises from the many contextual nuances of natural language. Many words have different meanings according to their context, and the wording of sentences can change the message of the speech.

When comparing the complexity these cases to the classification problem of this project the differences are many. First of all, the database only contains **21** data columns (not counting ICAO code), as seen on table 3.1. Moreover, many of the data columns are discrete in nature, meaning that their range of values is quite limited. For example, the *onground* column is either 0 or 1, the *prcp* and *snow* columns are 0 most of the times. The nature of the database used in the previous project is more thoroughly discussed in the previous chapter.

Most importantly, it is difficult to observe the need for complex layers or the presence of hidden patterns between the different variables. Let us use an example to illustrate this point. In image recognition if a single pixel is of a certain colour, this in itself does not communicate any sort of information. However, the presence of this pixel in a set of pixels of the same colour creates a certain shape, and the disposition of different shapes creates a whole image. In the case of image recognition, there are complex layers of information hidden in each data point. This can not be said the same for the problem of this project. For example, precipitation may correlate with a higher chance of defect, and other variables may increase this value. Maybe high precipitation with high temperatures almost guarantees the detection of a defect by the end of the flight. However, it seems ludicrous to suggest that there are multiple hidden and complex layers that relate the different variables and the

class of the data points. For such degree of complexity, many more columns are to be expected. For example, this could be the case if dozens of sensor all around the aircraft measured the structure vibrations. Some pattern could be unveiled that related some kind of vibrations with a higher risk of suffering a defect.

From the perspective of deep learning, the classification problem of this project is not complex enough. It is too simple, and there is little reason to think that there are really complex patterns in the data for the deep learning algorithm to discover.

5.2.2 Available data

The database used to train, validate and test the deep learning algorithm was obtained from **18** different aircraft which suffered **893** defects, creating over **700.000** data points, each related to a minute of flight, which corresponds to approximately **12.000** flight hours. Due to the way the data was collected, it is difficult to obtain the total number of flights. By estimating an average flight time of 2 hours, the total number of flights can be estimated to **6.000** more or less. This way, it can be reasoned that the classification problem of this project consists of **6.000** different data points and two classes, one with roughly **1000** points (flights that resulted in a defect) and the other with roughly **5000** (flights that did not result in a defect).

A question that should have been asked is if this sample size is enough to obtain good results using deep learning. This was not considered at all in the previous work. Instead, the limiting factor was not how big the sample size should be, but what was the maximum that could be done with the available resources. To download all the data, a total of 40 hours of computational time was spend, with 20h allocated to the flight database and the other 20h to the meteorological database. Since this process had to be closely monitored and it was impractical to achieve a high degree of automation, it was totally impractical to obtain a larger amount of data.

When pressed for answers, many data scientists quote that, as a rule of thumb, to start getting results with a deep learning algorithm a sample size of at the very least **1000** for each category is required [13]. However, this is a lower end estimate that is more based on experience than on an actual calculation. A higher end estimate can be obtained using a worst-case calculation method [14]. For a simple network structure, this method yields a sample size of **4000** for each class. Therefore, it could be understood that to obtain results with a deep learning algorithm, the aim should be to obtain a between 1000 and 4000 samples for each class.

Going back to the database used in this project, the two classes have different sample sizes, which should not be a big problem on itself. The flights that did not result in a defect consist of **5000**, more than the required range required obtain

results with deep learning. However, the flights that resulted in a defect consist of only **1000** samples. As such, no definitive conclusion can be reached in this aspect. Although a sample size of a 1000 is within the range of values needed to obtain results, it's on the lower end. It could be possible that deep learning is unable to produce good results due to the low sample size of the database. However, it is also feasible that some results could be obtained. In any case, a bigger sample size should be recommended in any case, either to start getting results or to improve the classification of the algorithm. The sample size seems to be the bare minimum, and maybe less than that.

5.2.3 Available resources

One of the main drawbacks of deep learning is that algorithms are extremely resource expensive to train. This is due to the complex data models inherent to deep learning. Powerful computational resources need to be available in order to produce results within a sensible time frame.

To give an idea of what this entails, let us illustrate this problem with real data. In the previous project, the computer used to run the code would take around **100 minutes**. For this project, in order to better run the deep learning algorithm training code, two different computers where used. The first one was a medium scale laptop, more than capable of using all the typical office programs and with decent hardware that could hold its own with some resource intensive applications. Running the code to train the algorithm from beginning to end would take around **60 minutes**. The other computer used was a large desktop, with a great graphics card and expensive hardware. It would take around **10 minutes** to run the code. Although it was close to 6 times faster than the other, the desktop was not as readily available as the laptop, as it had other taking its time.

This values seem to be very low at a first glance. If the deep learning code had be run only once, they would prove no issue at all. However, having to spend this amount of time on every run of the code ends up hampering the development of the deep learning tool. If the code is to be changed or adjusted in anyway, the computer takes the aforementioned amount of time to generate the results. As such, if an other faster classification tool had been used, better results could have been expected. Not because of the tool in itself, but because the developers of the tool could have spent more time tuning and changing the code. In the end, the execution time of the code can severely limit the number of changes to the training algorithm and the database. New things could have been tried, more options could have been explored, other ways to do the same thing could have been looked into.

In conclusion, although there were enough computational resources to run the

deep learning algorithm, the results could have been hampered by their amount. In order to reduce the computational strain, either the number of resources should be increased or another classification method could have been used.

5.2.4 Algorithm insight

With the correct framework, deep learning can be very good at solving classification problems with a high degree of complexity. However, there is always one issue where deep learning is severely lacking. This aspect is the interpretability and explainability of the algorithm. In a certain way, deep learning acts as a sort of black box. The input and output data are know and easily accessed. There is little problem to interpret the data that enters and exits the algorithm. However, the processes that happen in-between prove to be much harder to interact with.

When using deep learning to solve a classification problem, the trade-off between higher accuracy in the results and lower interpretability of the decision-making process has to be considered. In the case of this project, algorithm insight seems to be considerably important. Understanding the causes of the defect can prove to be more important than an increase in the classification accuracy. For instance, if the chance for a defect was highly increased when flying at high speeds on rainy days, this information would be very helpful for the user of said aircraft. Having a more accurate classification of the defects would lead to better outcomes and a better predictive maintenance process. However, it could be argued that it is more important for the final costumer to be able to learn the the different aspects that influence the appearance of a defect, to understand what tends to increase and what tends to decrease the probability of failure, and to obtain the necessary data in order to find the possible ways to reduce malfunction risk.

5.2.5 Decision assessment

According to what has been exposed to this point, the choice of using deep learning seems to not have been well substantiated:

- The classification problem that is to be solved does not appear to be complex enough for deep learning.
- The available data gathered up to this stage may be not enough to correctly train the algorithm.
- There increase in required processing power has proven to be a strain in the project.

- The trade-off between lower interpetability and higher result accuracy does not seem to be justifiable.

As such, it can be concluded that in the case of the classification problem of this project, the advantages of deep learning do not seem to justify its inconveniences. It can be argued that another classification method better suited for the characteristics of this project should have been chosen. However, the decision to use deep learning as the classification method was taken before the start of the project. It can be argued that the previous project did not have much choice in this matter.

5.3 Correlation and causation

The ultimate goal of this project is to create a predictive maintenance tool capable of detecting if any given flight would result in a defect on any part of the aircraft. As the name implies, PdM is based on predicting faults in advance. In other words, the algorithm should be able to discern if there is a high chance for a defect **before** the take off of the aircraft. Therefore, from a certain point of view, the main objective of this project is to indirectly study the causes of aircraft defects.

However, deep learning (and the rest of classification methods) is only focused on correctly classifying the input data. As such, it tries to find the **correlation** between the variables of the data point and its label. This means that the algorithm is totally incapable of discerning correlation from the real objective of the project, **causation**. In order to successfully create a predictive maintenance tool, understanding the causes of the defects is key, while the results of the defects are of little importance.

5.3.1 The importance of causation over correlation

In the typical classification problems of deep learning (speech recognition, image categorization and natural language processing), there is no problem of finding causation versus correlation. Each data point has a set of data that has to be correlated with its label. Finding the correlation is the only objective, and this is because an accurate classification does not need anything else. Besides, in these kinds of problems, the very same concept of causation bears little significance. An image of a car can be categorized in many ways, and the same can be said of voice recording of a single word. There is no causation here, only correlation between variables and labels.

In the classification problem of this project, finding causation versus correlation presents an important challenge. Understanding the causation of the defects can

be used to predict when the aircraft has a higher chance of failure and act before it is too late. However, if the algorithm only looks at correlation, it may end up being useless at predicting failures. This is because in order to correctly classify if a flight would result in a defect, better results in accuracy could be obtained by scanning and looking for the results of defects, instead of looking at the causes. As such, the algorithm would be of no use to aircraft operators because it would be impossible to obtain the required data to predict a defect.

Let us illustrate this with an example. Imagine an aircraft that each day goes from point A to point B, except when a failure is detected. In that case, the aircraft would land at point C in order to be repaired. If the deep learning algorithm presented in the previous project was given this theoretical case, it would easily achieve a high degree of accuracy. On any given flight, the algorithm would look at where did the aircraft landed, and classify the flight as a defect if it was on point C or as a no defect if on point B. However, this would be totally useless to the aircraft operator, because all flights were scheduled to land at point B. Relying on data that is unknown before the flight results on the algorithm being totally useless for predictive maintenance.

Obviously, such an extreme case as explained above is very unlikely to be found in the real world. But the idea presented here can be reproduced in many ways. For example, having a fault in the engine could result in a lower aircraft speed, and having a fault in the ailerons could result in wider turns. However, these results would be of no use to make engine or aileron defect predictions. Instead, the algorithm should look at the possible causes of the defects. For example, it could be possible that, before the flight, having flown at high speeds or having conducted tight turns were correlated with a high chance for defect. This would be useful for the operator, since this could enable them to perform some kind of predictive maintenance.

5.3.2 Causation in Machine Learning

The problem of finding causation is inherent to Machine Learning (ML), and as such, it is also an issue in deep learning. Since machine learning finds the correlation relationships in the database, the goal would be to find a method to go from correlation to causation. The issue is that causation always entails correlation, but correlation does not necessary entails causation. In other words, if a variable change causes another to change, then the first is correlated with the second. However, if a variable is correlated with another, it does not necessary mean that a change in the first variable will provoke a change in the second.

This happens because of confounders, which refers to variables that influence

other variables and the categorization of the data point at the same time. For example, a correlation could be found between the number of ice creams sold and the number of people that report having suffered a sunburn. It would be difficult to argue that buying ice cream increases the chance of suffering a sunburn, or vice versa, that suffering a sunburn increases the chance of buying an ice cream. Instead, the level of solar radiance could be the cause behind both variables. A higher solar radiance could be causing both an increase in the number of ice cream sales and the number of sunburns. In this case, solar radiance would be considered a confounder variable.

There are numerous real world examples where discerning correlation from causation has proven to be a problem for the ML methods.For instance, an American algorithm used to predict the re-offending probability turned up to be racially biased against blacks [15]. Even if there existed some correlation between race and re-offending, it is totally ridiculous to even suggest that the race of a person, when the same other background factors are equal, could influence the re-offending probability of said person.

In conclusion, although the problem of correlation versus causation is inherent to machine learning, this has to be considered. Said aspect has to be discussed and its consequences analyzed if this project aims to be successful.

5.3.3 Correlation without insight

As explained before, the causation is more important than correlation when trying to make predictions. However, correlation on its own, that is, without any kind of regard for causation, is not entirely worthless. Obtaining the correlation between some aspects of the problem can prove to be very useful data for the operator, even if there is no causation between them. For example, learning that after occurring a defect during flight the aircraft tends to arrive later than expected (because the pilot flies more carefully, or the aircraft performance is impaired, or whatever reason), could be very interesting for the operator, even if it does not help in predicting the occurrence of failures.

However, as explained on section 5.2.4, using deep learning has the inconvenience of diminishing the interpretability of the results. As such, due to the choice of this classification method, it is more difficult to take advantage of the correlations found by the algorithm.

5.4 Usefulness of defect prediction

Even after all the problems and different issues that have been outlined to this point, let us imagine that, somehow, the deep learning algorithm is a total success. A new algorithm is developed that is capable of predicting aircraft failures with a high degree of accuracy based on the required input data. It would be expected that this PdM tool had a high degree of usefulness for all aircraft operators. However, even with this near ideal results, the usefulness of the tool could be considerably impaired due to a number of reasons.

5.4.1 Lack of probability

Even with the best possible conditions, in the real world there is always a randomness factor. As such, when a classification method predicts a class for a given data point, this should not be understood as a pure prediction but as the most probable class. In other words, based on the data, the classification method calculates the probability of that point belonging to each class, and as such, it predicts that said point belongs to the class with the higher probability. It is impossible to get a total assurance that a point belongs to a certain class, as in the real world there always exists some randomness that has to be accounted for.

However, the algorithm developed for this project only predicts if a certain data point would result in an aircraft defect, or not. In other words, the output is totally binary, it's either a 0 for a no defect, or a 1 for a defect, with no in-between. This means that the operator would always lack a key part of the data, which is the chance of the occurrence of a defect. The degree of certainty of the prediction should be communicated in some way to the user, because the algorithm always has some doubt, it can't ever be totally sure.

This is important if the operator is to make the correct decision. For example, imagine that on a certain instant, the collected data points to a 5% chance of a defect occurring on the single engine of the aircraft. Using the algorithm as it is now, the tool would communicate that there would be no defect on the next flight, since there is a 95% chance that there is no defect. But if the operator knew that there is a 5% chance of loosing the single engine, and as such, suffer a very dangerous failure, they would probably perform some kind of maintenance on the engine before attempting to fly.

Although it might seem minor, knowing the degree of certainty of the failure prediction of the aircraft is important. Communicating the result of the prediction on its own is missing the degree of certainty.

5.4.2 Lack of defective part

In the current state of affairs in the aviation industry, when a defect of a component is detected (reactive maintenance) or when the expiry date of a part is passed (preventive maintenance) the operator of the aircraft performs the replacement of the affected part. Usually, the operator has a considerable stock of spare parts, ready to be used as need be. The defective component can be replaced in a very quick and easy manner, with little need for machinery or tools. Most of the times, only a simple check is performed on the part before sending it back to the manufacturer, who will repair the part as required and return it. Thanks to this process, the down-time of the aircraft is minimized.

When using the algorithm to detect if the next flight of an aircraft would result in a defect, the resulting output does not state what part or system will be in fault. Airplanes are complex machines with many separate systems and thousands of different parts. Knowing that a part of the aircraft will fail but not knowing which one in particular will render defect prediction totally useless for the operator. If the part that is about to be defective had to be found, the operator would have to strip down the entire aircraft and all of its components. The search for the defective part would be arduous and time-consuming. When it comes to maintenance, aircraft operators are interested in reducing costs and minimizing down-time. This maintenance operation would be very expensive and would render the aircraft essentially inoperative for a long span of time.

On top of that, the maintenance operation needed to replace the defective part would be even more difficult when taking into account that the part that is to be replaced has not failed yet. It may be difficult to find a broken blade, an oxidized valve or a faulty actuator when the operator has no idea what to look for. However, it is nearly impossible to find a component that has not failed yet. It has to be admitted that some components show signs of wear, become rusty or lower their performance before failing completely. But the ultimate truth is that it is impossible to distinguish components that will fail in the next flight from those that will not. Moreover, there are components that even after failing look completely normal, and only through testing can be verified as being defective.

Furthermore, not knowing what part or system is about to fail also presents a problem when analyzing the severity of the defect. For example, it is obvious that if the toilet is defective it could be a little inconvenient for the passengers, but if the aircraft loses an engine, the effects from that would be much more severe. From the operator's point of view, the information of what part is on the verge of defect is key to make the right decision. And it is even more crucial if the aircraft is to be maintained with a reasonable amount of time and resources.

5.4.3 Lack of input data

When talking about the usefulness of a predictive maintenance tool, a certain balance has to be reached between the required input data and the accuracy of the prediction. Usually, the more data the classification algorithm is provided, the more accurate predictions is able to make. However, if the amount of data needed to be inputted into the algorithm is too large or the process of collecting and processing the data is too long and complex, the aircraft operator might be unable to effectively integrate the tool into their maintenance process.

With this in mind, the current algorithm is structured in a way that makes it even more difficult to use effectively. The deep learning algorithm has been trained to detect the failures in aircraft flights from the flight parameters and meteorological data of said flight. Therefore, in order to predict the upcoming failure in a flight, the algorithm needs to be inputted the data of the flight. This presents a problem because this data will not been recorded or measured until the flight has already ended. In other words, in order to effectively use the predictive maintenance tool, the aircraft operator would need to collect some data that does not exist yet.

As explained on section 3.2, the algorithm database is composed of the flight data and the meteorological data. Taking into account the type of data, it has to be admitted that these parameters could be estimated before the flight:

- The flight plans provided by the pilot before departure could be used to estimate the flight parameters (speed, attitude, location, and others) at the required points during the flight. However, this would not take into consideration path deviations or missions without a meticulously detailed plan (training exercises, military sorties).

- The meteorological data (temperature, wind speed and direction, amount of rain) could be estimated by using weather previsions. The main issue would be that weather previsions are more accurate the closer they are to the present time. As such, the operator could be forced to conduct predictive maintenance very close to the departure.

In the end, even if the required parameters can be estimated, this presents a big problem. First, some kind of tool would have to be developed in order to estimate the required data. But more importantly, the accuracy of the defect prediction would become reliant on the accuracy of the data prediction. In other words, this would become a prediction based on a prediction of some data. It could be perfectly possible that, even if the deep learning algorithm is somehow made to work on a

theoretical environment, it would totally under-perform when put in practice in the real world.

5.5 Database capabilities

A question that should have been asked when designing and building the database is whether the data that was being gathered could ever be capable of fulfilling its intended goal. For example, a database containing all kinds of weather data could be used to predict the chance of precipitation. Maybe in the end the training is ineffective, but it seems feasible that by using weather data an algorithm could predict the presence rain. However, the very same database would be incapable to predict the day of the week (Monday, Tuesday, Friday). It would seem totally unfeasible that an algorithm could predict (with a reasonable degree of accuracy) which day of the week is currently in by just looking at the weather.

In order to correctly analyze the capability of the project database to accomplish its intended objective, the cause and indications of defects have to be understood. In the end, the goal that has to be reached is discerning whether the database can possibly provide data causing or indicating an upcoming failure in the aircraft.

5.5.1 Malfunction patterns

Under the guidance of the FAA, United Airlines conducted a study on source of failures in aviation. As it stands now, six different failure patterns are used to explain the defects in equipment. These are shown in figure 5.3, where the Y axis shows the operational time of the equipment and the X axis shows the chance of failure.

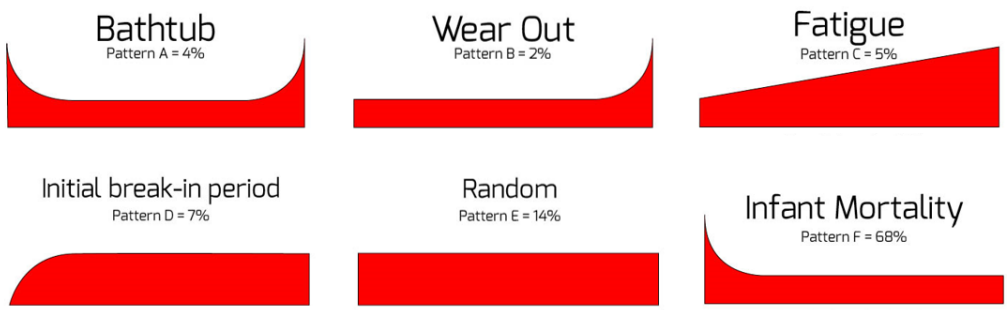


Figure 5.3: Previous confusion matrix [6]

The patterns A, B and C are age related, meaning that as the age of the part increases so it does its chance of failure. Calculating the expected time of failure of this parts is considerably easy thanks to their malfunction patterns. However,

most parts (up to 89% of all equipment) do not fall under any of this patterns, but on patterns D, E and F, where the age of the equipment does not significantly affect its chance of failure. In these cases, the failures are explained as being essentially random in nature. The ultimate goal of this project is to find a pattern in this randomness in order to predict equipment failures.

5.5.2 The causes of malfunctions

Usually, when an aircraft part is found to be defective the manufacturer of said parts conducts an investigation into the causes of the failure, as explained on section 2.1.2. By looking into the reports of these investigations, a series of repeated causes of equipment failure can be found, which are shown as follows. This list is not comprehensive and all encompassing. It should also be taken into consideration that only the causes related to equipment itself are displayed, and that those related to external factors (human error, bird-strike, hail, wrong maintenance, foreign object damage, and many others) are not shown.

- Age deterioration.
- Wear and tear.
- Software glitch.
- Mechanical dis-adjustment.
- Leakage.
- Fatigue failure.
- Degradation and corrosion.
- Contact failure.
- Short-circuit.
- Chaffing.
- Database corruption.
- Others.

The different aspects of malfunctions vary among the different systems of the aircraft. Analyzing the behaviour of defects in each and every system would be too long, since aircraft are really complex machines with many different systems. Instead of doing this, the electrical failures have been further studied to give an idea of what is to be expected.

5.5.3 Aircraft electrical failures

A study conducted to evaluate the causes of aircraft electrical failures [16] provides some very interesting data that can be used to understand the nature of electrical defects.

The study report provides a summary of electrical failures in aircraft. Connector failures account for the most defects, followed by parts and connections on printed wiring boards (PWBs). Connectors and PWBs are identified as the main contributors to electrical failure. The data can be seen as follows.

- 40% Connectors
- 30% Parts on PWBs
- 20% Connections on PWBs
- 10% Other

The main problem in the failure of connectors was identified as the formation of surface films, which caused the connectors to be non-conductive. By studying the defective PWBs in more detail, the study was able to rank the PWB components based on their replacement frequency, which is showed below.

- 27% ICs
- 14% Transistors
- 12% Hybrid circuits
- 12% Capacitors
- 12% Resistors
- 10% Diodes
- 3% Soldered joints
- 10% Others

Admittedly, the replacement frequency is not the same as the failure rate of the component, but it can be used as an estimation since they are closely related. In the end, that investigation was able to conclude the following.

- The majority of electrical failures in aviation are caused by interconnection problems, which are primarily due to wiring and connector failures.

- The dominant failure mechanism appears to be chafing, which results in electrical arcing of wiring and corrosion.
- At the printed wiring board level, the components primarily responsible for failure are connectors and semiconductors.

5.5.4 The indications of malfunctions

In the previous sections, the causes of malfunctions have been looked into. But it is important to remark that there is another aspect of malfunctions that can be of use when trying to predict them. And this is the possible indications of an upcoming malfunction.

In order to predict if a failure is incoming, one can look at the causes of said failure to determine if the chance of those causes braking a component. However, another way to predict failures is to look for indications that a malfunction is incoming. For example, if a component is about to fail, maybe it sees its performance reduced, it starts making some noise or it does not respond as quick as it used too. These aspects on their own are not the causes of the defect, but they may be indications that a failure is on the making.

In other words, when dealing with the prediction of defects one has to take into account that there are some aspects that may indicate the future presence of a failure even if they are not their direct cause. This field is not as studied as the causes of malfunctions, and as such not much is known yet.

5.5.5 Usefulness of flight data

After understanding more about the causes and indications of malfunctions, the goal of this section is to analyze the capabilities of the flight data to detect upcoming failures in aircraft. As it can be seen on section 3.3, the part of the database formed by the flight data contains the following variables.

- **Latitude and longitude:** On its own, the location of an aircraft during the flight does not seem to either be the cause or indicate the presence of failure in any way. At best, if the database was filled with enough flights from all around the world (which is not the case), the algorithm might identify the zones in the globe where aircraft are more prone to failure. However, it is difficult to argue that the aircraft location during flight is the cause of or indicates any kind of failure in the equipment.
- **Altitude from barometer and GNSS:** Similarly to the location on the globe, the altitude of the aircraft seems improbable to be the cause or indication of

many defects. Maybe some components are more prone to failure at certain altitudes, or to sudden changes in altitude, but this is unlikely and the scope of parts somewhat reduced.

- **Aircraft velocity:** Obviously, high speeds above a certain threshold are bound to cause numerous defects in the aircraft. However, this does not happen in normal operation of the aircraft. In this case, the aircraft velocity is unlikely to indicate any kind of failure (if the aircraft speed is reduced, the defect has already happened) or be the cause of many failures (maybe some very specific defects in the airplane surfaces or engine).
- **Direction of movement:** It is very unlikely that the direction of the aircraft has any kind of effect on the appearance of defects. The pointing of the aircraft does not convey any kind of interesting information. However, it is true that the integration of the aircraft direction could cause some sort of minor defects (tighter turns could cause problems in the control surfaces for example), but the scope would be small.
- **Vertical speed:** It is possible that the vertical speed of the aircraft has some influence on the appearance of defects in the wings of the airplane or in the landing gear, but it has to be noted that on normal operation aircraft fly at speeds set within limits set by the manufacturer.

In conclusion, it is unlikely that the flight data can be used to predict many failures on aircraft flights. It is possible that these variables are the cause behind some very specific defects on the exterior of the plane or the engines. But this is both unlikely and reduced in scope.

5.5.6 Usefulness of meteorological data

After analyzing the flight data, the goal of this section is to study the capabilities of the meteorological data in a similar fashion. As it can be seen on section 3.3, the part of the database formed by the meteorological data contains the following variables.

- **Current temperature:** It is possible that extreme temperatures influence the chance of failure of external components, the engines or the anti-icing system. However, on most operation, the temperature tends to fall within a comfortable range.
- **Relative humidity:** Maybe some external components are more likely to be oxidized when exposed to certain humidity conditions.

- **Dew point temperature:** This variable can be calculated from the temperature and humidity of the air. On top of that, it seems very unlikely that this obscure magnitude can be used to detect any kind of failure.
- **Precipitation:** Maybe some external components are more likely to be oxidized when exposed to rain.
- **Snow depth:** Maybe some external components are more likely to be oxidized when exposed to snow.
- **Wind direction:** On its own, the direction of the wind is very unlikely to be the cause behind any kind of defect. However, it could be interesting to cross this variable with the direction and velocity of the aircraft in order to obtain the wind direction from the point of view of the aircraft.
- **Wind speed:** It seems reasonable to think that extreme wind speeds could cause some sort of damage to exterior components, but this is a rare occurrence.
- **Peak wind gust:** This obscure variable seems very unlikely to be useful in any kind of way. Besides, the database already has the wind speed, so there is little reason to store the peak wind gust.
- **Sea-level pressure:** Maybe the pressure around the aircraft somewhat influences the appearance of defects in external components, and it is reasonable to think that sea-level pressure combined the altitude of the airplane can be correlated with the pressure around the aircraft.
- **Total sunshine:** It is highly unlikely that the sunshine exposure of the aircraft meaningfully affects the appearance of defects.
- **Weather condition:** Probably rainy weather increases the chance of corrosion and oxidation on the exterior components. However, it has to be noted that usually these parts are protected with some paint or varnish to expressively avoid this.

In conclusion, it is unlikely that the weather data can be used to identify a sizeable number of defects. Reasonably, maybe some defects on the exterior of the aircraft could be predicted, but this is very reduced in scope.

5.5.7 Usefulness assessment

In the end, after studying the defects in aircraft and the capabilities of the database, one conclusion has to be reached. This is that the database intended to be used to

train the algorithm appears to be totally incapable of predicting the majority defects. Trying to find any kind of useful correlation between variables and defect labels seems to be incredibly difficult if not impossible just because of the nature of the database.

One can not expect to meaningfully detect failures in an aircraft by just looking at its flight path and the environmental conditions. At best, only a minor part of all defects could be expected to be predicted.

5.6 Validation of correlation

According to the previous project report [1], one of the main aims of the study was to "show that there is a clear correlation between failures and the conditions that the aircraft endures in its life cycle". This would be done to capture the attention of potential customers, since it was implied that if it was possible to find a correlation between the defects and recorded data for this project, it would be possible to find a correlation between the defects and the data of new clients.

However, this is not necessarily true. This statement, which was taken as a fact, is not based on any kind of concrete evidence. This is because if one is able to find a correlation within a database, this does not in any way entail that a similar correlation could be found on an entirely separate database. In other words, finding a correlation between the defects and the recorded variables in the dataset of this project would not prove that the very same could be repeated with the dataset of a potential client, which would obviously be totally different from the present dataset.

It has to be admitted that finding a correlation within one dataset (which has not happened in this project) proves that one is well versed in the field of data analysis. But in the end, this would not prove that one can find a correlation in the database of a potential client, or even that there is one correlation present there waiting to be found. It is entirely possible that a new database is totally uncorrelated and there is no way to create a good classification model.

In any way, it is expected that one should be careful with their words and not make deceitful statements that could end up deceiving potential clients.

5.7 Summary of the critique

As a summary of this chapter, many issues and problems have been found in the work and results of the previous project. These are briefly shown in the list below.

- It has been impossible to recreate the results shown in the report of the previous project using the very same code and data.

- The decision of choosing deep learning as the classification algorithm seems questionable. Probably another method could have been chosen if possible.
- The problem of correlation versus causation in the database is a complex problem that has not been solved yet.
- Even if the predictive tool was made to work, it would have several important issues. Predictive maintenance would be very difficult with the current inputs and outputs.
- The database created in the previous project is arguably incapable of predicting the majority of aircraft defects.
- Even if correlation was found in the database, this would not prove that it could be repeated on a different database.

As a final note, one thing has to be made clear. Many things were done correctly by the previous project. And this work has been very useful for the current project. This chapter, and all that has been exposed, should be taken as constructive criticism aimed at trying to identify and solve the issues that have been detected.

Chapter 6: Data analysis and processing

An important part of the effort for this project has been devoted to performing an analysis of the data, with the intention of better understanding the database created by the previous project and to obtain better results. From this first database, a new database was created based on the findings of the analysis, and the results from the two datasets were compared. This chapter explains the process of data analysis, the reasoning behind each decision taken during said process, and the results obtained from all of this.

6.1 Data exploration

The first part of any kind of data analysis is the exploration of the data, which is a first step in the data preprocessing. Although some data exploration was conducted in the previous project, it has to be considered insufficient and too much superficial. A good exploratory analysis is usually key to visualize the data structure, the different values that said data can take, and the type of the variables. This section presents the new exploratory analysis process and its results.

6.1.1 ICAO field

After the most basic initial exploration of the data, a first issue was noted with the "icao" field, which is a categorical variable containing the ID (ICAO code) of each aircraft. Since the goal of the project was to create an algorithm that works for most aircrafts (and not a single specific aircraft), it is unwise to feed the algorithm training an aircraft identifier, as the algorithm could end up relating the target variable with individual airplanes.

Therefore, it was decided to use some information that lies behind the ICAO code but in a way that is less specific to individual aircraft. The "icao" field was transformed to the "borndate" field, which contains the born year of the aircraft. This variable is more generic and less specific, and, on the surface, it seems that it could be used by the algorithm to predict failures in aircraft. For example, it could be possible that an aircraft with more active years is more prone to failure than an aircraft that has just entered its operative cycle.

6.1.2 Defects over time

In order to visualize the occurrence of defects over time, a plot was created in which the X-axis displayed the time and the Y-axis displayed the defects (1 for defects, 0 for no-defects). This was repeated for each separate aircraft present on the database, and can be seen on figure 6.1 below.

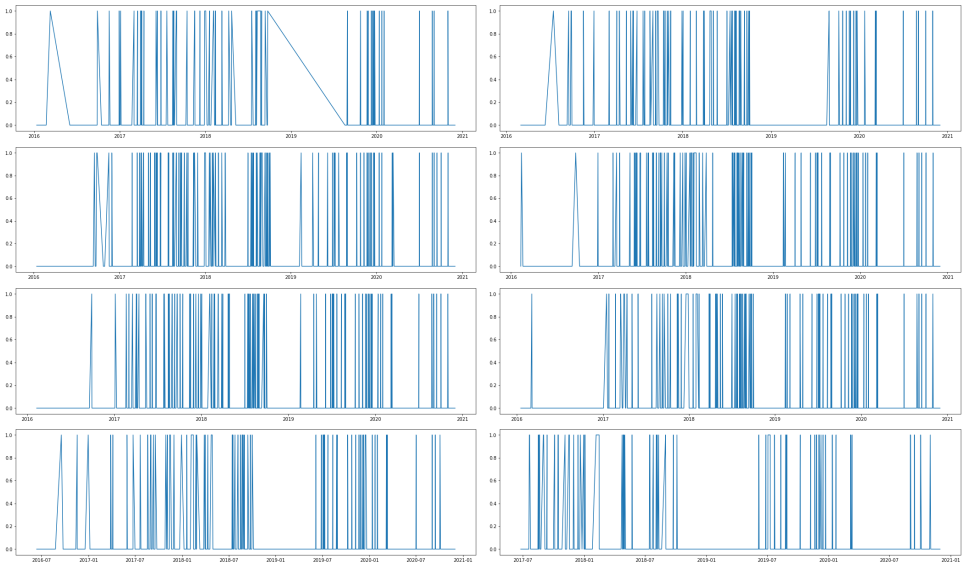


Figure 6.1: Defect occurrence over time

The previous figure 6.1 already displays some bits of very interesting data. First of all, there are big gaps on the data, meaning that there were long periods of time when the aircraft were not flying. Secondly, the database ranges from 2016 to 2021, and the total number of aircraft is 8. As explained in section 3.2, the first part of this statement agrees with what is said in the report of the previous project. However, the second part of this statement contradicts the report of the past project, since in that case it was stated that the total number of aircraft was 18.

6.2 Variables distribution and processing

The next step in the analysis of the data was the investigation of each variable in particular, separate from the others. The goal of this section was to study the distribution of each field and, based on that data, transform the database if necessary.

6.2.1 Born year

As explained in the section before, the variable "born year" was created from the "icao" variable. The distribution of this field can be seen on figure 6.2. As it can be seen, this is a categorical variable, where each year is a different category. Not

much can be said from this distribution except that in the fleet that composed the database, the aircraft born on 2014 accumulated the most flying time.

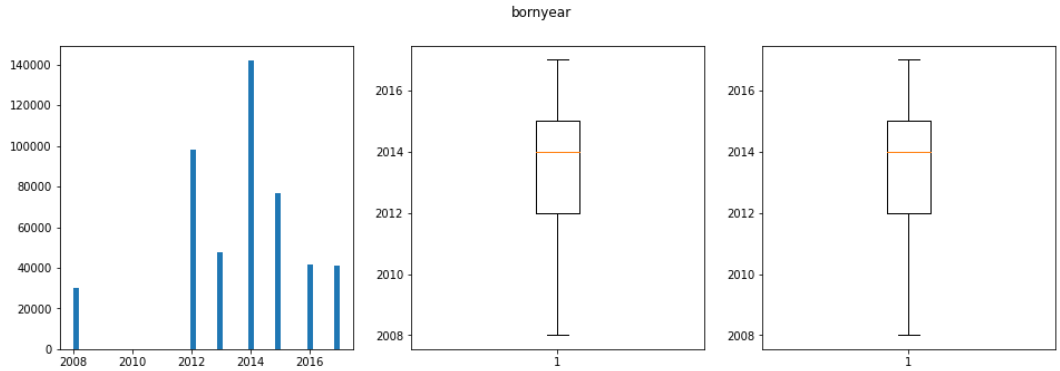


Figure 6.2: Histogram and box-plot of born year

6.2.2 Days since new

The variable of "days since new" is a new variable that was created by combining the "time" and the "born year" variables. The variable "time", which this new variable is intended to be a replace of, only displayed the time of the data point. The exact time at which the data was recorded does not convey any valuable information to the algorithm. Instead, this data should be related in some way to the aircraft. Therefore, by subtracting the born year of the aircraft from the time of the recording, the new variable "days since new" has been created. This could be useful because the age of the aircraft could influence the appearance of defects. From the distribution plot shown in figure 6.3, it can be seen that this variable follows a Gaussian distribution with some outliers.

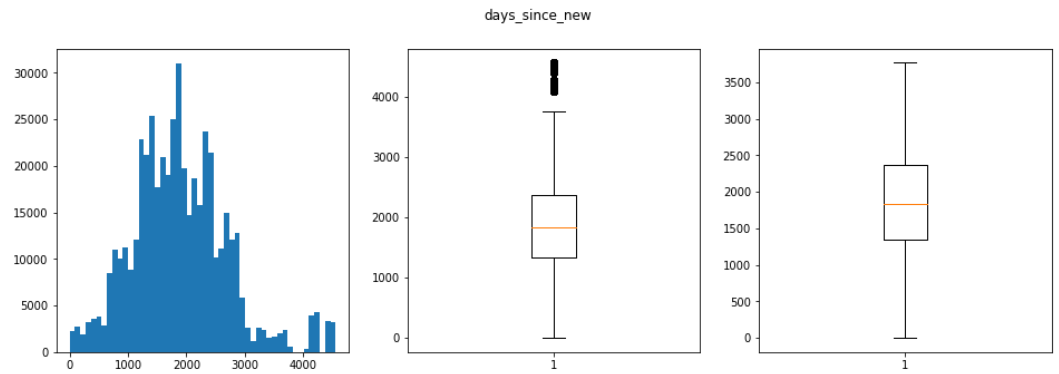


Figure 6.3: Histogram and box-plot of days since new

6.2.3 Latitude and longitude

The variables of "latitude" and "longitude" are two fields that should not be observed separately, since they are two numbers that are used together to describe the position of the aircraft. The figures 6.4 and 6.5 display the distribution of the latitude and longitude respectively. From this plots, the only information that can be extracted is that these two variables are highly concentrated and that there is a significant number of outliers. However, as explained above, they should not be studied independently.

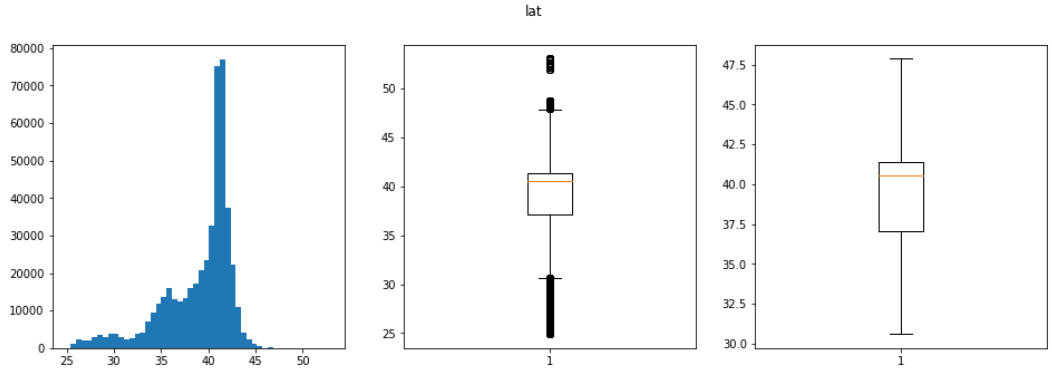


Figure 6.4: Histogram and box-plot of latitude

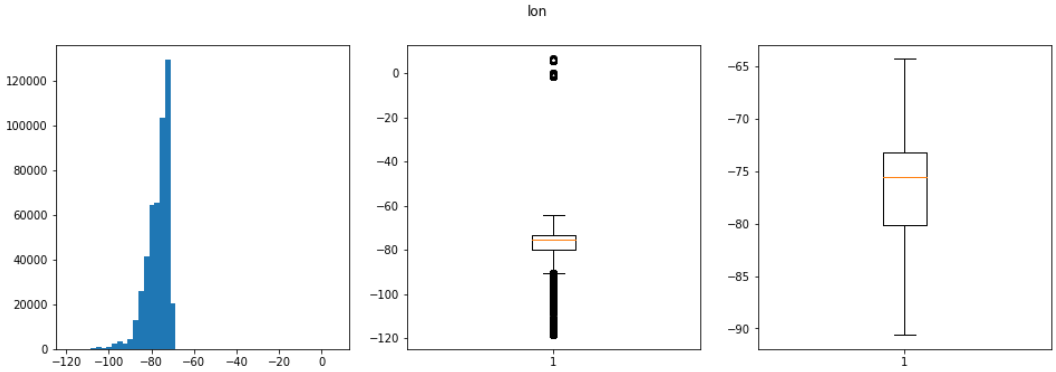


Figure 6.5: Histogram and box-plot of longitude

In order to learn more about the latitude and longitude variables, they have been plotted on a white canvas versus the locations of meteorological stations all over the world. This can be seen on figure 6.6, where each red dot represents a meteorological station and each blue dot represents the location of a recording in the database. From this plot it can be inferred that the vast majority of flights recorded in the database were conducted in the mainland of the United States of America and southern Canada, with some flights in England and Switzerland.

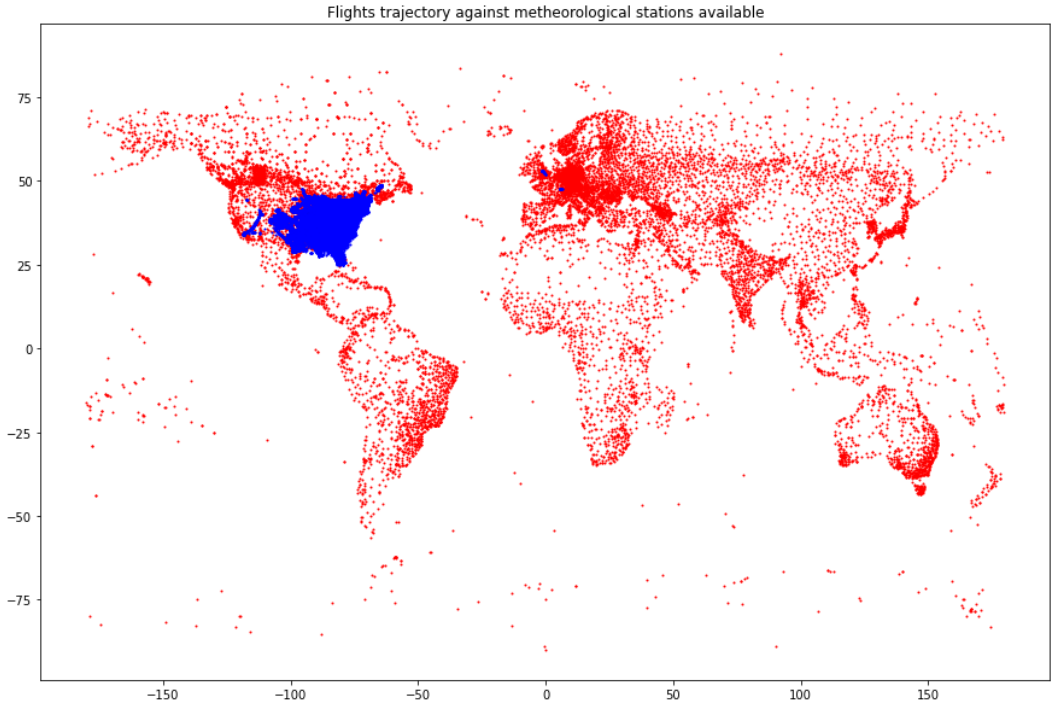


Figure 6.6: Trajectory of flights and station locations

6.2.4 On ground

As explained section 3.2, the variable "on ground" should indicate if the aircraft is on ground "1" or flying "0". However, as it can be seen on the figure 6.7 displaying the distribution of "on ground", this field contains zeros for the vast majority of instances. Out of the more than half a million entries, just 1300 points contain anything other than a "0". The reason for this could be that when the database was being created only the recordings taken during flight were saved in order to reduce the size of the dataset. In the end, it has to be concluded that "on ground" cannot be used by the algorithm to predict defects and is removed from the dataset altogether.

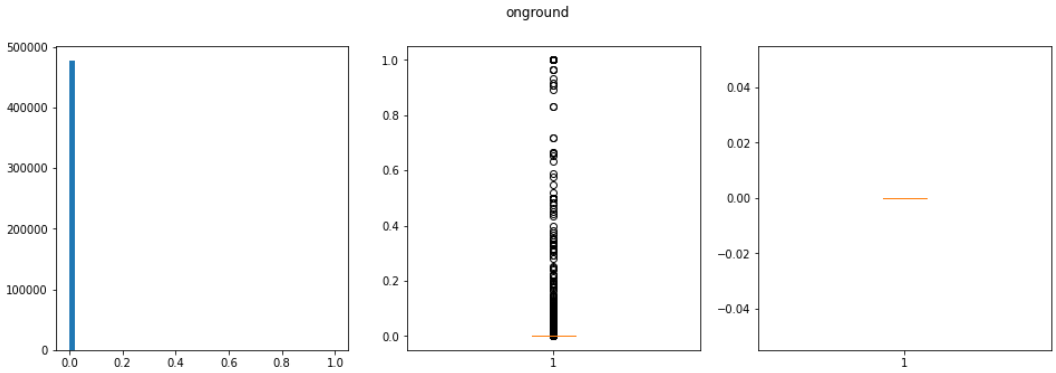


Figure 6.7: Histogram and box-plot of on ground

6.2.5 Barometric and GNSS Altitudes

On the one hand, the variable "baroaltitude" shows the altitude of the aircraft measured by the barometer. On the other hand, the variable "geoaltitude" shows the altitude of the aircraft provided by the GNSS. Which is to say that these two fields are measuring the same thing. As on could expect, the variable distributions shown in figures 6.8 and 6.9 are totally similar, with only some minor difference between the two fields. Observing the different values of the database in detail, one thing becomes clear: in the vast majority of cases, there is only a small gap between the two variables tends to stay pretty much constant. Since the two variables have a high degree of correlation, it's not worth to have two separate fields in the database. It has been considered that the altitude provided by the GNSS would be more precise than that provided by the barometer. For this reason, only the "geoaltitude" variable is used as an input for the algorithm, and the "baroaltitude" variable is discarded.

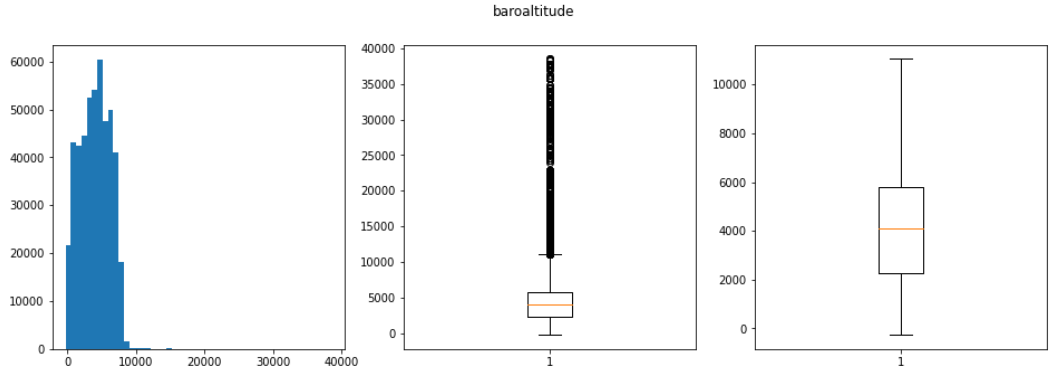


Figure 6.8: Histogram and box-plot of barometric altitude

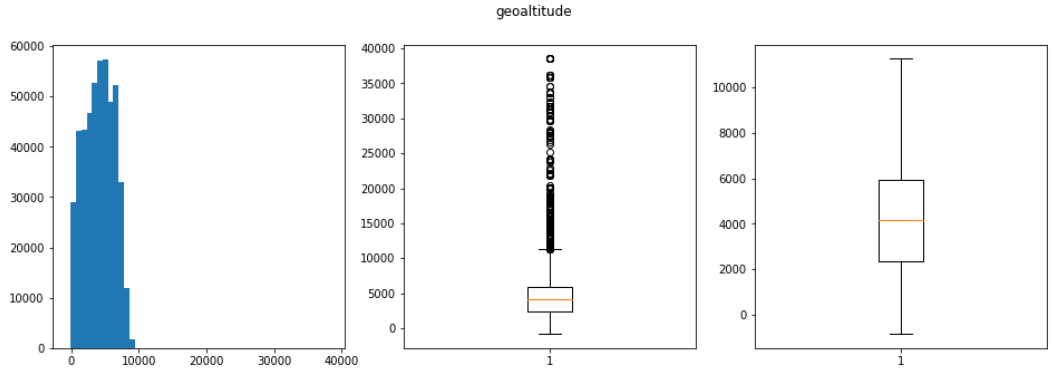


Figure 6.9: Histogram and box-plot of GNSS altitude

It has to be noted that most altitude values shown in the previous distributions seem to be consistent with what should be expected. Most values are located in the range of $18,000ft$ $(5,000m)$ and $30,000ft$ $(9,000m)$, which consist of the altitudes

at which turboprop aircraft are the most efficient. It has to be admitted that there is a significant number of outliers with altitude values that are simply not possible to accomplish. However, their number is very small, close to insignificant when compared to the total number of entries.

6.2.6 Velocity

Simply put, the "velocity" variable displays the total aircraft speed. The distribution of this field, which is shown in figure 6.10, is clearly Gaussian with very small variance. Most values are within the range of $50m/s$ and $200m/s$, which is as expected since these are the typical velocities of a turboprop aircraft. The outliers with lower values can be explained as aircraft velocities recorded during take off or landing. However, there are some outliers with ridiculous speeds that are simply unachievable with a turboprop airplane. But again, these are very few in number, close to insignificant.

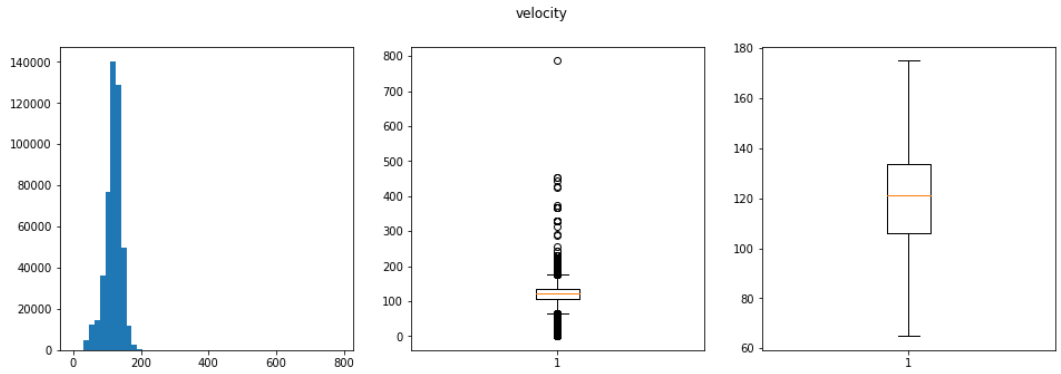


Figure 6.10: Histogram and box-plot of velocity

6.2.7 Heading

The "heading" variable represents the direction of the aircraft in the compass, being 0 degrees the north direction and 180 degrees the south direction. As it can be seen in the distribution of this field, shown in figure 6.11, the "heading" variable is very uniform all around the compass. There are two peaks that roughly correspond to a north-east and a south-west trajectories. Since this variable is very uniform and it was reasoned that the simple heading of an aircraft seems to be unlikely to affect the appearance of defects, it was decided to discard this field from the database.

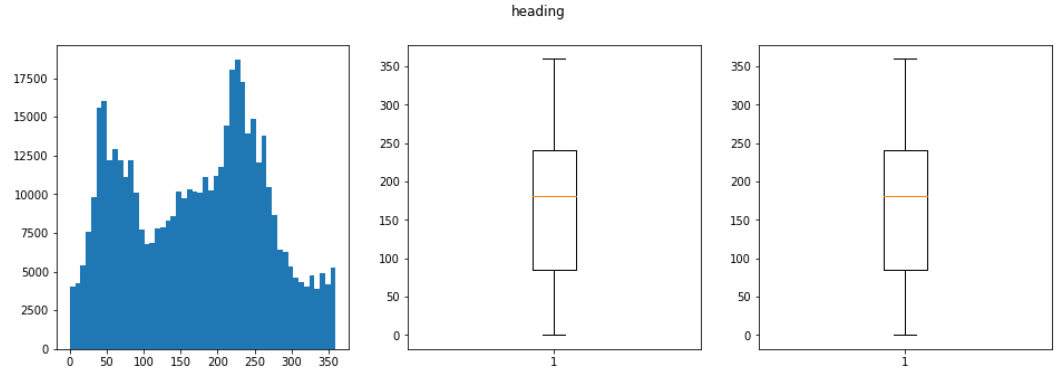


Figure 6.11: Histogram and box-plot of heading

6.2.8 Vertical speed

As its name indicates, the variable "vertical speed" represents the vertical velocity of the aircraft, with a positive value when ascending and negative when descending. The distribution of this field, which is shown in figure 6.12, is very concentrated on the value "0", which is understandable since most of the times aircraft tend to fly on a level plane. In order to improve the performance of the algorithm when reading this variable it was decided to transform this field from numerical to categorical. With this change, when the aircraft is flying steady a "0" appears on the vertical speed field, if it's ascending a "1" and a "-1" when descending.

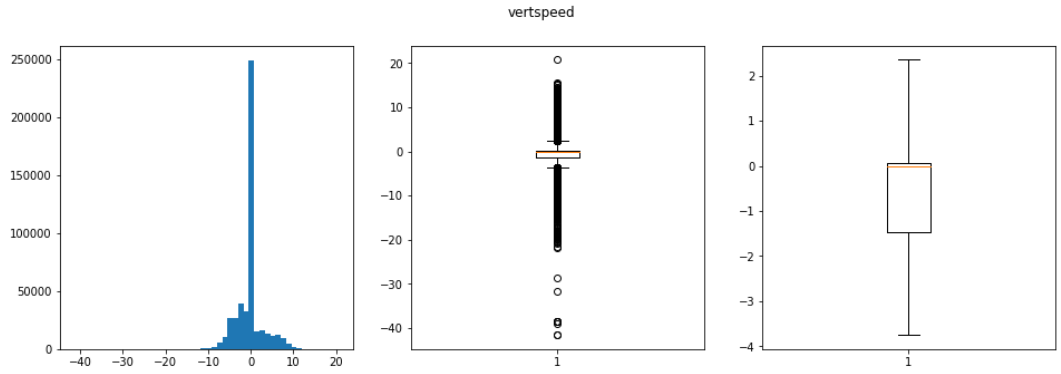


Figure 6.12: Histogram and box-plot of vertical speed

6.2.9 Temperature

As obviously as it seems, the variable "temperature" displays the current air temperature. The distribution of this field is shown on figure 6.13. Not much can be extracted from this plot, and the values that are displayed seem to be believable and consistent.

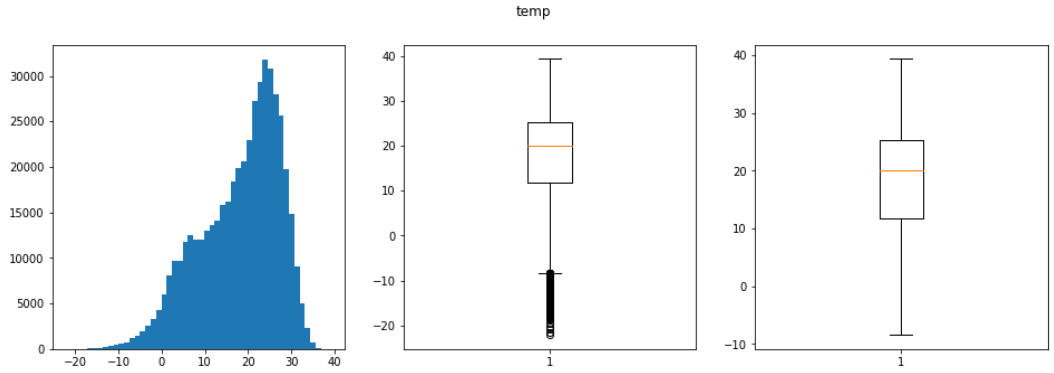


Figure 6.13: Histogram and box-plot of temperature

6.2.10 Dew point temperature

The variable "dew point temperature" measures the temperature at which point the air would become saturated with water considering a constant pressure and water volume. When observing the distribution of this variable, shown in figure 6.14, one thing becomes clear: this field is extremely correlated with the "temperature" variable. Because of this reason, and because it seams unreasonable to think that the dew point temperature of air can affect in any way the appearance of defects, it was decided to discard this variable from the database.

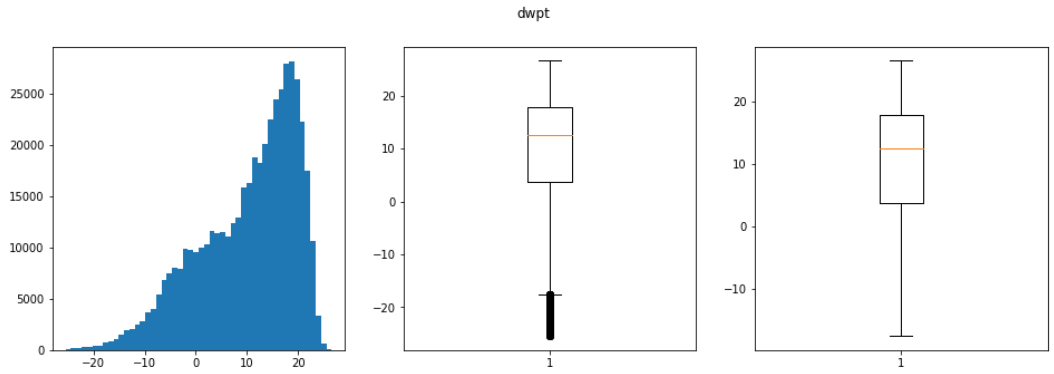


Figure 6.14: Histogram and box-plot of dew point temperature

6.2.11 Relative humidity

The distribution of the variable "relative humidity", which has a self-explanatory name, is shown on figure 6.15. Not much can be deduced from this and as such, no action has been deemed necessary for this field.

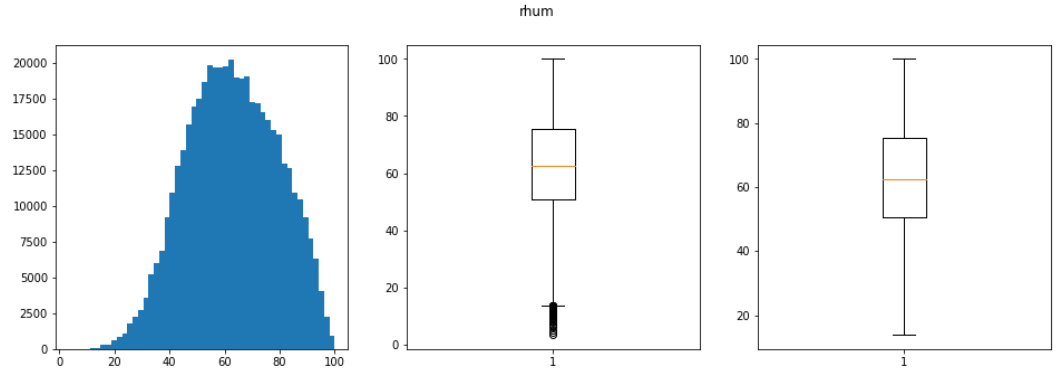


Figure 6.15: Histogram and box-plot of relative humidity

6.2.12 Precipitation

The variable "precipitation" presents a very unique distribution (fig 6.16), where there is a huge concentration of points with a value of "0". This can be understood since most of the time there is no rain. The number of outliers, that is, instances where the precipitation is greater than none, is very small, totally insignificant. For this reason, this variable has been discarded from the database.

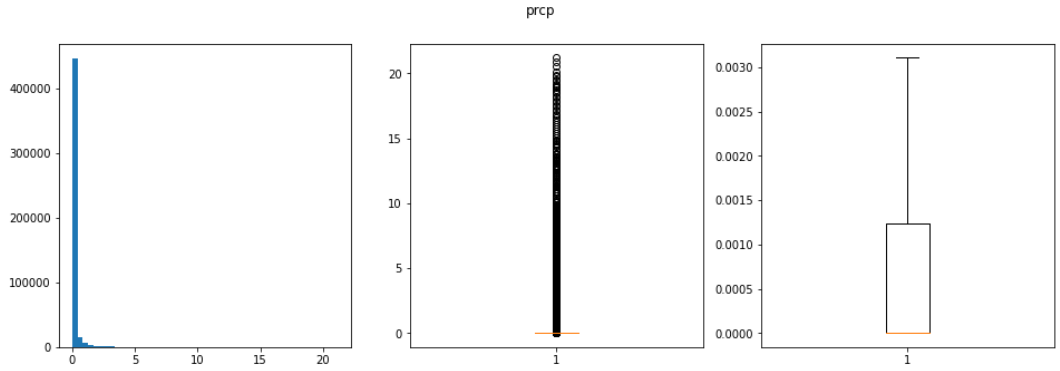


Figure 6.16: Histogram and box-plot of precipitation

6.2.13 Snow

It could be expected that this field has a very similar distribution to the "precipition" variable. However, the figure 6.17 shows that this field only contains zeros, with no outliers whatsoever. For this reason, this variable has been discarded from the database. Since this field contains no data, it can not train the algorithm.

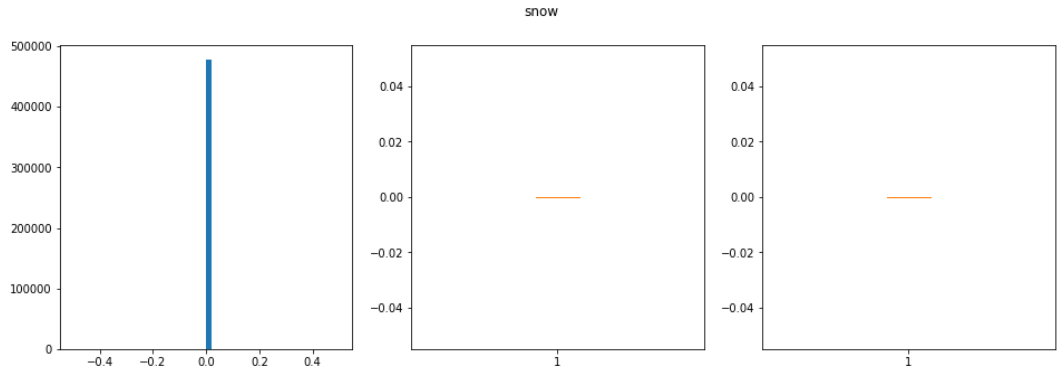


Figure 6.17: Histogram and box-plot of snow

6.2.14 Wind direction

Similarly to the field "heading", the variable "wind direction" displays the direction of the wind on the compass. It could be expected that this field follows a very uniform distribution. However, the distribution of this field (fig 6.18) is Gaussian, and centered on a southwest direction. In order to validate the believability of these results, a study could be conducted on the wind patterns in north America. Ultimately, it was decided to discard this variable for the same reason that the variable "heading" was discarded. It is unreasonable to think that the direction of the wind has any kind of effect on the appearance of defects in the aircraft.

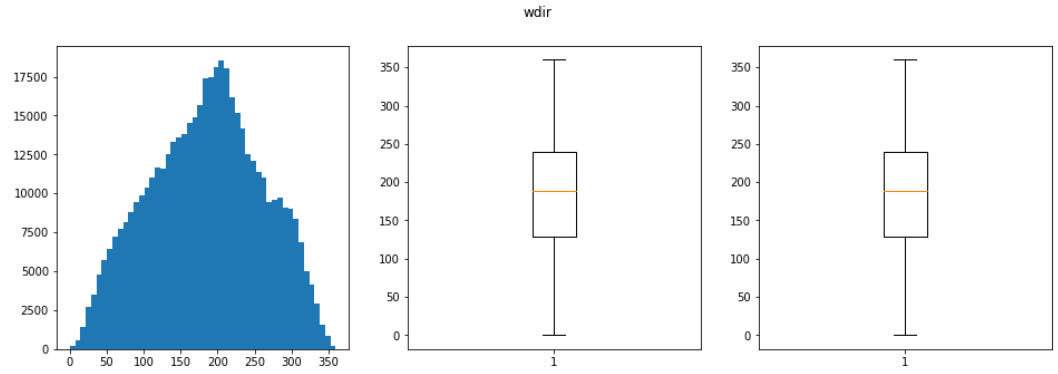


Figure 6.18: Histogram and box-plot of wind direction

6.2.15 Wind speed

The "wind speed" field distribution (fig 6.19) seems to be consistent. The distributions looks to be Gaussian with a lower variability and a concentration in the values close to zero, which could be explained as most days having a calm weather. Admittedly, there is a number of outliers with ridiculous high wind speed values, only obtainable if flying inside a typhoon or hurricane. Similarly to the aircraft velocity

field, the number of extreme outliers is very low and should not hamper excessively the training of the algorithm.

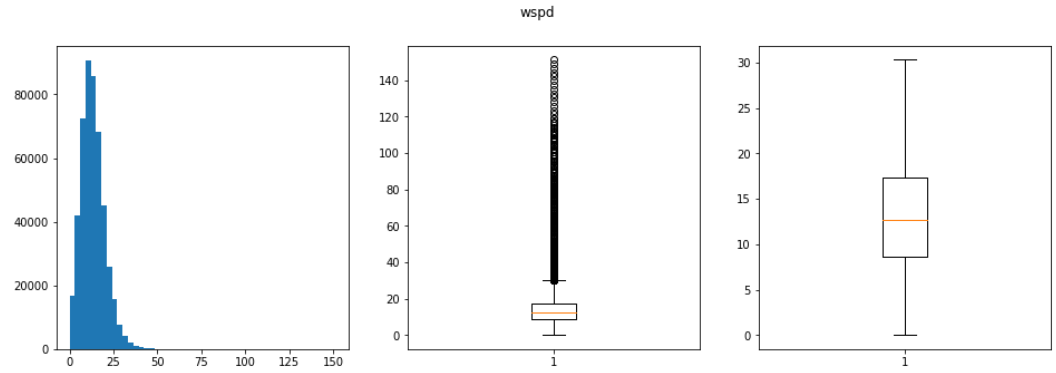


Figure 6.19: Histogram and box-plot of wind speed

6.2.16 Wind peak gust

The distribution of the variable "wind peak gust" is very similar to that of the "precipitation" field. The vast majority of values contain zeros, and only a very small number of outliers contain anything other than a zero. For this reason, it was decided to discard this column from the database.

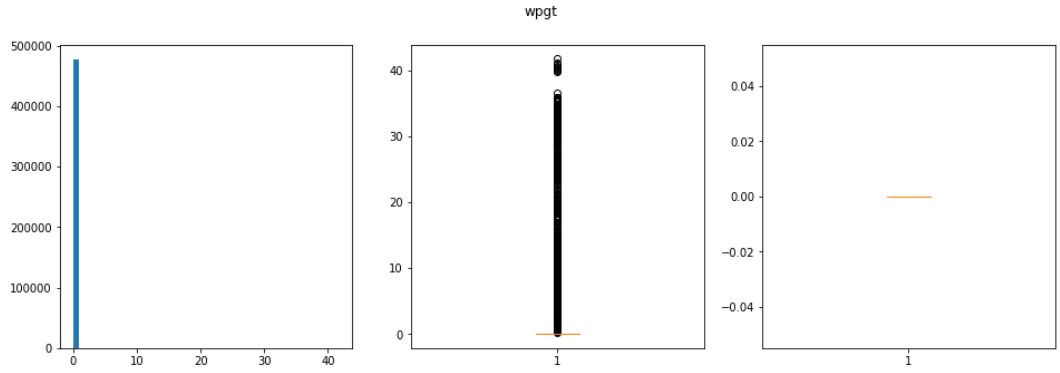


Figure 6.20: Histogram and box-plot of wind peak gust

6.2.17 Sea-level pressure

At first glance, it might seem that air pressure can have some influence on the appearance of defects in the aircraft. However, it has to be taken into account that this field tracks the pressure at sea altitude, and not where the airplane is flying. However, the main issue with this variable is the inconsistent range of values. As seen on the distribution of this variable (fig 6.21), the majority of values are concentrated around the expected $1000hPa$. However, an important number of entries

contain a sea-level pressure that makes no sense whatsoever. To put things into perspective, the lowest sea level pressure ever recorded was $870hPa$. It is therefore impossible that such an amount of data lies below that number. Because of this important inconsistency, it was decided to discard this column from the database.

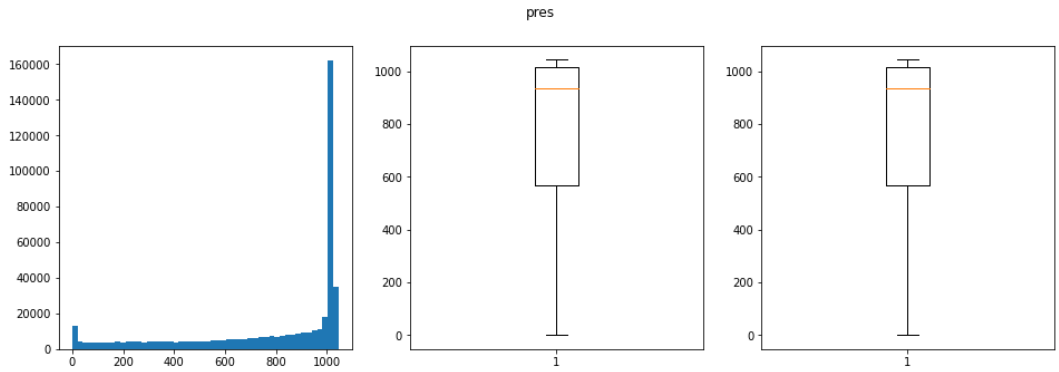


Figure 6.21: Histogram and box-plot of sea-level pressure

6.2.18 Total sunshine

This variable should contain the total number of sunshine minutes within each hour. So one could expect this variable to be filled with values from "0" (totally cloudy) to "60" (totally sunny). However, as seen in the distribution of this variable (fig 6.22), the vast majority of the field only contains zeros. There is a large number of outliers, but overall the data does not make any sense and for this reason, the "total sunshine" field was discarded from the database.

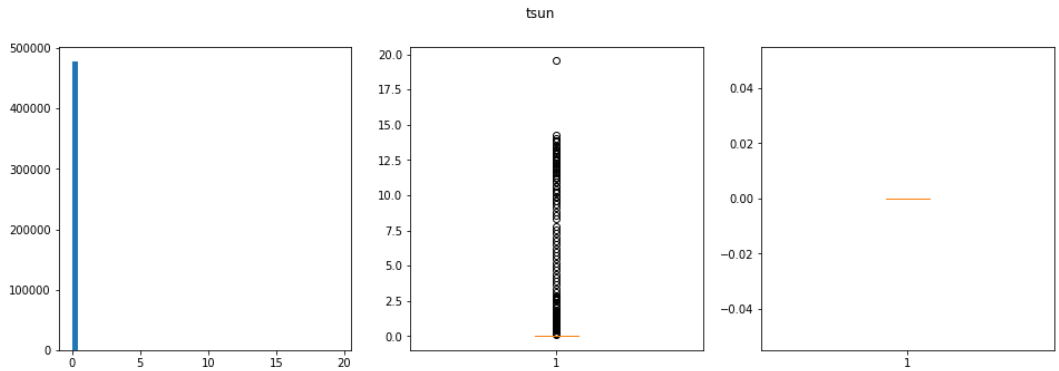


Figure 6.22: Histogram and box-plot of total sunshine

6.2.19 Weather condition code

This variable provides a categorical value that indicates the current weather condition. Each number can range from "1" to "25" and indicates a different type of

weather. The distribution of this field can be seen on figure 6.23. At first glance, the distribution of the weather condition seems to be consistent since lower code values are associated with calmer conditions, which should be more widespread than bad weather.

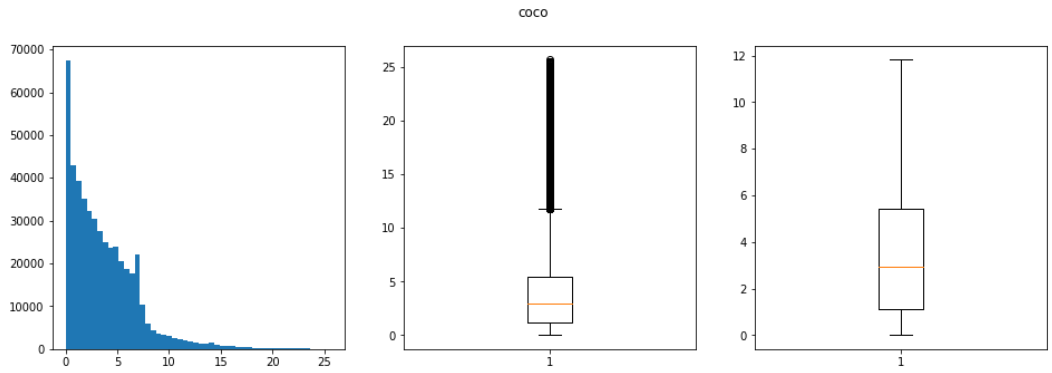


Figure 6.23: Histogram and box-plot of weather condition code

It was reasoned that the number of different categories, a total of 25, was a bit excessive. Many conditions were very similar from one another, and in order to help the algorithm training, it was decided to reduce the number of categories to a more manageable total of 7. This can be seen on the table 6.1. Instead of having a very specific label for each and every kind of weather, the new field contained one of seven different generic weather conditions.

COCO	Weather condition	Reduction
1	Clear	Fair
2	Fair	Fair
3	Cloudy	Cloudy
4	Overcast	Cloudy
5	Fog	Fog
6	Freezing Fog	Fog
7	Light Rain	Rain
8	Rain	Rain
9	Heavy Rain	Rain
10	Freezing Rain	Rain
11	Heavy Freezing Rain	Rain
12	Sleet	Sleet/Hail
13	Heavy Sleet	Sleet/Hail

COCO	Weather condition	Reduction
14	Light Snowfall	Snow
15	Snowfall	Snow
16	Heavy Snowfall	Snow
17	Rain Shower	Rain
18	Heavy Rain Shower	Rain
19	Sleet Shower	Sleet/Hail
20	Heavy Sleet Shower	Sleet/Hail
21	Snow Shower	Snow
22	Heavy Snow Shower	Snow
23	Lightning	Storm
24	Hail	Sleet/Hail
25	Thunderstorm	Storm

Table 6.1: COCO weather condition reduction

It has to be noted that during the exploration of the data it was found that almost all values of this field recorded on the database were not integers but decimals.

This made little sense, since a categorical variable should only contain the specific set of possible values. Moreover, it should be totally prohibited to make any kind of average between values of a categorical variable that is not ordered. For example, the average of a clear sky and heavy rain would be a foggy weather, which is obviously senseless.

6.2.20 Flight hours

The distribution of the field "flight hours", which can be seen on figure 6.24, seems to be adequate. It approximately follows a normal distribution with a very reduced number of outliers. Therefore, this field has remained unaltered in any kind of way.

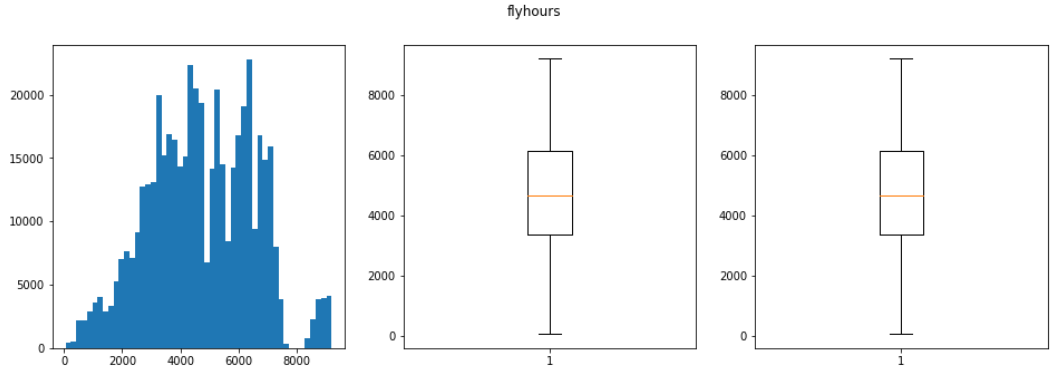


Figure 6.24: Histogram and box-plot of flight hours

6.2.21 Defect

The last variable to look at is arguably the most important one, the target variable. As expected, the distribution (fig 6.25) of the defect variable is entirely concentrated in two points, "0" (no defect) and "1" (defect). The distribution also shows that there is quite an imbalance in the data (90% to 10%), with a lot more cases of no defects than cases with defects.

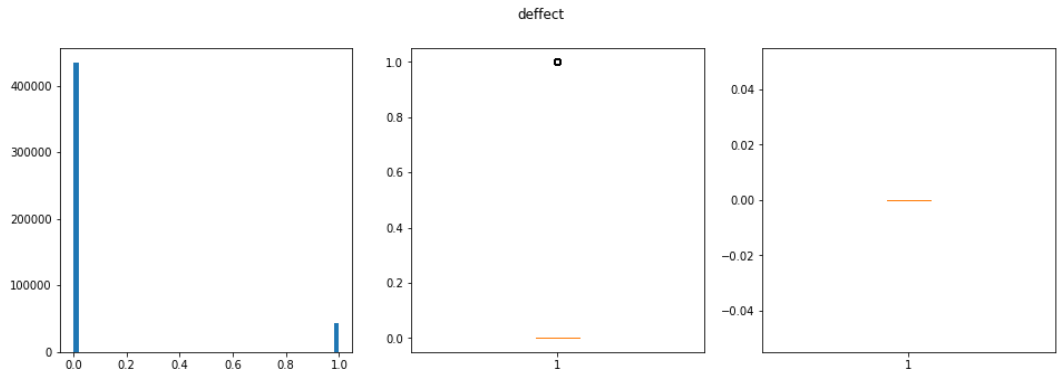


Figure 6.25: Histogram and box-plot of defect

6.2.22 Conclusion of variable analysis

After analyzing each and every variable individually, the correspondent database transformations have been performed and a new database has been obtained. This new database, created entirely from the previous database, is intended to perform better with the algorithm training.

6.3 Creation of extra variables

After the analysis explained before were performed, some further iterations on the dataset were performed in order to improve the overall performance. In the end, two more fields were added to the final database. These two newest variables really improved the output obtained from the training with the dataset.

6.3.1 AC pressure

As explained before, it was considered that the variables of sea-level pressure and altitude of the airplane were not meaningful enough, and as such they were discarded. However, by combining these two fields, a new variable is created called "Aircraft pressure". Basically, by using the following equation the two fields were transformed into a new one.

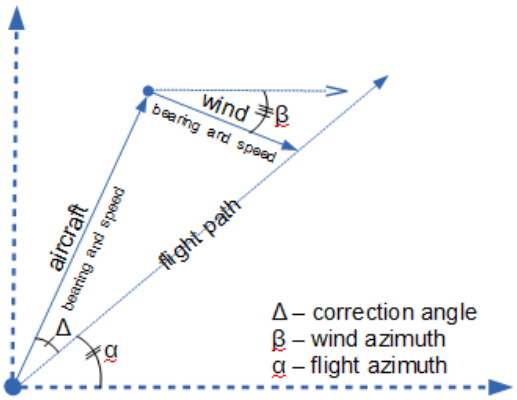
$$P = P_b \cdot \left[1 + \frac{L_b}{T_b} \cdot (h - h_b)\right]^{\frac{-g_0 \cdot M}{R \cdot L_b}}$$

It is reasonable to think that the value of the pressure at the altitude that the aircraft is flying in could somehow influence the appearance of defects. It is possible that there are some exterior components in the aircraft that are more prone to failure

when the air pressure is higher or lower than normal, or maybe some part can be broken when there are sudden changes in pressure.

6.3.2 Airspeed angle

In a similar way as the previous field, it was considered that the variables of "heading" and "wind direction" were not meaningful enough, and as such were discarded. However, if these two variables are combined, and also using the speed of the aircraft and of the wind, one can obtain a new field called "airspeed angle", which corresponds to the angle of wind seen by the aircraft. This is better illustrated in the figure below.



It is reasonable to think that the angle of the incident wind on the aircraft could influence the appearance of defects. Maybe there are some surfaces of the plane that are sensitive to winds coming from the sides or rear of the aircraft. In order to better help the algorithm training, it was decided to transform this field from numerical to categorical, with each category representing a different side from where the wind was coming from, using the perspective of the aircraft.

6.4 Variables correlation

Up to this point, each variable has been studied independently without looking at its behaviour with the other variables. The goal of this section is to study the correlation between variables and transform the database if deemed necessary.

Two different types of plots are shown in the following pages. First is the scatter plot, or simply scattering. And secondly is the correlation plot. This is done for the flight data, the meteorological data and finally for all the data together.

6.4.1 Data scatter plot

The flight data scatter plot (fig 6.26) shows only the variables related to the flight data. By looking at this figure it seems that there is little correlation between all these variables, with a notable exception. It looks like the variables "days since new" and "flight hours" are highly correlated. This makes sense, since if an aircraft is used in a regular basis, the values of these two fields should increase at a similar rate.

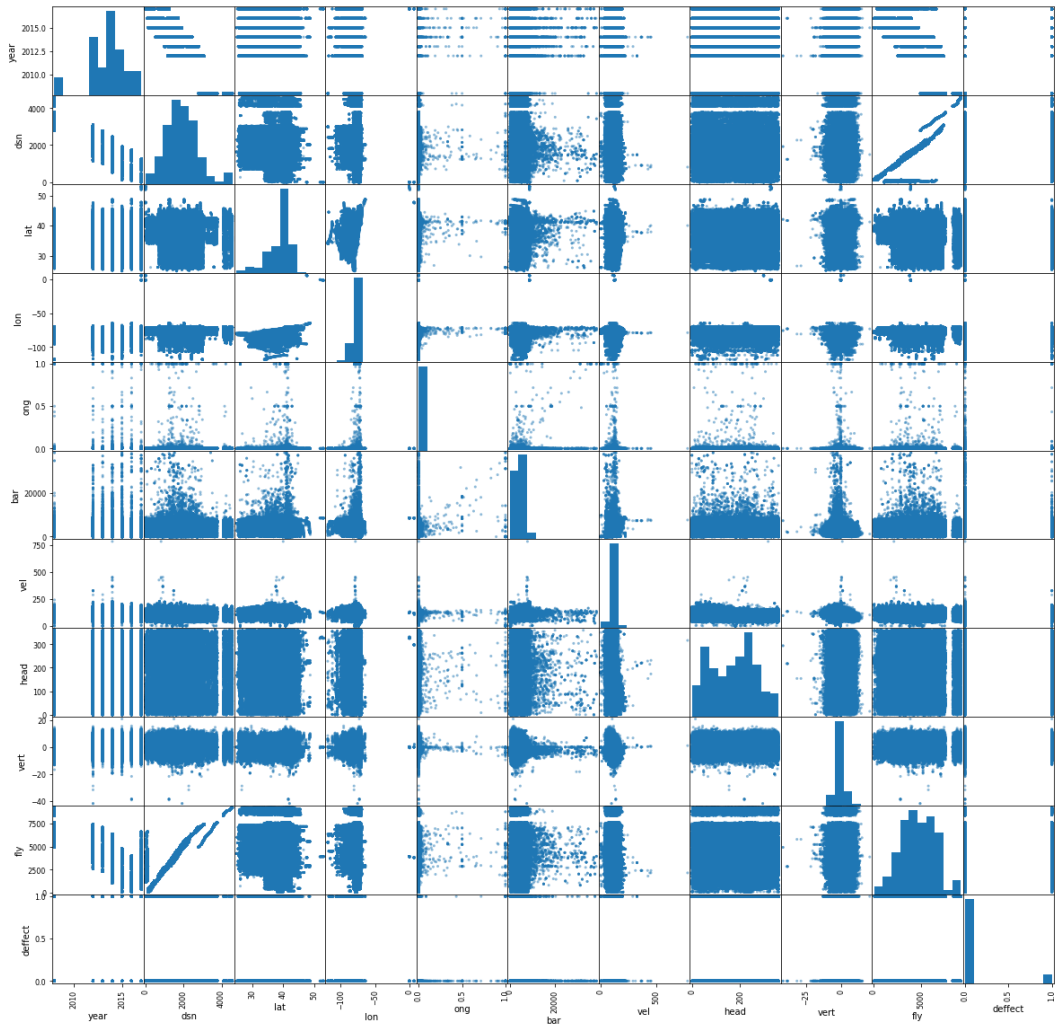


Figure 6.26: Flight data scatter plot

The meteorological data scatter plot (fig 6.27) shows only the variables related to the meteorological data. It is interesting to note that the variables that were discarded such as "snow" and "total sunshine" appear full of zeros in this figure too. This plot also shows that the dew point temperature and the air temperature are highly correlated, as expected. The rest of variables seem to be uncorrelated between one another.

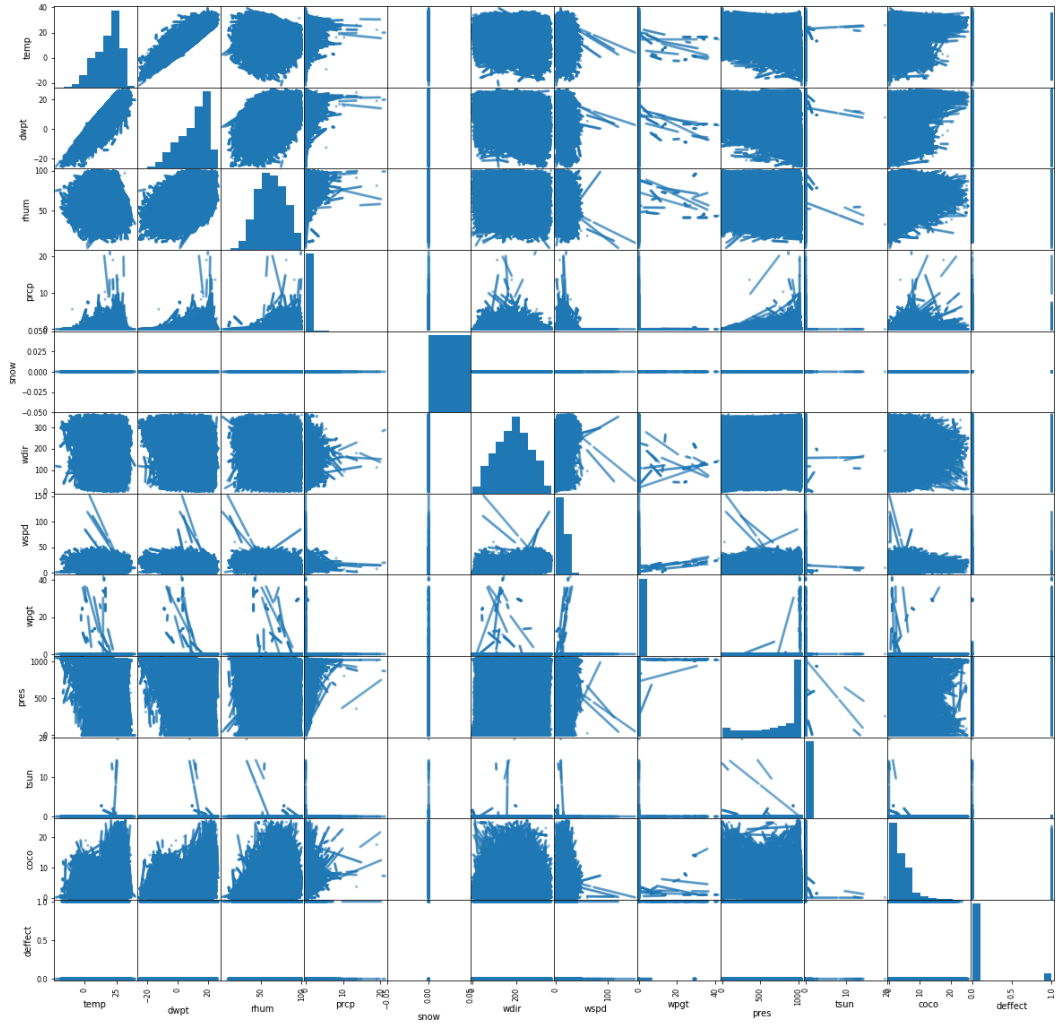


Figure 6.27: Meteorological data scatter plot

The last scatter plot (fig 6.28) displays the scattering of all the variables. This is done because, although it is very unlikely, it could be that some field from the flight data could have some kind of correlation with another field from the meteorological data. As seen in the figure, this is not the case. All the different fields seem uncorrelated enough.

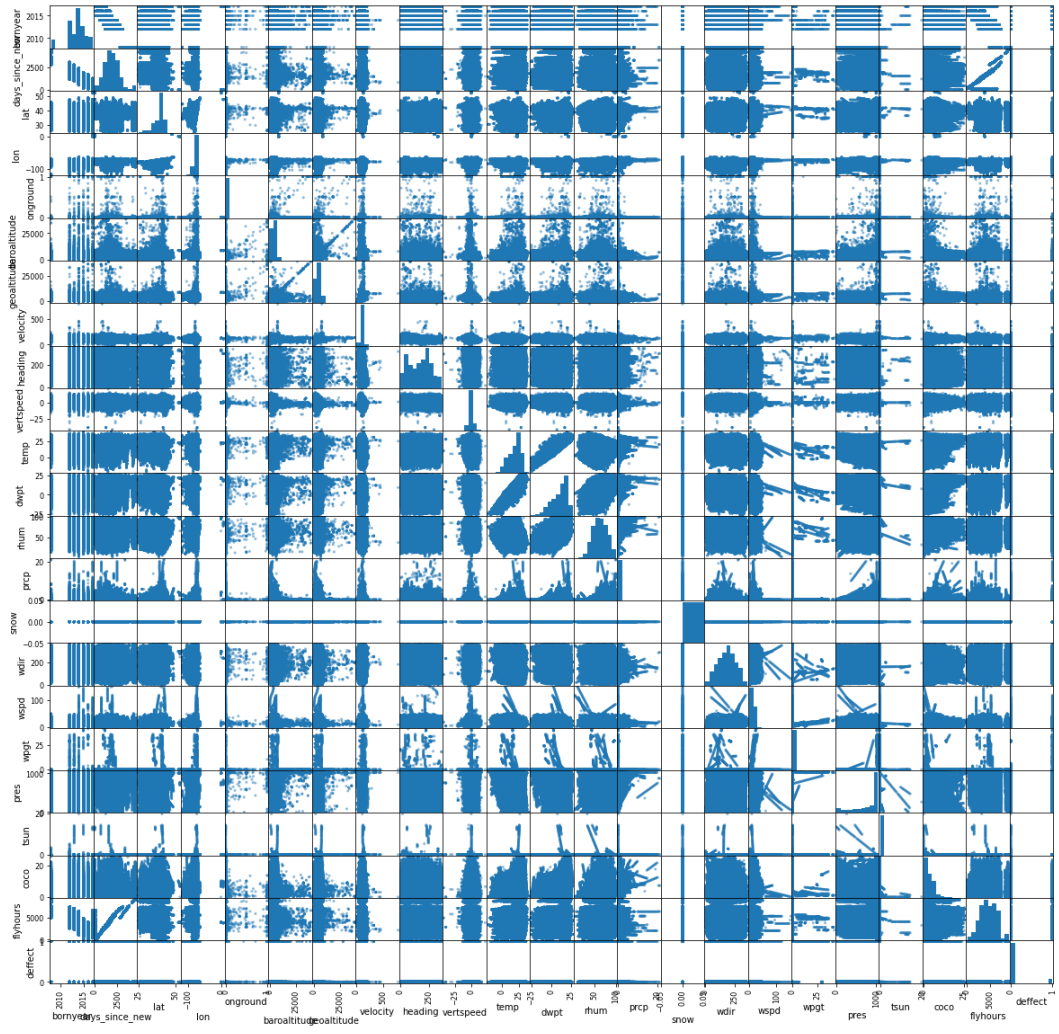


Figure 6.28: All data scatter plot

6.4.2 Data correlation plot

The flight data correlation plot (fig 6.29) basically corroborates the conclusions obtained from the scatter plot shown before, mainly that the variables "days since new" and "flight hours" present a high degree of correlation. On top of that, this figure also presents some additional information that was not detected in the scattering. The aforementioned two variables are inversely correlated to the variable "bornyear", which seams reasonable. There is also some small correlation between longitude and latitude, and other smaller correlation between variables so small that can be considered not meaningful.

The meteorological data correlation plot (fig 6.30) corroborates the observations obtained from the scatter plot, mainly that the fields of "temperature" and "dew point temperature" are highly correlated. Not much can be said from the other variables, since the correlation values shown in the matrix are not big enough to be deemed

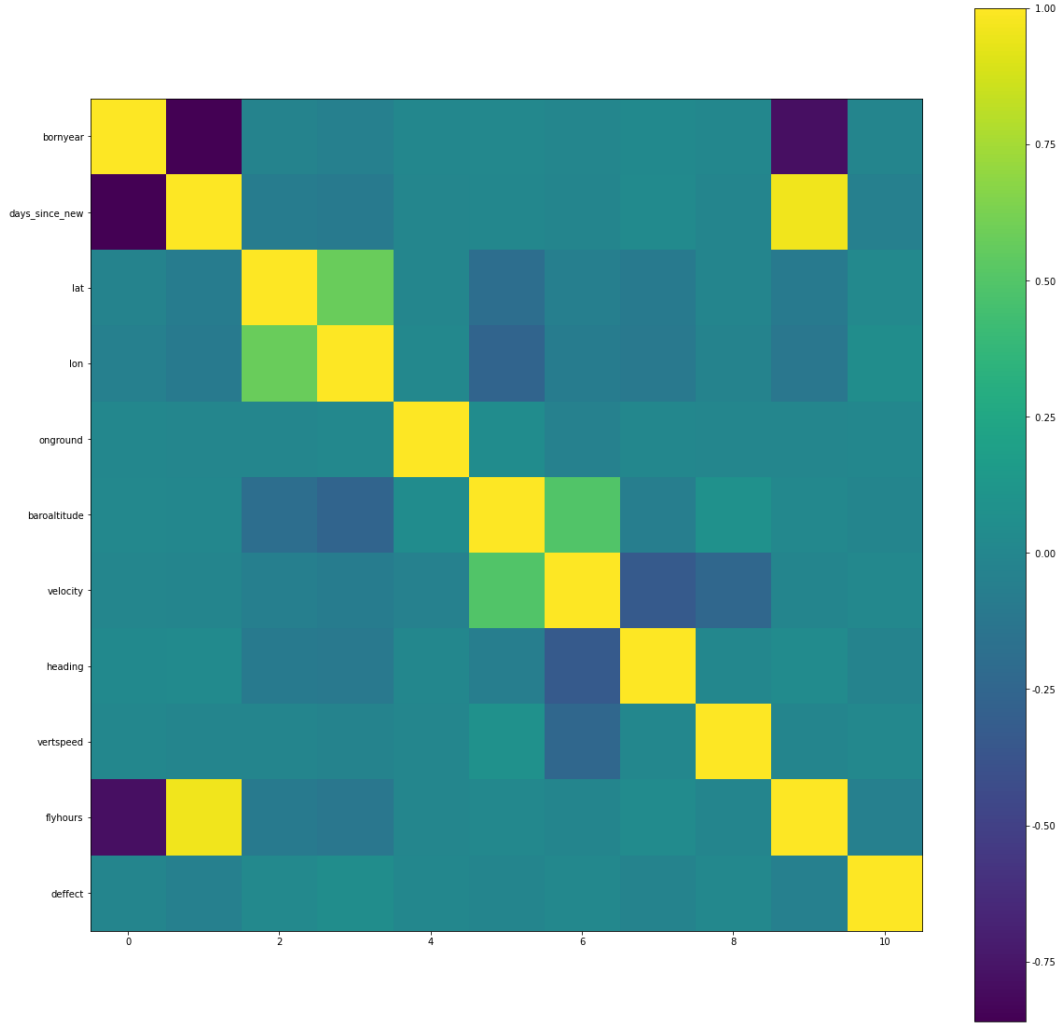


Figure 6.29: Flight data correlation plot

meaningful.

Similarly to what was explained before, a correlation plot with the entire dataset is also shown. Not much can be said that wasn't already known before. This plot shows no surprises, since there seems to be no meaningful correlation between variables from the flight data or the meteorological data.

All in all, after better analyzing the correlation between all the various fields of the dataset, it has to be concluded that the different decisions taken to create the new dataset seem to be correctly solid and reasonable. It does not look like there will be problems of variable correlation in the new dataset.

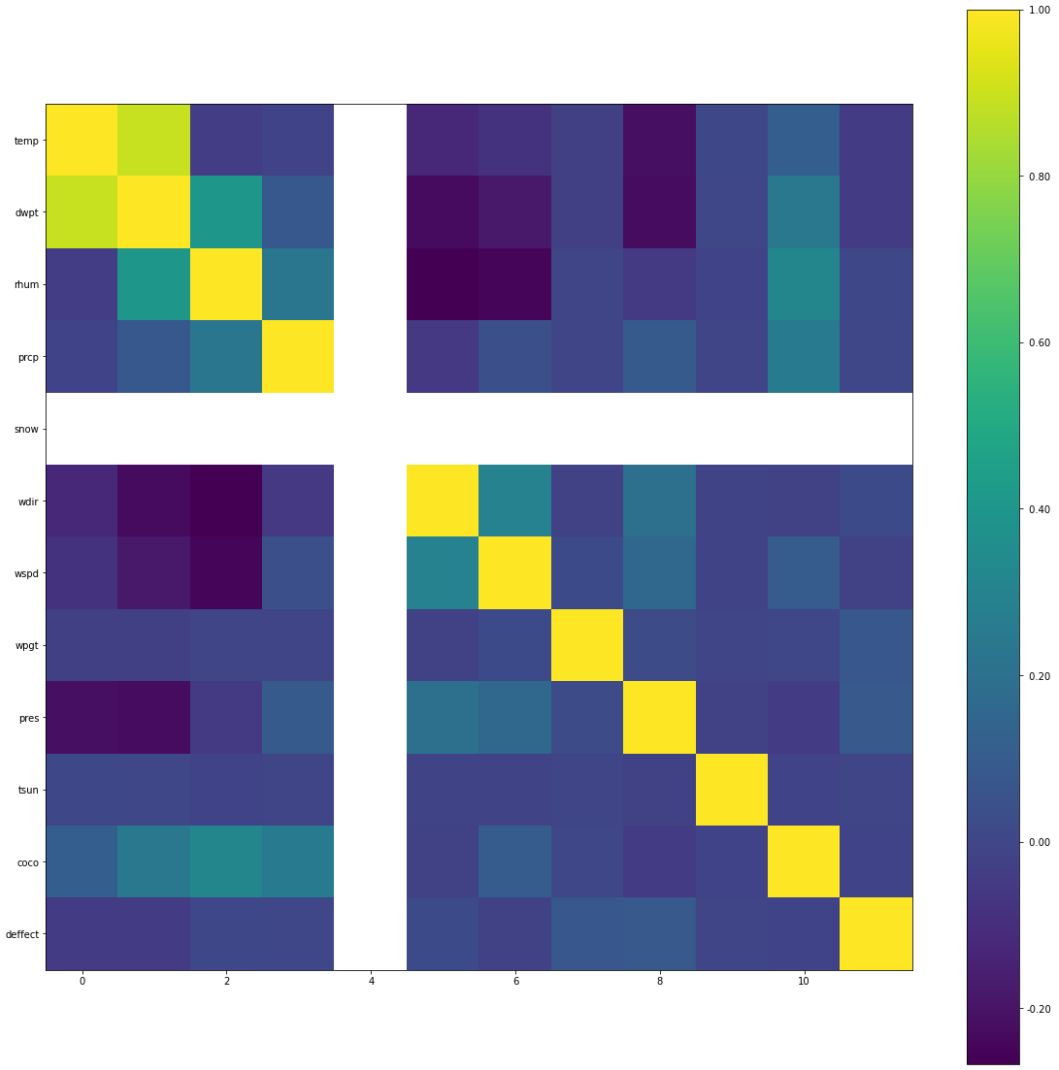


Figure 6.30: Meteorological data correlation plot

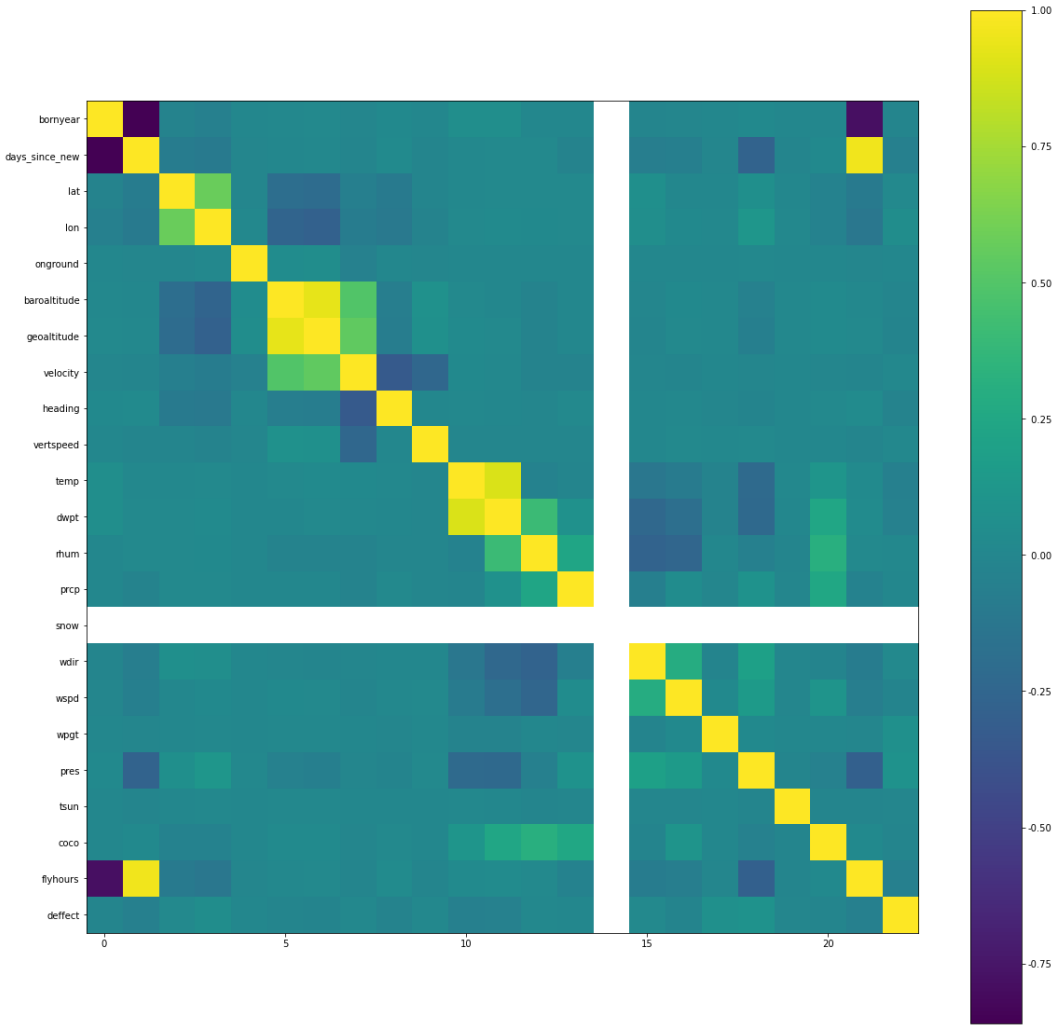


Figure 6.31: All data correlation plot

6.5 Data analysis conclusion

From the original database created during the course of the previous project, a new and improved database has been created by transforming the different fields and discarding useless variables. To do this, a more exhaustive analysis of the data has been performed. Now, it remains to be seen if this new database will produce better results than the old one.

Chapter 7: Data training and modeling

Creating a new dataset based on the previous one was the first action taken to improve the results obtained when predicting aircraft failures. The second action taken to improve the prediction performance was the creation of a new neural network, which was to be used to train a new model using the new database. This chapter presents the process of creating the new ANN and all the results obtained during said process.

7.1 Initial baseline

First of all, the baseline of the project has to be established. In other words, what are the results that should serve as a reference for the output of the algorithm. This is useful to assess how good are the results and whether the project can be considered successful or not.

At first glance, one could assert that the baseline of this project should be the results shown in the report of the previous project, since the goal of this project was to improve the performance of the predictive maintenance tool developed before. However, as explained on section 5.1, the results shown on the report were unable to be reproduced using the very same code and database. As such, instead of using those values, the baseline for this project will be the results that were recreated using the code and database from the previous project. These are shown again below.

Performance Indicators	**Test (%)**
Accuracy	30.450
Precision	50.214
Recall	51.482
F-score	50.840

Table 7.1: Replicated ANN performance

To reiterate, it was decided the results shown in the previous report were not to be used as the baseline of the project because they were un-replicable using the very same code and data. As such, since they can not be replicated, it was considered that they were superseded by the results obtained using the database and algorithm from the project.

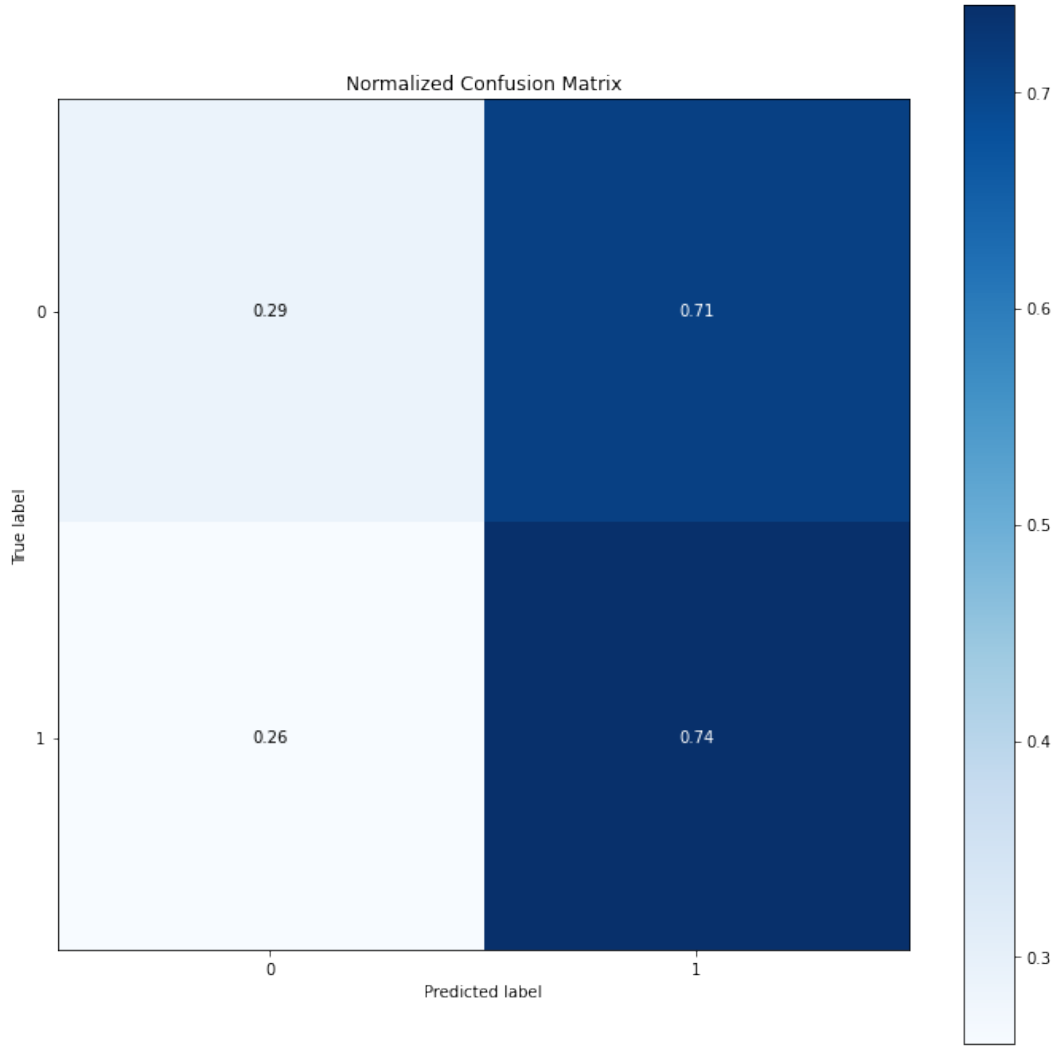


Figure 7.1: Replicated confusion matrix

It is also worth noting that, as explained on chapter 5, the results that one could expect should be much more conservative than those shown in the previous project. The report showed a trained model with excellent levels of precision and accuracy. It is not reasonable to expect a similar outcome due to the nature of the data. This is because the algorithm can only be expected to predict a small fraction of all the different defects, mainly those that occur on the exterior of the aircraft or on very sensitive components.

7.2 Previous ANN - new dataset

After obtaining the new dataset, the first thing that was tried was to input this new dataset into the code of the previous ANN. The performance indicators are shown on table 7.2 and the confusion matrix is shown on figure 7.2.

Performance Indicators	Test (%)
Accuracy	54.054
Precision	56.604
Recall	51.724
F-score	54.054

Table 7.2: confusion 1

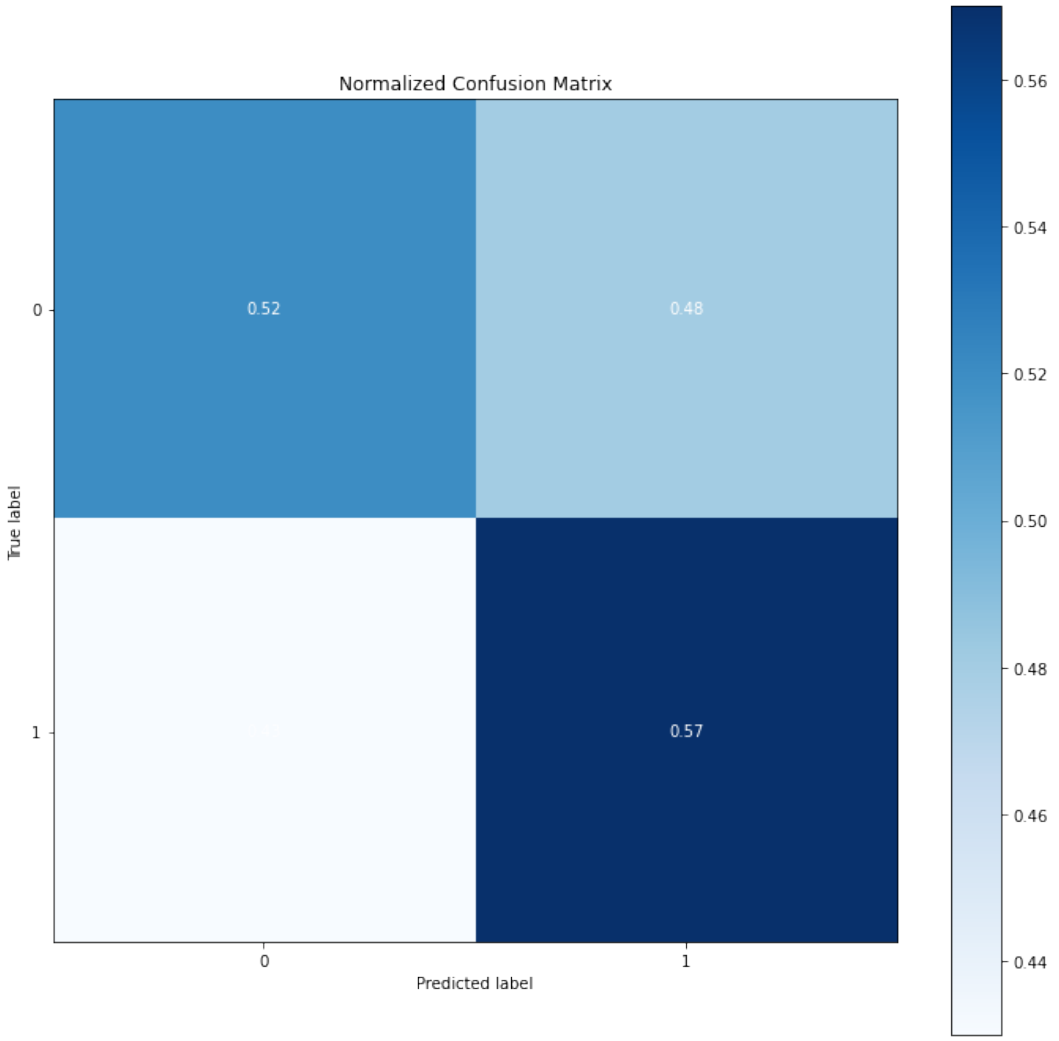


Figure 7.2: confusion 1

These results show a slight improvement in the performance of the defect predictions. Compared to the baseline, the precision and F-score have been slightly improved, while the recall value difference can be considered totally marginal. The accuracy of this model is slightly above 50%. Overall, it can be considered that slightly better results were obtained using the ANN of the previous project and the dataset created in this project.

7.3 Creation of the new ANN

The goal of this section is to explain in a general manner the process of creating a new ANN. Basically, what where the decisions made and the reasons behind them. Afterwards, the results will be shown and analyzed.

The network of the project consisted of two layers. First, a LSTM (Long-Short Term Memory) layer that was provided by the "pytorch" library. And second, a MLP (Multi-Layered Perceptron) layer formed of two fully connected layers. The first layer had ReLu as the activation function, while a Sigmoid was used as the activation function for the second layer.

Long Short Term Memory refers to a special type of recursive neural network. The thing that differentiates this type of network from other RNNs is the way it transforms the data. Unlike regular Recurrent Networks, which only perform a concatenation of the previous state of the network and afterwards a set of non-linear transformations, LSTM layers perform a total of five non-linear transformations to the data at each state of the network.

The functioning of the LSTM layers deserve further explanation. Basically, the network only performs a pointwise multiplication and a sum of vectors to an otherwise unaltered flow of data. The first multiplication computes how much of the information of the previous state is passed to the next state and how much of that information is lost or forgotten. This is decided by a "Sigmoid" that elaborates an output from zero to one. The addition of vectors computes how much new information will be added to the previous cell state. This is done by passing the input values through a "tanh" function and multiplying them by the values of another "Sigmoid" function, which works similarly to the first one. In a similar way, this function ponderates how much information will be actually added or lost to the new state. The result is a cell state that keeps changing subtly through time but keeps information about previous states until a certain point which happens when all the information from a previous state is lost.

Another recursion performed on the network occurs by keeping information of previous hidden states to compute the new ones. A hidden state is obtained by concatenating the previous hidden state with the input matrix, using a "Sigmoid" function and performing pointwise multiplication with a "tanh" function. The result is the hidden state, or the total output of the network if at the end of the recursion. This contains either information about all previous states or as many states as the cell state can store.

In the project, the network has been divided in two parts. The first one is the LSTM network from the Pytorch library. the second one is a regular Multilayer Per-

ceptron. Between these two parts a Dropout layer was added to prevent overfitting. The input size is a batch of 64x27 values and the RNN size is a matrix of 64x1024. Optimally, recursive part of the network should be as big as possible so it can store as much information about the past as necessary and extract many features from the data.

It was found that the length of the dataset too high. This meant that the dimension of the hidden layers was also too large. Since the intention was for the network to keep track of the previous observations, it was decided to train the network with fifty epochs. In total, this took 10 hours of computation. By observing some of the raw results, it could be understood that the network showed signs of some minor overfitting. Moreover, it was also observable some oscillation in the loss function. To solve this, the different parameters of the code were tweaked. After numerous iterations, the results obtained with the new ANN were considered acceptable.

7.4 New ANN - old dataset

After creating the new ANN, the next step that was taken was inputting the newly created ANN with the previous dataset. The results obtained are shown below, with the performance indicators being displayed on table 7.3 and the confusion matrix on figure 7.3.

Performance Indicators	**Test (%)**
Accuracy	52.553
Precision	53.642
Recall	47.929
F-score	50.625

Table 7.3: confusion 2

Compared to the baseline, it cannot be argued that this results present a meaningful improvement of the performance of the algorithm. Although it is true that both precision and accuracy have been somewhat increased from the reference values, they barely surpass the 50% value, meaning that they are little better than a coin-flip. However, the values of recall and F-score are lower than those shown in the baseline. This difference is not very big but it is noticeable. For this reason, it has to be concluded that the results obtained using the new ANN and the old dataset do not demonstrate a meaningful improvement or worsening of the algorithm performance.

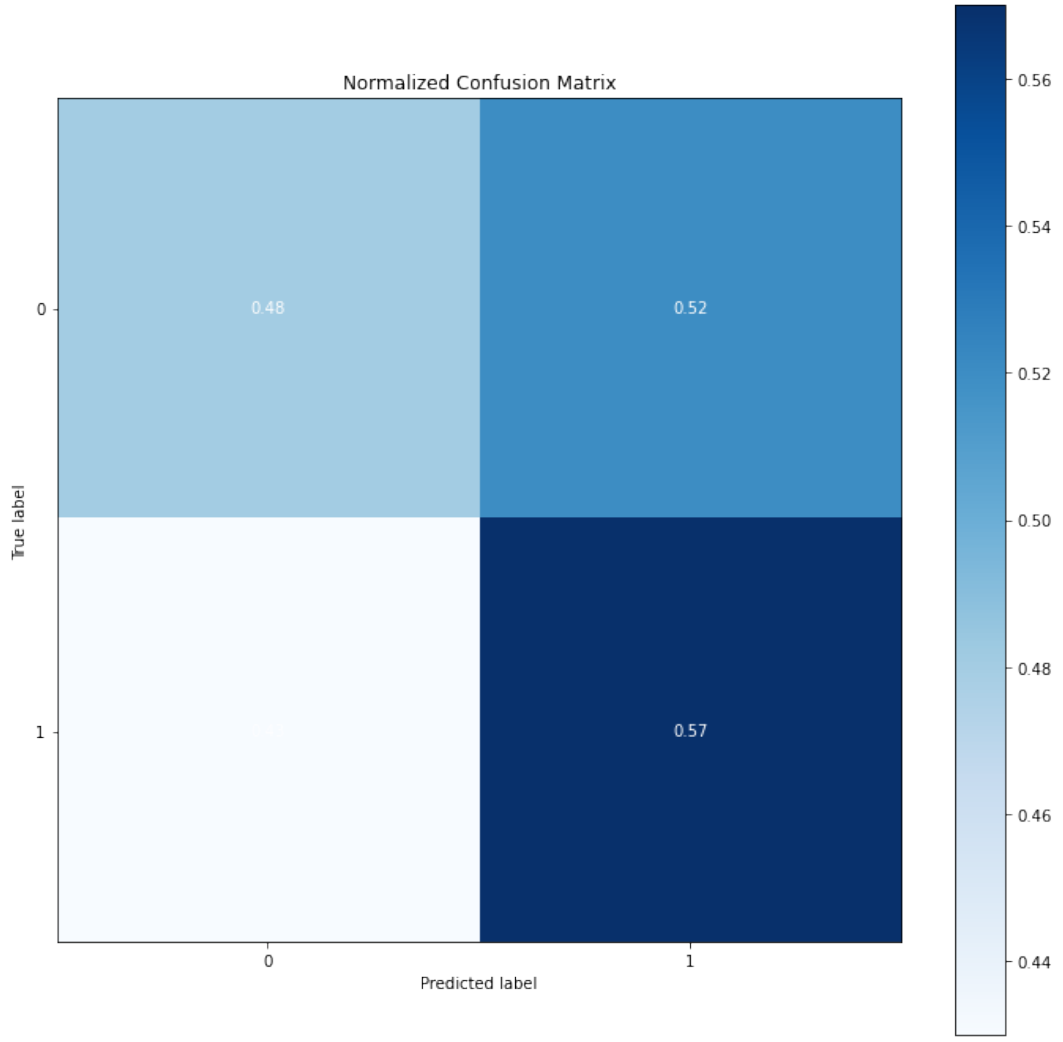


Figure 7.3: confusion 2

7.5 New ANN - new dataset

Finally, the last thing that was tried was using the new dataset and the new ANN together. The performance indicators are shown on table 7.4 and the confusion matrix is shown on figure 7.4.

The results obtained from the database and ANN created for this project are the best among all the different combinations. Compared to the baseline, the performance is considerably improved, with performance values that rise above 60% on all values. It has to concluded that the model trained with the new ANN and with the new dataset provides the best performance.

Performance Indicators	Test (%)
Accuracy	60.360
Precision	65.929
Recall	60.081
F-score	62.869

Table 7.4: confusion 3

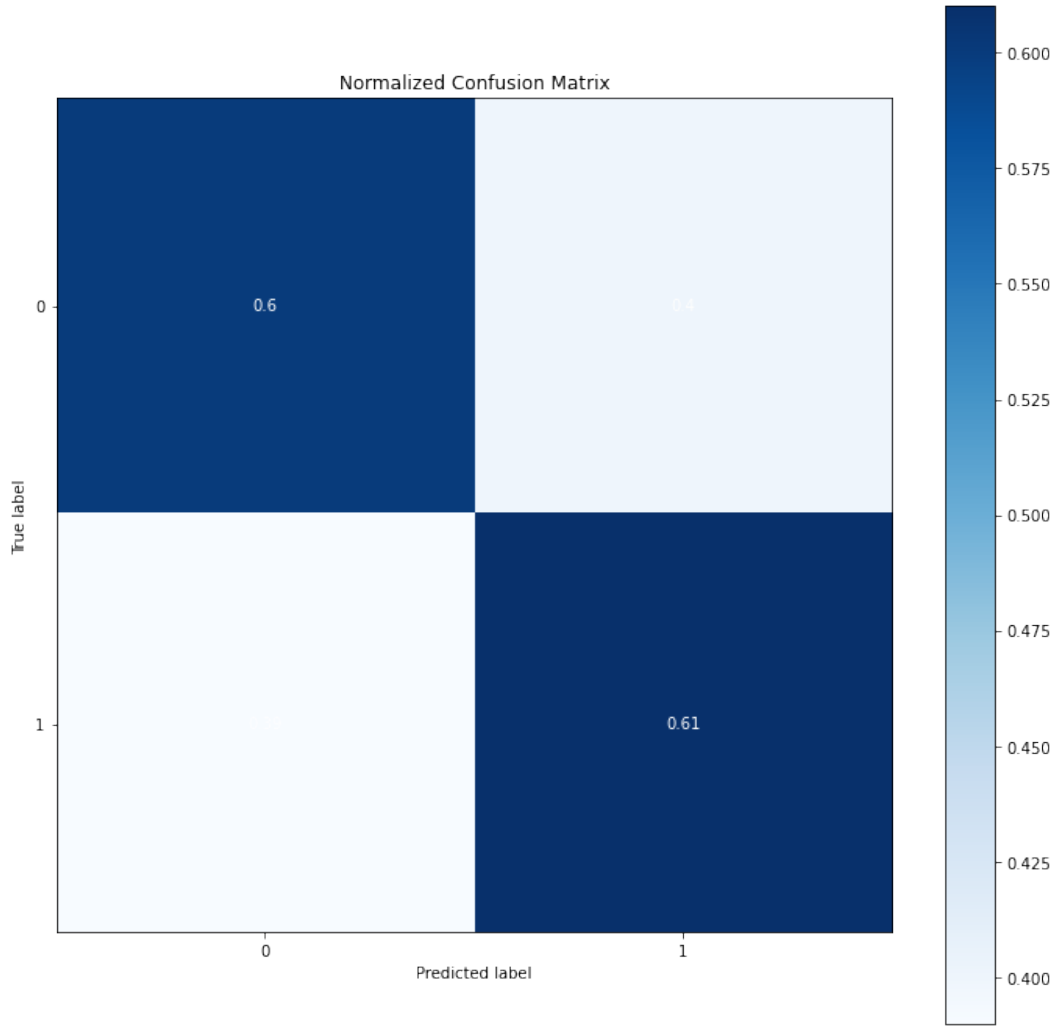


Figure 7.4: confusion 3

7.6 Conclusions on modeling

After training and modeling, it has to be concluded that both the creation of the new dataset and the creation of the new ANN have been successful. The performance has been improved when compared to the baseline.

One might argue that the precision of these models seem very low, since the

best results only barely surpass an accuracy of 60%. However, it has to be considered that, as explained before, with the data available to this project, only a small fraction of the total number of defects is expected to be found by the algorithm.

In other words, the implementation of the new dataset and ANN have managed to basically obtain more information usable to predict defects than before. It can be argued that it is increasingly difficult to improve the performance of the modeling with the available data. All the juice has been extracted from the database.

Chapter 8: Conclusions

At the beginning, the intention of this project was pretty clear. Build upon the previous project, use what was already obtained and developed to increase the performance of defect detection, implement a first working version of the interface and obtain new and better data to improve the algorithm training.

First of all, the implementation of the predictive maintenance tool as a new module in Robin can be considered successful. This was built with intend in mind to be easily changed, so when a new deep learning algorithm is developed it can be easily ported to the interface. When the algorithm is finally ready to be commercialized, it will be very simple and fast to transport it to the interface where the user will end up using it. Although this part of the project took many hours of design and coding, and is of huge importance for the company, it was decided to limit its size in the report in order to focus on more relevant matters.

Secondly, it was tried on many different occasions to obtain more data to feed the deep learning algorithm. The more data available to train the algorithm, the better results. Many actions were taken with many different partners in order to obtain real aircraft data from manufacturers and operators. However, it was impossible to conclude an agreement with any client by the end date of the project (there are clients who have already agreed to provide us with more data in the coming months). It was decided to omit this part from the report even if many hours were put towards this matter because in the end there is no result to show for.

Thirdly, at a mid-point in the project, many different issues started to arise from the previous project. On the surface, the results obtained and shown in the report were outstanding. However, there was no way to replicate the using the same code and data. On closer inspection, the data collected and code developed for the previous project were found to contain several important problems. From all of this, it was concluded that the project was to be developed basically from the bottom up if it was to be solid. If this project is to be built upon, its bases have to be rock solid. In a certain kind of way, the previous project was found to be a huge structure that had some very flimsy bases.

Fourthly, after closely analysing the database from the previous project, many ways to improve the algorithm performance were found. By applying all these different improvements on the database, a new dataset was obtained by transforming various fields and variables from the old dataset. In other words, the present on the database was the same, however it was reorganized and rearranged so the deep

learning algorithm would provide a better performance. The way variables are presented to the algorithm have proven to heavily influence the effectivity of training and modelling.

Fifthly, a new deep learning was developed from scratch. This new training algorithm was carefully created to optimize the performance in defect detection. The results obtained from this new ANN and the new database were considerably better than those obtained from the other sources. Although the precision of the model was not perfect, it can be argued that it detected all the possible defects that could be found with the data that was provided.

And finally, to sum up, the previous project displayed an excellent facade, with some amazing results. However, this was created on top of flimsy foundations. By basically re-starting the work from the ground up, this project was able to create a strong base. Future work on this predictive maintenance tool will enjoy solid foundations, which are crucial to build a project with the scale of this one.

Bibliography

[1] Lovejinder Singh Kaur. Study of the statistical correlations for aircraft reliability prediction. Master's thesis, UPC, Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa, Departament de Física, May 2021.

[2] Robin RAMS. https://robinrams.com. Accessed on May 2022.

[3] Mean Time Between Failures. https://en.wikipedia.org/wiki/Mean_time_between_failures. Accessed on May 2022.

[4] R Keith Mobley. *An introduction to predictive maintenance*. Elsevier, 2002.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[6] James Kovacevic. How equipment fails, understanding the 6 failure patterns. https://hpreliability.com/how-equipment-fails-understanding-the-6-failure-patterns/. Accessed on September 2022.

[7] Wilbur wright weeps. aviator's brother grieved by fatal accident. deplores officer's death. first thought is safety of passengers, he says, when news of orville's mishap at fort myer reaches him in france. countermands orders for flights, to regret of waiting crowd. https://www.newspapers.com/newspage/28878483/. September 19, 1908. Washington Post.

[8] YOSHIHIKO Tanaka, SHINICHI Nagai, MASANORI Ushida, and TSUYOSHI Usui. Large engine maintenance technique to support flight operation for commercial airlines. *Mitsubishi Heavy Industries Technical Review*, 40(2):5, 2003.

[9] Alessandro Birolini. *Reliability engineering: theory and practice*. Springer Science & Business Media, 2013.

[10] Joel Levitt. *Complete guide to preventive and predictive maintenance*. Industrial Press Inc., 2003.

[11] Press release: Sas picks skywise predictive maintenance for a320s. https://runwaygirlnetwork.com/2022/04/sas-skywise-predictive-maintenance-a320s/. Accessed on May 2022.

[12] Django Software Foundation. Django documentation. https://docs.djangoproject.com/en/4.1/. Accessed on September 2022.

[13] Ryan L. Melvin. Sample size in machine learning and artificial intelligence. https://sites.uab.edu/periop-datascience/2021/06/28/sample-size-in-machine-learning-and-artificial-intelligence/. Accessed on August 2022.

[14] Eric Baum and David Haussler. What size net gives valid generalization? In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

[15] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. Accessed on September 2022.

[16] Donald Galler and George Slenski. Causes of aircraft electrical failures. *Aerospace and Electronic Systems Magazine, IEEE*, 6:3 – 8, 09 1991.