Real-time anomaly detection with in-flight data : streaming anomaly detection with heterogeneous communicating agents

Nicolas Aussel

université PARIS-SACLAY

TELECOM SudParis

NNT : 2019SACLL007

Thèse de doctorat

Real-time anomaly detection with in-flight data: streaming anomaly detection with heterogeneous communicating agents

Thèse de doctorat de l'Université Paris-Saclay
préparée à Télécom SudParis

Ecole doctorale n°580 Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Évry, le 21 juin 2019, par

NICOLAS AUSSEL

Composition du Jury :

Mme Mathilde Mougeot Professeure, ENSIIE	Présidente
M. Mustapha Lebbah Maître de Conférences HDR, Université de Paris 13	Rapporteur
M. Pierre Sens Professeur, Université de Paris 6	Rapporteur
M. Éric Gressier-Soudan Professeur, CNAM	Examinateur
Mme Sophie Chabridon Directrice d'Études, Télécom SudParis	Directrice de thèse
M. Yohan Petetin Maître de Conférences, Télécom SudParis	Co-encadrant de thèse

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Détection des anomalies sur les données en vol en temps réel avec des agents communicants hétérogènes

Mots clés : Maintenance prédictive, apprentissage réparti, apprentissage automatique, aéronautique

Résumé : Avec l'augmentation constante du nombre de capteurs embarqués dans les avions et le développement de liaisons de données fiables entre l'avion et le sol, il devient possible d'améliorer la sécurité et la fiabilité des systèmes aéronautiques à l'aide de techniques de maintenance prédictive en temps réel. Cependant, aucune solution architecturale actuelle ne peut s'accommoder des contraintes existantes en terme de faiblesse relative des moyens de calcul embarqués et de coût des liaisons de données.

Notre objectif est de proposer un algorithme réparti de prédiction de pannes qui pourra être exécuté en parallèle à bord d'un avion et dans une station au sol et qui fournira des prédictions de panne à bord en quasi-temps réel tout en respectant un budget de communication. Dans cette approche, la station au sol dispose de ressources de calcul importantes ainsi que de données historiques et l'avion dispose de ressources de calcul limitées et des données de vol récentes.

Dans cette thèse, nous étudions les spécificités des données aéronautiques, les méthodes déjà développées pour prédire les pannes qui y sont associées et nous proposons une solution au problème posé. Nos contributions sont détaillées en trois parties principales.

Premièrement, nous étudions le problème de la prédiction d'événement rare, conséquence de la haute fiabilité des systèmes aéronautiques. En effet, de nombreuses méthodes d'apprentissage et de classification reposent sur des jeux de données équilibrés. Plusieurs approchent existent cependant pour corriger les déséquilibres d'un jeu de données. Nous étudions leurs performances sur des jeux de données extrêmement déséquilibrés et démontrons que la plupart sont inefficaces à ce niveau de déséquilibre.

Deuxièmement, nous étudions le problème de l'analyse de journaux d'événements textuels. De nombreux systèmes aéronautiques ne produisent pas des données numériques faciles à manipuler mais des messages textuels. Nous nous intéressons aux méthodes existantes basées des scripts ou sur l'apprentissage profond pour convertir des messages textuels en entrées utilisables par des algorithmes d'apprentissage et de classification. Nous proposons ensuite notre propre méthode basée sur le traitement du langage naturel et montrons que ses performances dépassent celles des autres approches sur un banc d'essai public.

Enfin, nous proposons une solution d'apprentissage réparti pour la maintenance prédictive en mettant au point un algorithme s'appuyant sur les paradigmes existants de l'apprentissage actif et de l'apprentissage fédéré. Nous détaillons le fonctionnement de notre algorithme, son implémentation et démontrons que ses performances sont comparables avec celles des meilleures techniques non réparties.

Title : Real-time anomaly detection with in-flight data: streaming anomaly detection with heterogeneous communicating agents

Keywords : Predictive maintenance, distributed learning, machine learning, aeronautics

Abstract : With the rise of the number of sensors and actuators in an aircraft and the development of reliable data links from the aircraft to the ground, it becomes possible to improve aircraft security and maintainability by applying real-time analysis techniques. However, given the limited availability of on-board computing and the high cost of the data links, current architectural solutions cannot fully leverage all the available resources limiting their accuracy.

Our goal is to provide a distributed algorithm for failure prediction that could be executed both on-board of the aircraft and on a ground station and that would produce on-board failure predictions in near real-time under a communication budget. In this approach, the ground station would hold fast computation resources and historical data and the aircraft would hold limited computational resources and current flight's data.

In this thesis, we study the specificities of aeronautical data and what methods already exist to produce failure prediction from them and propose a solution to the problem stated. Our contribution is detailed below in three main parts.

First, we study the problem of rare event prediction created by the high reliability of aeronautical systems. Many learning methods for classifiers rely on balanced datasets. Several approaches exist to correct a dataset imbalance and we study their efficiency on extremely imbalanced datasets and demonstrate that most of them are ineffective at this scale.

Second, we study the problem of log parsing as many aeronautical systems do not produce easy to classify labels or numerical values but log messages in full text. We investigate existing methods based on a statistical approach and on Deep Learning to convert full text log messages into a form usable as an input by learning algorithms for classifiers. We then propose our own method based on Natural Language Processing and show how it outperforms the other approaches on a public benchmark.

Last, we offer a solution to the stated problem by proposing a new distributed learning algorithm that relies on two existing learning paradigms, namely Active Learning and Federated Learning. We detail our algorithm, its implementation and demonstrate that its performances are comparable with those of existing non-distributed methods.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Contents

List of Figures

List of Tables

Glossary

Acronym	Description
ADMM	Alternating Direction Method of Multipliers
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DT	Decision Tree
EASA	European Union Aviation Safety Agency
FAA	Federal Aviation Agency
GBT	Gradient Boosted Tree
GD	Gradient Descent
GPU	Graphical Processing Unit
HDD	Hard Drive Disk
HDFS	Hadoop File System
IoT	Internet of Things
IPLoM	Iterative Partitioning Log Mining
IID	Independent and Identically Distributed
L-BFGS	Limited memory Broyden-Fletcher-Goldfarb-Shanno
LCS	Longest Common Subsequence
LDA	Latent Dirichlet Allocation
LR	Logistic Regression
LSTM	Long Short-Term Memory
MIL	Multiple Instance Learning
ML	Machine Learning
MTBF	Mean Time Before Failure
NLP	Natural Language Processing
PCA	Principal Component Analysis
RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Unit
RF	Random Forest
SGD	Stochastic Gradient Descent
SMOTE	Synthetic Minority Oversampling TEchnique
SVM	Support Vector Machine

Chapter 1

Introduction

Contents

This chapter presents the problematic of the thesis, offers a definition of the general and industrial concepts and details the roadmap that was followed for the completion of the thesis.

Section1.1 details the general context of the thesis and the problematic it is meant to address.

The next section 1.2 starts with a characterization of aeronautical systems with a focus on the specificities that justify the need to study them separately from other industrial systems. Then, an explanation of the concept of predictive maintenance for industrial systems and the reason why Machine Learning is considered particularly promising to achieve it are given.

Finally, by cross-examining the problematic of the thesis, the specificities of aeronautical systems and the contributions of Machine Learning, the essential steps of the thesis are identified and the contribution roadmap formalised in section 1.3.

1.1 General context and problem statement

The goal of this thesis is to propose a real-time predictive maintenance system adapted to the specific challenges of aeronautical systems. The corporation Safran, of which my hosting company TriaGnoSys is a subsidiary, is an aeronautical equipment supplier and , in this capacity, has many integrated sensors deployed in a variety of aircraft. The measurements made by the sensors during the flight are collected and stored after the flight. Currently, they are usually

analysed by an operator if and only if an anomaly is detected.
In order to improve the reliability of our equipments and to reduce maintenance costs, it would be preferable to make use of those measurements to predict anomalies before they happen. Several aeronautical constructors such as Airbus and Boeing have already adopted this approach by using Machine Learning in order to deal with the vast amounts of data but in a strictly offline manner, that is to say that they make use of their data only after landing. Some specificities of aeronautical systems, detailed in the next section, already need to be addressed at this stage. This application is not trivial and though predictive maintenance models for aeronautical systems already exist, the detail of their implementation is not publicly available.
To extend this work further, the goal of this thesis is to set a predictive maintenance model for aeronautical systems that can operate during the flight. This would increase the security and reliability of aeronautical systems and improve the organization of the maintenance operations after landing. The main difference imposed by this goal concerns the computation resources. Offline models can use as many computation servers as necessary in the very favourable context of a data center whereas a model used in flight must use the on-board resources and follow a number of constraints to be allowed on board. Those constraints are imposed by aeronautical certification agencies and include, among others, fault tolerance, high reliability and isolation with regards to other on-board software in case of malfunction.
The solution to those limitations that is investigated in this thesis is the use of a data link between the aircraft and the ground. Indeed, it is becoming the norm for commercial flights to offer connectivity with the Internet and it is one of the services in which the hosting company is specialised in. This connectivity can be achieved in different ways with different kind of satellite connections or through direct air-to-ground communication. Figure 1.1 presents a simplified representation of the systems involved.
The issue that remains is that, no matter what connectivity solution is used, compared to ground connections, the data link between the aircraft and the ground is always financially more expensive, has a very limited bandwidth and a limited availability. Therefore, transmitting all the measurements made by the on-board sensors to the ground following a cloud computing paradigm is not realistic. As such, some computations have to take place on-board to transmit only relevant data. The core of this thesis is to determine an ensemble of two learning models, one on board and the other on the ground that can collaborate on the anomaly prediction problem and a software architecture able to support them in order to produce accurate estimates of anomaly risks on board without saturating the data link.

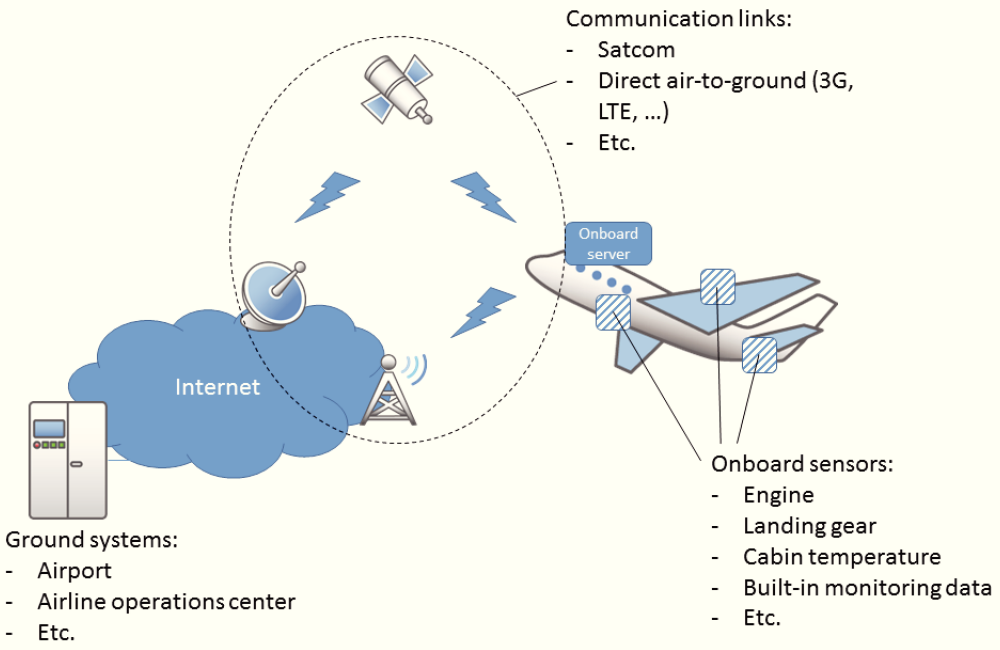


Figure 1.1: Overview of aeronautical components

1.2 Definition of general and industrial concepts

1.2.1 Characterization of aeronautical systems

By definition, we call system a group of interdependent devices that form a unified whole. By extension, an aeronautical system would simply be a system that is found in an aircraft. While being semantically accurate, this definition overlooks important aspects of modern avionics systems and would imply that fitting a system designed for a ground application in an aircraft is sufficient for it to become an aeronautical system which is inaccurate.

The aeronautics supplier industry is heavily regulated by multiple national or international agencies throughout the world. Some of the most well-known are the Federal Aviation Agency (FAA) that regulates aeronautical systems in the United States[1] and the European Union Aviation Safety Agency (EASA) for the European Union[2]. In order to be allowed on board of an aircraft in the aerial space under the jurisdiction of those agencies, it is mandatory for an aeronautical system to be certified. The details of the certification process and the requirements vary depending on the agency and on the criticality of the aeronautical system, in the sense that the

[1] https://www.law.cornell.edu/cfr/text/14/part-25
[2] https://www.easa.europa.eu/sites/default/files/dfu/Easy%20Access%20Rules%20for%20Part-21.pdf

more important a system is to the safety of the aircraft, the higher the requirements will be. The exact steps needed to certify an aeronautical system will therefore vary slightly from one system to another and are outside the scope of this thesis however there are enough similarities in the certification process to lead to similarities in the aeronautical systems and that is of interest for this work.

- The first and most obvious requirement, characteristic of aeronautical systems, is their high reliability. Reliability in this case is defined as $1 - probability\ of\ failure\ over\ a\ period\ of\ time$. Mean Time Before Failure (MTBF) is often used as a way to express the expected lifetime of an aeronautical system based on its reliability.
- A second specificity of aeronautical systems induced by the certification process is the need to closely monitor their performances of aeronautical system and ensure the conservation of the data obtained through this monitoring. This is required in order to demonstrate that the aeronautical system meets the certification specifications.
- A third specificity of aeronautical systems is the difficulty to change them. The certification process is costly, both financially and in terms of time, and introducing changes to a system during the process creates additional costs. Similarly, an update to a previously qualified system also has to go through a certification process, though simpler than the process for a new system. Because of this, there is a considerable delay between the design of an aeronautical system and its actual implementation in the field.
- Lastly, though desirable, the goal of perfect isolation of aeronautical systems from each other is, by design, impossible to achieve as they are co-located with every other onboard system and have to share resources such as power supply. Aeronautical systems are always interdependent at some level.

1.2.2 Predictive maintenance

As explained in the problem statement, the goal of this thesis is to achieve a better predictive maintenance for aeronautical systems. Predictive maintenance is a popular topic for industrial applications as a way to increase reliability and reduce costs and waste. Its principle is to monitor the condition of the system to maintain in order to repair or replace it just in time before it fails. There are two other approaches to maintenance.

The first one, corrective maintenance, replaces a system after it has started malfunctioning. It has the advantage of minimizing the number of required maintenance operations but does not allow for the planning of those operations and, in the case of interdependent systems, for example in an assembly line, the consequences of a dysfunction can have an impact beyond the system to maintain that might not be acceptable, like a complete halt of production in the case of the assembly line.

The second approach, called preventative maintenance, replaces and repairs systems in a scheduled manner based on statistics of their reliability. This approach is favoured for critical systems where the consequences of a failure are considered unacceptable such as medical systems or for systems where the cost of a planned maintenance is significantly lower than an unplanned one. For the example of an assembly line, making a planned maintenance when the line is not being operated for example. The preventative approach however leads to the repair and replacement of systems that are still in working condition which implies that too many maintenance operations are taking place and that there is a waste of spare parts. It is also possible that failures happen before the scheduled maintenance as the condition of the system is not actively monitored so the number of unplanned maintenance operations is reduced but is not zero.

In contrast, the predictive maintenance approach aims at replacing parts only when it is necessary like in the corrective approach but without letting the system fail like in the preventative approach by modelling its behaviour based on monitoring data. When perfect accuracy is not possible, it is also often possible depending on the model used to balance the sensitivity of the prediction with regards to the relative cost of planned versus unplanned maintenance. It has the downside of requiring the implementation of sensors to monitor the equipment. The table 1.1 summarizes the strengths and drawbacks of the three maintenance approaches identified.

Maintenance approach	# of maintenance operations	impact on operations	complexity
Corrective	$+$	$-$	$++$
Preventative	$-$	$+$	$+$
Predictive	$++$	$++$	$-$

Table 1.1: Summary of maintenance paradigms

Predictive maintenance is therefore very desirable and, in theory, superior to the other approaches when sensors are easily implementable. It is also however technically more complex as it relies on a failure prediction model that is potentially difficult to obtain whereas corrective maintenance only requires to detect the malfunction and preventative relies on the system life expectancy. The failure prediction model can be obtained from a formal analysis of the system by a domain expert in the order to determine what can be deduced about the state of the system from the monitored parameters. However, when studying the interactions of multiple interdependent systems, the complexity of a formal analysis quickly becomes intractable with the number of potential cross-system interactions. Such an approach can still be undertook at considerable costs for extremely critical applications to ensure, for example, the safety of nuclear power plant installations but such endeavours are not possible for every industrial system.

Even though formal analysis are not always possible, there is another approach that consists in studying statistical correlations between the system failure events and the monitored parameters. Until recently, this approach required considerable computation resources to achieve

acceptable levels of accuracy on complex systems. With the latest advances in Machine Learning applications, it is however becoming increasingly simple to sift through amounts of data that would have been considered intractable a decade earlier.

1.3 Presentation of the thesis

In the two previous sections, we have identified the goal of the thesis, the specificities of the aeronautical application field and the most suitable approach to create the failure prediction model. Based on this, we identify the problems that this thesis needs to address in order to reach a conclusion and structure it around these problems.

A natural first step as a preamble to this work is to present the background that it is necessary to follow the contributions presented therein. As such, chapter 2 contains a synthetic overview of the state of the art of the domain.

Then, the first problem that we identify is tied to the fact that aeronautical systems are reliable. While obviously being a desirable trait for any industrial system, it also carries the implication that there are few examples of failure to observe. That is a problem from a Machine Learning perspective because for the statistical model to be accurate, it requires as many examples of failures as possible. If the failures are too rare, it will be difficult to generalize correctly the model from the observations. A simple example and a common situation for aeronautical systems such as engines or navigation sensors would be a critical system that only had a single failure in several years of operation. Because of the criticality of the system, a failure prediction model as accurate as possible is desired but since there is only a single known example of failure it is not possibly to obtain a statistically conclusive result. A second aspect of this problem is the balance aspect. Many Machine Learning classification algorithms are designed to learn from a balanced dataset, that is to say a dataset where the classes to discriminate are in equal proportions. For aeronautical systems, because of the scarcity of failures, this ratio can typically be in the range of $1{:}100,000$. A first step in the thesis is therefore to understand the extent of this problem and study which methods or combination of methods from Machine Learning can operate on aeronautical systems in this situation of rare events prediction. We do so in chapter 3. The results of this contribution have been published in the proceedings of the 16$^{\text{th}}$ IEEE International Conference on Machine Learning and Applications (ICMLA 2017) as "Predictive models of hard drive failures based on operational data" by Aussel, N., Jaulin, S., Gandon, G., Petetin, Y., Fazli, E. and Chabridon, S. [Aussel et al., 2017].

The second problem that we need to acknowledge is that because of the costs, financial and in terms of time, associated with updating aeronautical systems it is not reasonable to make changes necessary for them in order to enable the work of this thesis. To put it in another way, we cannot decide which parameters to monitor for the failure prediction model. We have to use the parameters that are already monitored. Otherwise the delay in the update process and the time needed to gather a sufficiently large sample that contain multiple failures would widely

exceed the duration of the thesis. Nevertheless a positive aspect is that aeronautical systems are already closely monitored so there is already data to work with but the data collection process cannot be changed. A second important step in the thesis is therefore to study how to adapt Machine Learning method to the pre-existing data format. We conduct this study and propose our own approach in chapter 4. The results of this contribution have been published in the proceedings of the 26th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2018) as "Improving Performances of Log Mining for Anomaly Prediction Through NLP-Based Log Parsing" by Aussel, N., Petetin, Y. and Chabridon, S. [Aussel et al., 2018].

Finally the last problem to address in the thesis is the distributed learning between the aircraft and the ground station. Here the contribution is a new algorithm that can be run in parallel on one host with real-time data sources and limited computational power, the aircraft, and another host with high computational power but no access to recent data, the ground station. The requirements of the algorithm are to be able to run with a limited communication budget and to offer guarantees on the accuracy of its decision on the aircraft side. This is done in chapter 5.

Chapter 2

Fundamentals and State of the Art

Contents

2.1 Introduction

In this chapter, we present a synthetic review of related work. This review is organised by themes in three sections. The first one focuses on the current state-of-the-art in terms of predictive maintenance distinguishing frameworks and methods applicable in a generic industrial context and in an aeronautical context. We will rely on it to determine what are the best practices and useful metrics used in this field to measure our results.

Then an overview of Machine Learning fundamentals is given. Machine Learning is used in this thesis to determine the statistical model that enables anomaly detection so we provide a definition of Machine Learning and we detail the standard methods used to build a Machine Learning pipeline step-by-step from data pre-processing to post-processing.

Finally, we review the current state-of-the-art in the matter of Distributed Learning. This is necessary to set the context of the solution to propose to the thesis problem statement.

2.2 Predictive maintenance of industrial systems

2.2.1 General case

Predictive maintenance is about monitoring the state of a system to trigger maintenance just in time in order to maximize maintenance efficiency. A good introduction to the field of predictive maintenance can be found in [Mobley, 2002] with an explanation of predictive maintenance theory, its financial and organisational aspects, concrete examples focused on industrial machinery and guidelines on how to set up and sustain a predictive maintenance program.

There are plenty of studies about predictive maintenance focusing on its organisational or logistics aspects such as [Shafiee, 2015]. They are essential to ensure that the changes induced on the maintenance policy actually translates into a net measurable improvement for the operator of the system or, to put it in another way, how the failure predicted by the model can be translated into a concrete business process and how the efficiency of this process can be measured. For this, aspects like cost of unplanned versus planned maintenance need to be considered as well as location and availability of spare parts and maintenance personnel and associated costs. A notable study, [Horenbeek and Pintelon, 2013], focuses on the interaction of monitored components and how optimizing the cost of the maintenance policy of the whole system is not as simple as optimizing the maintenance policy at component level. This study models a stochastic dependence between components to show that the degradation of a component and its failure has an impact on other components with possible cascading effects of primary failure of a component inducing secondary failures of other components. It also introduces a structural and economic dependence to model the fact that simultaneous maintenance of several components at once is cheaper than individual of each component one by one. Nevertheless, since the focus of this thesis is on the production of the failure prediction model, the organisational and logistics

aspects are not explored further here and are left for future work.

There is also plenty of work focusing on the technical implementation of predictive maintenance, that is to say how to get from the observations from the sensors monitoring the system to the failure prediction model. Multiple approaches to this problem exist. In [Hashemian, 2011], the authors propose a review of the state-of-the-art techniques available in 2011 with a particular focus on industrial plants, the type of sensors that are available for various systems and the effect on the observations for various types of anomaly. In general there are two different ways to establish a failure prediction model based on observations. The so called model based approach starts from a theoretical physical model of the system to determine what would observations indicative of an imminent failure look like or the statistical approach which starts from a history of observations to try to infer a failure prediction model.

A good example of predictive maintenance based on a physical model can be found in [Mazzoletti et al., 2017]. This study leverages expert knowledge about the system it examines, here a permanent magnet synchronous machine, to build an analytical theory of how a malfunction may impact sensor measurements and then validates experimentally the theory. This approach has been historically adopted in many other predictive maintenance studies such as [Bansal et al., 2004] or [Byington et al., 2002]. However the more complex the system the more difficult it is to apply this approach. In the case of interdependent systems, which, as we have seen, aeronautical systems generally are, building an exhaustive analytical theory would require experts for every system involved as well as experts in the interaction of those systems making it impossible to scale this approach to, for example, an entire aircraft.

The other approach to failure prediction for predictive maintenance that has been gaining in popularity lately is the statistical approach found for example in [Li et al., 2014a] and [Ullah et al., 2017]. The goal of this approach is to find correlations between sensor measurements and system failures and build the failure prediction model based on those without necessarily resorting to the physical interpretation of the measurements. A very simple illustration would be the following: let us assume that our system is equipped with a sensor S_A measuring the parameter A and let us assume that we found that when $A < 5$ the daily failure rate of our system is 1% and when $A > 5$ the daily failure rate is 99%. Without knowing what A represents or what unit it is expressed in, we can already intuitively propose a simple threshold rule for failure prediction, when $A > 5$ predict failure and else predict non-failure. Of course, situations are rarely so clear cut and, even so, the next question to answer would be can we predict when the threshold will be exceeded so there is an abundance of work on how to generate the failure prediction model from the observations. In [Grall et al., 2002], a specific category of system is studied, deteriorating systems, monitored by a sensor measurement characterized by a stochastic increasing process and a failure threshold. A method is then proposed to identify the parameters of the stochastic process from historical observations and to derive failure prediction rules from them. This approach is very interesting as it foregoes the need for expert knowledge of the system but as it still has analytical components there is still a difficulty in

scaling it up when the number of sensor measurements increases or when we cannot make assumptions a priori on their behaviour. The latest solution that has been found in that case is to use a Machine Learning approach to automate the extraction of relevant features and rules when there is a large volume of data available. [Li et al., 2014a] and [Ullah et al., 2017] provide two recent examples of complex models, a Decision Tree and a Multi-layer Perceptron, learned through Machine Learning techniques. We explain the principle of those techniques in the next section 2.3 dedicated to Machine Learning fundamentals.

2.2.2 Aeronautical systems

The situation for machine learning techniques applied to aeronautics is quite complicated. There is ample documentation of products being sold by aircraft manufacturers such as Boeing AnalytX[3] or Airbus Skywise[4] but few details about them are public. Datasets are obviously not publicly available and, even when a scientific publication exists, neither are the implementation of the algorithms used. An illustration of this can be found in [Burnaev et al., 2014] and [Kemkemian et al., 2013].

It is also possible to find additional references in patents held by manufacturers but the level of detail available is lower yet. The following patents [Song et al., 2012] and [Kipersztok et al., 2015] illustrate that fact. It is also worth noting that the volume alone of patents held by the aircraft manufacturers, more than 54.000 for Boeing alone as of 2019, makes it impossible to conduct an exhaustive review.

For these reasons, despite the existing concrete applications of machine learning for predictive maintenance of aeronautical systems, it is still necessary to carefully study them in this thesis.

A notable exception is the thesis recently made public [Korvesis, 2017]. Taking the slightly different point of view of an aircraft manufacturer, it focuses on the issue of automation of Machine Learning for predictive maintenance processes while this thesis considers the problem of software architecture and distributed execution. It is nevertheless worth noting that similar observations regarding the applicability of Machine Learning techniques to aeronautical systems with regards to rare event predictions were reached concurrently in this thesis and in [Korvesis, 2017].

In the table 2.1, we summarize the contributions of the articles we discuss here for this thesis.

[3] https://www.boeing.com/company/key-orgs/analytx/index.page
[4] https://www.airbus.com/aircraft/support-services/skywise.html

Reference	Comment
[Mobley, 2002]	General introduction
[Shafiee, 2015]	Logistics and process aspects
[Horenbeek and Pintelon, 2013]	Multi-component complexity
[Hashemian, 2011]	State-of-the-art for industrial plants; focus on sensors
[Mazzoletti et al., 2017]	Predictive maintenance bqsed on physical models
[Bansal et al., 2004]	Predictive maintenance bqsed on physical models
[Byington et al., 2002]	Predictive maintenance bqsed on physical models
[Li et al., 2014a]	Statistical approach with Decision Tree
[Ullah et al., 2017]	Statistical approach with Multi-layer Perceptron
[Grall et al., 2002]	Predictive maintenance for deteriorating system
[Burnaev et al., 2014]	Aeronautical application; dataset not released
[Kemkemian et al., 2013]	Aeronautical application; dataset not released
[Korvesis, 2017]	Similar topic but focus on automation; datasets not released

Table 2.1: Summary of related work in predictive maintenance

2.3 Machine Learning fundamentals

This section is meant to give an overview of Machine Learning terminology and the techniques that are discussed in this thesis. Machine Learning is a very active field of research at the moment. For a more detailed introduction to the field from a statistical perspective, we recommend [Friedman et al., 2001]. To quote it, Machine Learning is about extracting knowledge from data in order to create models that can perform tasks effectively. Unpacking that statement, we can identify three points:

- Machine Learning is working from data. This means that the inputs are observations, measurements, recordings with no assumption at this stage on their format. They can be numerical or categorical, continuous or discrete, text files or video recordings. The only thing that is certain is that we do not have access to a model to generate this data and have to progress empirically towards it. This variety of inputs partially explains the variety of learning methods that have been developed. A first difference that is usually made at this stage is whether the data is labelled or not which, in this context, means that for a given observation, we know the desired output. When the desired output is known, the data is labelled and the learning is called supervised. When the desired output is not known, the learning is called unsupervised. A hybrid situation where some of the data is labelled and some is not is possible in which case the learning is called semi-supervised.

- The expected output is a model that performs a task. Once again, this formulation is very generic and carries no assumption on the nature of the task. A generic denomination is

often made depending on whether a categorical or a numerical output is expected. In the first case, the model is called a classification model and in the second it is called a regression model.

- There is an expectation of efficiency for the model. The question of performance metrics is essential for a Machine Learning approach as it is necessary to guide the learning process. A function called the loss function is used to determine how well the current model fits the data.

The categories defined here are very broad and do not necessarily describes every method available. For example, reinforcement learning is a popular sub-field of Machine Learning interested in learning action policies using algorithms such as Q-learning [Watkins and Dayan, 1992] which is traditionally considered to be neither a classification nor a regression model. Nevertheless these categories are part of the standard terminology in Machine Learning and are a useful way to quickly describe the application range of a learning method.

Regarding the notations that we will use, roughly speaking, we have at our disposal a set of multidimensional observations $x \in \mathbb{R}^d$, $(x_1, x_2, ..., x_d)$. For example, x can represent the data or a transformation of the data acquired by the sensors. For a given x, we associate a label $y \in \mathbb{R}$ for a regression problem or $\{0, 1\}$ for a binary classification problem with 0 coding for no event and 1 for event. The objective consists in predicting the label y associated to a data x.

2.3.1 Logistic Regression

Logistic Regression (LR) [Friedman et al., 2001] is a method widely used for regression and classification where predictions rely on the following logistic function:

$$\phi(x) = \frac{1}{1 + \exp^{-w_0 - \sum_{i=1}^{d} w_i x_i}} \tag{2.1}$$

In the regression case, $\phi(x)$ is interpreted as the estimate for y. In the classification case, $\phi(x)$ is interpreted as the probability that $y = 1$ given x. Consequently, the objective is to estimate $\Pr(y = 1|x)$ and so $(w_0, \cdots, w_d)$ from $\{(x^1, y^1), \cdots, (x^N, y^N)\}$.

LR can be trained several methods such as Stochastic Gradient Descent or Limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm (L-BFGS) [Bottou, 2010] depending on the computing resources available. A known limitation of the LR model is that it works poorly with time series, where the assumption that the observations $\{(x^1, y^1), \cdots, (x^N, y^N)\}$ are independent is challenged.

2.3.2 Support Vector Machine

Support Vector Machine (SVM) is a technique that relies on finding the hyperplane that splits the two classes to predict while maximizing the distance with the closest data points [Cortes

and Vapnik, 1995]. With N the number of samples, x_i the features of a sample, y its label and $\vec{w}$ the normal vector to the hyperplane considered, b the hyperplane offset and λ he soft-margin, the SVM equation to minimize is:

$$f(\vec{w}, b) = \lambda \|\vec{w}\|^2 + \frac{1}{N} \sum_{i=1}^{N} max(0, 1 - y_i(w.x_i - b)) \tag{2.2}$$

It is worth noting that even if it was initially designed with separable data in mind, the equation remains valid even when some points are on the wrong side of the decision boundary [Cortes and Vapnik, 1995, Friedman et al., 2001] which means SVM is applicable even without the assumption that the data is separable. SVM can also be trained online using, for example, active learning techniques [Bordes et al., 2005].

2.3.3 Deep Neural Networks

Fully connected Deep Neural Networks (DNN) are popular architectures which aim at approximating a complex unknown function $f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^n$ is an observation, by $f_\theta(\mathbf{x})$ [Rosenblatt, 1957] [Negnevitsky, 2001]. θ consists of the parameters of the DNN, the bias vectors $\mathbf{b}^{(i)}$ and the weight matrices $\mathbf{W}^{(i)}$ for all i, $1 \leq i \leq P$, and $f_\theta(\mathbf{x})$ is a sequential composition of linear functions built from the bias and from the weights, and of a non-linear activation function $g(.)$ (eg. the sigmoid $g(z) = 1/(1 + exp(-z))$ or the rectified linear unit (ReLu) $g(z) = \max(0, z)$) [Cybenko, 1989][LeCun et al., 2015].

Parameters $\theta = \{\mathbf{b}^{(i)}, \mathbf{W}^{(i)}\}$ are estimated from the back-propagation algorithm via a gradient descent method based on the minimization of a cost function $\mathcal{L}_\theta((\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N))$ deduced from a training dataset [Rumelhart et al., 1988].

However, when the objective is to classify high dimensional data such as colour images with a large number of pixels, fully connected DNN are no longer adapted from a computational point of view.

2.3.4 Convolutional neural networks

Convolutional Neural Networks (CNN) aim at dealing with the previous issue by taking into account the spatial dependencies of the data [Albawi et al., 2017]. More precisely, data are now represented by a 3-D matrix where the two first dimensions represent the height and the width of the image while the depth represents the three colour channels (R,G,B).

Next, as fully connected DNN, CNN consists in building a function $f_{\mathbf{t}}heta(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \times \mathbb{R}^{d_3}$, by the sequential composition of the following elementary steps :

- a convolution step via the application of convolution filters on the current image. Each filter is described by a matrix with appropriate dimensions;

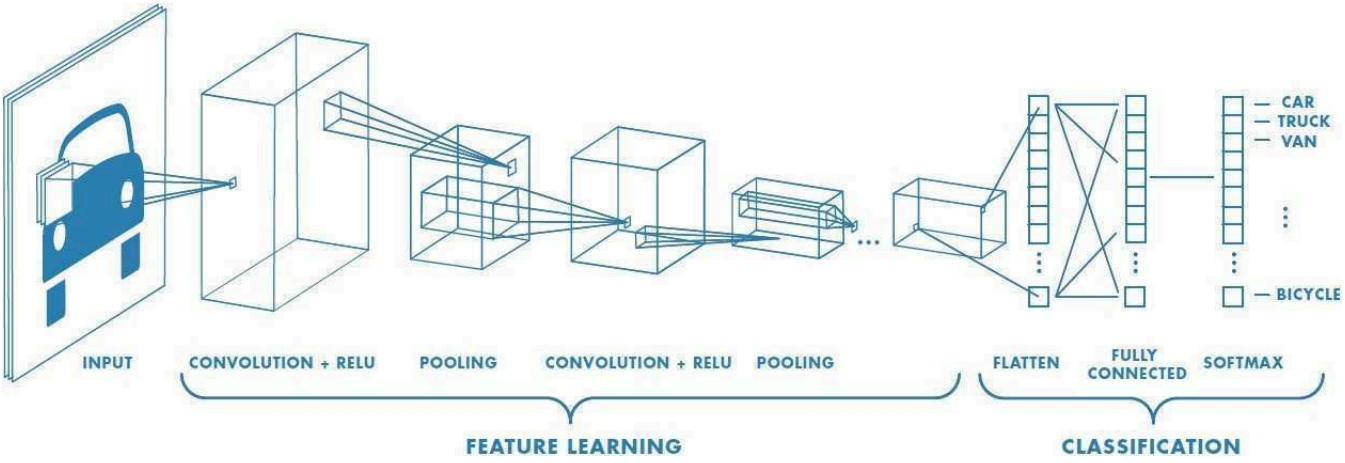


Figure 2.1: Example of a CNN architecture

- the application of an activation function $g(.)$ such as the ReLu;
- a pooling step to reduce the dimension of the resulting image.

After the recursive application of these three steps, the output image is transformed into a vector of $\mathbb{R}^n$ and is classified via a fully DNN described in the previous paragraph [Krizhevsky et al., 2012]. A general CNN architecture is displayed in Fig. 2.3.4.

Again, the back-propagation algorithm estimates the parameters (weights and bias of the final DNN and of the convolution matrices) of the CNN.

2.3.5 Decision Tree

Decision Tree (DT) [Breiman, 2017] is a supervised learning method that can be used for both classification and regression. Its goal is to recusively create simple decision rules to infer the value of the target. To do so, it determine each split as follow: given m the current node to split, Q_m the data available at that node, N_m the number of samples in Q_m, θ the candidate split, n_{left} and n_{right} the amount of sample split respectively to the left and right of the node and H the impurity function,

$$\frac{n_{left}}{N_m} H(Q_{left}(\theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\theta)) \tag{2.3}$$

This step is repeated until $N_m = 1$, i.e. the split is pure, composed of a single class, or the maximum depth is reached.

2.3.6 Random Forest

Random Forest (RF) improves on DT by combining several decision trees each trained on bootstrapped samples with different attributes. A process called bagging is used. Its principle is that to train N trees, for each tree, a subset of the features and a subset of the observations are sampled with replacement. New predictions are then made based on a vote among the different decision trees [Criminisi et al., 2012].

2.3.7 Gradient Boosted Tree

Finally, Gradient Boosted Tree (GBT) is another ensemble technique based on decision trees. Instead of training random trees like in RF, the training takes place in an iterative fashion with the goal of trying to minimize a loss function using a gradient descent method [Criminisi et al., 2012]. Example of log loss function characterized by the formula: $2 \times \sum_{i=1}^{N} log(1 + \exp(-2y_i F(x_i)))$ where N is the number of samples, x_i and y_i respectively the features and the label of the sample i and $F(x_i)$ the predicted label for sample i. This function is minimized through 10 steps of gradient descent.

2.4 Applied Machine Learning

In this section we take a more specific interest into three sub-fields of Machine Learning that make use of the techniques described in section 2.3 to tackle specific challenges that are of use to this thesis.

2.4.1 Distributed Learning

Many Machine Learning techniques such as DNNs are notorious for requiring large amount of data to train effectively [Krizhevsky et al., 2012]. As a result, it is not unusual when employing these techniques to have to consider the question of computational resource usage. An obvious solution to reduce this usage would be to simply invest in a more powerful processing unit but this solution is not always practical. Another way that can be investigated is to share the computation load across multiple cores or multiple processing units.

The use of multiple cores to take advantage of the embarrassingly parallel nature of the computations in DNNs, largely credited for the rise of deep learning methods in popularity, can be found in the implementation of Machine Learning on Graphical Processing Units (GPUs). It is sometimes called Parallel Learning and an example of a study on this can be found in [Sierra-Canto et al., 2010]. We will not investigate this further as we are more specifically interested in learning that happens between processing units in different hosts. The approach of using sharing the computation load across multiple processing units is called distributed learning. It

can take different shapes depending on how the processing units are allowed to communicate with each other. [Peteiro-Barral and Guijarro-Berdiñas, 2013] is a survey of some of the work done on this topic as of 2013.

Regarding the different characteristics of processing unit communications to take into account that are important to correctly frame the problem, several borrows from the field of distributed systems. We can mention:

- Centralized and decentralized learning: this question, very common in distributed systems, is about the roles of the hosts and whether one of them has a privileged role of coordinator or master. A well-known example of a centralized learning method is the parameter server for centralized asynchronous Stochastic Gradient Descent (SGD) described in [Li et al., 2014c] for example. In this approach, a central server called the parameter server distributes data and workload asynchronously to a group of workers while maintaining a set of global parameters. After receiving updates from the workers, the parameter server aggregate them and update the global model. An example of work in centralized learning on an algorithm other than SGD can be found in [Zhang and Kwok, 2014] where the method of Alternating Direction Method of Multipliers (ADMM) is used to find a distributed consensus asynchronously by using partial barriers. In contrast, in decentralized learning, every host has the same role. In [Watcharapichat et al., 2016], a variant of GD for Deep Learning is presented with the explicit goal of getting rid of the parameter servers in order to remove potential performance bottlenecks. This variant relies on exchanging partitions of the full gradient update called partial gradient exchanges. Another example of decentralized learning can be found in [Alpcan and Bauckhage, 2009] where the authors propose a decentralized learning method for SVM by decomposing the SVM classification problem into relaxed sub-problems.

- Data and model parallelism: as a flip side to distributed system bit-level parallelism and task-level parallelism, the expression data parallelism and model parallelism are used. They do, however, cover very similar concepts. Data parallelism expresses the idea that the tasks between the hosts are shared by distributing the dataset between them while for model parallelism operations are executed in parallel by the hosts on the same dataset. They are illustrated in figure 2.4.1. A notable difference between model parallelism in distributed learning and task parallelism in distributed systems is that model parallelism does not necessarily imply that the operations are done on the same data. It is also worth noting that data and model parallelisms are not mutually exclusive. In [Li et al., 2014c], for example, the parameter server distributes both the data and the tasks between the workers. This makes sense because fragments of a given dataset are assumed to represent the same distribution and hence the results can be combined in the same model.

- Synchronous and asynchronous: the distinction between synchronous and asynchronous communication in distributed learning is completely analogous to the one in distributed

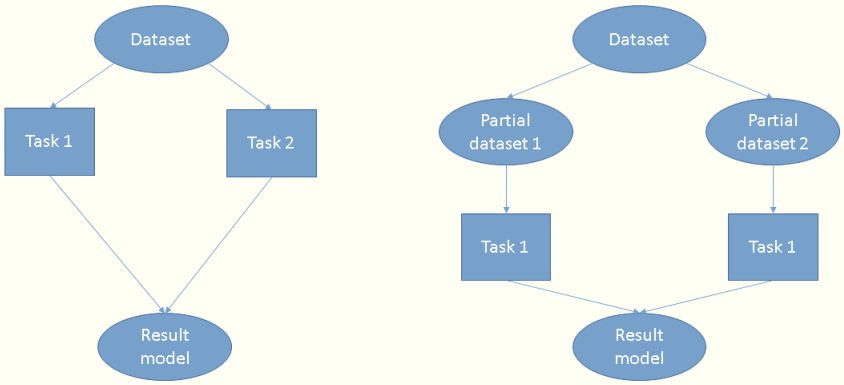


Figure 2.2: Illustration of model parallelism (left) and data parallelism (right)

systems. Asynchronous models such as citezhang2014asynchronous and [Li et al., 2014c] do not assume that all hosts have access to the same clock time. Concretely, it implies that hosts are not required to wait for their neighbours to be at a certain point in their execution to proceed with their own. The vast majority of distributed learning algorithms are asynchronous but a few application of synchronous learning can be found, notably in neurocomputing such as in [Kubacki and Sosnowski, 2017].

On the matter of distributed learning publications that are of particular interest to this thesis we have to mention [Valerio et al., 2017]. In this article, an Internet of Things (IoT) distributed learning framework for classification is presented and evaluated on network overhead and model accuracy. They present a solution called Hypothesis Transfer Learning based on an implementation of Greedy Transfer Learning found in [Kuzborskij et al., 2015]. In this decentralized approach, every host compute a local model based on the data accessible to it, sends its model to all its neighbours in a synchronization phase and collect their models. A second round of training is done, using Greedy Transfer Learning from neighbour models to improve the local model. The synchronization phase is then repeated until the host receives the updated local model of every host that has contributed to the distributed learning. Finally, the models are aggregated in an ensemble model using majority voting, that is to say the result of the classification is the most frequent prediction among the aggregated models. This work consider several hypothesis that are relevant for aeronautical systems namely it avoids data transfer between hosts, provides an explicit way to trade-off network overhead and model accuracy by tuning how many nodes take part in the Greedy Transfer Learning training and it considers the case of

class imbalance (though with a much less drastic ratio). The most significant difference with the prediction in aeronautical systems, beside the order of magnitude change in class imbalance ratio, is that it is a decentralized algorithm with communication between hosts which is not a realistic hypothesis for aircraft. Another problem that it does not address is the fact that the performance of individual models is not controlled meaning that it is not lower bounded and that there are no mechanisms to correct a potential drift of a local model. It makes sense in the situation considered in [Valerio et al., 2017] but it is a requirement that needs to be addressed in the aeronautical use case.

2.4.2 Active Learning

Another field of Machine Learning that we want to detail is Active Learning. Active Learning is a field of semi-supervised learning where a third-party called an oracle can provide missing labels on request. It is traditionally not considered a sub-field of Distributed Learning and is applied in a very different context where data labelling requires the intervention of a human operator that is both costly and slow and the goal is to provide a strategy to get the best model accuracy under a fixed request budget. A summary of the principles of Active Learning from a statistical perspective can be found in [Cohn et al., 1996]. From a practical perspective, active learning has historically been used with success for use cases such as spam filtering [Georgala et al., 2014], image classification [Joshi et al., 2009] or network intrusion detection [Li and Guo, 2007]. Different approaches are available for Active Learning and a survey summarizing them can be found in [Settles, 2009]. The most common approach for concrete applications is called pool-based active learning and described in [Lewis and Gale, 1994]. The assumption in this approach is that there is a large pool of unlabelled data and a smaller pool of labelled data available. The labelled data is used to bootstrap a tentative model and a metrics to measure the amount of information gain one can expect from a sample is defined. Some examples of metrics can be a distance metrics between the sample and the decision boundary of the model, the density of labelled data in the vicinity of the sample or the difference in the learned model with regards to the possible labels. Once the metrics is defined, the unlabelled samples that are expected to maximize the information gain are queried from the oracle then the tentative model is updated and the process is repeated until the expected information gain falls below a pre-defined threshold, meaning that no new label is expected to significantly change the model, or the query budget is spent.

A recent example of active learning can be found in [Gal et al., 2017] where a pool based approach is used on a dataset of skin lesion images with the goal to classify the images between benign or malign lesions (melanoma). The unlabelled images that are selected by the active learning approach using the metrics BALD ([Houlsby et al., 2011]) are queried. In this benchmark, the labels are known so they are simply disclosed but in a real world application a dermatologist could be consulted.

Another example of active learning relevant to this thesis can be found in [Miller et al., 2014]. In the context of malware detection, the authors introduce the concept of adversarial active learning. In this situation, the oracle is a human annotator that can be either a security expert that is assumed to always provide the correct labels, a crowd-sourced non-expert that is less expensive from a query perspective but that can make mistakes which are modelled as noisy labels or an adversarial agent masquerading as a crowd-sourced non-expert that provides the wrong labels. While an adversarial hypothesis is not directly applicable to our context, the notion of having different oracles available that have different levels of reliability is relevant.

Finally, another approach to active learning in decision trees can be found in [De Rosa and Cesa-Bianchi, 2017]. Here the approach is not pool-based but stream-based, that is to say that the algorithm is receiving a stream of unlabelled samples and does not have access to the complete pool which proscribes the greedy approach of evaluating every sample for information gain and selecting the best. In this article, a decision tree is trained and when a new sample is classified through it the confidence that is evaluated is not the confidence in the classification of the sample but in the optimality of the model. To put it differently, when a new leaf is added to the tree, the comparison is made between the ideal decision tree that could have been made if the true distribution of the samples were known and the expected performance of the current tree and, if the difference is found to exceed a certain threshold, labels are requested to minimize to risk of selecting a sub-optimal split for the new leaf. The points that are the most relevant for this thesis is that, by moving away from a greedy approach, we can ensure that the computation cost remains manageable, making this algorithm suitable for an on-board implementation and the notion of approaching the uncertainty in terms of bounded risks for the model to be sub-optimal instead of considering sample misclassification makes it possible to directly manage the trade-off between the communication budget and the quality of the model.

2.4.3 Federated Learning

A last sub-field of Distributed Learning that we would like to detail here is called Federated Learning. This term was coined rather recently in [McMahan et al., 2016]. The first use case described was the distributed learning of a centralized model for image classification and language modelling on mobile devices. In this situation, several assumptions made on previous works in distributed learning are not satisfied anymore. In particular, in this context, there is a privacy concern with the data proscribing any transfer of raw data between the central server and the clients, the data is massively distributed between a very large number of clients whose availability may change suddenly and the data cannot be assumed Independent and Identically Distributed (IID) between clients. Each one of these issues has already been studied individually in Distributed Learning but they cumulatively make for a challenging problem. In [McMahan et al., 2016], a solution to learn a DNN in this context is proposed. It uses iterative averaging of synchronous SGD over a random subset of clients to update a central model that is then

forwarded to the queried clients at each iteration.

Several publications are focused on improving different aspects of this method. In [Konečnỳ et al., 2016], the communication efficiency is improved through several tricks like random masks, update quantization and structured updates. In [Bonawitz et al., 2016], the focus is put on establishing how to ensure that the data used by the client in an update cannot be deduced from the update itself from the server-side by using a client-side differential privacy approach with double masking. Conversely, [Geyer et al., 2017] also establishes how to mask whether a client participated at all in a differential update.

In general, the constraints characterizing the federated learning approach are also applicable to aircraft which are also numerous, with limited availability that can dynamically change and with data that is not IID from one aircraft to another.

It is worth noting however that Federated Learning has been developped so far with DNN applications in mind and, in particular, it is also compatible with SGD and not with DT based approaches that do not rely on SGD.

In the table 2.2, we summarize the contributions of the articles we discuss here for this thesis.

Reference	Comment
[Peteiro-Barral and Guijarro-Berdiñas, 2013]	Distributed Learning survey
[Sierra-Canto et al., 2010]	Parallel but not distributed example
[Li et al., 2014c]	Centralized learning example with asynchronous SGD
[Zhang and Kwok, 2014]	Centralized learning example with ADMM
[Watcharapichat et al., 2016]	Decentralized Learning example with GD
[Alpcan and Bauckhage, 2009]	Decentralized Learning example with SVM
[Kubacki and Sosnowski, 2017]	Synchronous Learning example, biocomputing
[Valerio et al., 2017]	IoT communication-efficient Distributed Learning
[Cohn et al., 1996]	Principles of Active Learning
[Settles, 2009]	Active Learning survey
[Lewis and Gale, 1994]	Pool-based Active Learning example
[De Rosa and Cesa-Bianchi, 2017]	Stream-based Active Learning example
[Miller et al., 2014]	Active Learning uncertain oracles at varying costs
[McMahan et al., 2016]	Federated Learning original contribution
[Konečnỳ et al., 2016]	Federated Learning communication efficiency
[Bonawitz et al., 2016]	Federated Learning differential privacy

Table 2.2: Summary of related work in applied machine learning

2.5 Conclusion

In this chapter, we have presented several works related to the problem of this thesis focusing on Machine Learning fundamentals that will be useful for the next chapters and on state-of-the-art contributions to the problem at hand. The state-of-the-art contributions provide interesting clues to individually solve many of the problems we are faced with but none of the contributions offer a global solution that would fit all the constraints of aeronautical systems. Starting from this observation, we can further refine the questions that we address in the next chapters.

For rare event prediction in chapter 3, we have identified that most methods in applied Machine Learning are not particularly concerned with class imbalance but are compatible with multiple fundamental learning models and sampling methods. Therefore we will set out to compare the performance of these learning models and sampling methods in situations of class imbalance in order to determine which one would be the best choice to be at the core of the applied Machine Learning method.

For chapter 4, dedicated to identifying the best way to adapt to the data format that is already available and that is very often for aeronautical systems free form text, we are also looking for pre-processing techniques that would not in theory depend on the applied framework. Thus, the approach will be similar in trying to find the best methods without considerations such as distributed execution at this stage.

For chapter 5, however, we have seen that existing methods, in particular in Federated Learning and Active Learning, already provide multiple relevant solutions but not to every problem at the same time. Our approach will therefore be to figure out a new way to combine those existing solutions in the same applied Machine Learning framework.

Chapter 3

Rare Event Prediction for Operational Data

Contents

In this chapter, we examine the applicability of different failure prediction techniques on operational data. To do so, we consider a public dataset of hard drive data to use as a proxy for confidential aeronautical data. We try to apply existing methods to this problem, observe that their performances on operational data greatly differ from previous experiments and conclude on the relevant techniques for rare event prediction.

3.1 Introduction

3.1.1 General problem

One of the characteristics distinguishing aeronautical systems is the high importance that is given to their reliability as a way to ensure very high safety standards and minimize the cost of maintenance. As a result, failures of aeronautical systems are extremely rare. This makes the task of predicting them more difficult for two reasons.

First, being rare events, it is more difficult to collect a dataset that includes a number of failures significant enough for statistical inference to be effective. In other words, if a given system fails on average once per year of operation and is deployed on a fleet of 20 aircraft, in order to collect a dataset containing at least 100 failures the data collection process should last on average 5 years. Such a duration is not reasonable in most cases. In order to work around that limitation, a practical solution is to work with artificially aged systems that are created by submitting systems in a controlled environment to stressful constraints that are believed to lead to precocious wear similar to that of older systems. For example, one could operate a system in the presence of high vibrations, temperature or with an unusual workload in order to artificially age it. However, this approach has been shown to introduce bias ([Pinheiro et al., 2007]) as the failures of artificially aged systems do not exactly match those of operational systems. Whenever it is possible, it is preferable to work with systems that are already extensively monitored for which a significant volume of data is already available. Those systems are, fortunately, not uncommon in aeronautics.

The second point making predictions difficult in that situation is that the ratio between failure samples, that is to say samples of a system that is about to fail, and healthy samples, from normally operating systems, is heavily skewed in favour of the latter. This is potentially problematic for classification because many learning techniques such as Logistic Regression or Decision Tree operate best when the ratio between the two classes to predict is close to 1 ([Blagus and Lusa, 2010]). However, if we take the example of a system failing once per year on average again with a sampling rate, already pretty low, of 1 measure per hour we have on average about 8760 healthy sample for every failure sample for an imbalance ration in the range of 10^{-4}. The two general ways to approach this difficulty is to either use sampling techniques to correct the imbalance by changing the sample population with the risk of introducing biases or by working

with techniques that are more robust to imbalance.

Investigating these issues and more specifically the second point is necessary to design failure detection for aeronautical systems but given the confidential nature of industrial dataset and to ensure the reproducibility of the results, the investigation was conducted on a public dataset of another very reliable system, hard drive disks (HDDs).

3.1.2 Hard Drives

Hard drives are essential for data storage but they are one of the most frequently failing components in modern data centres [Schroeder and Gibson, 2007], with consequences ranging from temporary system unavailability to complete data loss. Many predictive models such as LR and SVM, analysed in section 3.2, have already been proposed to mitigate hard drive failures but failure prediction in real-world operational conditions remains an open issue. One reason is that some failures might not be predictable in the first place, resulting, for example, from improper handling happening occasionally even in an environment maintained by experts. However, this alone cannot explain why the high performances of the failure prediction models that appear in the literature have not mitigated the problem further. Therefore, the specificities of hard drives need to be better taken into account.

First of all, the high reliability of a hard drive implies that failures have to be considered as rare events which leads to two difficulties. The ideal application case of many learning methods is obtained when the classes to predict are in equal proportions. Next, it is difficult to obtain sufficient failure occurrences. Indeed, hard drive manufacturers themselves provide data on the failure characteristics of their disks but it has been shown to be inaccurate (see e.g. [Schroeder and Gibson, 2007], [Pinheiro et al., 2007]) and often based on extrapolating from the behaviour of a small population in an accelerated life test environment. For this reason, it is important to work with operational data collected over a large period of time to ensure that it contains enough samples of hard drive failures.

Another challenge is that the Self-Monitoring, Analysis and Reporting Technology (SMART) used to monitor hard drives is not completely standardized. Indeed, the measured set of attributes and the details of SMART implementation are different for every hard drive manufacturer. From a machine learning point of view, there is no guarantee that a learning model trained to predict the failures of a specific hard drive model will be able to accurately predict the failures of another hard drive model. For this reason, in order to draw conclusions on hard drive failure prediction in general, it is important to ensure that the proposed predictive models are estimated from a variety of hard drive models from different manufacturers and also tested on a variety of hard drive models. Until now, this constraint was not taken into account properly, probably because gathering a representative dataset has been a problem for many previous studies, impairing the generality of their conclusions. We rather focus on the Backblaze public

dataset[5] consisting of several years worth of measurements on a large drive population operated in an expert-maintained environment. It has been made available recently, with the earliest measurements done in 2013.

The objective of this contribution is to offer an insight as to why many previous studies are not directly applicable when considering class imbalance, data heterogeneity and data volume and next to adapt predictive models based on machine learning methods for pattern recognition of hard drive failure prediction. We also compare the proposed models and discuss their performances on the Backblaze dataset that includes hard drives from different manufacturers in order to determine if they are robust to the differences in SMART parameters. The contribution is organized as follows. In section 3.2, we review the state-of-the-art while paying a particular attention to the datasets that were used. In section 3.3, we detail the new dataset that we use for this study and the specific challenges associated with it. In section 3.4, we describe the different machine learning techniques that we applied to the dataset, and we underline the different steps of pre-processing, feature selection, sampling, learning and post-processing. In section 3.5, we present and discuss our experimental results obtained with the three most relevant learning models: SVM, RF and GBT. Finally, we end the contribution with a conclusion and we discuss possible ways to extend this study.

3.2 Related Work

In this section, we will first describe the results obtained in the literature with a strong focus on the numeric performances and then explain and comment them.

3.2.1 Previous studies

Several studies on the subject of hard drive failure prediction based on SMART data have already been carried out. In particular, [Murray et al., 2003], [Murray et al., 2005], [Zhao et al., 2010] and [Wang et al., 2011] all used the same dataset. The models were tested on a dataset of 369 hard drives of the same model with healthy drives and failed drives in equal proportions. The data from healthy drives was collected in a controlled environment by the manufacturer.

In [Murray et al., 2003], several methods are proposed to build a prediction model: SVM, unsupervised clustering, rank-sum test and reverse arrangements test. This study found the best method among those tested to be the rank-sum test by detecting 24% of the failed drives while maintaining a false alarm rate below 1%.

In [Murray et al., 2005], a subsequent study from the same authors, the best performances were obtained with a SVM with a detection rate of 50.6% and a false alarm rate below 0.1%.

[5] https://www.backblaze.com/b2/hard-drive-test-data.html

In [Zhao et al., 2010], hidden Markov models and hidden semi-Markov models are tested. The best model reaches a detection rate of 52% and a false alarm rate below 0.1%.

In [Wang et al., 2011], a health monitoring method based on the Mahalanobis distance is developed. It yields a detection rate of 67% while maintaining the false alarm rate below 0.1% still.

In [Hamerly and Elkan, 2001], two Bayesian methods are tested, a Bayesian clustering model based on expectation maximization and a supervised naive Bayes classifier. The dataset used was collected from 1927 drives including 9 failed drives. The performances reached are 60% detection rate and a false alarm rate of 0.5%.

In [Zhu et al., 2013] and [Li et al., 2014b], a dataset comprising samples from $23,395$ drives operating in a data center is studied. Two different hard drives models from the same manufacturer are used. The methods used are back-propagation recurrent neural networks, SVM, classification and regression trees. The best results are obtained with classification trees achieving over 95% detection with a false alarm rate of 0.09%.

In [Ma et al., 2015], a population of $1,000,000$ drives is studied. 6 hard drive models are considered. The method used is threshold-based classification with only 1 SMART parameter. It reaches 70.1% recall and 4.5% false alarm rate. The dataset is unfortunately not publicly available and neither are the precise models and the environmental conditions in which the results are obtained making it difficult to comment or draw conclusions from these results.

3.2.2 Discussion

As we see, most previous studies were conducted on small datasets collected in a controlled environment using manufacturer data. Moreover, [Schroeder and Gibson, 2007] and [Pinheiro et al., 2007] have shown that manufacturer data on disk reliability is not accurate as it is relying on accelerated life tests and stress tests that appear to consistently underestimate the actual disk failure rate. As such, hard drive failure prediction models trained on manufacturer data have a high risk of being biased and cannot be relied on. Additionally, in [Murray et al., 2003] manufacturer data on hard drives is mixed with data from hard drives returned by users. The authors highlight in their paper the importance of understanding the induced limitations. However, given how often this dataset is used in other studies such as [Zhao et al., 2010] or [Wang et al., 2011], it is preferable to rely on a dataset without such mixing.

The most notable exceptions to these issues are the studies [Zhu et al., 2013] and [Li et al., 2014b]. Unfortunately the associated dataset used is not publicly available and is limited to two drive models.

Very recently, some studies started to exploit the Backblaze dataset. [Botezatu et al., 2016] considers a large subset of over $30,000$ drives from the Backblaze dataset to train several classifiers. However, the results (98% for the detection and 98% for the precision) are obtained on a limited and different subset of filtered data. In the industry, [El-Shimi, 2017] shows promising

results but the lack of implementation details prevents comparison.

Another limitation of previous studies is the choice of evaluation metrics which generally coincide with the detection and false alarm rates. This is a relevant choice for balanced datasets but, as operational datasets are extremely unbalanced in favour of healthy samples, even a low false alarm rate in the range of 1% could translate into poor performances. Therefore, we rather report precision and recall metrics.

In order to overcome these issues related to the dataset and to provide reproducible results, we consider a large, operational and publicly available dataset from Backblaze, and compute the precision and recall metrics, rather than detection and false alarm rates, on unfiltered samples. This enables us to draw conclusions for operational data.

3.3 Dataset

As discussed in the previous section, choosing the right dataset is essential in order to draw accurate conclusions on hard drive failure prediction. In this section, we provide details about SMART parameters and how they can be interpreted and we describe the Backblaze dataset that we work on.

3.3.1 SMART parameters

SMART is a monitoring system implemented in most hard drives that is meant to allow users to anticipate hard drive failures by reporting the value of several indicators that are believed to be representative of the health of the hard drive. Each SMART parameters reported shows up as an ID, a normalized value ranging between 1 and 253 meant to indicate the health of the drive with regards to the parameter as set by the manufacturer of the hard drive disk according to its specifications and a raw value whose meaning depends on the implementation of the SMART parameter but is usually associated with an event count or a physical measurement

It is important to note that this monitoring system has not been fully standardized resulting in significant differences in implementation between hard drive manufacturers. As a result each manufacturer is free to decide which SMART parameters should be monitored and how the monitoring should be implemented. Details of their implementation are not disclosed meaning there is no guarantee that SMART parameters from two different models or manufacturers have the same meaning. Concretely, it follows that SMART parameters with the same ID from two different hard drive models can actually represent slightly different phenomenons. It is also possible for SMART parameters with different IDs from two different hard drive models to represent the same phenomenon.

3.3.2 Backblaze dataset

This work relies on operational data that the Backblaze company started to release at the end of 2013. It gathers daily measurements of SMART parameters of each operational hard drive disk of this company data centre. Updates to the dataset are provided quarterly. The fields of the daily reports are composed as follows: the date of the report, the serial number of the drive, the model of the drive, the capacity of the drive in bytes, a failure label that is at 0 as long as the drive is healthy and that is set to 1 when the drive fails and finally the SMART parameters. For the rest of the study, we focus on the data from January 2014 to December 2014 in order to enable a comparison between prediction performances. Over this period, 80 fixed SMART parameters are collected among those defined by the manufacturers. They include, for example, counts of read errors, write faults, the temperature of the drive and its reallocated sectors count. However, it should be noted that, as explained in the previous subsection, most drives do not report every parameter resulting in many blank fields.

Finally, the dataset contains over 12 million samples from $47,793$ drives including 81 models from 5 manufacturers. Among those 12 million samples, only $2,586$ have their failure labels set to 1 and the others are healthy samples, for an overall ratio of about 2 failure samples for every $10,000$ healthy samples that is below 0.022%.

3.4 Data processing

We identify in this section a set of classification techniques that are best suited for an extremely unbalanced training set and a loosely controlled environment in real operation as opposed to laboratory experimentations. Considering the size of the dataset, we focus on classification computations that can be distributed across several nodes.

3.4.1 Pre-processing

As noted in [Pinheiro et al., 2007], traditionally used outlier filtering techniques such as Extreme Value Filtering are inadequate for SMART values of an operational set as it is difficult to distinguish between exceptional values caused by a measurement artefact or by an anomalous behaviour that may lead to a hard drive failure. A classical approach is to limit filtering to obvious errors. In our case, this corresponds to physically impossible values such as power-on hours exceeding 30 years. We filter on two SMART parameters, SMART 9, power-on hours, and SMART 194, temperature. It turned out that this filtering is negligible and does not noticeably impact the dataset with only 5 drives concerned, matching the observation in [Pinheiro et al., 2007] that less than 0.1% of the hard drives are concerned.

Similarly to what is done in [Zhu et al., 2013], we define a time window for failure in the following manner: after a drive fails, we relabel a posteriori the N previous samples from this

particular drive as failures where N is the length of the time window in days. In other words, with our classification model we try to answer the question "Is the hard drive going to fail in the next N days?". This length will be optimized as a hyper parameter of the model to determine its optimal value if it exists.

3.4.2 Feature selection

Not every SMART parameter is indicative of a failure. In [Pinheiro et al., 2007], a systematic study of the predictive power of several SMART parameters such as temperature that were previously believed to be indicative of imminent failures shows that they are not useful predictors on the population they study. Therefore, we can conclude that a subset of SMART parameters have to be considered as noise for the purpose of failure prediction and thus the question of feature selection is crucial in order to design an efficient failure prediction model. As such, we consider different strategies for feature selection. We considered using Principal Component Analysis (PCA) [Jolliffe, 2011] and Restricted Boltzmann Machines (RBM) [Hinton, 2002] in order to identify the SMART parameters the most closely correlated with hard drive failures however the results were consistently and significantly inferior to the other approaches so we have chosen not to conduct a full study on them in order to minimize the computation time.

The other feature selection strategy that we consider is based on the pre-selection used in related studies [Pinheiro et al., 2007, Zhu et al., 2013, Li et al., 2014b] of SMART parameters highly correlated to failure events. With this scheme, we consider only the nine raw SMART parameters number 5, 12, 187, 188, 189, 190, 198, 199 and 200. The description of those SMART parameters is reported in 3.1.

SMART parameters	Description
5	Reallocated Sector Count
12	Power Cycle Count
187	Reported Uncorrectable Errors
188	Command Timeout
189	High Fly Writes
190	Temperature Difference
198	Offline Uncorrectable Sector Count
199	UltraDMA CRC Errors Count
200	Multi-Zone Error Rate

Table 3.1: Description of SMART parameters

Finally, based on the fact that different hard drive manufacturers implement SMART parameters differently and do not necessarily implement the same SMART parameters in their products, and based on the observation that the performances of feature selection strategies we

tried are below the expectations set by other studies such as [Pinheiro et al., 2007, Zhu et al., 2013, Li et al., 2014b], we emit an hypothesis. We postulate that even SMART parameters with an overall low correlation to failure events that are discarded by most feature selection strategies are still in fact important to create an accurate failure prediction model. Our interpretation is that considering that hard drive failures are rare events and that multiple modes of failure can be found across a large population of hard drives, it is possible for predictors of failure modes with low prevalence to be distinct from predictors of failure modes with high prevalence while the total number of failure across different low prevalence modes can be higher that the higher prevalence failure mode. For example, assume the parameter X_1 is a good predictor for the failure F_1 that represents 30% of all failures, the parameters $X_2, X_3 ... X_{100}$ are good predictors of failures $F_2, F_3 ... F_{100}$ that each represents 0.5% of all failures and the remaining 20.5% of failures cannot accurately be predicted by any parameter available. Most sensible feature selection strategies will classify parameter X_1 as relevant and discard parameters $X_2, X_3 ... X_{100}$ as noise but accepting such a selection will discard important information and severely limit the performances of the failure prediction model from a potential 79.5% of failures correctly predicted to a mere 30%. It is important in that situation to preserve low information features.

We therefore take special interest in algorithms such as Random Forests that are not vulnerable to noisy features and are, in fact, able to extract information from features with low correlation with failure events. This also provides the additional benefit of not relying on sources external to the dataset to select the features. When it is relevant, we thus use these algorithms on the complete set of features.

3.4.3 Sampling techniques

In order to reduce the impact of the class imbalance issue on the learning algorithms, we investigate the use of sampling techniques. Given the extreme imbalance factor of the dataset, naive oversampling and undersampling were excluded as potential sources of overfitting. In [Wallace et al., 2011], they have been found to introduce statistical bias in populations with imbalance factors in the order of $1/100$. Given that the imbalance factor in our dataset is more than an order of magnitude higher we do not investigate those techniques further. Contrary to [Botezatu et al., 2016], we limit the application of sampling techniques to the same subset of data used for training.

SMOTE (Synthetic Minority Oversampling TEchnique) aims at alleviating class imbalance by generating additional training samples of the minority class through interpolation [Chawla et al., 2002]. SMOTE first selects a random minority sample, then determines its k nearest neighbours, selects one of them and places randomly on the segment between the two samples a new artificial minority sample. This process is then repeated as many times as necessary before reaching a pre-selected oversampling factor. By creating artificial instances distinct from the existing minority samples, it partially avoids the overfitting problem observed with naive

oversampling. It has been shown by [Chawla et al., 2002] to reliably improve classification performances on a variety of benchmark datasets with imbalance ratio up to 1 for 50. However, as acknowledged in the study, some datasets show much higher imbalance ratio. This is the case of the Backblaze dataset with an imbalance ratio 100 times higher than the most imbalanced dataset tested in [Chawla et al., 2002].

Another step to improve the training set is to filter a certain category of failure samples. We observe that some failure samples share the exact same feature values as healthy samples leading to the impossibility for any classifier based on those features to discriminate them. Thus, filtering those hard-to-classify failure samples leads to a trade-off of a higher precision at the expense of recall. Early results showed that the trade-off was not satisfactory with a loss of at best more than 5% in recall for a gain of 1% in precision. Therefore we have chosen not to conduct a full study of this method to minimize computation time.

3.4.4 Machine Learning algorithms

The theory of each Machine Learning algorithm described in this subsection are detailed in chapter 2

All the parameters of the learning methods are optimized through grid-search.

Previous studies such as [Botezatu et al., 2016] have considered Logistic Regression for hard drive failure prediction. We implemented it but results show a constant prediction in favour of the majority class, which is understandable given the sensitivity of Logistic Regression to class imbalance. Further work on sampling techniques is needed in order to use Logistic Regression on the Backblaze dataset. We therefore focus our experimentations on solutions able to deal with extreme class imbalance.

We more specifically investigate three approaches. The first is Support Vector Machine that have been shown to produce good results in related studies such as [Zhu et al., 2013] and [Li et al., 2014b]. However, given our additional requirement that the method we use need to be compatible with distributed computing, we limited ourselves to the use of linear kernels as distributed versions of non-linear kernels algorithms were not available. The learning took place via 100 steps of stochastic gradient descent to ensure a reasonable computation time (see table 3.2).

The second approach we use is the Random Forest. We set the number of decision trees at 50 with a maximum depth of 8 constructed with the Gini impurity metric.

Finally, we use Gradient boosted tree. The iterations are done with a log loss function characterized by the formula: $2 \times \sum_{i=1}^{N} log(1 + \exp(-2y_i F(x_i)))$ where N is the number of samples, x_i and y_i respectively the features and the label of the sample i and $F(x_i)$ the predicted label for sample i. This function is minimized through 10 steps of gradient descent.

3.5 Experimental results

3.5.1 Post-processing

In order to ensure the accuracy of the results, two additional steps are taken. First, we perform cross-validation through a customized stratified k-folding algorithm. The samples are first regrouped by HDD serial number so that the samples measured on one HDD are always in the same fold. HDDs are then split between those that reported a failure during the study and those that did not. The stratified k-sampling then takes place on the HDD level and not on the individual sample level in order to ensure that samples from a given HDD are always in the same fold. For this study, the number of folds has been fixed to 3. On top of that, in order to account for the optimization of the length of the time-window as an hyper-parameter of the models, every measurement is rerun three times by selecting new sets of folds for the cross-validation and the mean value is reported.

Two metrics are measured on the failed samples, precision and recall respectively defined as the number of successfully predicted failures divided by the total number of predictions and the number of failures successfully predicted divided by the total number of failures observed. If we define the failure sample as positive and the healthy samples as negative we have:

$$precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{3.1}$$

$$recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{3.2}$$

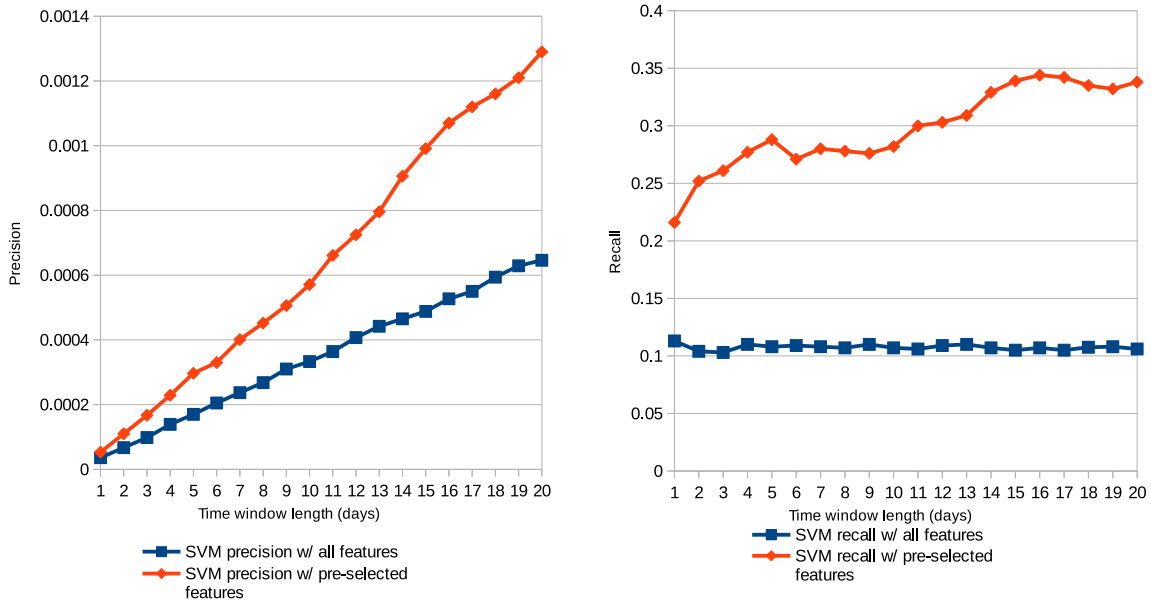
Note that contrary to similar studies referenced in section 3.2, we decide to report precision instead of false alarm rate: due to the high class imbalance, even a small false alarm rate could translate into poor performances. Indeed, a misclassification of only 1% of the healthy samples would result in 100 false alarms for every $10,000$ healthy samples, on average; since there are only 2 failure samples for every $10,000$ healthy samples, it means that we have 50 false alarms for every detected failure if we assume 100% recall on the failure samples, and consequently a precision below 2%. Similarly, we can note that a constant prediction in favour of the majority class would result in an accuracy of 99.98%.

3.5.2 Results and discussion

The performances in terms of precision and of recall are displayed in Fig. 3.1 (SVM), Fig. 3.2 (RF) and 3.3 (GBT) as a function of the time window length. The SMOTE sampling strategy is then tested on the best performing model, RF with all features.

The experimental setup we use is a cluster of 3 computers running Apache Spark v2.1 using a total of 24 cores. Due to the various size of the time window parameter, cross-validation and

repetition of the tests, every technique is run a total of 180 times, not including the grid-search executions to optimize the parameters. The execution time is reported in Table 3.2.



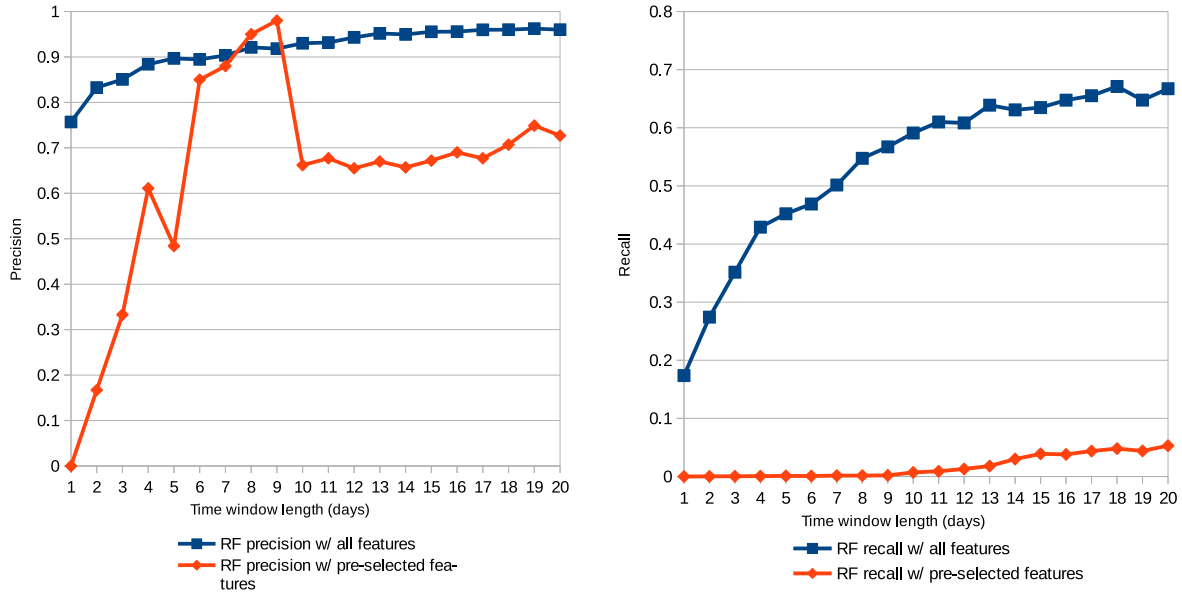
(a) Precision

(b) Recall

Figure 3.1: Precision and Recall of SVM for varying time window length with and without feature selection

There are several interesting points to notice on the graphs. First, on figure 3.1 for the linear SVM, we note that the performances are low. This is likely a sign that the classes to predict are not linearly separable. The solution for this would be to use another kernel for the SVM but unfortunately this would be at the cost of the parellelization of the implementation which would push the computation time beyond acceptable range. Regarding the length of the time window, it should be noted that the linear increase in precision with preselected features is likely only a side-effect of the relabelling. Further investigation reveals that the support vectors are not changing when the time window changes so the model learnt is the same. This is most likely because the inertia of the SVM model is too high to be affected by changing less than 0.1% of the labels.

Second, for RF, on figure 3.2, we can note that the usage of pre-selected features does not improve the performances but decreases them. The fact that feature selection does not bring improvement is understandable given that RF models have been shown to be resilient to noisy features. However the decrease also implies that the features that were not pre-selected are not pure noise but also contain useful information to predict an impending failure. This

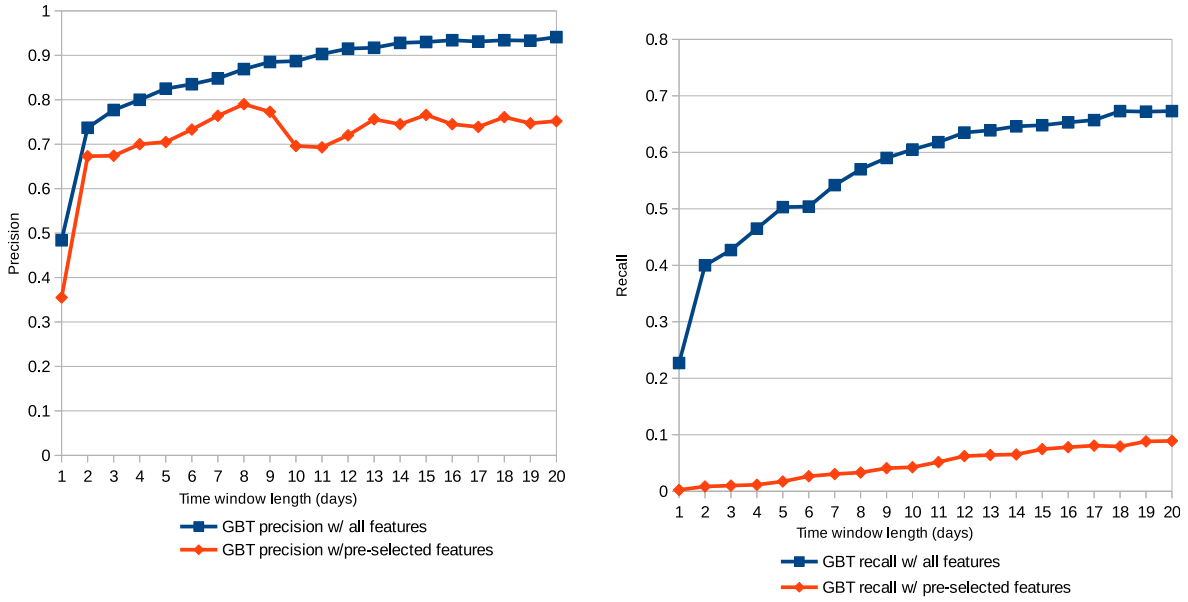


(a) Precision (b) Recall

Figure 3.2: Precision and Recall of RF for varying time window length with and without feature selection

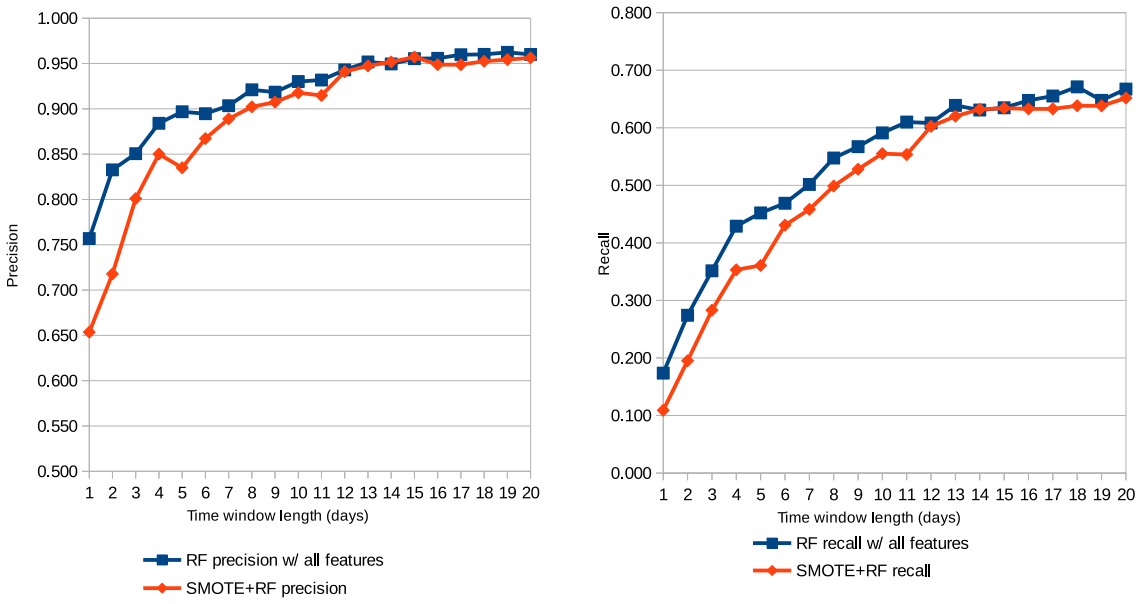
Method	Execution time (1 run)	Execution time (180 runs)
SVM preselected	11 min	33 hours
SVM all features	24 min	72 hours
GBT preselected	10 min	30 hours
GBT all features	41 min	123 hours
RF preselected	14 min	42 hours
RF all features	37 min	111 hours
RF+SMOTE	42 min	126 hours

Table 3.2: Execution time of the methods



(a) Precision (b) Recall

Figure 3.3: Precision and Recall of GBT for varying time window length with and without feature selection



(a) Precision (b) Recall

Figure 3.4: Precision and Recall of the RF model for varying time window length with and without feature selection

highlights the fact that SMART features are implemented differently on different drives and thus that conclusion regarding features useful for predicting failures of a specific type of hard drive model cannot easily be generalized to every type of hard drive. Additionally, for small values of the time window, the RF model struggles because the scarcity of the failure labels is further aggravated by the bagging technique used to learn the individual decision trees that compose the RF model. This is especially true with the pre-selected features.

Finally, for GBT, on figure 3.3 it does not display the same limitation as RF for time windows higher than one day, likely because it is not based on a bagging technique. The performances when using all available features are mostly similar to RF.

Overall, the best performances are reached by RF and GBT when using all available features with RF reaching up to 95% precision and 67% recall and GBT reaching up to 94% precision and 67% recall. The reported precision of 95% and 94% would translate on average with daily measurements on a single hard drive into a single false alarm respectively every 100.000 days and 83.333 days. In the mean time, over two thirds of the failures would be predicted correctly.

In figure 3.4, the last experiment with the SMOTE sampling technique resulted in negative results. The precision and recall of RF are decreased by SMOTE for smaller values of the time window and are similar at higher values. This is likely the result of an imbalance factor of 5000, much higher than the one used when developing sampling techniques which remains in the range of 2 to 100 in [Galar et al., 2012].

3.6 Conclusion

In this contribution, we work on hard drive failure prediction with a publicly available, large and operational dataset from Backblaze with relevant metrics, which is crucial to bridge the gap between laboratories studies and real-world systems and for reaching reproducible scientific results. This implies to address the class imbalance between failures and non-failures and the large dataset size. We have selected the precision and recall metrics, most relevant to the problem, and tested several learning methods, SVM, GBT and RF. With 95% precision and 67% recall, the best performances were provided by RF with all features while GBT was a close contender with 94% precision and 67% recall. SVM performances were unsatisfactory with a precision below 1%. We have also shown that when studying different hard drive models from different manufacturers, selecting the features classically used for hard drive failure prediction leads to a drop in performances.

Contrary to what was expected, SMOTE did not improve the prediction performances highlighting the difficulties stemming from the extreme unbalanced ratio of the Backblaze dataset. Additional work on sampling techniques is needed to balance the dataset. Ensemble-based Hybrid sampling techniques [Galar et al., 2012] such as SMOTEBagging, an improvement of the SMOTE sampling technique, could help to improve our learning models and might enable to use learning methods more sensitive to class imbalance such as logistic regression.

Finally, we plan to evaluate on the Backblaze dataset other learning techniques which have demonstrated promising results on other samples such as back-propagation recurrent neural networks [Zhu et al., 2013], Bayesian classifiers [Hamerly and Elkan, 2001] or Mahalanobis distance [Wang et al., 2011].

In general, with regards to the goal of studying rare event prediction, this study helps us establish that sampling techniques are not performing sufficiently well yet for the level of imbalance that interest us. Superior results are achieved by using learning techniques that are already resilient to imbalance and the best results are achieved by Random Forests.

Chapter 4

Automated Processing of Text-based Logs: Log Mining and Log Parsing in Industrial Systems

Contents

In this chapter, we present the approach we designed to process text-based logs, the most common type of data available for our aeronautical system, in an efficient manner in terms of computational resources and model accuracy. To that end, we study existing methods in log parsing and log mining, propose a new approach based on Natural Language Processing (NLP) and validate our method on both an aeronautical dataset and a public benchmark.

4.1 Introduction

4.1.1 General problem

There is a discrepancy between the public datasets often used as benchmark for classification algorithms and the data that is collected in actual aeronautical systems. Benchmark datasets consist of well-known measurements and clean numerical or categorical values. This enables standardized pre-processing and let the studies that uses them focus on the actual learning algorithms. Meanwhile, the actual inputs from an industrial dataset are not necessarily compatible with those methods. They take, for our use case, the form of full text event logs similar to syslogs of Linux systems. An illustration can be found in table 4.1. Such logs cannot be directly used as inputs by classical classification algorithms which rely on numerical values. The solution to change the data collection mechanism to fit the needs of the learning algorithm is not practical because of the long time needed to collect a significant amount of failure samples. Therefore, we want to use already existing data as much as possible. This means that we need to find a mapping between the available industrial dataset and the inputs needed by the available models. In order to achieve that, we focus on log parsing and log mining methods to determine an efficient way both in terms of accuracy of the model and in terms of computation to find this mapping from text to numerical values. Given the confidential nature of the industrial dataset and to ensure the reproducibility of the results we validated our findings in parallel on a public benchmark.

2015-28-Dec 15:20:13	TPSD	516	1	0	0	Resident SD Card Read/Write test passed on SD5
2015-28-Dec 15:20:18	TINF	435	1	0	0	CIDS communication established
2015-28-Dec 15:20:21	TPSD	463	1	0	0	eth0:UP. Connected

Table 4.1: Illustration of an aeronautical system text log

4.1.2 Log parsing and log mining

The usage of logging systems is one of the most widespread solutions for monitoring complex systems and applications. One such example would be the syslog of Linux machines. Such logging systems offer both a high level of flexibility for developers and a high level of expressiveness for users making them very useful to understand potential anomalies. However the high volume of log messages generated by complex systems makes it difficult for human operators to effectively monitor every event. This often leads to a reliance on high level signalling of the gravity of log messages such as "information","warning" and "error". This system has the obvious weakness of being only accurate for events and situations that were accounted for at the time of the implementation of the logging system. In a situation of interaction between multiple subsystems developed independently, the signalling might not be representative of the state of the entire system. That is to say an "error" message from a subsystem could be the result of a normal operation of the system, for example a subsystem reporting a loss of connectivity when the network services are being reset at the system level. Conversely, a "normal" message from a subsystem could indicate an anomaly in the system, for example, a subsystem reporting communication taking place when the network services are shutdown at the system level. As a result of this limitation and of the high volume of log messages, in practice, many logging systems are often limited in their use as tools for postmortem diagnosis.

In order to enable a better usage of logs, many data mining techniques have been adapted to automate log analysis. This process can be split into two main steps: i) how to transform the raw unstructured text from the log messages into a suitable input for data mining; this is called log parsing; ii) how to automatically exploit this structured information with data mining techniques; this is called log mining. Log mining has been the subject of many publications [Xu et al., 2009, Fu et al., 2009, Nagaraj et al., 2012], log parsing as well [Vaarandi, 2003, Makanju et al., 2012]. However, until recently there was no benchmark of publicly available datasets and implementations to systematically evaluate the performances of the log parsing methods. As a result, the interactions between log mining and log parsing have been only partially explored. Moreover, as logs are automated messages, even raw messages display repetitive patterns. Consequently, the first parsing approach adopted has often been rule-based and aimed at distinguishing between constant parts and variable parts to categorize log messages. The constant part being the string typed by the programmer when the software was written and the variable part the variables that appear in that string. However, the log messages are made of sentences or partial sentences understandable by a human operator and as such it is also possible to take into consideration their semantic aspect in order to parse them and categorize them. Thus it would make sense to investigate the use of NLP which is surprisingly scarcely studied [Bertero et al., 2017, Kubacki and Sosnowski, 2017] and absent from state-of-the-art surveys such as [He et al., 2016].

The goal of this contribution is to study the influence of a selection of log parsing techniques from the field of NLP on the efficiency of a fixed log mining process aimed at providing

log-based failure prediction of an aeronautical system with a focus on applicability. The results are evaluated on an industrial dataset collected on live systems and, in the end, the best log parsing method found is evaluated on a new benchmark dataset and compared to existing solutions found in [He et al., 2016]. We find that our solution achieves substantial performance improvement on the benchmark dataset while also being more robust and easier to parametrize than state-of-the-art solutions.

The contribution is organized as follows. First, we study existing works in log parsing and log mining in section 4.2. Next, we detail the log parsing techniques that we implemented in section 4.3. We explain the log mining process that we use in section 4.4. We present our results and interpret them in section 4.5. Finally, 4.6 concludes the contribution.

4.2 Related work

There are many studies pertaining to both log mining and log parsing. Some of the recent works include [Xu et al., 2009, Fu et al., 2009, Nagaraj et al., 2012]. In [Xu et al., 2009], the authors first parse the constant and variable parts of the logs messages using either the source code of the logging application or the messages themselves, then filter the messages on frequency and perform anomaly detection by using PCA, treating outliers as anomalies. It is however not directly applicable to complex interdependent aeronautical systems as the study of the static source code might not be possible in the case of co-developed systems.

In [Fu et al., 2009], the authors filter out the variable parts of log messages using manually defined rules and cluster the rest with a similarity measure based on a custom weighted edit distance. They model the sequence between messages in a Finite State Automaton and perform anomaly detection by identifying sequences where impossible transitions are made. For an aeronautical system, it is however not plausible to create a FSA of reasonable size for all transitions.

In [Nagaraj et al., 2012], the authors first distinguish between state messages and event messages before engineering features from the results of statistical tests run on each distribution separately. Finally, the anomaly detection is performed by analysing outliers. In the case of aeronautical systems with a very large variety of messages and extremely rare failures the assumptions that outliers correspond to anomalies is not verified.

The same issue appears in [Vaarandi, 2003]. It describes a log parsing method that is still widely used to this day based on multiple passes over the logs to determine the most frequent words and the messages in which they occur.

[Sipos et al., 2014] relies on Multiple Instance Learning (MIL) to predict hard drive failures. Daily aggregates are used to simplify the model. Even though this appears promising for reducing the amount of data, it is not applicable to all kinds of data and requires a preliminary domain analysis.

[Makanju et al., 2012] describes another state-of-the-art log parsing algorithm, IPLoM, for Iterative Partitioning Log Mining. It is of a particular interest for this since it has been found in [He et al., 2016] to be the most effective log parsing algorithm available for the Hadoop File System (HDFS) benchmark dataset which shares several characteristics with the industrial dataset that this study is based on and in particular its size and the distributed nature of the system being studied. IPLoM works by performing several steps of hierarchical partitioning of the log messages before splitting them in constant and variable parts.

The study [He et al., 2016] reviews many of the recent works on log mining and log parsing and points out that very little is available in terms of benchmark, be it from the perspective of the datasets or of the implementation of the log parsers. This leads to difficulties in performance evaluation for researchers designing new methods and difficulties in applying log mining methods for potential users. This study also highlights the fact that the existing log parsing solutions are not distributed and tend to be very costly to run on large datasets in terms of execution time. Subsequent studies from the same authors in [He et al., 2017a] and [He et al., 2017b] propose two new log parsing algorithms POP and Drain. POP is based on recursive partitioning and optimized for low time complexity and Drain is an online algorithm which uses fixed depth parse tree. Though they both perform extremely well for log parsing with performances nearly optimal, once evaluated on a log mining task benchmark, their performances are similar to those of IPLoM. Finally, it is worth noting that the proposed methods are all rule-based and none of them are based on Natural Language Processing.

The article [Bertero et al., 2017] presents an approach based on the NLP method of word embedding for log parsing, sharing our observation that off-the-shelf NLP techniques could potentially yield significant improvement over traditional rule-based log parsing. The authors then use three different classifiers for log mining, Random Forest, Naive Bayes and Neural Networks, evaluated on one of the public benchmark datasets from [He et al., 2016] that we also use in our work reaching up to 90% accuracy with the Random Forest technique.

In [Kubacki and Sosnowski, 2017], the authors propose a parsing algorithm that uses detailed semantic analysis and custom dictionaries defined through statistical analysis of their text corpus, taken from logs of several servers at their disposal. Evaluation is then conducted on the same servers rendering comparisons with benchmark performances difficult.

[Du and Li, 2017] describes a log parser called Spell for Streaming Parser for Event Logs using Longest Common Subsequence (LCS). It is based on the LCS algorithm run in a streaming fashion to identify constant and variables part of log messages and classify them into message types. It does not however contain an evaluation on log mining metrics but only on log parsing. In [Du et al., 2017], the Spell log parser is used in combination with a Long Short-Term Memory (LSTM) deep neural network to evaluate the full pipeline on two public datasets, including the HDFS benchmark that we also use, and reaching up to 96% F-measure on it. It is worth noting that the deep learning approach used is considerably more complex to implement and parametrize than our proposed solution.

The table 4.2 summarizes the main principles discussed in these studies.

Reference	Rule-based	ML-based	NLP-based	Main difficulties for aeronautical application
[Xu et al., 2009]	×			Requires system expertise
[Fu et al., 2009]	×			High model complexity
[Nagaraj et al., 2012]	×			Frequentist anomaly detection
[Vaarandi, 2003]	×			Frequentist anomaly detection
[Sipos et al., 2014]	×			Requires system expertise; high model complexity
[Makanju et al., 2012]	×			Bad complexity scaling with message variety
[He et al., 2017a]	×	×		Some system expertise required
[He et al., 2017b]	×	×		Some system expertise required
[Bertero et al., 2017]		×	×	High model complexity
[Kubacki and Sosnowski, 2017]		×	×	Highly customized semantics
[Du and Li, 2017]	×			Bad complexity scaling with message variety
[Du et al., 2017]		×		Complex parametrization

Table 4.2: Summary of related work in log parsing and log mining

4.3 Log parsing

This section presents in detail the log parsing methods that we selected and implemented. They are organized in five categories, namely tokenization, semantic techniques, vectorization, model compression and classification methods.

Log parsing is understood here as the combination of methods used to transform the unstructured character strings that compose the log messages in a structured form suitable for failure prediction. Since the character strings form sentences or partial sentences understandable by a human operator, we take a special interest in methods from the field of NLP which aim specifically at extracting structure from documents written in natural languages by opposition to formal languages. By contrast with the parsing of formal languages, it means that we are parsing sentences that admit multiple interpretations as different words can cover multiple meanings depending on the grammatical and semantic context of the sentence.

The goal of this section is to present several easy to implement log parsing methods that can be used to engineer the features necessary for log mining.

4.3.1 Tokenization

The first method that we apply is a tokenizer. Its role is to process a text and break it into individual words in order to enable comparisons between sentences. In our case, we split the words based on whitespaces and punctuation marks and convert the characters to lower case. As an example, the sentence "Temperature too high: the CPU is overheating" would get broken down into the words "temperature", "too", "high", "the", "cpu", "is", "overheating". It is worth noting that this tokenizer has difficulties especially with contracted English forms such as "don't" or "it's" in which case other solutions based on regular expressions can be used. However, such forms are absent from our logs so these solutions were not considered.

4.3.2 Semantic techniques

The next methods we can apply on the individual words are stemming, synonym replacement and stopwords removal. Those methods are not strictly necessary to perform failure prediction as they simply aim to clean up the input text to reduce noise by looking into the meaning of the words considered. We study their impact on the prediction performances in section 4.5. Stemming is the process of removing inflected forms at the end of words to find out the stem of the word. We use it to conflate similar words such as "failure" and "fail" which are semantically close and convey a similar sense under the same stem "fail". This allows us to reduce the vocabulary and increase the similarity between sentences whose meanings are close. In our case, we apply a standard stemmer, the English snowball stemmer from the nltk package [Loper and Bird, 2002].

With the synonym replacement method, we replace words by their most common lemma corresponding to their most common meaning based on the standard wordnet module of the nltk package. This allows us to conflate words with similar meaning but different stems that cannot be caught by the stemming method. An example of that would be the words "present" and "detected" that are quite common in the log messages, the synonym replacement method can conflate them to the same representation, reducing further the vocabulary.

Stopwords removal is the process of removing stopwords. Stopwords are words that are frequent but have a low relevance for the classification, one such example is the article "the". If kept, the stopwords will induce noise in the classification model as two unrelated sentences could have many such words in common. In our case, we use the standard list of English stopwords found in the Apache Spark MLLib library [Meng et al., 2016] from which we remove the words "no", "not", "nor", "on", "off" and "any" (i.e. we will leave them in the logs) which are meaningful in the failure messages we are looking for. A tokenizer and stopwords

removal on the previous sentence "Temperature too high: the CPU is overheating" would result in "temperature", "high", "cpu", "overheating".

4.3.3 Vectorization

The next step in adapting the logs to prediction algorithms is to determine how to transform the sentences from a list of words into a vectorized representation. The first method that we try is the bag-of-words model. In this model, a sentence is represented as a vector of dimension equal to the total vocabulary size of the corpus. Each dimension corresponds to a given word and the value of the vector is equal to the number of times the given word appears in the sentence. This representation is easy to generate and the resulting model is easy to manipulate from a machine learning perspective but it loses the information contained in the ordering of the words in the sentence. For example, the sentences "Error: no network connection" and "Network connection: no error" would have the same representation in this model despite having different meanings.

The next model we try is the bi-gram model. In this model, instead of considering individual words, we consider pairs of consecutive words. This reduces the ambiguity generated by the loss of ordering. In the previous example, the vocabulary of the sentences would be respectively "Error no", "no network", "network connection" and "network connection", "connection no", "no error". This would capture the fact that both sentences are about network connection but have different conclusions. The drawback of this approach is that it increases the size of the vocabulary: given n different words, we can generate n^2 different bi-grams so, theoretically, a sentence should be represented in the bi-gram space by a vector of dimension n^2. Not all possible bi-grams are encountered which reduces this effect. In the example sentences, starting from a vocabulary of 4 words, we only get 5 different bi-grams.

We also study the tri-gram model where tuples of three consecutive words are considered. Given the limitations induced by the volume of logs and the increase in dimensionality of the n-gram model, we do not study n-grams for $n > 3$.

4.3.4 Model compression

The final step before we run the classification is the hashing trick method. This method reduces the size of the models in memory by applying a hashing function to the words and replacing the words with their computed hash. It considerably reduces the memory footprint of the models enabling faster and more complex computations at the expense of possible hash collisions and a reduction in the interpretability of the resulting models. In our case, we use the default hashingTF implementation of Apache Spark MLLib library.

4.3.5 Classification

The final step is the classification itself. The goal is to automatically categorize the log messages encoded in the previous steps so that failure prediction can be made based on the message categories. Two unsupervised clustering algorithms have been considered for this step, bisecting k-means [Steinbach et al., 2000] and Latent Dirichlet Allocation (LDA) [Blei et al., 2003].

Bisecting k-means is a variant of the k-means clustering algorithm. It starts with a unique cluster that regroups all samples and then iteratively splits the clusters starting with the one with the highest within set sum of squared errors. In our case, $k = 50$ was found to be the optimal value after a grid search step.

LDA is a topic modelling approach which attempts to generate topics associated with frequently co-occurring words and assign to every sentence a set of weights corresponding to the topic distribution inside the sentence. In our case, we set the number of topics to 50, as with the bisecting k-means and the number of iterations to define the topics to 100.

The final outcome of all the previous steps are features whose format depends on the classification technique used in the last step. In the case of bisecting k-means, the text associated to each message is replaced by a single value corresponding to its cluster label. An intuitive way to explain how it works is that the messages classified in the same cluster are very similar and are thus supposed to be about the same topic such as "Temperature test" or "Network failure", it is worth noting that topics are in this representation mutually exclusive. In the case of LDA, the log message is replaced by a vector of topic distribution. In that case, we extend the previous model with the possibility to be related to multiple topics. Hence if we have a "Network" topic and a "Failure" topic, we would find network failures by selecting messages with a topic distribution vector with a high value in both of these topics.

4.4 Log mining

This section describes the log mining process that we follow. We first present the modelling of a failure case that we want to predict. We then describe the classification step.

4.4.1 Modelling

In order to evaluate the efficiency of the log parsing, we target a well-known failure in one of the data storage sub-system of the aeronautical system that appears in our logs. The industrial dataset used is a record of one year of logs from an aeronautical system currently in use. It contains over 4.5 million log messages including 302 messages pertaining to the failure that must be predicted. We manually label every instance of that failure. The failure messages appear in burst because of retry mechanisms and the propagation of the consequences of the failure to several subsequent tests. The repeated failures have no practical interest since they

always happen in fixed sequences and do not reveal actual new events. Thus we filter them out and only keep the failure label of the first message in each sequence. After this filtering, we end up with 188 messages labelled as failures out of over 4.5 million log messages for an imbalance ratio of over $23.000 : 1$ in favour of the non-failure cases. We model the log messages as a time series. The features we associate with each message depend on the classification step that was used. If a bisecting k-means was performed, we associate to each message the sequence of cluster labels of the 20 previous messages. If a LDA was performed, we associate to each message the sequence of topic distribution vectors of the 20 previous messages. If less than 20 previous messages are available, we do not consider the current message for prediction. The size of the slilding time window has been selected as a compromise between the size of the model and the availability of computational resources.

4.4.2 Classification

The technique that we use to transform the features we engineered into predictions is the Random Forest technique [Criminisi et al., 2012]. It is one of the state-of-the-art techniques for classification. It requires little parametrization and is very efficient even in the case of extreme class imbalance and in the presence of noise. In this study, we set the number of decision trees at 20 with a maximum depth of 8, constructed using the Gini impurity metric. The parameters were chosen through grid-search on the industrial dataset and kept unchanged for the public benchmark.

Two other methods were considered, Gradient Boosted Trees [Criminisi et al., 2012] and Support Vector Machine [Cortes and Vapnik, 1995]. Their performances however did not match RF on any configuration. This is in line with the observation from [Bertero et al., 2017].

4.5 Results and interpretation

The results on the operational dataset are reported in the tables 4.3, 4.4 and 4.5. The comparison with the existing method on the benchmark dataset is presented in Table 4.6. All of the following results were validated through 3-fold cross-validation.

4.5.1 Metrics

We have chosen to report the standard performance metrics, precision, recall and F-score to evaluate our results. Precision and recall are also used in the benchmark performance evaluation in [He et al., 2016] and F-score is a hybrid metrics that will enable the comparison of techniques that trade-off between precision and recall. Defining the failure messages that we manually

labelled as positives and the non-failure ones as negatives, we have:

$$precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{4.1}$$

$$recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{4.2}$$

$$F - score = \frac{2 \times precision \times recall}{precision + recall} \tag{4.3}$$

4.5.2 Industrial dataset

In Table 4.3, the first observation that we can make is that no matter what log parsing techniques are used, the precision remains quite high with its lowest value achieved with the combination "LDA, bi-gram, stemming and synonyms replacement" at 75.8% and the next lowest at 85.7%. This is likely due to the high precision and resilience to noise of the RF learning algorithm. However, significant improvements can still be achieved as two combinations achieve 0 false positive for a reported precision over 99.5%. Those two combinations are "LDA, tri-gram, stemming and synonym replacement" and "LDA, bag-of-words, stemming, synonym replacement and stopwords removal".

Concerning recall in Table 4.4, the differences between the different combinations are more significant and the motivation for using the different models is more obvious. The most simple combination, bag-of-words and bisecting k-means only achieves 45.4% recall with only one other combination significantly lower, bisecting k-means, bi-gram, stemming and synonym replacement at 32.0% and a few combinations at comparable recall. The highest recall is achieved with LDA and bi-gram at 89.9%.

Finally regarding the overall performances summarized with the F-score in Table 4.5, the improvement brought by the log parsing techniques is even more obvious with every combination besides the previously mentioned "bisecting k-means, bi-gram, stemming and synonym replacement" having a higher score than the basic "bisecting k-means and bag-of-words". That is to say the top left combination bisecting k-means, bag-of-words can be considered a basic log parser with little to no NLP applied to it and, beside one specific combination, every NLP technique improves on it. The highest score of 0.937 is achieved by the combination which also had the highest recall "LDA and bi-gram" followed by "LDA, bi-gram, stopwords removal, stemming and synonym replacement" at 0.897.

Overall, it can be noted that the interactions between the different models and techniques are complex as none of them can be highlighted as systematically increasing or decreasing the metrics used. It is only when the models and methods are combined with each other that their usefulness becomes apparent. This is not surprising considering that these NLP methods are also in use in the field of speech processing and, similarly, the optimal combination of these

Table 4.3: Log mining precision with different log parsing schemes on the industrial dataset

		Bisecting k-means			LDA		
		Bag-of-words	Bi-gram	Tri-gram	Bag-of-words	Bi-gram	Tri-gram
Nothing	Nothing	0.865	0.930	0.971	0.962	0.978	0.988
	Stopwords removal	0.955	0.941	0.954	0.989	0.979	0.953
Stemming and synonym replacement	Nothing	0.927	0.883	0.945	0.857	0.758	>0.995
	Stopwords removal	0.944	0.980	0.967	>0.995	0.974	0.906

Table 4.4: Log mining recall with different log parsing schemes on the industrial dataset

		Bisecting k-means			LDA		
		Bag-of-words	Bi-gram	Tri-gram	Bag-of-words	Bi-gram	Tri-gram
Nothing	Nothing	0.454	0.670	0.723	0.666	0.899	0.436
	Stopwords removal	0.782	0.643	0.674	0.471	0.512	0.671
Stemming and synonym replacement	Nothing	0.604	0.320	0.723	0.586	0.499	0.459
	Stopwords removal	0.552	0.589	0.642	0.552	0.832	0.486

Table 4.5: Log mining F-score with different log parsing schemes on the industrial dataset

		Bisecting k-means			LDA		
		Bag-of-words	Bi-gram	Tri-gram	Bag-of-words	Bi-gram	Tri-gram
Nothing	Nothing	0.595	0.779	0.829	0.787	0.937	0.605
	Stopwords removal	0.860	0.764	0.790	0.638	0.672	0.788
Stemming and synonym replacement	Nothing	0.731	0.470	0.819	0.696	0.602	0.628
	Stopwords removal	0.697	0.736	0.772	0.710	0.897	0.633

techniques is still highly dependent on the dataset used with parameters as varied as the target language, the vocabulary size, the number of available audio samples and on target task. The key point is that the best combination we found with regard to F-score is quite robust, displaying the best recall over every other combination and one of the best precision. To ensure further that its performances are not coincidental beyond the cross-validation, in the next subsection, we proceed to test it on the public benchmark with the same parameters we used for the industrial dataset.

4.5.3 Public dataset

Table 4.6: Performance comparison against state-of-the-art approaches on the HDFS benchmark dataset

	Precision	Recall	F-score
SLCT	0.593	0.649	0.620
LogSig	0.963	0.634	0.765
IPLoM	0.975	0.665	0.791
Drain	0.975	0.665	0.791
POP	0.975	0.665	0.791
Best log parsing combination	>0.999	0.985	0.992

We present in Table 4.6 is the comparison of the best combination "LDA and bi-gram", with existing results from the studies [He et al., 2016], [He et al., 2017a] and [He et al., 2017b] on a public benchmark dataset from an HDFS cluster[6]. This HDFS dataset was chosen among the available benchmark sets in [He et al., 2016] as the largest dataset and also the closest one to the industrial dataset we used. This dataset consists in over $11,000,000$ messages belonging to 29 different categories generated from a 203-node cluster. The datasets contains over $16,000$ messages marked as WARN that are considered failures. The public benchmark is an extract of $2,000$ messages from this dataset. Though the problem of class imbalance is less pronounced than on the industrial dataset, the HDFS dataset is very similar in terms of size and structure to the aeronautical logs. An example of log messages from this set can be found in table 4.7 The parameters used for the algorithms were kept at the same value as with the industrial dataset. The comparison shows clearly that the natural language processing inspired approach improves recall significantly while maintaining a high precision leading to a very significant increase of the F-score from 0.791 for rule-based methods and 0.96 for Deep Learning based methods to 0.992.

[6]https://github.com/logpai/logparser/tree/master/logs/HDFS

2008-11-11 03:40:58	BLOCK*NameSystem.allocateBlock:/user/root/randtxt/[...]
2008-11-11 03:40:59	Receiving block blk_904 src: /10.251.43.210:55700

Table 4.7: Example of raw HDFS text log

This could be seen as surprising because the traditional rule-based log parsing methods put focus on producing a structure that perfectly transposes the information in the log messages with performances such that significant improvements seemed unlikely. Our interpretation is that building a perfectly unambiguous parsing is actually not a desirable outcome when dealing with log messages because of the logs inherent ambiguity. The messages are implemented by a human operator to be read and understood by another human operator with natural language that is inherently ambiguous and flexible. This flexibility enables a human reader to understand when several messages, though they might be worded differently, refer to a similar issue. For example, when dealing with network problems, it is common that the actual log message vary depending on the subsystem that attempted to connect. It is obvious for someone reading the log messages that they are related but an unambiguous rule-based approach will not detect it because it operates under the assumption that one spelling is equivalent to one lemma. Through the use of natural language processing and clustering techniques in our approach, we introduce ambiguity and flexibility by foregoing the word order and voluntarily confusing semantically similar messages imitating in that sense the approach of a human operator and leading to better results.

Finally, it is worth noting that these excellent performances were obtained by using the exact same methods and parameters used on the industrial dataset which implies that this pipeline is reliable for log parsing and rules out the possibility that the results were the product of a coincidence enabled by the many combinations tested and further demonstrate the robustness of our method.

4.6 Conclusion

In this contribution, we propose a general scheme for the failure prediction problem in the context of industrial systems. Our approach is based on log mining, which is a promising approach to achieve failure prediction as many systems already implement detailed logs and on log parsing which is a critical preliminary step. We focus on simple NLP techniques or combination thereof and on the interaction between log parsing and log mining and their effect on the performances of failure prediction. To that end, we provide a detailed performance analysis on both an industrial dataset of a system currently in use and on a large publicly available benchmark dataset in order to compare the performances of our approach with state-of-the-art algorithms. On the industrial dataset, we achieve 97.8% precision and 89.9% recall using the "LDA and bi-gram" log parsing combination and a simple RF-based log mining. On the HDFS dataset,

the same method improves the precision and recall from respectively 97.5% and 66.5% to over 99.9% and 98.5% and the F-score improves over the study [Du et al., 2017] from 96% to 99.2% with a more simple and robust pipeline. We finally offer an interpretation of the improvement yielded by NLP based methods over traditional rule-based methods. We considered the possibility to use more complex NLP techniques such as word embedding but, given the excellent performances of the simpler methods, any possible performance increase over them would most likely not be measurable and deemed that obtaining the same performances with simpler methods is a stronger result. Several more advanced NLP techniques such as word embedding, topic segmentation or a more elaborate synonym replacement process would be worth investigating in the future however the most pressing issue would be to gather a more complex benchmark dataset in order to evaluate said techniques.

In general, with regards to the objective of this chapter to study how to efficiently map full text event logs to suitable inputs for classification algorithms, we have determined that a combination of simple natural language processing techniques provides the best results on both our own dataset and on benchmark sets.

Chapter 5

Combining Federated and Active Learning for Distributed Ensemble-based Failure Prediction

Contents

In this chapter, we propose a new Distributed Learning method to answer the problem stated at the beginning of this thesis of distributed learning with strong constraints on client-side computational resources and communication budget. To do so, we start by putting the problem in context. We review existing works and their limitations with regards to our goal; then we describe the algorithm we propose and, finally, we demonstrate its performance on a public benchmark.

5.1 Introduction

5.1.1 General problem

Throughout the two previous contributions, we have focused on concretely determining which ML would work the best with the constraints of aeronautical data and proposing our own when it was relevant. This has enabled us to set the frame needed for this final contribution by determining which base methods would be available to us. In this contribution, we go back to the initial goal to monitor aeronautical systems for failure prediction using Machine Learning. The problem remains that aeronautical systems generate too much data to be able to handle the prediction on the aircraft for lack of computational resources. Data links are available between the aircraft and the ground but they are too expensive and not fast enough to transfer all the data. Therefore, we cannot directly apply a cloud computing paradigm which would consist in offloading all the data to a remote location such as a data centre with sufficient computational resources and collecting the results.

In this situation, we can identify characteristics that the solution will need to demonstrate. Firstly, it will need to be able to classify data from normal readings, i.e. situations where there are no signs of imminent failures, using a minimal amount of computation power. This is key to be able to process the large volume of data generated by the aeronautical systems without overloading the data link. Secondly, the solution for failure prediction will need a mechanism to identify interesting data, i.e. data that could be indicative of an imminent failure with regards to a pre-defined threshold, and transfer this data from the aircraft to a central server on the ground without spending too much communication budget. This is a direct consequence of the observation that the amount of computational power on board is too limited to give an accurate prediction of failure risks. Thirdly, we need a way to train a prediction model on the ground without accessing the data from the aircraft. This is a direct consequence of the previous two requirements as only a small amount data can be sent to the central server. But, as accurate predictions are expected from it, we need a mechanism to enable centralized learning without accessing the raw data.

Different solutions already exist for each of the characteristics we have identified taken individually. The main concern here is to figure out how to fulfil all the requirements simultaneously but the most immediate source of inspiration can be found in existing paradigms already discussed in this thesis.

5.1.2 Distributed Learning paradigms

Two sub-fields of Machine Learning that provide elements of answer have been identified in section 2.4.1 as Active Learning and Federated Learning. They have been designed with slightly different use cases in mind which obviously leads to slightly different requirements but that nevertheless match our own quite well.

The Active Learning paradigm offers a way to model the envisioned relationship between the aircraft and the central server on the ground. Its premiss is that the model being trained has access to a third-party source of information, called an oracle, to reveal labels when necessary and its goal is to maximize the performance of the trained model under a request budget. We can immediately draw a parallel with the second characteristic of the solution we identified with the aircraft playing the role of the active learning model in training and the central server playing the role of the oracle albeit an imperfect one (an unusual situation but already studied, for example in [Miller et al., 2014]). In that case, the active learning paradigm offers an approach to select the hard-to-classify data and balance the amount of requests with the communication budget. Regarding the first characteristic however, further investigations are required as a number of works in Active Learning favour a greedy approach incompatible with the idea that normal readings need to be processed quickly.

The Federated Learning paradigm is directly related to the third identified characteristic of the solution, i.e. enabling centralized learning without accessing raw data. The communication budget and the computational resources of the hosts are also concerns that are considered in this approach. However, it is purely driven by the central server and does not provide any mechanism for the clients to assess their performance and request clarifications from the central server.

The goal of this contribution is to propose a new approach that fulfils all three requirements by combining Active Learning and Federated Learning. The performances are evaluated on a public benchmark and compared to existing solutions.

The contribution is organized as follows. First, we study existing works in Active Learning and Federated Learning in section 5.2. Next, we detail our solution in section 5.3. We present our results and interpret them in section 5.4 and section 5.5 concludes the contribution.

5.2 Related Work

In this section, we review state-of-the-art works related to our problem. A more comprehensive review of Active Learning and Federated Learning can be found in section 2.4 but the goal here is to focus on the applicability of the techniques reviewed with regards to the three identified requirements.

Note that for aeronautics applications, following the results in chapter3, we would like to use RF because of its ease of interpretation, resilience to noise and that it can be proved reliable in a range of safety scenarios. An issue remains in that RF is easy to distribute but there is no guarantee on the performance of individual trees.

5.2.1 Federated Learning

The terminology of Federated Learning was coined recently in [McMahan et al., 2016]. It describes a Distributed Learning situation where the following observations apply:

- There is a massive number of clients with limited computational resources each holding a unique fraction of the dataset. Their availability is subject to changes. This matches well the aeronautical use case where individual aircraft can be seen as clients. Their availability is not perfect and depends on the satellite connection.
- The data are not independent and identically distributed between clients. In the original use case because it is assumed that different users will interact differently with their mobile device and type differently. In the aeronautical use case, this holds true as different aircraft can face different flight conditions depending on their route and the way their systems are used by the crew and the passengers.
- Communications are allowed between the clients and a central server but they are limited by a communication budget and, for privacy concerns, no exchange of raw data is allowed. In the aeronautical use case, the privacy requirement between the client and the server is relaxed but the communication constraint is certainly relevant.

The original solution proposed is to initialize randomly a deep learning model at the server level and to gather incremental SGD updates from the clients while exchanging only the model parameters with them.

Since the publication of the original article, several contributions have followed and improved different aspects of this method. More specifically, [Konečnỳ et al., 2016] proposes a way to improve the communication efficiency and [Bonawitz et al., 2016] and [Geyer et al., 2017] offer new insights as how to better guarantee that the privacy concerns are respected.

The Federated Learning approach is of high interest for us as it describes a framework for massively distributed and communication-efficient learning in a context very close to ours. In particular, Federated Learning would be a great answer to our third requirement, learning in a centralized model without downloading data from the clients. There are however limitations. We have so far identified RF as our most promising choice for a base learning model. Also, Federated Learning is developed for DNN and uses incremental SGD updates from the clients. However, RF does not use SGD so we cannot apply Federated Learning directly. Also, Federated Learning does not offer any solution for our first identified characteristic. For that we turn to Active Learning.

5.2.2 Active Learning

Active Learning is a sub-field of semi-supervised learning where a third-party, called an oracle, can provide missing labels on request. The goal is then to learn the most accurate model possible

under a given request budget. A survey of different existing approaches can be found in [Settles, 2009] and a summary of the statistical foundations can be found in [Cohn et al., 1996]. A more developed discussion about this is available in section 2.4. The general principle of Active Learning is interesting for our problem because it allows us to dynamically balance uncertainty on client-side and communication budget which would offerus a solution to fulfil the second requirement we identified of allowing requests from the clientto the server for uncertain samples under a communication budget.

However, a first obstacle that we have to overcome is that most Active Learning works are pool-based and use a greedy sampling approach meant to minimize the intervention of the oracle. Such oracle is often a human operator with the limitation of being prohibitively expensive in terms of computation; a human operator is indeed still orders of magnitude slower and more expensive than a complex computation. Examples of this can be found in [Georgala et al., 2014], [Joshi et al., 2009] and [Li and Guo, 2007] illustrating the versatility of pool-based Active Learning with three applications on very different use cases of respectively spam detection, image classification and network intrusion detection. In our case, however, we envision the oracle as another more powerful model faster and less expensive than a human operator and we assume that complex computations are not possible on the client-side.

There are however other works on online active learning that adopt a stream-based approach. In that approach, the decision to request a label from the oracle must be done immediately and leads to more efficient decision from a computation point of view. Examples of this approach can be found in [Bouguelia et al., 2013] for an application to document classification and in [Smailović et al., 2014] for an application to sentiment analysis. [De Rosa and Cesa-Bianchi, 2017] is an article of particular interest to this thesis as it proposes a new way to train decision trees using an active learning approach that focuses on minimizing the risk of selecting a sub-optimal split when a new leaf is added to the tree by computing confidence intervals. This approach is both computationally cheap and provides an easy way to trade-off the uncertainty on the model and the communication budget. The only limitation is that this model is not distributed so we still need to implement a mechanism to leverage remote data sources to create a centralized model.

5.3 Proposed Algorithm

The approach we propose to combine Active Learning and Federated Learning is to use the confidence based active online DT from [De Rosa and Cesa-Bianchi, 2017] as a base model that is learned on the client side and sent to the server. The server aggregates the client DTs in an ensemble model similar to a RF to learn an accurate model without accessing the raw data. Finally, the individual DTs use the ensemble model from the server as an oracle for label requests. The figure 5.1 provides a high-level illustration of the approach envisioned.

To get more into details, each client has access to its local dataset and train a confidence

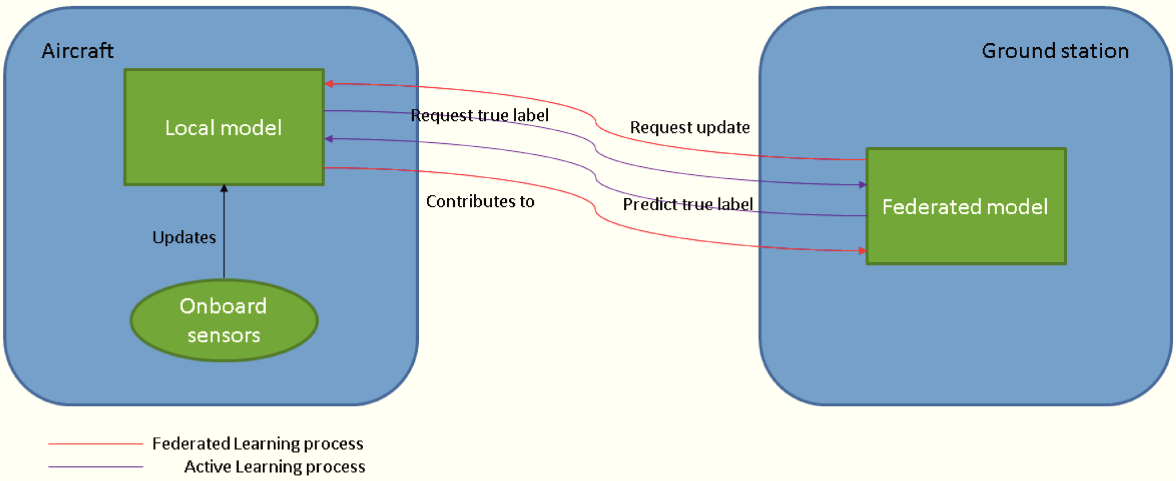


Figure 5.1: Illustration of Federated Active Learning

based active online DT locally. Regularly, the central server sends an update request to the clients. The clients reply with their current DT. The server creates an ensemble model by collecting all the DTs. The ensemble model relies on majority voting to classify new samples, that is to say a new sample is run through each DT and its classification is the most frequent result among all DTs. A first parameter to control the communication budget of the approach will therefore be how often the central server request updates from the clients.

From the client perspective, they have at their disposal a stream of observations to train their local DT. As described in [De Rosa and Cesa-Bianchi, 2017], whenever the uncertainty surrounding a new split in the DT exceeds a certain confidence threshold, a request is made to reveal the true label of the samples needed to decide on the right split and a part of the request budget is spent. Here the requests are sent to the server and the labels are decided using the ensemble model. There is second parameter to control the communication budget of the approach in the request budget of each client.

There is a last point that needs to be clarified in the initialization of the model. Indeed, the necessary labels for the training of the DT are provided by the ensemble model but the ensemble model can only provide labels after it has received base models from the clients. In order to get around this constraint, we assume that some amount of labelled historical data is already available and can be used in the first round of update to initialize the DTs. This assumption is perfectly justified in the aeronautical use case where there is plenty of historical data available and the problem of label availability only concerns recent data where the true status of a system has not been clarified through a maintenance operation yet.

The proposed process is described in algorithm 1.

5.4 Experimental Results

5.4.1 Experimental settings

In order to enable a meaningful comparison with other works, we have selected for our experiments the MOA airlines dataset[7], one the of the public benchmarks that was used in [De Rosa and Cesa-Bianchi, 2017]. Though it is related to the air transport industry, it is not strictly speaking a dataset from an aeronautical system but it presents the advantages of being public and enabling comparisons with other solutions. This dataset consists of over $500,000$ samples each with a label and seven features. Each sample corresponds to real flight and the task is to predict if the flight has taken off on schedule or if it was delayed. The seven features available are summarized in table 5.1. The imbalance ratio in this dataset is quite even at 0.80 in favour of the flights that are not delayed. This means that some of the techniques such as SVM or LR ruled out in chapter 3 could have acceptable performances on this particular but, since the target

[7] https://moa.cms.waikato.ac.nz/datasets/

Algorithm 1 Federated Active Forest

▷ F_t is the forest at round t, $T_{k,t}$ is the k^{th} tree of F_t, C is the set of clients, C_k is the k^{th} client of C, $n_{k,t}$ is the dataset gathered by the k^{th} client between rounds t and $t-1$, B_{max} is the maximum request budget and $B_{k,t}$ is the request budget remaining for the k^{th} client at round t

Start
  $F_0 \leftarrow$ SEEDFOREST(C)
  for each round $t = 1, 2, ...$ **do**
    for each client C_k in C **do**
      $T_{k,t} \leftarrow$ CLIENTUPDATE($(T_{k,t-1})$)
    end for
    $F_t \leftarrow \bigcup_{\forall k}(T_{k,t})$
  end for
End
function SEEDFOREST(C) ▷ Server-side function
  for each client C_k in C **do**
    $T_{k,0} \leftarrow$ SEEDTREE()
  end for
  $F_0 \leftarrow \bigcup_{\forall k}(T_{k,0})$
end function
function CLIENTUPDATE($T_{k,t-1}$) ▷ Client-side function
  $B_{updated} \leftarrow B_{k,t-1} +$ UPDATEBUDGET$(sizeof(n_{k,t}))$ ▷ Request budget increased to match dataset growth
  $T_{k,t}, B_{k_t} \leftarrow$ CONFIDENCEDECISIONTREE$(T_{k,t-1}, n_{k,t} B_{updated})$
  return $(T_{k,t})$
end function
function SEEDTREE ▷ Client-side function
  $T_{k,0}, B_{k,0} \leftarrow$ CONFIDENCEDECISIONTREE$(Root_Tree, n_{k,0}, B_{max})$ ▷ $Root\_Tree$ is a tree limited to a root node, $n_{k,0}$ is assumed to be fully labelled
  return $(T_{k,0})$
end function

application for aeronautical systems would not display this level of class balance and since our proposed method is DT based, we do not consider them further.

Feature	Description	Format
Airline	Unique identifier of the airline operating the flight	alphanumeric
Flight numeric	Unique flight ID	numeric
AirportFrom	Origin airport unique IATA code	string
AirportTo	Destination airport unique IATA code	string
DayOfWeek	Day of the week	numeric
Time	Timestamp (scale not disclosed)	numeric
Length	Length of the flight (min)	numeric

Table 5.1: Description of features in the MOA airlines dataset

To evaluate the performance of our algorithm, we report the precision, recall and F-score as in the previous contributions. Given the relatively balanced nature of this dataset and to provide as many elements of comparison as possible with other works done on this benchmark, we also report the accuracy. Defining the delayed flights as positives and the flights on schedule as negatives as the labels in the dataset suggest us to do, we have:

$$accuracy = \frac{true\ positive + true\ negative}{true\ positive + true\ negative + false\ positive + false\ negative} \tag{5.1}$$

$$precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{5.2}$$

$$recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{5.3}$$

$$F - score = \frac{2 \times precision \times recall}{precision + recall} \tag{5.4}$$

In order to ensure the validity of our results, we apply a 2-fold cross-validation by randomly splitting out the dataset into two sets, one for training and one for testing, measuring the performances a first time and repeating every measurement a second time after inverting the training and testing sets. Regarding the parameters of this experiment, we examine more specifically the performance of our method with regards to the number of hosts among which the samples will be split, the maximum request budget available for the active learning process and the number of communication rounds for the federated learning process. When studying the variations of a parameter, others were kept at constant with 20 communication rounds, 5 clients and a request budget of 10%. The numbers of communication rounds and clients were chosen so that the constraints of distributed learning would be visible and the request budget was chose so that comparisons with other methods are facilitated.

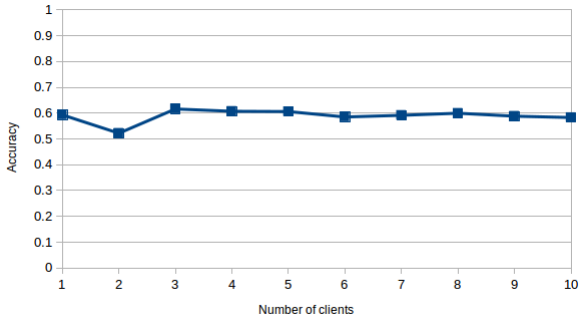
5.4.2 Results and discussion

The results are regrouped by variable parameter and displayed in Figure 5.2 for the number of hosts, 5.4 for the request budget and 5.3 for the number of communication rounds. As additional information, table 5.4.2 provide additionnal insights on the average performances of individual clients for varying number request budgets.

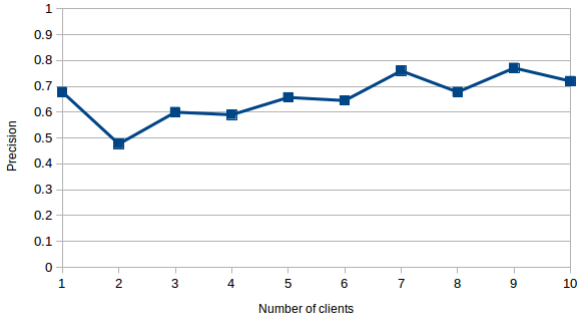
In Figure 5.2, the first observation that we can make is that the results seem slightly unstable, especially considering that cross-validation has already been applied to them so one could expect smoother curves. In fact, the relative instability is a side effect of the majority voting employed in the ensemble model. When the number of clients is even, there is a limit case where each classification outcome gets the same number of vote. In such a case, our model was configured to favour the delay prediction, resulting in an increase in recall and a decrease in precision. This limit case is more frequent for small numbers of clients. Beside this, we can observe that the accuracy is stable at about 61% which is an excellent level of performance comparable to state-of-the-art results in the non-distributed case with [De Rosa and Cesa-Bianchi, 2017] reporting about 62% of accuracy with a similar request budget. This validates the potential of the approach we propose for communication-efficient distributed learning. For the other metrics, we can observe that they decrease slightly for high number of clients which illustrate the fact that the distributed learning constraint does have a cost on the performance even it is relatively small with the F-score decreasing from about 49.75% with a single host to 41.3% with 10 hosts.

Concerning Figure 5.3, the results are more straightforward. We observe that as the number of communication rounds increase, the model accuracy is quite stable but it hides the fact that precision is increasing at the expense of recall leading to an overall decreasing F-score from 51% to 30%. The interpretation to this behaviour is that the ensemble model is being updated too frequently. The base DT model makes use of a grace period parameter which defines a minimum amount of samples that need to be collected before a request can be made. The default value used was 100, the same as in the original publication. However, as the dataset is split between multiple clients and further down between multiple communication rounds. It is possible when the number of communication rounds is too high that between consecutive updates of the ensemble model, no updates of the local DTs have been made and consequently no change is made on the ensemble model leading to "skipped" communication rounds that eventually lead to lower the overall performances. We tested this interpretation by lowering the grace period to 30 and observed that performances for a high number of communication rounds were increased.

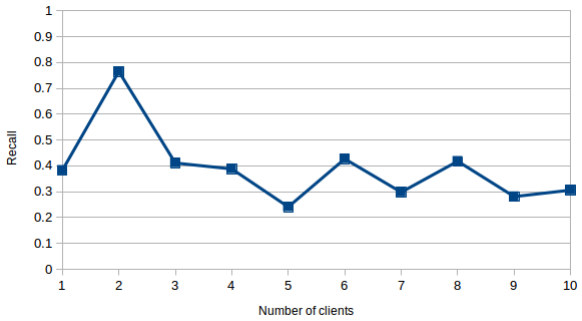
Regarding Figure 5.4, it is worth noting that it is using a different scale than the other figures as the request budget is presented as a logarithmic scale. The reason for that is to highlight the extreme stability of the performances with a very slight increase of the accuracy for high request budget from 56% to 59%. The interpretation for the very low values is largely due to the initialization step that relies on fully labelled data. Given 5 clients, 20 rounds and a $50/50$



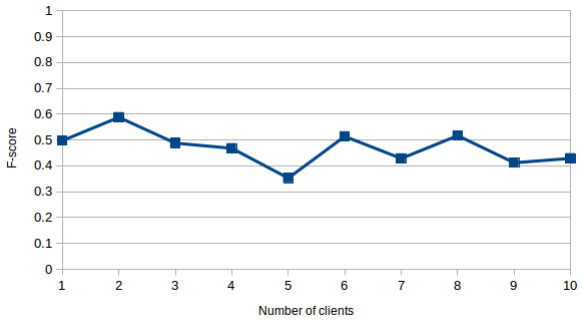
(a) Accuracy



(b) Precision

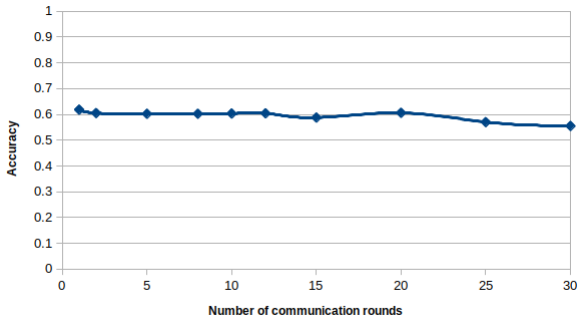


(c) Recall

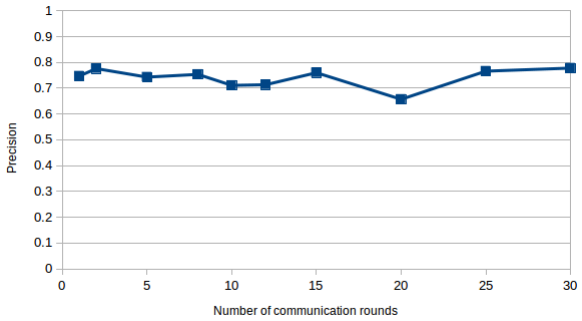


(d) F-score

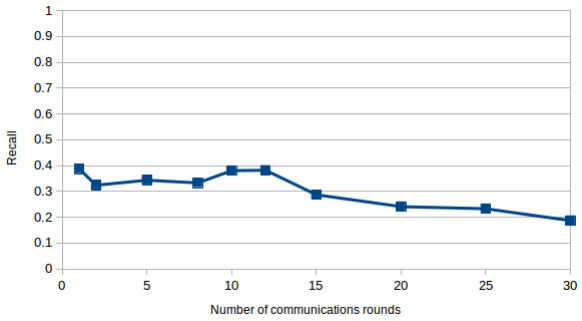
Figure 5.2: Performance with regards to number of hosts



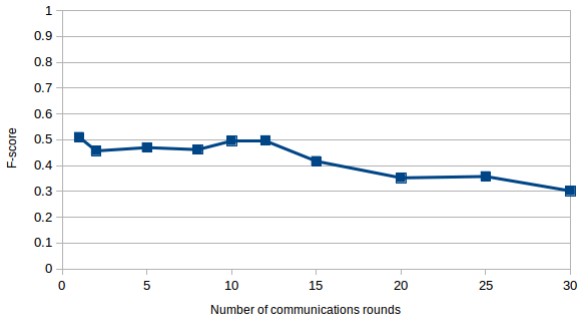
(a) Accuracy



(b) Precision

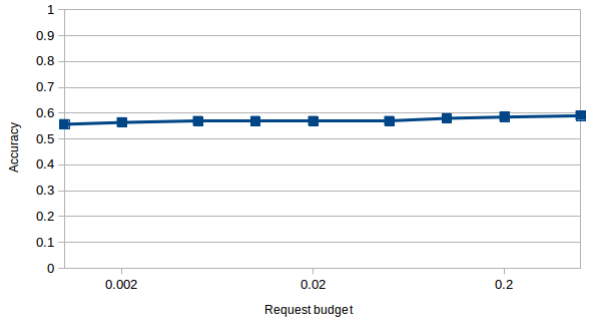


(c) Recall

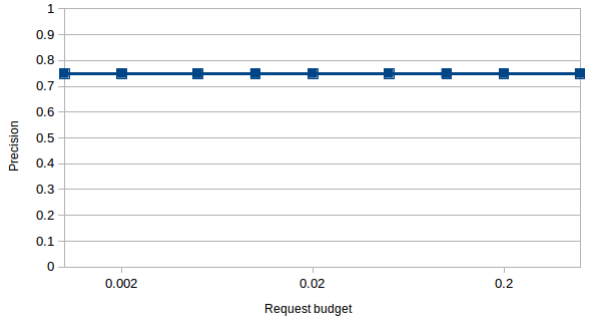


(d) F-score

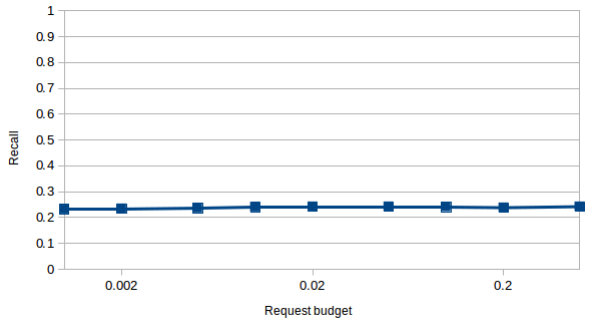
Figure 5.3: Performance with regards to the number of communication rounds



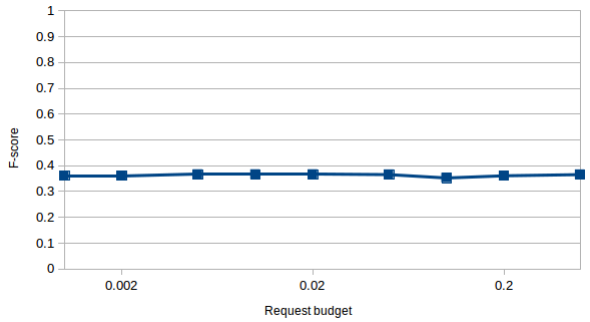
(a) Accuracy



(b) Precision



(c) Recall



(d) F-score

Figure 5.4: Performance with regards to the request budget

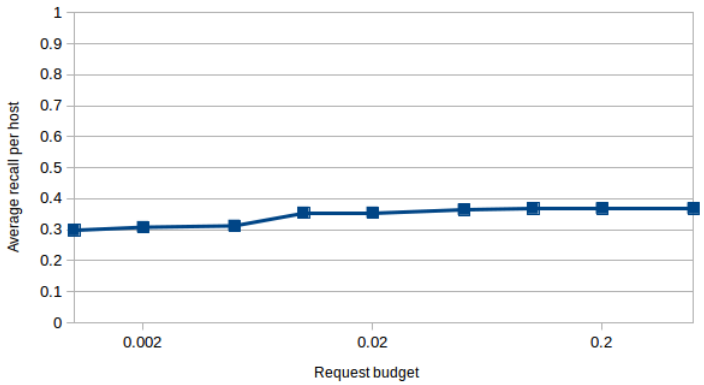


Figure 5.5: Average recall per client with regards to the budget request

training/test split, the amount of data used in the initialization only represents 2.5% of the total dataset so even for an extremely low request budget of 0.1%, it is necessary to label 2.51% of the dataset. Still, given that typical range for request budget in similar studies such as [De Rosa and Cesa-Bianchi, 2017] starts at 10% this is a very promising result, showing that our approach is efficient for extremely low request budgets thus fulfilling one of the requirements for applicability to aeronautical systems. Finally, we have included Figure 5.4.2 to give insight as to what changes in the model for different request budget. Indeed, if the performance of the ensemble model is very stable, it is hiding the fact that the performance of the local DTs. Selecting recall as the best illustration of this, we see that the average recall of individual DTs increases from about 30% to 37% when the request budget increases. This is significant as local DTs would act in a real-world implementation as back-up in situations where the data link between the aircraft and the ground would be unavailable, as such, being able to guarantee a certain level of performance of the local model is important.

Overall, it can be noted that the performances of the Federated Active Learning approach that we propose are very stable and very close to state-of-the-art level of performance for non-distributed learning.

5.5 Conclusion

In this contribution, we propose a new applied Machine Learning framework to enable real-time learning for aeronautical systems. To do so, we identify three necessary characteristics for a solution, review existing solutions for each of them and propose a new method to fulfil all of them at the same time. Our method is a combination of Active Learning and Federated Learning. It relies on training confidence decision trees at client level and aggregating them at server level to create an ensemble model to act as an oracle. We provide a detailed performance analysis of

our method on a public benchmark and compare our results to state-of-the-art classifiers.

Our approach achieves an accuracy of up to 61% very close to state-of-the-art levels of performance of 62% despite the additional constraint of distributed learning and with a very consistent level of performance with label budgets as low as 3% thanks to the initialization step and the use of the ensemble model.

In general, with regards to the objective of this chapter, we have determined a new method that fulfils all the requirements for real-time distributed failure prediction of aeronautical systems by combining existing works in an original way.

Chapter 6

Conclusion

Contents

6.1 Thesis outcome

One of the main challenges of the aeronautical industry is to balance the requirements to guarantee the highest level of reliability, so that air transport remains the safest way to travel, and to keep the maintenance costs down in order to survive in a competitive market environment. Those two requirements are often at odds but predictive maintenance offers a chance to reconcile these conflicting goals.

In our work, we have sought to address the problem of applying ML-based failure prediction algorithms to aeronautical systems in real-time. To do so, we determined what are the unique challenges that prevent offline failure prediction algorithms to be applied to aeronautical systems. We identified three main challenges:

- The high level of reliability of aeronautical systems leads to extremely rare failures. This complicates the situation from a ML perspective as it is difficult to draw statistically relevant conclusions with so few examples and from the learning model perspective it means that we need to deal with an extreme case of rare event prediction.
- The process to get a new aeronautical system certified and deployed is very demanding and time-consuming. This implies that we have to deal with the existing data collection as any suggested change would take too long to be deployed and gather a sufficient amount of failures.

- The computational resources available on-board are limited and so is the data link between the aircraft and the ground. This means that we can neither do all the computations on-board nor can we move all the data to the ground and do the failure prediction there. We have to find a hybrid solution.

The contributions of this thesis addressed all these problems point by point. In chapter 3, we studied the problem of rare event prediction and extreme class imbalance from the perspective of applied ML. We analysed the performances of existing sampling techniques and learning models and established that sampling techniques are not performing sufficiently well yet for the level of imbalance present in aeronautical systems. Superior results are achieved by using learning techniques that are already resilient to imbalance and the best results are achieved by Random Forests.

In chapter 4, we took interest in the data format problem. In particular, we studied how to map the most widespread format in our systems, full-text logs, to a numerical input that would be suitable for classification algorithms. We proposed a ML pipeline based on a combination of simple natural language processing techniques and showed that it was robust, easy to parametrize and provided the best results on both our industrial dataset and on public benchmarks.

Finally, in chapter 5, we addressed the problem of distributed learning with limited resources. We reviewed existing works and proposed a new solution based on the combination of Active Learning and Federated Learning techniques that is efficient from a computation perspective and that offers a way to trade-off model accuracy and communications budget. We then evaluated our approach on a public dataset to show that it maintains high levels of performance even when compared to state-of-the-art non-distributed solutions.

Finally, a key point of this thesis that is worth noting is the constant concern of relating every problem with a public dataset. It is unfortunately not always the case with industrial applications in general and aeronautical works in particular and the lack of reproducibility it leads to has been a source of difficulties for us and undoubtedly for other researchers around the world as well. Consequently, we made a point of always including results with public data in order to ensure that our contributions can be reproduced, compared with other works and generally built upon.

6.2 Perspectives

Along this thesis we detailed several questions that we have not been able to investigate in the timeframe of the thesis. We identified what we consider the most promising problems in this section.

We have concluded in chapter 3 that existing sampling techniques are not adapted to extreme class imbalance in the range of $1 : 10,000$ or more. An interesting research direction would

be to try to figure out the requirements for such a technique to exist and determine if it would be possible to adapt the existing sampling techniques for that problem or possibly to propose new sampling techniques for extreme class imbalance. This contribution could be focused around a comparative performance studies of the different approaches for varying levels of class imbalance. This could reasonably be achieved in a matter of months.

Regarding our contribution on Federated Active learning, it has only been tested so far on a public benchmark but a case study of its application to an industrial system, preferably an aeronautical system, would be interesting. We are currently investigating what would be the best possible target for this study with our industrial partner and what would be good public dataset to serve as a proxy for this contribution. We cannot accurately estimate the timeframe for this as it would highly depend on the specificities of the system chosen but a broad assessment from previous work done would place it between six months and a year provided a good proxy dataset is identified.

Finally, this thesis was aimed at producing a failure prediction model but another aspect of predictive maintenance that we have not addressed is the organisation of the logistics and business processes. Additional contributions could be made on how to design a predictive maintenance program and a maintenance policy around the Federated Active Learning method we proposed. The main questions would probably be centred around i) the real-time aspect of the failure predictions and how to best take it into account in the aircraft and at the destination airport, ii) the possibility offered by the failure prediction model to trade-off increased accuracy of the prediction at the expense of communication costs and how to optimize the use of this feature from a business perspective and iii) the question of the varying maintenance capabilities at different airports and how to best take it into account when planning the maintenance of systems whose failure is not critical. A thorough study of this problem is quite ambitious and could be a suitable starting point for another thesis.

Appendices

Appendix A

Résumé substantiel

A.1 Introduction

A.1.1 Contexte et problématique

Le but de cette thèse est de proposer un système de maintenance prédictive en temps réel adapté aux contraintes propres aux systèmes aéronautiques. De nombreux capteurs permettent de mesurer et d'enregistrer la progression de multiples variables à bord. Ces mesures permettent aujourd'hui d'analyser les pannes s'étant déjà produites. Afin d'améliorer la fiabilité des systèmes aéronautiques et de réduire leurs coûts de maintenance, il serait préférable d'utiliser ces mesures afin d'anticiper les pannes. Plusieurs constructeurs et équipementiers aéronautiques ont déjà adoptés cette approche mais de manière strictement hors-ligne, c'est à dire en exploitant les données uniquement après atterrissage. Cette application est déjà intéressante en soi, les détails d'implémentation de ces méthodes n'étant pas publics. Pour aller plus loin, l'objectif de cette thèse est de proposer est de proposer un modèle de maintenance prédictive de systèmes aéronautiques capable de fonctionner pendant le vol. Cet objectif impose des contraintes fortes sur les ressources de calcul: là où les modèles hors-lignes peuvent disposer d'autant de ressources que nécessaires dans le contexte très favorable d'un data center notre modèle embarqué doit utiliser les ressources disponibles à bord et respecter plusieurs contraintes pour être autorisé dans l'avion par les autorités de certification aéronautique. La solution que nous explorons pour contourner ces limitations est l'utilisation d'une liaison de données entre l'avion et le sol. En effet, il devient courant pour les vols commerciaux d'offrir de la connectivité avec Internet et il s'agit de l'une des spécialités de l'entreprise accueillant cette thèse. Cette connectivité peut-être obtenue de différentes manières à l'aide, par exemple, d'une connexion satellite ou d'une connexion directe air-sol. Le problème qui demeure malgré tout est que peu importe la solution de connectivité utilisée, comparé aux connexions disponibles au sol, la connexion entre l'avion et le sol est couteuse financièrement et ne dispose que d'une bande passante limitée. C'est

pourquoi transmettre au sol l'ensemble des mesures des capteurs embarqués selon le paradigme du cloud computing n'est pas réaliste. Ainsi certains traitements doivent nécessairement avoir lieu à bord pour ne transmettre que des données pertinentes. Le cœur de cette thèse est de déterminer un ensemble de deux modèles d'apprentissage, l'un à bord et l'autre au sol, qui collaboreront sur le problème de prédiction d'anomalies et une architecture logicielle capable de les supporter afin de produire des prédictions aussi précises que possible sans saturer la liaison de données.

A.1.2 Définitions des concepts généraux et industriels

A.1.2.1 Caractérisations des systèmes aéronautiques

Un système est un ensemble de composants interdépendants formant un tout unifié. On pourrait définir un système aéronautique comme un système qui se trouve dans un avion. Cette définition bien que correcte d'un point de vue sémantique est inexacte. L'industrie aéronautique est soumise à un grand nombre de contraintes réglementaires fixées par des autorités de certification qui ont la responsabilité d'autoriser ou non un système à bord des vols commerciaux de son espace aérien. La nature exacte de ces contraintes varie selon la criticité du système envisagé et l'autorité de certification consultée. L'objectif de cette thèse n'est pas de faire une revue exhaustive de ces contraintes mais il est néanmoins intéressant d'observer que plusieurs caractéristiques des systèmes aéronautiques en découlent et sont pertinentes dans ce cadre. Les plus importantes sont la fiabilité élevée, la nécessité de mesurer et d'enregistrer les performances, la durée importante des cycles de développement et l'interdépendance des systèmes.

A.1.2.2 Maintenance prédictive

Les approches possibles des politiques de maintenance des systèmes industriels peuvent se ranger en trois catégories. La plus simple à mettre en place est la maintenance corrective. Il s'agit d'intervenir uniquement lorsqu'un système tombe en panne. Cela conduit à minimiser le nombre d'interventions de maintenance et donc les coûts associés mais impacte la fiabilité des systèmes et dans le cas de systèmes interdépendants, comme une chaîne d'assemblage, peut générer des impacts au-delà du système étudié. La maintenance préventive vise à remplacer et réparer les systèmes avant qu'ils ne tombent en panne suivant un agenda fixe basé sur l'étude de la fiabilité des systèmes en terme de durée de vie. Cette approche permet de diminuer le nombre de maintenances non planifiées et les impacts des pannes mais conduit à remplacer des systèmes fonctionnant encore correctement. Elle est préférée dans les circonstances où les conséquences d'une panne sont considérées comme inacceptables, par exemple avec des systèmes médicaux. Enfin la troisième approche à laquelle nous nous intéressons plus particulièrement dans cette thèse est la maintenance prédictive dont l'objectif est de planifier les opérations de maintenance "juste à temps", c'est-à-dire juste avant que le système ne tombe en panne, de

sorte que le nombre d'opérations soient minimisées comme dans l'approche corrective mais sans laisser le système effectivement tomber en panne comme dans l'approche préventive. La maintenance prédictive s'appuie sur des mesures de l'état du système par des capteurs. Son principal défaut est la complexité de sa mise en œuvre. Dans les cas de systèmes complexes et interdépendants en particulier, une approche formelle se basant sur une étude physique du système pour en déduire son état devient rapidement trop complexe nécessitant des connaissances pointues sur tous les systèmes impliquées et sur leurs interactions. Une approche statistique peut être employée pour étudier la corrélation entre l'état du système et les résultats mesurés mais cette approche demande des ressources de calcul considérables dans le cas de systèmes complexes pour atteindre des niveaux de performance acceptables. Les dernières avancées en apprentissage automatique permettent néanmoins de manipuler de façon de plus en plus simple des volumes de données qui auraient été considérés trop massif il y a encore une décennie.

A.1.2.3 Présentation de la thèse

Cette thèse est structurée autour de la problématique de mise en place d'un système de maintenance prédictive temps réel de systèmes aéronautiques. Trois principaux problèmes ont été identifiés pour atteindre cet objectif. Premièrement, la fiabilité des systèmes aéronautiques implique qu'il y a peu de données disponibles concernant les pannes. D'un point de vue statistique nous sommes donc confrontés à problème de détection d'événements rares. Plusieurs solutions statistiques existent pour pallier à ce problème. Une première étape de cette thèse sera l'étude de ces solutions et de leur efficacité. Deuxièmement, les cycles de développement très longs des systèmes aéronautiques et les coûts financiers associés rendent impossible l'implémentation de changement dans les systèmes étudiés et plus spécifiquement des paramètres mesurés. La solution proposée devra donc s'appuyer sur les formats de données existants. En conséquence, une deuxième étape dans la thèse sera d'étudier comment adapter les méthodes d'apprentissage automatique aux données disponibles. Enfin, le dernier problème à adresser sera l'apprentissage distribué entre l'avion et la station au sol. La contribution ici sera un nouvel algorithme pouvant être exécuté en parallèle entre un hôte disposant de sources de données importantes mais de ressources de calcul limitées et un hôte avec des ressources de calcul importantes mais aucun accès aux données récentes sous contraintes de communication.

A.2 Conclusion

A.2.1 Résultats de la thèse

L'un des grands défis de l'industrie aéronautique est de concilier l'exigence du plus haut niveau de fiabilité avec la nécessité économique de maintenir les coûts de maintenance à un niveau raisonnable pour survivre dans un environnement hautement compétitif. La maintenance prédictive offre un moyen de réunir ces deux objectifs contradictoires. Dans cette thèse, nous avons identifiés trois principaux obstacles à l'application des techniques de maintenance prédictive en temps réel aux systèmes aéronautiques. Le premier est le haut niveau de fiabilité des systèmes aéronautiques entraînant des difficultés à l'application de méthodes statistiques par l'extrême rareté des événements à prédire. Le second est la nécessité de s'adapter aux formats de données existants souvent éloignés des standards de l'apprentissage automatique afin de ne pas imposer de changements coûteux à mettre en place financièrement et en termes de temps. Le troisième obstacle est la nécessité de s'adapter aux contraintes propres de cet environnement distribué que sont la rareté des ressources de calcul embarqués et la présence d'un budget de communication maximal. Les contributions de cette thèse adressent chaque obstacle point par point. Nous avons étudié dans le chapitre 3 les méthodes statistiques dans une situation de prédiction d'événements extrêmement rares. Nous avons établis que les solutions classiques notamment d'échantillonnage développées dans des contextes moins difficiles ne sont pas suffisamment efficaces. Les meilleurs résultats sont obtenus en utilisant des techniques d'apprentissage résistantes au problème de déséquilibre par construction et la meilleure méthode pour notre problème est celle des Random Forest. Dans le chapitre 4, nous nous sommes intéressés au problème du format des données. En particulier, nous étudiés comment faire correspondre le format le plus répandue dans nos systèmes aéronautiques, les journaux d'événements textuels, avec le format numérique attendu en entrée des méthodes de classification. Nous avons proposé une chaîne de traitement complète d'apprentissage automatique basé sur une combinaison de méthodes simples issues du traitement du langage naturel puis nous avons montré que cette chaîne de traitement était robuste, simple à paramétrer et produisait les meilleurs résultats à la fois sur notre jeu de données industriel et sur un banc d'essai publique. Enfin dans le chapitre 5, nous adressons le problème de l'apprentissage distribué avec contraintes. Nous proposons une méthode originale basée sur la combinaison de techniques issues de l'apprentissage actif et l'apprentissage fédéré. Nous évaluons notre solution sur un jeu de données publique et montrons qu'elle maintient des niveaux de performances équivalents aux solutions à l'état de l'art qui n'intègrent pas les contraintes d'apprentissage distribué. Pour finir, un point clé de cette thèse que nous souhaitons souligner est la préoccupation constante que nous avons eu de mettre en avant des jeux de données publiques pour valider nos résultats. Ce n'est malheureusement pas toujours le cas pour les applications industrielles en général et aéronautiques en particulier ce qui conduit à des grandes difficultés à reproduire les résultats mis en avant. Cela a été une source de difficulté pour nous comme ça l'est sans aucun doute pour d'autres chercheurs. En

conséquence toutes nos contributions incluent des résultats obtenus sur des données publiques afin qu'elles puissent être reproduites, comparées et généralement servir de base de travail pour d'autres.

A.2.2 Perspectives

Tout au long de cette thèse nous avons souligné plusieurs questions que nous n'avons pas été en mesure d'adresser dans le cadre de notre travail. Nous identifions ici les plus prometteuses. Nous avons conclus dans le chapitre 3 que les méthodes d'échantillonnages n'étaient pas adaptés aux situations de déséquilibre de classe extrême. Il serait intéressant d'étudier les conditions nécessaires à la mise au point de telles techniques en adaptant des techniques existantes ou en proposant de nouvelles. En ce qui concerne l'apprentissage actif fédéré proposé dans le chapitre 5, il a été testé uniquement sur un jeu de données publique pour le moment. Nous souhaitons valider ses performances sur un système aéronautique et travaillons à identifier un jeu de données candidat prometteur. Enfin, cette thèse a eu pour objectif de proposer un modèle de prédiction de panne mais un autre aspect de la maintenance prédictive que nous n'avons pas étudié est l'aspect de la logistique et des processus industriels. Des contributions supplémentaires pourraient permettre d'identifier comment mettre en place une politique de maintenance autour de notre modèle afin de tirer parti de ses spécificités pour maximiser son impact sur les coûts de maintenance.

Bibliography

[Albawi et al., 2017] Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. 15

[Alpcan and Bauckhage, 2009] Alpcan, T. and Bauckhage, C. (2009). A distributed machine learning framework. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 2546–2551. IEEE. 18, 22

[Aussel et al., 2017] Aussel, N., Jaulin, S., Gandon, G., Petetin, Y., Fazli, E., and Chabridon, S. (2017). Predictive Models of Hard Drive Failures Based on Operational Data. In *16th IEEE Int. Conf. on Machine Learning and Applications, ICMLA, Cancun, Mexico, December 18-21*, pages 619–625. 6

[Aussel et al., 2018] Aussel, N., Petetin, Y., and Chabridon, S. (2018). Improving Performances of Log Mining for Anomaly Prediction Through NLP-Based Log Parsing. In *26th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS, Milwaukee, WI, USA*, pages 237–243. 7

[Bansal et al., 2004] Bansal, D., Evans, D. J., and Jones, B. (2004). A real-time predictive maintenance system for machine systems. *International Journal of Machine tools and manufacture*, 44(7-8):759–766. 11, 13

[Bertero et al., 2017] Bertero, C., Roy, M., Sauvanaud, C., and Trédan, G. (2017). Experience report: Log mining using natural language processing and application to anomaly detection. In *28th International Symposium on Software Reliability Engineering (ISSRE 2017)*, page 10p. 45, 47, 48, 52

[Blagus and Lusa, 2010] Blagus, R. and Lusa, L. (2010). Class prediction for high-dimensional class-imbalanced data. *BMC Bioinformatics*, 11(1):523. 26

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022. 51

[Bonawitz et al., 2016] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. (2016). Practical secure aggregation for federated learning on user-held data. *CoRR*, abs/1611.04482. 22, 64

[Bordes et al., 2005] Bordes, A., Ertekin, S., Weston, J., and Bottou, L. (2005). Fast kernel classifiers with online and active learning. *J. Mach. Learn. Res.*, 6:1579–1619. 15

[Botezatu et al., 2016] Botezatu, M. M., Giurgiu, I., Bogojeska, J., and Wiesmann, D. (2016). Predicting Disk Replacement Towards Reliable Data Centers. In *22d ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*. 29, 33, 34

[Bottou, 2010] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer. 14

[Bouguelia et al., 2013] Bouguelia, M.-R., Belaïd, Y., and Belaïd, A. (2013). A stream-based semi-supervised active learning approach for document classification. In *2013 12th International Conference on Document Analysis and Recognition*, pages 611–615. IEEE. 65

[Breiman, 2017] Breiman, L. (2017). *Classification and regression trees*. Routledge. 16

[Burnaev et al., 2014] Burnaev, E. et al. (2014). Rare event prediction techniques in application to predictive maintenance of aircraft. In *Proceedings of ITaS conference*, pages 32–37. 12, 13

[Byington et al., 2002] Byington, C. S., Roemer, M. J., and Galie, T. (2002). Prognostic enhancements to diagnostic systems for improved condition-based maintenance [military aircraft]. In *Proceedings, IEEE aerospace conference*, volume 6, pages 6–6. IEEE. 11, 13

[Chawla et al., 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357. 33, 34

[Cohn et al., 1996] Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1996). Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145. 20, 22, 65

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297. 15, 52

[Criminisi et al., 2012] Criminisi, A., Shotton, J., and Konukoglu, E. (2012). Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2–3):81–227. 17, 52

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314. 15

[De Rosa and Cesa-Bianchi, 2017] De Rosa, R. and Cesa-Bianchi, N. (2017). Confidence decision trees via online and active learning for streaming data. *Journal of Artificial Intelligence Research*, 60:1031–1055. 21, 22, 65, 67, 70, 74

[Du and Li, 2017] Du, M. and Li, F. (2017). Spell: Streaming parsing of system event logs. In *Proceedings - 16th IEEE International Conference on Data Mining, ICDM 2016*, pages 859–864, United States. Institute of Electrical and Electronics Engineers Inc. 47, 48

[Du et al., 2017] Du, M., Li, F., Zheng, G., and Srikumar, V. (2017). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1285–1298, New York, NY, USA. ACM. 47, 48, 59

[El-Shimi, 2017] El-Shimi, A. (2017). Predicting Storage Failures. In *VAULT - Linux Storage and File Systems Conference, Cambrige (MA)*. 29

[Friedman et al., 2001] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin. 13, 14, 15

[Fu et al., 2009] Fu, Q., Lou, J.-G., Wang, Y., and Li, J. (2009). Execution anomaly detection in distributed systems through unstructured log analysis. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, ICDM '09, pages 149–158, Washington, DC, USA. IEEE Computer Society. 45, 46, 48

[Gal et al., 2017] Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1183–1192. JMLR. org. 20

[Galar et al., 2012] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. (2012). A Review on Ensembles for the Class Imbalance Problem: Bagging, Boosting, and Hybrid-Based Approaches. *IEEE Trans. on Systems, Man, and Cybernetics*, 42(4):463–484. 40

[Georgala et al., 2014] Georgala, K., Kosmopoulos, A., and Paliouras, G. (2014). Spam filtering: An active learning approach using incremental clustering. In *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*, WIMS '14, pages 23:1–23:12, New York, NY, USA. ACM. 20, 65

[Geyer et al., 2017] Geyer, R. C., Klein, T., and Nabi, M. (2017). Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*. 22, 64

[Grall et al., 2002] Grall, A., Dieulle, L., Berenguer, C., and Roussignol, M. (2002). Continuous-time predictive-maintenance scheduling for a deteriorating system. *IEEE Trans. on Reliability*, 51(2):141–150. 11, 13

[Hamerly and Elkan, 2001] Hamerly, G. and Elkan, C. (2001). Bayesian Approaches to Failure Prediction for Disk Drives. In *18th Int. Conf. on Machine Learning*. 29, 41

[Hashemian, 2011] Hashemian, H. M. (2011). State-of-the-art predictive maintenance techniques. *IEEE Trans. on Instrumentation and Measurement*, 60(1):226–236. 11, 13

[He et al., 2016] He, P., Zhu, J., He, S., Li, J., and Lyu, M. R. (2016). An evaluation study on log parsing and its use in log mining. In *Dependable Systems and Networks (DSN), 2016 46th Annual IEEE/IFIP International Conference on*, pages 654–661. IEEE. 45, 46, 47, 52, 57

[He et al., 2017a] He, P., Zhu, J., He, S., Li, J., and Lyu, M. R. (2017a). Towards automated log parsing for large-scale log data analysis. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1–1. 47, 48, 57

[He et al., 2017b] He, P., Zhu, J., Zheng, Z., and Lyu, M. R. (2017b). Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 33–40. 47, 48, 57

[Hinton, 2002] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800. 32

[Horenbeek and Pintelon, 2013] Horenbeek, A. V. and Pintelon, L. (2013). A dynamic predictive maintenance policy for complex multi-component systems. *Reliability Engineering & System Safety*, 120:39 – 50. 10, 13

[Houlsby et al., 2011] Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. (2011). Bayesian Active Learning for Classification and Preference Learning. Technical Report arXiv:1112.5745. 20

[Jolliffe, 2011] Jolliffe, I. (2011). *Principal component analysis*. Springer. 32

[Joshi et al., 2009] Joshi, A. J., Porikli, F., and Papanikolopoulos, N. (2009). Multi-class active learning for image classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379. 20, 65

[Kemkemian et al., 2013] Kemkemian, S., Enderli, C., and Larroque, A. (2013). New industrial testing and maintenance for aesa radars. In *2013 7th European Conference on Antennas and Propagation (EuCAP)*, pages 806–810. IEEE. 12, 13

[Kipersztok et al., 2015] Kipersztok, O., Nodelman, U., and Swayne, M. (2015). Recommendations for time to replace parts on machines. US Patent 9,218,694. 12

[Konečnỳ et al., 2016] Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*. 22, 64

[Korvesis, 2017] Korvesis, P. (2017). *Machine Learning for Predictive Maintenance in Aviation*. Theses, Université Paris-Saclay. 12, 13

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA. Curran Associates Inc. 16, 17

[Kubacki and Sosnowski, 2017] Kubacki, M. and Sosnowski, J. (2017). Holistic processing and exploring event logs. In Romanovsky, A. and Troubitsyna, E. A., editors, *Software Engineering for Resilient Systems*, pages 184–200, Cham. Springer International Publishing. 19, 22, 45, 47, 48

[Kuzborskij et al., 2015] Kuzborskij, I., Orabona, F., and Caputo, B. (2015). Transfer learning through greedy subset selection. In Murino, V. and Puppo, E., editors, *Image Analysis and Processing — ICIAP 2015*, pages 3–14, Cham. Springer International Publishing. 19

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. E. (2015). Deep learning. *Nature*, 521(7553):436–444. 15

[Lewis and Gale, 1994] Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In Croft, B. W. and van Rijsbergen, C. J., editors, *SIGIR '94*, pages 3–12, London. Springer London. 20, 22

[Li et al., 2014a] Li, H., Parikh, D., He, Q., Qian, B., Li, Z., Fang, D., and Hampapur, A. (2014a). Improving rail network velocity: A machine learning approach to predictive maintenance. *Transportation Research Part C: Emerging Technologies*, 45:17–26. 11, 12, 13

[Li et al., 2014b] Li, J., Ji, X., Jia, Y., Zhu, B., Wang, G., Li, Z., and Liu, X. (2014b). Hard Drive Failure Prediction using Classification and Regression Trees. In *44th IEEE/IFIP Conf. on Dependable Systems and Networks*. 29, 32, 33, 34

[Li et al., 2014c] Li, M., Anderson, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., Long, J., Shekita, E. J., and Su, B.-Y. (2014c). Scaling distributed machine learning with the parameter server. In *Operating Systems Design and Implementation (OSDI)*, pages 583–598. 18, 19, 22

[Li and Guo, 2007] Li, Y. and Guo, L. (2007). An active learning based tcm-knn algorithm for supervised network intrusion detection. *Computers and Security*, 26(7):459 – 467. 20, 65

[Loper and Bird, 2002] Loper, E. and Bird, S. (2002). Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*, pages 63–70. Association for Computational Linguistics. 49

[Ma et al., 2015] Ma, A., Traylor, R., Douglis, F., Chamness, M., Lu, G., Sawyer, D., Chandra, S., and Hsu, W. (2015). RAIDShield: Characterizing, Monitoring, and Proactively Protecting Against Disk Failures. *ACM Trans. on Storage, Special issue FAST'2015*, 11(4):17:1–17:28. 29

[Makanju et al., 2012] Makanju, A., Zincir-Heywood, A. N., and Milios, E. E. (2012). A lightweight algorithm for message type extraction in system application logs. *IEEE Transactions on Knowledge and Data Engineering*, 24(11):1921–1936. 45, 47, 48

[Mazzoletti et al., 2017] Mazzoletti, M. A., Bossio, G. R., De Angelo, C. H., and Espinoza-Trejo, D. R. (2017). A model-based strategy for interturn short-circuit fault diagnosis in pmsm. *IEEE Transactions on Industrial Electronics*, 64(9):7218–7228. 11, 13

[McMahan et al., 2016] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. (2016). Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*. 21, 22, 64

[Meng et al., 2016] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al. (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241. 49

[Miller et al., 2014] Miller, B., Kantchelian, A., Afroz, S., Bachwani, R., Dauber, E., Huang, L., Tschantz, M. C., Joseph, A. D., and Tygar, J. D. (2014). Adversarial active learning. In *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, pages 3–14. ACM. 21, 22, 63

[Mobley, 2002] Mobley, R. K. (2002). *An introduction to predictive maintenance*. Elsevier. 10, 13

[Murray et al., 2003] Murray, J. F., Hughes, G. F., and Kreutz-Delgado, K. (2003). Hard Drive Failure Prediction using Non-parametric Statistical Methods. In *ICANN/ICONIP*. 28, 29

[Murray et al., 2005] Murray, J. F., Hughes, G. F., and Kreutz-Delgado, K. (2005). Machine learning methods for predicting failures in hard drives: A multiple-instance application. *J. Mach. Learn. Res.*, 6:783–816. 28

[Nagaraj et al., 2012] Nagaraj, K., Killian, C., and Neville, J. (2012). Structured comparative analysis of systems logs to diagnose performance problems. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 26–26. USENIX Association. 45, 46, 48

[Negnevitsky, 2001] Negnevitsky, M. (2001). *Artificial Intelligence: A Guide to Intelligent Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition. 15

[Peteiro-Barral and Guijarro-Berdiñas, 2013] Peteiro-Barral, D. and Guijarro-Berdiñas, B. (2013). A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11. 18, 22

[Pinheiro et al., 2007] Pinheiro, E., Weber, W.-D., and Barroso, L. A. (2007). Failure Trends in a Large Disk Drive Population. In *FAST*, volume 7, pages 17–23. 26, 27, 29, 31, 32, 33

[Rosenblatt, 1957] Rosenblatt, F. (1957). The perceptron–a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory. 15

[Rumelhart et al., 1988] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA. 15

[Schroeder and Gibson, 2007] Schroeder, B. and Gibson, G. A. (2007). Disk Failures in the Real World: What Does an MTTF of 1,000,000 Hours Mean to You? In *5th USENIX Conf. on File and Storage Technologies*, FAST '07, Berkeley, CA, USA. USENIX Association. 27, 29

[Settles, 2009] Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison. 20, 22, 65

[Shafiee, 2015] Shafiee, M. (2015). Maintenance logistics organization for offshore wind energy: Current progress and future perspectives. *Renewable Energy*, 77:182–193. 10, 13

[Sierra-Canto et al., 2010] Sierra-Canto, X., Madera-Ramirez, F., and Uc-Cetina, V. (2010). Parallel training of a back-propagation neural network using cuda. In *2010 Ninth International Conference on Machine Learning and Applications*, pages 307–312. 17, 22

[Sipos et al., 2014] Sipos, R., Fradkin, D., Mörchen, F., and Wang, Z. (2014). Log-based predictive maintenance. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, August*, pages 1867–1876. 46, 48

[Smailović et al., 2014] Smailović, J., Grčar, M., Lavrač, N., and Žnidaršič, M. (2014). Stream-based active learning for sentiment analysis in the financial domain. *Information sciences*, 285:181–203. 65

[Song et al., 2012] Song, S., Lake, P. J., and McCullough, J. K. (2012). Fleet performance optimization tool for aircraft health management. US Patent 8,340,948. 12

[Steinbach et al., 2000] Steinbach, M., Karypis, G., Kumar, V., et al. (2000). A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Boston. 51

[Ullah et al., 2017] Ullah, I., Yang, F., Khan, R., Liu, L., Yang, H., Gao, B., and Sun, K. (2017). Predictive maintenance of power substation equipment by infrared thermography using a machine-learning approach. *Energies*, 10(12):1987. 11, 12, 13

[Vaarandi, 2003] Vaarandi, R. (2003). A data clustering algorithm for mining patterns from event logs. In *IP Operations & Management, 2003.(IPOM 2003). 3rd IEEE Workshop on*, pages 119–126. IEEE. 45, 46, 48

[Valerio et al., 2017] Valerio, L., Passarella, A., and Conti, M. (2017). A communication efficient distributed learning framework for smart environments. *Pervasive and Mobile Computing*, 41:46–68. 19, 20, 22

[Wallace et al., 2011] Wallace, B. C., Small, K., Brodley, C. E., and Trikalinos, T. A. (2011). Class Imbalance, Redux. In *IEEE 11th Conf. on Data Mining*. 33

[Wang et al., 2011] Wang, Y., Miao, Q., and Pecht, M. (2011). Health monitoring of hard disk drive based on Mahalanobis distance. In *Prognostics and System Health Managment Conf.*, pages 1–8. 28, 29, 41

[Watcharapichat et al., 2016] Watcharapichat, P., Morales, V. L., Fernandez, R. C., and Pietzuch, P. (2016). Ako: Decentralised deep learning with partial gradient exchange. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, pages 84–97. ACM. 18, 22

[Watkins and Dayan, 1992] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292. 14

[Xu et al., 2009] Xu, W., Huang, L., Fox, A., Patterson, D., and Jordan, M. I. (2009). Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 117–132. ACM. 45, 46, 48

[Zhang and Kwok, 2014] Zhang, R. and Kwok, J. (2014). Asynchronous distributed admm for consensus optimization. In *International Conference on Machine Learning*, pages 1701–1709. 18, 22

[Zhao et al., 2010] Zhao, Y., Liu, X., Gan, S., and Zheng, W. (2010). Predicting Disk Failures with HMM-and HSMM-based Approaches. In *Industrial Conf. on Data Mining*, pages 390–404. Springer. 28, 29

[Zhu et al., 2013] Zhu, B., Wang, G., Liu, X., Hu, D., Lin, S., and Ma, J. (2013). Proactive Drive Failure Prediction for Large Scale Storage Systems. In *IEEE 29th Symp. on Mass Storage Systems and Technologies*, pages 1–5. 29, 31, 32, 33, 34, 41