Machine Learning for Predictive Maintenance in Aviation

Panagiotis Korvesis

NNT : 2017SACLX093

Thèse de doctorat de l'Université Paris-Saclay préparée à l'école Polytechnique

Ecole doctorale n°580
Sciences et technologies de l'information et de la communication
Spécialité de doctorat : Informatique

par

M. Panagiotis Korvesis

Apprentissage Automatique pour la Maintenance Predictive dans le Domaine de l'Aviation

Thèse présentée et soutenue à Palaiseau, le 21 novembre 2017.

Composition du Jury :

M.	Albert Bifet	Professeur Telecom Paristech	(Président)
M.	Aristidis Likas	Professeur University of Ioannina	(Rapporteur)
M.	Andreas Stafylopatis	Professeur National Technical University of Athens	(Rapporteur)
M.	Themis Palpanas	Professeur University of Paris Descartes	(Examinateur)
M.	Jesse Read	Professeur chargé de cours École Polytechnique	(Examinateur)
M.	Stephane Besseau	Ingénieur AIRBUS	(Examinateur)
M.	Michalis Vazirgiannis	Professeur École Polytechnique	(Directeur)

ABSTRACT

The increase of available data in almost every domain raises the necessity of employing algorithms for automated data analysis. This necessity is highlighted in predictive maintenance, where the ultimate objective is to predict failures of hardware components by continuously observing their status, in order to plan maintenance actions well in advance. These observations are generated by monitoring systems usually in the form of time series and event logs and cover the lifespan of the corresponding components. Analyzing this history of observations in order to develop predictive models is the main challenge of data driven predictive maintenance.

Towards this direction, Machine Learning has become ubiquitous since it provides the means of extracting knowledge from a variety of data sources with the minimum human intervention. The goal of this dissertation is to study and address challenging problems in aviation related to predicting failures of components on-board. The amount of data related to the operation of aircraft is enormous and therefore, scalability is a key requirement in every proposed approach.

This dissertation is divided in three main parts that correspond to the different data sources that we encountered during our work. In the first part, we targeted the problem of predicting system failures, given the history of Post Flight Reports. We proposed a regression-based approach preceded by a meticulous formulation and data pre-processing/transformation. Our

method approximates the risk of failure with a scalable solution, deployed in a cluster environment both in training and testing. To our knowledge, there is no available method for tackling this problem until the time this thesis was written.

The second part presents our approach for analyzing logbook data, which consist of text describing aircraft issues and the corresponding maintenance actions and it is written by maintenance engineers. The logbook contains information that is not reflected in the post-flight reports and it is very essential in several applications, including failure prediction. However, since the logbook contains text written in natural language, it contains a lot of noise that needs to be removed in order to extract useful information. We tackled this problem by proposing an approach based on vector representations of words (or word embeddings). Our approach exploits semantic similarities of words, learned by neural networks that generated the vector representations, in order to identify and correct spelling mistakes and abbreviations. Finally, important keywords are extracted using Part of Speech Tagging.

In the third part, we tackled the problem of assessing the health of components on-board using sensor measurements. In the cases under consideration, the condition of the component is assessed by the magnitude of the sensor's fluctuation and a monotonically increasing trend. In our approach, we formulated a time series decomposition problem in order to separate the fluctuation from the trend by solving an optimisation problem. To quantify the condition of the component, we compute a risk function which measures the sensor's deviation from its normal behavior, which is learned using Gaussian Mixture Models.

RÉSUMÉ

L'augmentation des données disponibles dans presque tous les domaines soulève la nécessité d'utiliser des algorithmes pour l'analyse automatisée des données. Cette nécessité est mise en évidence dans la maintenance prédictive, où l'objectif est de prédire les pannes des systèmes en observant continuellement leur état, afin de planifier les actions de maintenance à l'avance. Ces observations sont générées par des systèmes de surveillance habituellement sous la forme de séries temporelles et de journaux d'événements et couvrent la durée de vie des composants correspondants. Le principal défi de la maintenance prédictive est l'analyse de l'historique d'observation afin de développer des modèles prédictifs.

Dans ce sens, l'apprentissage automatique est devenu omniprésent puisqu'il fournit les moyens d'extraire les connaissances d'une grande variété de sources de données avec une intervention humaine minimale. L'objectif de cette thèse est d'étudier et de résoudre les problèmes dans l'aviation liés à la prévision des pannes de composants à bord. La quantité de données liées à l'exploitation des avions est énorme et, par conséquent, l'évolutivité est une condition essentielle dans chaque approche proposée.

Cette thèse est divisée en trois parties qui correspondent aux différentes sources de données que nous avons rencontrées au cours de notre travail. Dans la première partie, nous avons ciblé le problème de la prédiction des pannes des systèmes, compte

tenu de l'historique des Post Flight Reports. Nous avons proposé une approche statistique basée sur la régression précédée d'une formulation méticuleuse et d'un prétraitement / transformation de données. Notre méthode estime le risque d'échec avec une solution évolutive, déployée dans un environnement de cluster en apprentissage et en déploiement. À notre connaissance, il n'y a pas de méthode disponible pour résoudre ce problème jusqu'au moment où cette thèse a été écrite.

La deuxième partie consiste à analyser les données du livre de bord, qui consistent en un texte décrivant les problèmes d'avions et les actions de maintenance correspondantes. Le livre de bord contient des informations qui ne sont pas présentes dans les Post Flight Reports bien qu'elles soient essentielles dans plusieurs applications, comme la prédiction de l'échec. Cependant, le journal de bord contient du texte écrit par des humains, il contient beaucoup de bruit qui doit être supprimé afin d'extraire les informations utiles. Nous avons abordé ce problème en proposant une approche basée sur des représentations vectorielles de mots. Notre approche exploite des similitudes sémantiques, apprises par des neural networks qui ont généré les représentations vectorielles, afin d'identifier et de corriger les fautes d'orthographe et les abréviations. Enfin, des mots-clés importants sont extraits à l'aide du Part of Speech Tagging.

Dans la troisième partie, nous avons abordé le problème de l'évaluation de l'état des composants à bord en utilisant les mesures des capteurs. Dans les cas considérés, l'état du composant est évalué par l'ampleur de la fluctuation du capteur et une tendance à l'augmentation monotone. Dans notre approche, nous avons formulé un problème de décomposition des séries temporelles

afin de séparer les fluctuations de la tendance en résolvant un problème convexe. Pour quantifier l'état du composant, nous calculons à l'aide de Gaussian Mixture Models une fonction de risque qui mesure l'écart du capteur par rapport à son comportement normal.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

Over the past decade, industry began investigating the potential of data that have been stored for long time but hardly exploited. Advances in data science (or machine learning?) turned very hard problems into feasible tasks that can bring great added value to almost every industrial domain. Nowadays, *learning* models from data in order to perform tasks such as predictions or information extraction has become very popular and therefore, machine learning algorithms are becoming indispensable tools in decision making. In this context, this CIFRE[1] thesis summarizes our efforts to exploit machine learning for problems arising in the aviation industry and more precisely, in *predictive maintenance*.

1.1 SCOPE OF THE THESIS

Civil aviation is one of the most popular means of transportation that serves billions of passengers each year (Figure 1.1). A key parameter for the efficiency of this industry is the availability of the aircraft, which is the percentage of time an aircraft is in good condition and capable to fly. The availability is usually reduced by unexpected failures, whose occurrences render the aircraft unable to perform a flight and therefore, sometimes leading to

[1] Conventions Industrielles de Formation par la REcherche a.k.a CIFRE, is an industrial Ph.D. program in France and is supported by ANRT (Association Nationale Recherche Technologie) http://www.anrt.asso.fr/

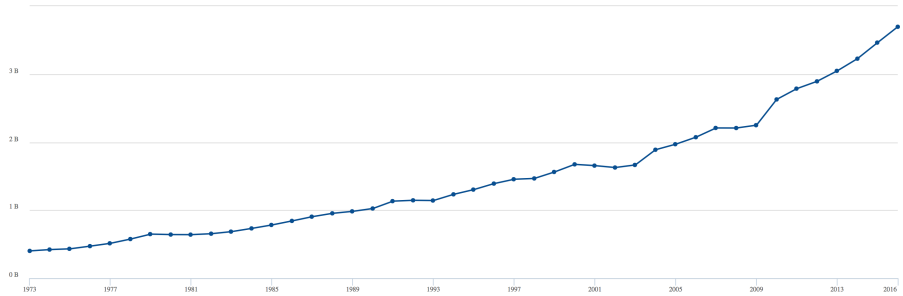


Figure 1.1: Number of passengers carried by air transport per year. Source: *The World Bank*[2]

severe delays causing financial damages to airlines and reduce the customers satisfaction. It is the goal of predictive maintenance to prevent such situations by making predictions about potential failures that could affect the normal aircraft operation.

1.1.1 *Predictive Maintenance*

Scheduled maintenance is the traditional way of maintaining equipment and consists of a set of maintenance procedures defined at aircraft design time and that must be planned in advance and performed periodically. However, whenever an unexpected failure occurs between two scheduled maintenance slots, the equipment becomes unavailable until the necessary maintenance actions are performed. These unexpected failures can be a costly burden to the equipment owners because during the downtime they may not be able to provide the expected services to their clients. On the other hand, the goal of *predictive maintenance* is to prevent such unexpected equipment failures by continuously observing

[2] https://data.worldbank.org/indicator/IS.AIR.PSGR

the status of the equipment and raising alerts well in advance. The time between the alert and the failure can be used by the experts to plan and perform the maintenance and to avoid any operational disturbance. In the case of aviation, where the status of the equipment can be reflected by data related to the operation of the aircraft, the challenge is to analyze such data, to translate high level predictive maintenance objectives into data science tasks (or data mining routines) and to achieve the best possible results.

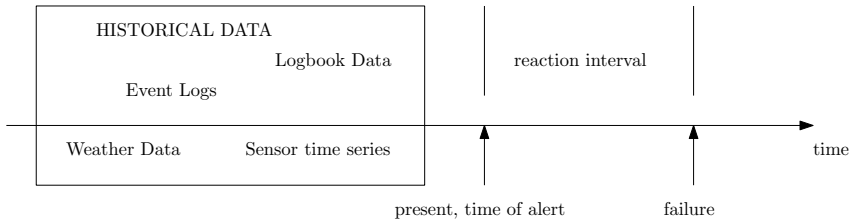


Figure 1.2: Data driven predictive maintenance

Given the increasing collections of available data generated by monitoring processes (e.g. sensors and event logs), the goal is to predict upcoming critical events or system failures. An important issue is that the interval between the prediction and the failure (Figure 1.2) should be sufficient for the planning and execution of the corresponding maintenance actions.

Predictive maintenance and machine learning have developed a very strong connection. However, it is not always easy or straightforward to perform effective predictive maintenance for several reasons:

- **Lack of predictive power in the data.** It is possible that the available data does not contain relevant or adequate information about the problem/task under consideration.

- **Lack of annotated data.** Although large datasets could be available for analysis, when dealing with supervised tasks one has to obtain annotated (or labeled) data. Their lack can be a major obstacle and the acquisition can be very expensive, since one has to consume several man-hours, in order to manually assign the ground truth labels.

- **Huge amounts of data.** In many real world scenarios one has to deal with many gigabytes or even petabytes of data in order to be able to extract useful knowledge about the domain.

These problems might be easier or harder to deal with depending on the application and the domain under consideration. Finally, recent advances in big-data and related technologies have made it possible to analyze very large datasets in distributed environments.

1.1.2 *Time Series Data*

The goal of this thesis is to employ modern machine learning techniques to analyze time series data from the aviation industry. Time series is one of the most common data types in many real world applications (e.g., finance, meteorology, medicine, sensor networks) that constantly draws the attention of the research community, and has been very well studied over the past decades. During this thesis, well-known problems such as prediction and modeling were addressed in the context of a set of predefined industrial requirements. Additionally, the research and development is being performed by employing modern distributed

computing frameworks, suitable for the big-data demands of the project due to the huge amount of information generated by the various aircraft components.

1.2 DATA RELATED TO AIRCRAFT OPERATION

An aircraft generates large amounts of data during its operation, most of which belong to one of two main categories:

- **Sensor time series**. These correspond to data coming from sensors on-board, such as pressure, temperature, speed etc. Hundreds of sensors with varying sampling frequencies emit measurements that can reach the size of a few gigabytes for every single flight.

- **Events Logs**. Events are generated by systems that monitor the condition of components on-board. These systems are designed to detect conditions that should be reported to the crew and the generated events correspond to failures.

Another very important category of data related to the operation of the aircraft is the *logbook*. The logbook contains information about the maintenance activities that were performed on the aircraft along with issues reported by the crew, replacements and repairs of components etc. Although these data are not generated by the aircraft and their size is limited, the information that they contain is invaluable for performing effective predictive maintenance.

1.2.1 *Tools and Libraries*

In order to obtain the results presented in this thesis we performed our experiments using available tools and libraries. The ones that were mostly used in the context of the thesis are the following:

- Apache Spark Zaharia et al. (2016). A big data processing engine which was extensively used to analyze the data involved in this thesis.

- MLlib Meng et al. (2016). The machine learning library of Apache Spark which includes efficient implementations of machine learning algorithms for the distributed environment.

- Scikit-Learn Pedregosa et al. (2011). A python library for Machine Learning which includes a plethora of both state of the art and recent algorithms.

- CVXOPT Dahl and Vandenberghe (2006). A convex optimization library in python.

- Matplotlib Hunter (2007). A python plotting library that was used to generate all the figures that present analysis results.

- IPE Schwarzkopf (1995). A drawing editor for creating vector graphics with which all schematic diagrams were created.

1.3 OVERVIEW OF CONTRIBUTIONS

The contributions made in the context of the thesis can be briefly summarized as follows:

- We proposed a method from predicting aircraft failures given the history of past ones. To our knowledge, this was the first approach on such data, we achieved 25% precision with very few false positives, and we showed the impact in cost reduction of our results. Furthermore, although our target data were aircraft failures, our method can be used for event logs coming from any other domain.
- We proposed a method for cleaning logbook data using recent advances in text mining. We demonstrated the problem of dealing with such data and snapshots of our results. Our work is the first step towards the embedding of such data in failure prediction.
- We addressed the problem of assessing the condition of sensors on board by proposing a method for quantifying the severity of the fault and consequently, the risk of its failure. Our approach was a novel modification of a recently proposed method, followed by the utilization of recent advances in neural network architectures.

For more detailed discussion of our contributions, please refer to the concluding sections in Chapters 4, 5 and 6.

1.4 OUTLINE OF THE THESIS

The rest of the thesis is organized as follows:

- **Chapter 2** presents the minimum background in order to facilitate the reader in understanding the next, technical chapters.
- **Chapter 3** demonstrates our initial steps to investigate the failure dataset.
- **Chapter 4** presents our method on failure prediction
- **Chapter 5** is about processing technical logs that contain information about maintenance activities using text mining approaches.
- **Chapter 6** presents our approach on sensor degradation monitoring using time series approaches.
- **Chapter 7** concludes this thesis with a short discussion about the achievements and the future steps.

2

BACKGROUND

In this chapter we present the minimum background required to follow the rest of the thesis. Due to the fact that we addressed several problems on different types of data, we had to investigate a wide range of methods and algorithms. However, in this section we describe only the key components used in each problem. For comprehensive material in the area, one should refer to some remarkable Machine Learning books such as Bishop (2006), Barber (2012), Goodfellow et al. (2016), Koller and Friedman (2009) and Theodoridis (2015). TP

2.1 LEARNING FROM DATA

Machine learning is about extracting knowledge from data in order to create models that can perform tasks effectively. A typical Machine Learning application consists of the following parts:

- A task and a metric for evaluation. The task comes inherently given a real problem and the metric quantifies effectiveness of a solution.
- A model family that we believe is capable of solving the problem in hand. The selection of the model type depends on several factors, such as the amount of available training data (i.e. size of the given dataset), the complexity of

the task and knowledge about its performance on similar problems Caruana and Niculescu-Mizil (2006).

- A dataset on which the *best* model will be trained in order to solve the task, aiming to the best performance with respect to the evaluation metric.

- A loss function that quantifies the goodness of fit. In contrast to the evaluation metric, the loss function is a differentiable and usually model-specific Bottou (2010).

- An optimization algorithm to train the model. The models consist of parameters (usually referred as θ), whose values reflect the performance on the loss function. Thus, an optimization strategy is required in order to select the parameter set that minimizes the loss function.

Note the difference between the first and the fourth point. In several cases these functions can be the same, for example for house price prediction, the *Mean Squared Error* (MSE) can be both the task objective and the loss function.

In Machine learning we are creating *models* that can *best* understand a set of available data (aka a dataset) in order to perform a specific task. A dataset $\mathcal{X}$ consists of a usually finite number of *samples* $x_i, i = 1, \dots n$, such as images, documents or users, depending on the application.

2.1.1 *Supervised Learning and Evaluation Metrics*

Supervised learning occurs when the model is trained using input-output pairs, like in classification and regression. In this

		Predicted Class: Positive	Predicted Class: Negative	Total
Actual Class	Positive		FN	$TP + FN$
	Negative	FP	TN	$FP + TN$
	Total	$TP + FP$	$FN + TN$	

Table 2.1: Confusion Matrix

scenario, the dataset consists of two subsets, the *feature set* $\mathcal{X}$ and the *label set* $\mathcal{Y}$, both having the same cardinality ($|\mathcal{X}| = |\mathcal{Y}|$), which is equal to the number of available samples.

On the other hand, unsupervised learning occurs when the model is trained using only the feature set $\mathcal{X}$. The most popular unsupervised tasks is clustering, since the cluster labels are not known in advance.

In the case of supervised learning, since the labels are known in advance, we can directly assess the performance of our approach using standard metrics. When the labels are continuous (Regression), $y_i \in \mathbb{R}^d, y_i \in \mathcal{Y}$, the popular performance metrics are the MSE, the Mean Absolute Error (MAE) and their weighted alternatives, where each sample comes with a different weight.

When the labels correspond to distinct categories (Classification) several metrics have been proposed and are mostly related to the distribution of the classes: Accuracy, Precision/Recall, F1-Score are among the most popular ones. In order to calculate these metrics, one has to compute the *confusion matrix* (Table 2.1).

In the context of predictive maintenance one of the most important problems is to predict equipment failures. If we consider this problem as a binary classification, our errors correspond to either false alarms (False Positives) or missed failures (False Negatives). Depending on the application, we have to decide the

importance of the *types of errors* that we encounter and we have to deal with the balance of False Positives and False Negatives according to industrial parameters such as the expected cost. For example, in cases where alarms trigger costly maintenance processes, avoiding many False Positives is very important.

2.2 PROBABILITY

2.2.1 *Survival Analysis*

Survival analysis Miller (2011) is widely used to extract temporal information about *events*. The most common application appears in biology and medicine, where common events of interest are population deaths, treatments etc. Furthermore, in the context of survival analysis, questions of major interest are whether there exist variables (covariates and interactions among them) that affect the time to an event and whether two or more populations exhibit significant difference on the time of occurrence of the event of interest. In predictive maintenance, events usually correspond to equipment failure and in engineering applications survival analysis is called *reliability theory*.

2.2.2 *Survival data and Censoring*

A typical dataset in survival analysis consist of subjects and their corresponding times to the event of interest (i.e. the survival time) and occasionally, other variables associated with the subjects if they are available. Figure (2.1), shows a small example of depicting a survival dataset, consisting of 6 subjects ($AC_1 - AC_6$)

and their times to the event of interest, e, $(t_1 - t_6)$. All the subject's observations are aligned in time and they begin at $t = 0$.

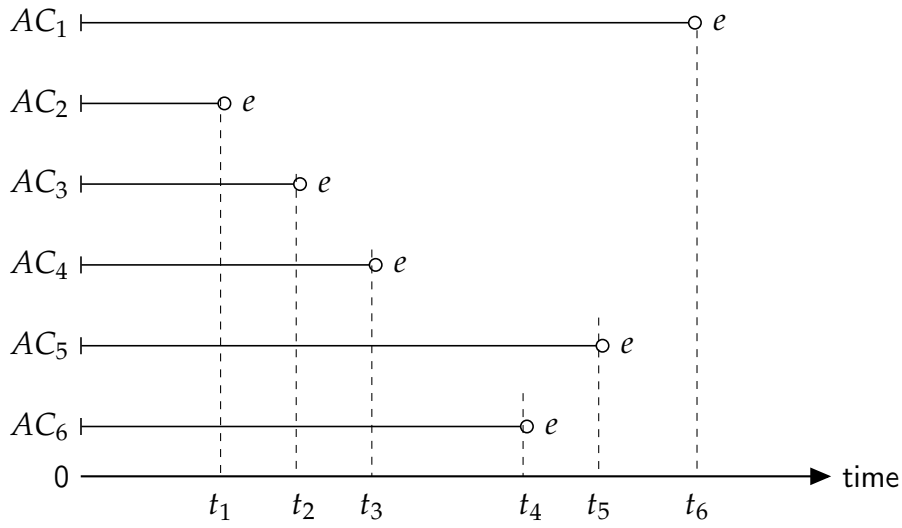


Figure 2.1: Survival data

A very important concept in survival analysis is censoring. Censoring occurs when we are not sure about the exact time of the event's occurrence due to incomplete data. For example, if the event of interest does not occur for a subject in the population during the observation period, then we know that the survival time for this subject is at least as long as the observation period. This is probably the most common case of censoring. The three types of censoring are:

- **right censoring** When the time to the event of interest is **more** than some known value. For example, when the observation period ended without observing the event.
- **left censoring** When the time to the event of interest is **less** than some known value.

- **interval censoring** When the time to the event of interest is **within** a known interval.

In survival analysis applications it is very important to handle the censored data and therefore, all types of censoring have been extensively studied Klein and Moeschberger (2005). However, in the case of equipment failures we deal mostly with right censoring.

2.2.2.1 *Survival and Hazard Functions*

The key components of survival analysis *survival* and the *hazard* functions. The survival function (2.1), $S(t)$, is the probability of an event occurring after time t (or equivalently surviving at least until time t), i.e. the probability of a failure not occurring before time t,

$$S(t) = p(T > t). \tag{2.1}$$

The most important properties of the survival function are presented in Table 2.2. In brief, it is a non-increasing function with respect to time and at the beginning of the observations it always equals to 1, because of the assumption that no subject *dies* at $t = 0$.

The hazard function (2.2), $h(t)$, is the rate of the event occurrence (a.k.a. death rate or hazard rate) at time t,

$$h(t) = \lim_{\Delta t \to 0} \frac{p(t \leq T < t + dt | T \geq t)}{\Delta t}. \tag{2.2}$$

The most important properties of the hazard function are a) it is always positive ($h(t) \geq 0 \; \forall t \geq 0$), and b) $\int_0^\infty h(t)dt = \infty$.

Property	Interpretation
$S(0) = 1$	The probability of a failure at the start of the observation is zero.
$\lim_{t\to\infty} S(t) = 0$	The event will certainly occur
$S(t_a) \geq S(t_b) \iff t_a \leq t_b$	It is monotonically decreasing
$S(t) = 1 - F(t) = 1 - \int_t^\infty f(\tau)d\tau$	

Table 2.2: Properties of the survival function

Consequently, it is straightforward to derive the following relations (equations 2.3, 2.4) between the hazard and survival functions:

$$h(t) = \frac{dS(t)/dt}{-S(t)} \tag{2.3}$$

$$S(t) = exp\left(-\int_0^t h(\tau)d\tau\right) \tag{2.4}$$

2.2.3 *Gaussian Mixture Models and the EM algorithm*

A Gaussian Mixture Model (GMM) is used to model the pdf of random vector x as a linear combination of densities instead of a single one (eq. 2.5),

$$p(X) = \sum P_k p(x|k). \tag{2.5}$$

Figure 2.2 demonstrates an example of a mixture model and a single Gaussian fit on a sample population generated by the sum of two random variables.

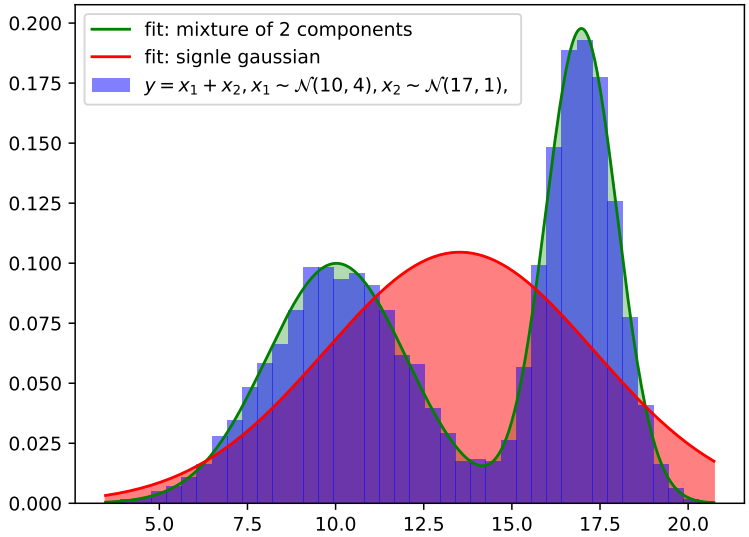


Figure 2.2: Gaussian Mixture Model example

The difficulty in estimating the GMM parameters using the maximum likelihood approach is due to the latent variable, k, which indexes the components from which sample is drawn. The solution comes from Expectation Maximization (EM) which is an algorithm for maximum likelihood estimation of parameters, in the presence of latent variables or missing data.

2.2.3.1 *Expectation Maximization*

The algorithm to estimate the parameters of a GMM is the EM Dempster et al. (1977). It is an iterative method to perform Maximum Likelihood estimates of parameters in the presence of latent variables. Its name refers to the two steps of the iterative process, the E-step which is the estimation of the latent/missing

variables and the M-step, which is the parameter update by maximizing the likelihood function.

The process can be summarized in equation 2.6,

$$\theta_{n+1} = \underbrace{\underset{\theta}{\operatorname{argmax}} \left\{ \underbrace{\mathbb{E}_{Z|X,\theta_n} \{ lnP(X, z|\theta) \}}_{E-step} \right\}}_{M-step}. \tag{2.6}$$

For the comprehensive description of of the EM steps in GMM estimation one can refer to Dempster et al. (1977).

2.3 REGRESSION

Regression is the task of learning (fitting) a model whose dependent variables (or output variables) are continuous, i.e. $\mathcal{M}(\mathcal{X}|\Theta) = \mathcal{Y}$ where $\mathcal{Y} = \{y_1, \dots y_d\}$ and $y_i \in \mathbb{R} \forall i$. It is one of the most studied problems with applications in almost every domain and thus, many regression models have been proposed, such as Linear Regression with regularization, Support Vector Regression, Spline and Polynomial Regression etc.

In predictive maintenance and consequently in this thesis, tasks like estimating the risk of a system failure, quantifying the condition of a components and predicting missing sensor values can be naturally modeled as regression problems and therefore extensive research and experimentation was performed to select the proper ones for each case.

2.3.1 *Random Forests*

Random forests Liaw and Wiener (2002) are one of the most popular models in Machine Learning and they have been widely used both for classification and regression. They belong to a broad class of algorithms called *ensemble learning* Zhou (2012); Zhang and Ma (2012), which is the combination of multiple, simple models in order to solve a hard, complex tasks. In this sense, a random forest is composed by a set of decision trees Breiman et al. (1984), each one estimating the output variable on a set of available features.

The most important advantages of random forests can be summarized as follows:

- **Scalability**. Random forests can be naturally trained in distributed environments Chen et al. (2017).

- **Non-linearity**. They can capture complex relationships among the features by traversing the tree and splitting the data based on feature values.

- **Interpretability**. Being composed by many simple decision trees, several techniques can be used to assess each feature's importance based on the effect of the splits in which it participates Archer and Kimes (2008).

- **Simple and intuitive hyperparameters**. The only[1] hyperparameters of a random forest is the number of trees and the maximum depth.

[1] Random forests can be tuned by other means, such as selecting different impurity functions, however, this is not considered as a parameter

2.3.2 *Model Evaluation*

Given a training dataset and a model class (or a class of models) one has to decide whether the trained model will perform well in production, i.e. will perform as well with new samples. Therefore, there are two types of errors: a) the *training error*, which is the error on the samples used to train the model and b) the *generalization error*, which is the error on new samples that were not used during training. The generalization error is our *estimation* of our model's performance once deployed.

Since the generalization error cannot be computed before the actual deployment of the model, we simulate it by dividing the available data into two sets, the training and the validation datasets. We fit the model on the training dataset and we evaluate its performance on the validation one, thus obtaining the *validation error*, which is our simulation for the generalization error.

The most popular approach is *k-fold cross-validation* (or rotation estimation) Kohavi (1995), in which we divide the training sets in k non overlapping segments. Then the model is trained using the $k-1$ segments and the validation error is estimated on the other one.

In general, the $k-1$ training subsets can be selected by random sampling the indices of the data points in the training set. However, in the special case of sequential data (such as time series), one has to comply with a simple and intuitive rule: the sequence of the data must be respected and thus, *past samples* should only be used for training and *future samples* for validation 2.3.

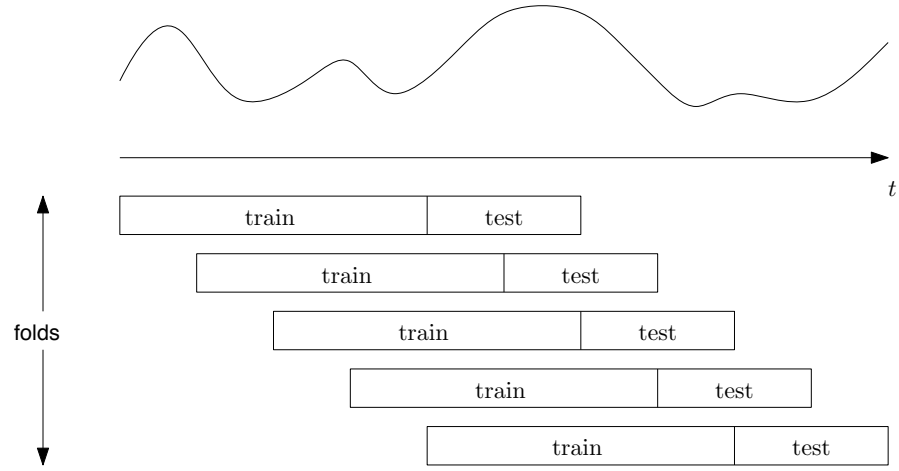


Figure 2.3: Cross validation in time series data

2.3.3 *Hyperparameter Selection*

Machine Learning models contain parameters that cannot be optimized during the training phase, such as the regularization term λ in LASSO Tibshirani (1996), C in Support Vector Machines Duan et al. (2003), the number of trees and the maximum depth in random forests. Parameters that cannot be optimized need to be selected by brute force approaches, such as *grid-search*. These parameters are known as hyperparameters.

In gird-search, one defines the ranges of the hyperparameters in which the search for the best model will be performed. At every step of the search, a combination of the values within the predefined ranges is assigned to the hyperparameters and a $k - fold$ cross validation is performed.

The values of the hyperparameters are closely related to overfitting since they control the complexity of the model. A *complex* model tends to overfit since it tends to have the capacity to mimic

the training set. However, if this complexity is not controlled, the model will perform poorly on the test set (see Figure 2.5).

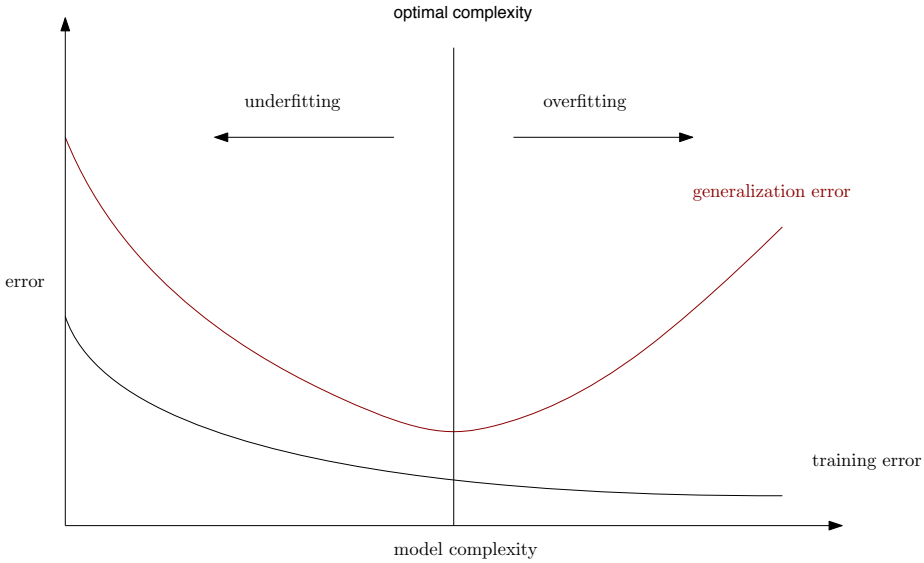
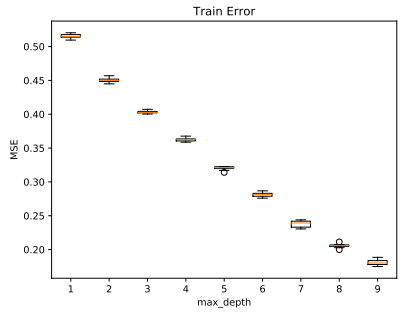


Figure 2.4: Errors and Model Complexity

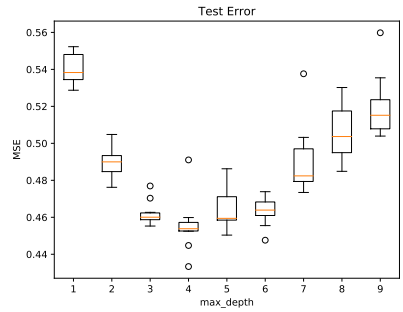
Consider for example the *max_depth* parameter in a random forest, which controls the maximum depth which each tree can reach by the iterative splits and thus, the larger the depth the more *complex* relationships among the features will be used for the split and this may lead to overfitting. In Figure 2.5 we present the impact of the *max_depth* parameter on the train and test errors using the *Wine Quality*[2] dataset.

This dataset Cortez et al. (2009) consists of 4898 instances (samples) with 12 attributes that represent characteristics of the wines. For our purpose of demonstrating the problem of overfitting, we formulated a regression problem aiming to predict each wine's score (it is one of the 12 attributes) using the other 11 features. The method was evaluated using the Mean Square Error. The

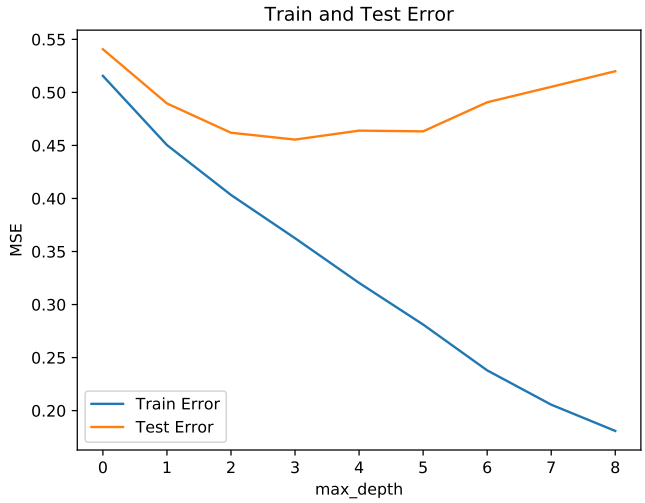
[2] https://archive.ics.uci.edu/ml/datasets/wine+quality



(a) train error



(b) test error



(c) means of train and test error

Figure 2.5: Impact of maximum depth on train and test error

box-plots in 2.5 were generated using 10 runs, each one performing a 5-fold cross validation. As the figures show, the train error drops by increasing the max-depth whereas the test error has a minimum value when the depth is 4. The figures show that for greater values of the depth the model starts overfitting the data.

2.4 LEARNING AS OPTIMIZATION

Learning a model $\mathcal{M}$ from the data requires a metric or a measure that quantifies the quality of the fit. This measure is usually called the *loss function* and quantifies errors made during the fitting process (i.e. the samples for which the model did not fit well) and is task dependent. Therefore, model training is an optimization process that aims to minimize the loss function by finding the model's parameters that achieve this minimum loss. A plethora of methods exist to perform this optimization, some of them being model-specific and some more general. In the next section we briefly present the concept of Gradient Descent (GD), which is the most widely used optimization approach in Machine Learning.

2.4.1 *The Gradient Descent*

The GD method is an iterative algorithm for optimization Ruder (2016) and it is widely used in learning parameters in numerous applications. In each iteration, the parameters are updated by following the opposite direction to the gradient of the loss function with respect to the current parameter values (Eq 2.7).

$$\theta_{i+1} = \theta_i - \eta \nabla_\theta \mathcal{L}(\theta_i) \tag{2.7}$$

The problem with this approach is that at each iteration, the gradient has to be computed on the whole training set and thus, in cases where it is very large, each step requires a lot of time. Alternatively, Stochastic Gradient Descent (SGD) Bottou (2010) is an extension of GD, which performs an update of the parameters

using a single sample. The term *stochastic* comes from the fact that the process depends on the samples randomly selected at each iteration.

2.4.2 *Convex Quadratic Programming*

When the loss function is convex, a single set of values for the model's parameters exists, which evaluates the function to its global minimum. In this section, we will briefly introduce a specific form of a convex optimization problem, which is called Quadratic Programming (QP). An optimization gram is Quadratic, if the objective function is convex quadratic, and the constraint functions are affine Boyd and Vandenberghe (2004). The quadratic program has the form:

$$\begin{aligned} \min_{x} \quad & x^T Q x + w^T x + c \\ \text{s.t.} \quad & Gx \leqslant h \\ & Ax = b \end{aligned} \tag{2.8}$$

where $Q \in \mathbf{S}_+^n$, $G \in \mathbb{R}^{m \times n}$ and $A \in \mathbb{R}^{p \times n}$. $\mathbf{S}_+^n$ is the space of Positive Semidefinite matrices of shape $n \times n$. A matrix $M \in \mathbb{R}^{n \times n}$ is positive semidefinite iff $x^T M x \geqslant 0, \forall x \in \mathbb{R}^{n \times 1}$.

The importance of convex optimization in Machine Learning lies in the fact that formulating the parameter learning as such, one can obtain a global minimum. Consider for example the problem of linear regression, where we aim to minimize the mean squared error $\|X\theta - b\|^2$. This problem can be formulated as:

$$\min_{w} \quad \theta^T \underbrace{X^T X}_{Q} \theta - \underbrace{2b^T X}_{-w} \theta + \underbrace{bb^T}_{c} \tag{2.9}$$

which is convex.

Finally, Quadratic Programming will be the key concept for developing the methodology in Chapter 6.

3

SURVIVAL ANALYSIS FOR FAILURE-LOG EXPLORATION

In this chapter we present our approach to explore Post Flight Report (PFR) datasets (which consists of aircraft failures) using state of the art survival analysis techniques. Failures are the most important entities in predictive maintenance because it's their occurrence that reflects problems in equipment on-board and triggers maintenance actions. Therefore, before building a machine learning approach for failure prediction, we extract basic information about their occurrence for two main reasons: a) we will use this information later to properly setup our learning process with no prior knowledge from experts and b) the outcomes are useful to investigate differences among aircraft types or equipment types with respect to failure rates, in order to discover relevant vulnerabilities.

3.1 INTRODUCTION

As mentioned in the previous section, survival analysis is widely used to investigate the time-to-event in several domains, mostly in medicine and biology and has been widely used for many years now Kelly and Lim (2000), Binet et al. (1981), Adams et al. (1991) Gross and Clark (1975). In engineering, suvival anaysis (which is named reliability theory Ma and Krings (2008)) methods

are used to extract knowledge about the remaining lifetime of components and the time to failures, Liao et al. (2006), Zhou et al. (2014).

All these approaches study the corresponding problem in hand using the main concepts presented in Section 2.2.1. Although these methods provide useful information about the termporal nature of the events, they are not capable of performing accurate predictions and therefore, machine learning approaches have been employed Wang et al. (2017). However, simple survival analysis methods consist of very effective tools for exploring failure log data.

3.1.1 *Random Variables in event logs*

An event log contains the sequence of failures that occurred during the operation of the aircraft. The alphabet of failures is finite and each failure is recorded using its unique identifier (id). Let $\mathcal{E} = \{e_1, \ldots, e_k\}$ be the alphabet of failures. Thus, each entry in the logbook is a tuple $< t_i, x_i >$, where t_i is the timestamp of the failure and $x_i \in \mathcal{E}$ is the failure that occurred. For each aircraft, the logbook forms a sequence of such tuples.

The alphabet of failures contains both important[1] and not important ones and thus, we are interested to analyzing and predicting the former ones. Let $e_{\mathcal{T}}$ be a critical failure that we aim to predict. In the history of event logs this failure may appear more than once and thus, the survival time corresponds to the time interval between two consecutive occurrences of $e_{\mathcal{T}}$ (i.e. the

[1] In our work, the importance of a failure is estimated by the amount of time the aircraft has to stay on the ground until the corresponding maintenance action is performed.

time interval starting from the maintenance action until its next failure). We define k random variables that correspond to these intervals $T^{e_i}, i \in [1 \dots k]$. Furthermore, we introduce $k \times k - 1$ random variables $T^{e_i}_{e_j}$ that correspond to the time between failure e_j and the next occurrence of e_i. We will use this information later in this chapter to introduce the concept of *predictors*. Figure 3.1 depicts these two categories of random variables with respect to the target failure $e_{\mathcal{T}}$.

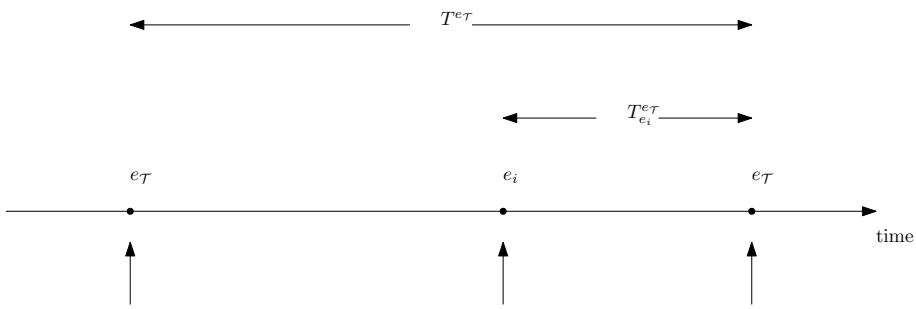


Figure 3.1: Survival time: Time interval between consecutive occurrence of a target failure $e_{\mathcal{T}}$ that occurs at times t_1 and t_3.

To be aligned with the concepts in survival analysis, we assume that after each occurrence of a target failure a maintenance action treats the failure in such way that it becomes independent of the previous one. Although this might not be always true, it is a reasonable assumption in most cases. Furthermore, throughout the rest of the thesis we consider the time as a discrete variable that is measured in *flights* for two main reasons. First, in order to avoid including possible time periods of non-utilization of an aircraft and second, the number of take-offs and landings are more important for many failure types than the amount of hours.

3.1.2 *Building a Dataset for Survival Analysis*

In a typical survival analysis experiment, one starts observing a population at $t = 0$, and monitors the time of occurrence of the events. In order to structure our failure data accordingly, we introduce a new subject of the population for interval between two consecutive occurrences. Moreover, we consider the interval from the beginning of the PFR log until the first occurrence of the target failure and the interval from the last occurrence of the target failure until the end of the PFR as right-censored observations. In Figure 3.2 we demonstrate this setup.

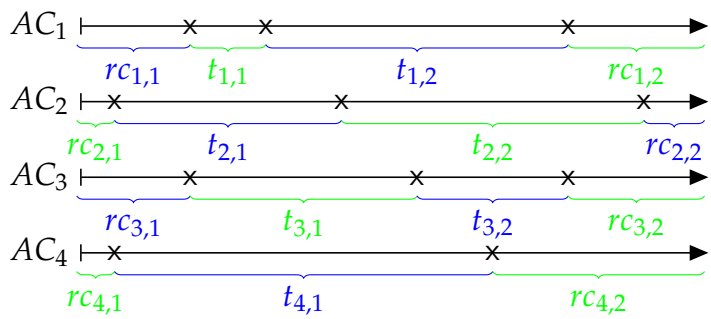


Figure 3.2: PFR survival data and censoring. 'rc' for right censoring and 'x' is the event of interest. Each interval $t_{i,j}$ and $rc_{i,j}$ is a subject in the population.

The time-aligned dataset created by the intervals depicted in Figure 3.2 is presented in Table 3.1. Each row is added for every timestep at which an event occurred (either censored or not) in a time-increasing order. The columns are:

- t is the duration from the beginning of the observation until the event
- $d(t)$ is the number of events at time t

- $q(t)$ is the number of right censoring at time t
- $n(t^-)$ is the number of subjects *at risk* just before time t, i.e. the number of subjects without the event just before time t.

intervals	t	$d(t)$	$q(t)$	$n(t^-)$
-	0	0	0	15
$rc_{2,1}, rc_{4,1}$	$rc_{2,1}$	0	2	15
$rc_{2,2}$	$rc_{2,2}$	0	1	13
$t_{1,1}$	$t_{1,1}$	1	0	12
$rc_{1,1}, rc_{3,1}$	$rc_{1,1}$	0	2	11
$rc_{1,2}, rc_{3,2}$	$rc_{1,2}$	0	2	9
$t_{3,2}$	$t_{3,2}$	1	0	7
$rc_{4,2}$	$rc_{4,2}$	0	1	6
$t_{3,1}, t_{2,1}$	$t_{3,1}$	2	0	5
$t_{2,2}, t_{1,2}$	$t_{2,2}$	2	0	3
$t_{4,1}$	$t_{4,1}$	1	0	1

Table 3.1: A table-structured survival dataset that records the occurrence of events of interest and censoring

This table-structured representation facilitates the Kaplan-Meier (KM) estimation of the survival function presented in the next section and the partial likelihood estimation for the Cox Proportional Hazard approach presented in Section 3.2.2.

3.2 TIME INTERVAL BETWEEN FAILURES

In this section we study the survival functions of target failures using the non-parametric Kaplan-Meier Kaplan and Meier (1958) approach and furthermore, we investigate the impact of some available covariates using the Cox Proportional Hazard method Cox and Oakes (1984a).

3.2.1 *Kaplan - Meier method*

The Kaplan-Meier is a non-parametric method of estimating the survival function. The KM estimation of the survival function is

$$\hat{S}(t) = \hat{S}(t^-)\hat{p}(T > t|T \geq t) \tag{3.1}$$

where

$$\hat{p}(T > t|T \geq t) = \begin{cases} 1 & \text{if } d(t) = 0 \\ \frac{n(t^-)-d(t)}{n(t^-)} & \text{otherwise} \end{cases} \tag{3.2}$$

and consequently, from (3.1) and (3.2) one can derive the final form of the survival function:

$$\hat{S}(t) = \prod_{t_{(i)} \leq t} \frac{n(t^-) - d(t)}{n(t^-)}. \tag{3.3}$$

In Figure 3.3 we show an example of survival estimated for four critical events. The dataset comes from PFR reports from 18 aircraft.

In Figure 3.4 we present the survival functions of the failures presented in Figure 3.3. The dashed lines correspond to the confidence intervals for 95% confidence.

Figures 3.3 and 3.4 show that events e_{2183} and e_{1987} have similar occurrence intervals and also, the same stands for events e_{2095} and e_{2156}. However, event e_{2136} has different behavior.

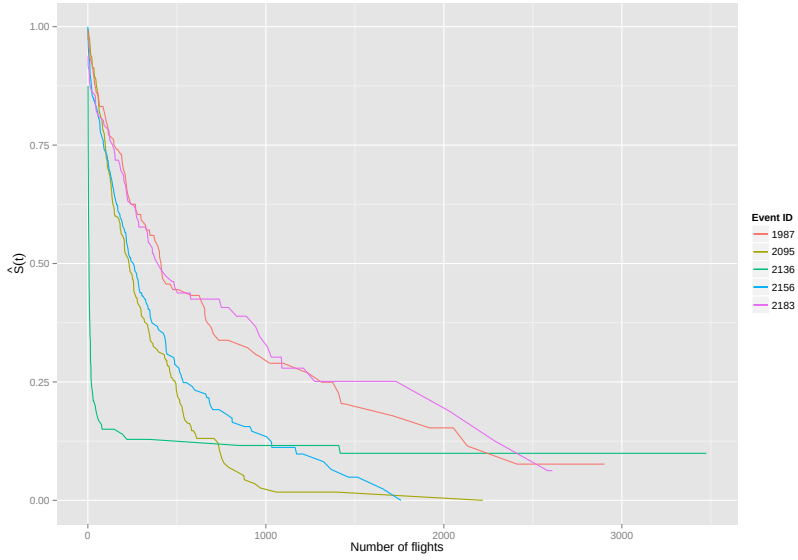


Figure 3.3: Kaplan-Meier estimates for four critical events

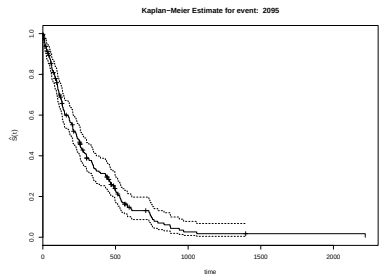
3.2.2 *Cox Proportional Hazards*

The CPH method enables the utilization of continuous and categorical covariates in the model of the hazard function and thus, it can be used to study the effect of each one on the hazard rate.

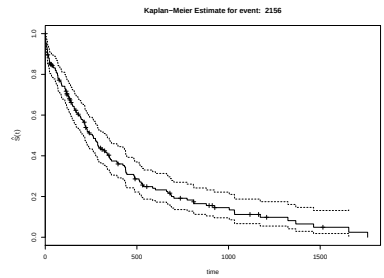
In CPH the hazard function is modeled as the product of a baseline hazard ($h_0(t,a)$) which depends on time, and a term ($exp(\beta^T x)$) that depends on the covariate (3.4).

$$h(t,x) = h_0(t,\alpha)\exp(\beta^T x) \tag{3.4}$$

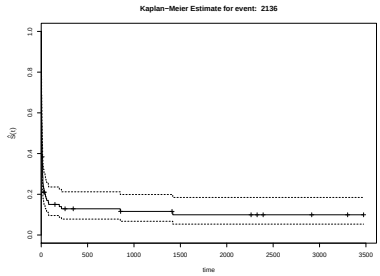
The advantage of this model is that if we are interested in investigating how the covariates affect the hazard function, then we do not need to compute the baseline hazard, since the ratio of



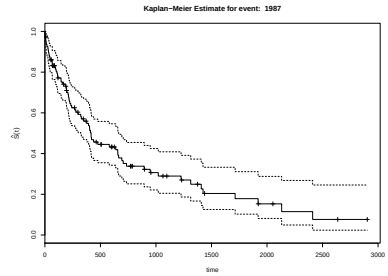
(a) Event e_{2095}



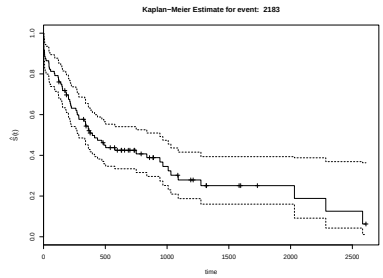
(b) Event e_{2156}



(c) Event e_{2136}



(d) Event e_{1987}



(e) Event e_{2183}

Figure 3.4: Estimated survival functions and 95% Confidence Intervals for five critical failures

two hazards at a specific time depends on the difference of the covariates and the corresponding coefficient (3.5).

$$\frac{h(t,x_1)}{h(t,x_2)} = \frac{h_0(t,\alpha)\exp(\beta x_1)}{h_0(t,\alpha)\exp(\beta x_2)} = \exp\{\beta(x_1 - x_2)\} \tag{3.5}$$

Finally, learning the coefficients is performed by maximizing the *partial log-likelihood* Lee and Wang (2013)

$$\log L(\beta) = \sum_{i=1}^{k} \left\{ \beta x_i - \log \left[\sum_{l \in R(t_i)} e^{bx_l} \right] \right\}, \tag{3.6}$$

where $R(t_i)$ are the subjects of the population that are *at risk* and x_i is the covariate vector of subject i that *died* at timestep i.

3.2.2.1 *Modelling with Cox Proportional Hazards*

In our 18-aircraft dataset we have only one categorical covariate, the aircraft label. We use this just to demonstrate how the CPH can be used in the future when we will add convenient covariates.

For the case of one single categorical covariate with k possible values, the model is was built using $k-1$ binary variables, for all but one possible values of the covariate.

In Table 3.2 we present the coefficients of the model extracted from the failure 'e_{2095}'. The column *exp(coef)* shows the exponential of the coefficient $\exp(\beta^T x)$ and the p column shows the p-value of the statistical test having null hypothesis that this coefficient is zero.

A value x of a coefficient (*exp(coef)*) means that the baseline hazard is multiplied by x, i.e. x times the hazard rate of the first category (the one left out for the $k-1$ binary variables). For example, the value 4.419 means that the hazard rate of AC17 this is 4.419-times the hazard rate of AC01.

	coef	exp(coef)	se(coef)	z	p
xAC02	1.2312	3.425	0.486	2.5330	0.0110
xAC03	-0.2349	0.791	0.611	-0.3842	0.7000
xAC04	0.8254	2.283	0.501	1.6473	0.0990
xAC05	0.5012	1.651	0.523	0.9580	0.3400
xAC06	1.0511	2.861	0.494	2.1299	0.0330
xAC07	0.2148	1.240	0.557	0.3852	0.7000
xAC08	-0.1975	0.821	0.655	-0.3015	0.7600
xAC09	-0.0471	0.954	0.609	-0.0774	0.9400
xAC10	0.5901	1.804	0.542	1.0888	0.2800
xAC11	0.8827	2.417	0.530	1.6659	0.0960
xAC12	1.0854	2.961	0.511	2.1231	0.0340
xAC13	0.6474	1.911	0.530	1.2210	0.2200
xAC14	1.2373	3.446	0.503	2.4607	0.0140
xAC15	0.9381	2.555	0.522	1.7985	0.0720
xAC16	0.9083	2.480	0.561	1.6184	0.1100
xAC17	1.4859	4.419	0.485	3.0629	0.0022
xAC18	1.1376	3.119	0.515	2.2093	0.0270

Table 3.2: Modelling with CPH for event e_{2095}

3.3 STUDYING INTER-EVENT TEMPORAL DIFFERENCES

In Section 3.1.1 we introduced the random variables $T_{e_i}^{e_j}$ that are the number of flights between an event e_j and the next occurrence of e_i. Here, given a target failure $e_{\mathcal{T}}$ we aim to investigate the random variables $T_{e_j}^{e_{\mathcal{T}}}$ in order to identify possible *predictors*, aiming to answer the following question: *How many flights before the occurrence of $e_{\mathcal{T}}$ are we able to make a prediction*? This question is very important since most machine learning methods for failure prediction form a binary classification problem Sipos et al. (2014a), with the positive class being generated by an interval before the failure. The determination of the length of this interval is usually

defined by experts, however, when this knowledge is hard to acquire an automated approach would be very essential.

In Figure 3.5 each point corresponds to one random variable $T_{e_i}^{e_\mathcal{T}}, i = [1, \ldots, |\mathcal{E}|]$ and its coordinates are $< E[T_{e_i}^{e_\mathcal{T}}], std\{T_{e_i}^{e_\mathcal{T}}\} >$. Intuitively, good candidate predictors appear in the bottom left corner of the graph, since small standard deviation of $T_{e_i}^{e_\mathcal{T}}$ means that once e_i occurs, one can make more confident prediction about $e_\mathcal{T}$.

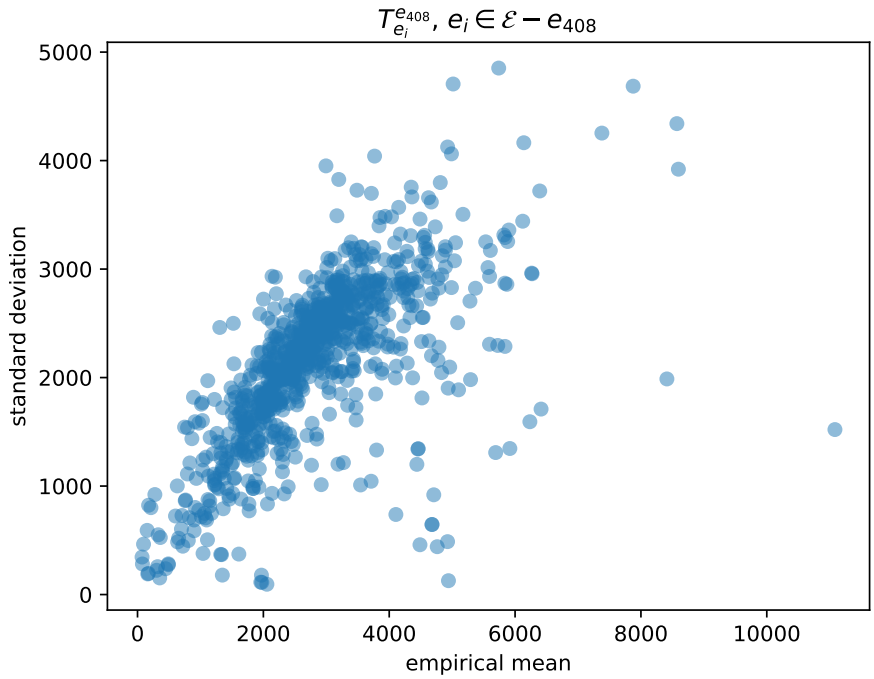


Figure 3.5: Empirical means and standard deviations of temporal difference between the critical failure and the other failures

In Figure 3.6 we present the random variables that correspond to four failures whose representation is on the bottom left corner of 3.5. The information coming from this figures gives us an

initial estimation on the prediction window for predicting $e_{\mathcal{T}}$ which in this specific case is 400 flights.

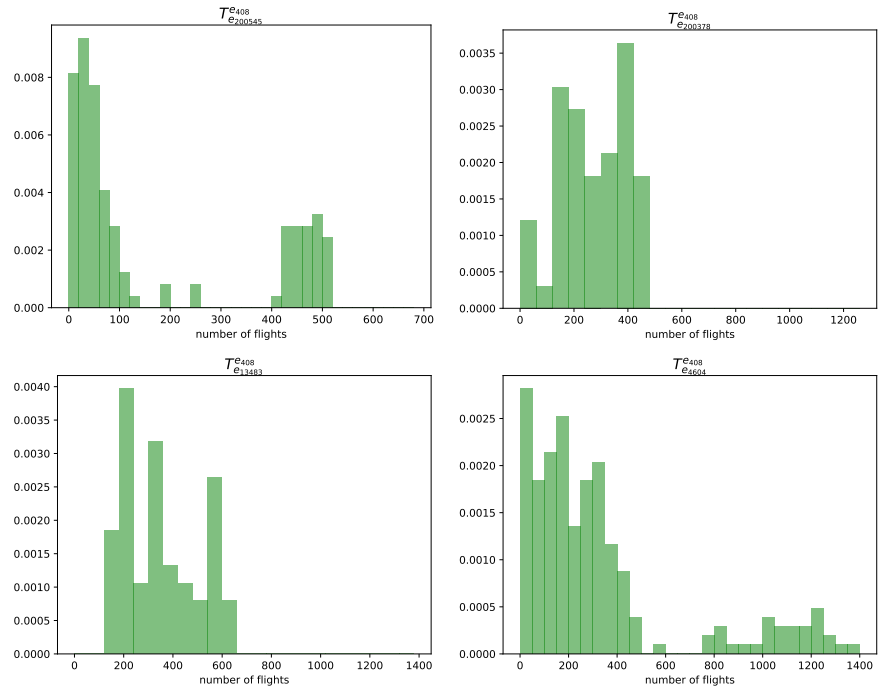


Figure 3.6: Four examples of random variables that correspond to good candidate predictors

In Figure 3.7 we present four random variables having the highest mean values and standard deviations. It is obvious that these failures do not contain useful information (when considered solely) about the critical failure.

Exploring the variables $T^{e_{\mathcal{T}}}_{e_i}$ for a critical failure gives essential exploratory information that can be used for prediction. However, in order to make accurate predictions given the PFR data one has to take into account predictors that are combinations of other failures and not single ones.

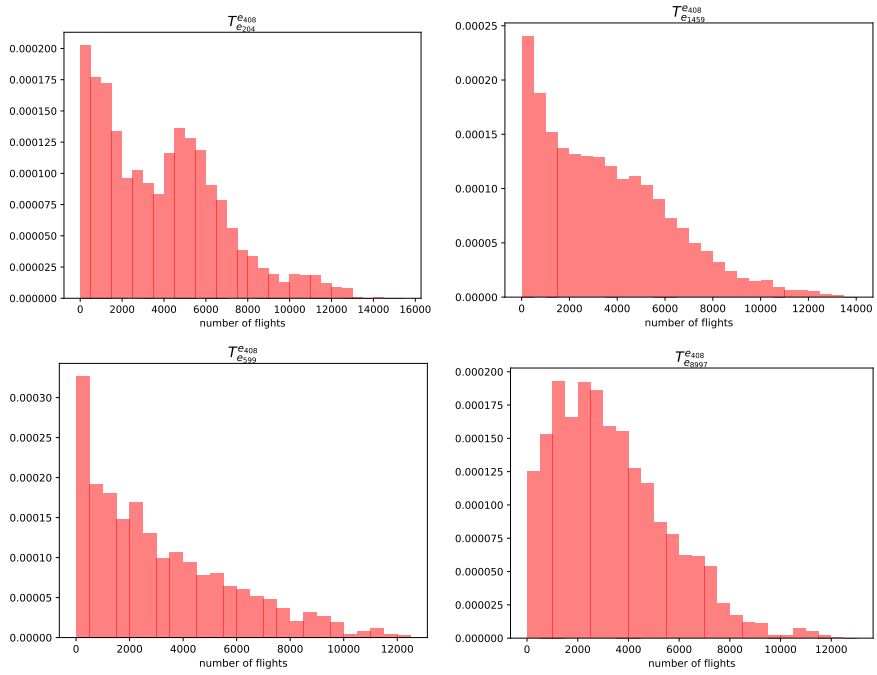


Figure 3.7: Four examples of random variables that do not provide useful information about the critical failure under consideration

3.4 SUMMARY

In this chapter we presented our exploratory analysis on the PFR dataset. We presented an approach using state of the art survival analysis methods in order to explore the time intervals between failures and the possible impact of available variables on the survival times. Moreover, we used the time intervals between failures in order to identify candidate predictors and also, to extract information that will be useful in failure prediction, which is presented in the next chapter.

4

FAILURE PREDICTION IN POST FLIGHT REPORTS

In this chapter we present an approach to tackle the problem of event prediction for the purpose of performing predictive maintenance in aviation. Given a collection of recorded events that correspond to equipment failures, our method predicts the next occurrence of one or more events of interest (target events or critical failures). Our objective is to develop an alerting system that would notify aviation engineers well in advance for upcoming aircraft failures, providing enough time to prepare the corresponding maintenance actions. We formulate a regression problem in order to approximate the risk of occurrence of a target event, given the past occurrences of other events. In order to achieve the best results we employed a multiple instance learning scheme (multiple instance regression) along with extensive data preprocessing. We applied our method on data coming from a fleet of aircraft and our predictions involve failures of components onboard, specifically components that are related to the landing gear. The event logs correspond to post flight reports retrieved from multiple aircraft during several years of operation. To the best of our knowledge, this is the first attempt on aircraft failure prediction using post flight report data and finally, our findings show high potential impact on the aviation industry.

The rest of this chapter is organised as follows: In section 4.2 we present the related work on event prediction from event

logs. Although there are plenty of methods for categorical time series prediction, we emphasise on methods that address the problem of predicting future occurrences of specific events (or failures in predictive maintenance). In section 4.3 we present the nature of event-log data along with our preprocessing steps and in section 4.4 we present our approach for prediction. In section 4.5 we discuss about our data, the experimental setup and the parameters and section 4.6 presents our results. Our conclusions and future work are in section 4.7.

4.1 INTRODUCTION

Monitoring systems generate events that reflect the condition of the equipment or the processes that are being observed. Hardware health, business processes and server activities are examples of cases where event logs are generated in order to provide the corresponding information. Given a set of event logs, questions of great importance are: how can we predict when a specific event will occur? How far before its occurrence are we able to predict it? In this chapter we address the problem of predicting the next occurrence of a specific event (target event), given the history of past events. This problem is extremely significant in predictive maintenance Kauschke et al. (2016); Susto and Beghi (2016), where the prediction of an upcoming failure of a system is very important.

In hardware monitoring applications, events are usually generated by software installed on the target equipment that regularly samples sensor measurements (e.g. temperature, pressure, etc). So far, the event logs are used mostly for debugging purposes

by the technical experts. It is challenging and efficient to be able to use the event logs for predicting failures instead of the raw, time-series data, because they are much smaller in size, easily accessible and can be transmitted in real time to a centralised infrastructure where the prediction can take place.

Motivated by the above, we present our approach for event prediction that could potentially be applied in a wide area of applications.

4.2 RELATED WORK

Prediction in event logs is well studied problem in a wide range of applications Salfner et al. (2010). Each application exhibits its own special characteristics that have great impact on the design of the corresponding algorithm. Hard drives Son et al. (2013); Murray et al. (2005), medical equipment Sipos et al. (2014b); Yuan et al. (2011) and web servers Zhang et al. (2016) are some application domains where failure prediction is important and prediction methods have been successfully applied.

The vast majority of failure prediction methods try to solve a classification problem; given the events that occur within a time window, the classifier decides if a failure will occur within a predefined interval. For example, in Weiss and Hirsh (1998) the authors use a genetic algorithm approach in order to discover prediction rules and perform the prediction, in Gu et al. (2008) the authors propose an online failure prediction scheme based on decision trees, in Yu et al. (2011) the authors employ Bayesian classification and more recently in Zhang et al. (2016) the authors use Long-Short Term Memory (LSTM) neural networks. Further-

more, in Laxman et al. (2008) the authors present a method that predicts if the next event (given a history of events) is a target event (an event of interest). Their approach is based on frequent episode mining and Hidden Markov Model (HMM).

In this setup, a multiple instance learning Fu et al. (2011) approach is more appropriate in order to tackle the problem of falsely assigning unrelated events to the positive class (i.e. events that randomly appeared before the failure). Towards this direction, the authors in Sipos et al. (2014b) create one positive sample by averaging the features extracted from the events for each window close to the failure and multiple negative ones for the windows far before the failure. They employ SVM Fan et al. (2008) to perform the prediction. Similarly, the authors in Murray et al. (2005) propose Multiple Instance Naive Bayes (miNB), that iteratively learns the labels (positive and negative) in each window, aiming to minimise the false positive rate.

Apart from the classification-based approaches, failure prediction has been tackled by statistical methods. In Yuan et al. (2011) the authors follow another approach based on survival analysis. More specifically, they use Cox Proportional Hazard (CPH) model Cox and Oakes (1984b) in order to approximate the survival function and they use the occurrence of other events/failures as covariates (features). In this setup, a prediction is performed by estimating an interval during which a failure is expected. Furthermore, in Watanabe et al. (2012) the authors propose a method that learns patterns from event log streams and the failure prediction is performed by calculating the probability of a failure occurring given the identified patterns.

Although a plethora of existing approaches is available in the literature, the particularities of our data prevented us from using any out of the shelf solution. Our target events are extremely rare, we have a large and sparse feature space and our goal is to develop a customisable method that could be easily adapted in an industrial environment. Therefore, our method differs from the ones presented above in several aspects. Initially, our primary goal is to create a model that outputs a risk function based on the present events and quantifies the risk of an upcoming failure. Thus, we formulate a time-to-failure regression problem. The value of our risk function is later employed in a simple thresholding scheme to make a final prediction. Furthermore, our method does not require the definition of the prediction interval parameter (i.e the amount time before the failure's occurrence that is considered as a positive class) for training our predictive model. On the contrary, we utilise a decision threshold that can be configured according to the desired precision and recall. Finally, several methods exist for predicting failures in aviation but they target specific aircraft components. The most relevant work appears in Huet et al. (2015), where the authors propose a graph-based method to predict the most probable failures in the upcoming flights.

4.3 EVENT LOG DATA & PREPROCESSING

Event logs consist of time-ordered events; Table 4.1 shows an example of an event log file. An event is composed by a timestamp that corresponds to the time of its occurrence and other attributes that contain information about the event. These attributes may in-

clude a system or subsystem identifier that generated the event, a task id (in information system logs), description about the activity (in server logs), failure description (in hardware monitoring) and a unique event identifier. In this work we target event logs and events come from a finite and known alphabet. A timestamp and an *event id* are the only attributes required by our methodology.

sys id	timestamp	event id	source	description
sys_1	2013-11-10 13:30	6226	3412	failure 1
sys_1	2013-11-10 14:33	3401	4902	failure 1
sys_2	2013-12-10 10:12	3401	5552	failure 2
sys_3	2013-12-10 11:40	408	4414	failure 2
⋮	⋮	⋮	⋮	⋮

Table 4.1: Event log example with renamed attributes

Let $\mathcal{E} = \{e_1, e_2, \ldots, e_k\}$, be the finite set of possible events. Our goal is to predict the next occurrence of a target event $e_{\mathcal{T}} \in \mathcal{E}$.

Initially we partition our dataset into *episodes*, that correspond to the periods between consecutive occurrences of the target event $e_{\mathcal{T}}$ for each system. Each episode ep_i^j of system j begins with the first event after the occurrence of $e_{\mathcal{T}}$ and ends with the next occurrence of target event (Figure 4.1).

The events within each episode are grouped into time-segments based on the nature of the log generation process. For example, a segment may correspond to a day or to a single usage of the equipment. This is essential in cases where the granularity of the timestamp is not high enough to distinguish the order of the events within the segment.

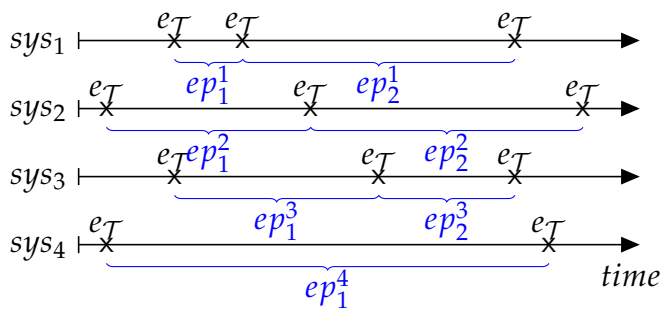


Figure 4.1: Partitioning event logs into episodes

Every segment is represented by an $x \in \mathbb{N}^{|\mathcal{E}|\times 1}$ vector, similar to the bag-of-words representation (in our case bag-of-events) in text mining Joachims (1998), and contains the number of times each event occurred within the segment. Thus, $x_{e_i} \geq 1$ if the event e_i occurred within that segment and $x_{e_i} = 0$ otherwise.

Every episode forms a $|\mathcal{E}| \times t$ matrix, $X^{ep_i} = [x^1, x^2, \ldots, x^t]$, where x^1 is the first segment of that episode and x^t the last (i.e. the segment that contains the target event $e_\mathcal{T}$, $x^t_{e_\mathcal{T}} \geq 1$). For simplicity, we use t for the duration of every episode although the duration of each episode is different. Also, for bag-of-events representation $x^i_{e_j}$ the superscript i is the time of the segment (starting from 1 at the beginning of the episode) and the subscript e_j corresponds to the event e_j.

4.3.1 *Preprocessing*

In this representation it is straightforward to apply standard preprocessing techniques Zheng et al. (2009). In this work we employ the following ones:

- Removing events that appear less that n times. Extremely rare events (e.g. ones that occur only a few times within millions of event entries) can potentially[1] be removed in order to reduce the dimensionality of the data.

- Event Collapsing: Removing multiple occurrences of events in the same segment. In other words, making the bag-of-events of each segment (x^i) binary. Depending on the application, multiple instances of an event at a timestep can either be noise or may not provide useful information.

- Penalising frequent events. Usually the most frequent events do not contain significant information since they correspond to issues of minor importance. A tf-idf (term-frequency - inverted document frequency) or similar approach can be used to penalise such frequent events.

- Removing events that occur in consecutive segments. This step can be significant in scenarios where events of minor importance occur and appear in every timestep until their underlying cause is treated by the technical experts. In such cases only the first time of the occurrence is important.

- Statistical Feature Selection. We propose another feature selection method that is based on the time intervals between the events $e_i \in \mathcal{E} - e_{\mathcal{T}}$ and the target event $e_{\mathcal{T}}$. Specifically, we introduce $|\mathcal{E}| - 1$ random variables, $T_{e_i}^{e_{\mathcal{T}}}$, with $P(T_{e_i}^{e_{\mathcal{T}}} \leq t)$ being the probability of event e_i occurring less than t timesteps before the next occurrence of $e_{\mathcal{T}}$. We fit Weibull distributions Kaiser and Gebraeel (2009); Nakagawa and

[1] Some rare events can be important for predicting the target event and thus, this step should be evaluated experimentally

Osaki (1975) on these random variables and we select the features e_i if $E\left\{T_{e_i}^{e_\mathcal{T}}\right\} \leq \gamma$. A rationale for this is that good predictors for $e_\mathcal{T}$ should occur within an expected interval before the occurrence of $e_\mathcal{T}$ (see Section 4.6.5).

One can easily employ text-based transformations on the feature matrix, such as Latent Semantic Indexing (LSI) Deerwester et al. (1990) and word embeddings Mikolov et al. (2013c) to reduce the dimensionality of the feature space and to exploit possible hidden structure. However, an initial evaluation of such transformations did not lead to improvement of our results and thus we leave it as future work.

4.4 METHODOLOGY

Our goal is to learn a function $r(x^i)$ for a target event $e_\mathcal{T}$, $r : \mathbb{R}^{|\mathcal{E}|\times 1} \mapsto [0, 1]$, that quantifies the risk of event $e_\mathcal{T}$ occurring in the near future, given the events that occurred at timestep i, x^i. One property of $r(x^i)$ is to be monotonically increasing in time ($r(x^i) \leq r(x^{i+1})$) and its maximum value would be at the end of each episode, when $e_\mathcal{T}$ occurs.

Our approach is driven by the assumption that *as we approach the occurrence of $e_\mathcal{T}$, one or several events that are correlated to $e_\mathcal{T}$ will start occurring* (i.e. predictors). For an episode that starts at timestep 1 and ends at timestep t, this function at timestep i should increase significantly as the distance from the target event ($t - i$) approaches to zero. Furthermore, $r(x^i)$ should be close to zero when the distance from the target event is relatively large. Thus, we formulate a regression problem $< x, y >$ as $< x^i, f(i; t) >$ where

$$f(i;t) = \frac{1}{1 + e^{s((t-i)+c))}} \tag{4.1}$$

and we learn our function by minimising the mean squared error: $||r(x^i) - f(i)||_2^2$.

In figure 4.2 we present the form of this function for an episode of length 300. Parameters s and c correspond to the steepness and the shift of this *sigmoid-shaped* function. Selecting a proper value for c is highly significant because it represents the expectation of the time before the target event at which correlated events will start occurring. Thus, this parameter depends on the dataset and should be tuned by the corresponding area experts.

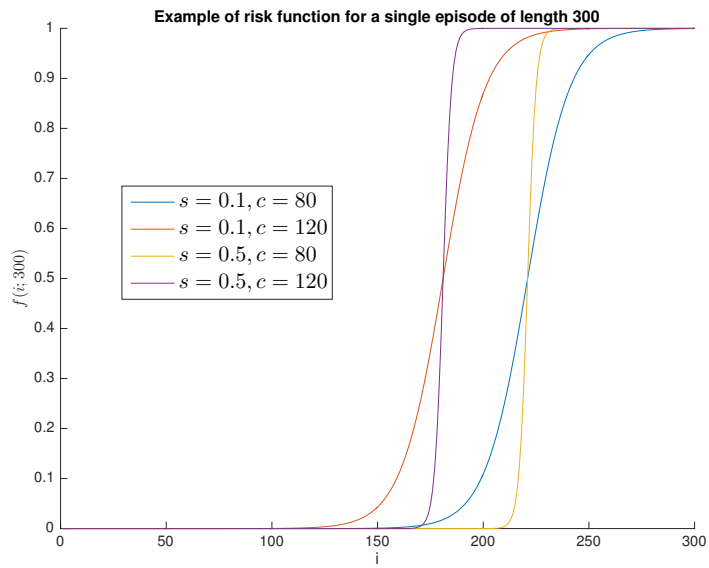


Figure 4.2: Sigmoid Function

For a collection of k episodes we can estimate this function by solving a regression problem $< X, Y >$ where $X = [X^{ep_1}; X^{ep_2}; \ldots; X^{ep_k}]$

is the set of feature (event) matrices and Y is the set of the k vectors with the risk values,

$$Y = [\underbrace{f(1;|ep_1|)\dots f(t;|ep_1|)}_{ep_1}\dots\underbrace{f(1;|ep_k|)\dots f(t;|ep_k|)}_{ep_k}]^T.$$

where $|ep_i|$ is the duration of episode i.

There are several off the shelf methods that can be used to solve this regression problem. In this work, we employed the random forest regression model Liaw and Wiener (2002) because it gave the best results compared to other regression techniques such as LASSO, Support Vector Regression (SVR) and Gradient Boosting.

Our feature matrix X^{ep_i} consists only of other events $(\mathcal{E} - \{e_\mathcal{T}\})$ but it can be extended in a straightforward manner by adding extra rows for other possible additional information, such as keywords extracted by text description, environmental variables and sensor time-series data.

Finally, we aim to identify the set of predictors with respect to $e_\mathcal{T}$, $\mathcal{P}^{e_\mathcal{T}} \subseteq \mathcal{E} - \{e_\mathcal{T}\}$, which is the subset of events that precede the target event and can lead to its prediction.

4.4.1 *Multiple Instance Learning Setup*

Multiple Instance Learning (MIL) Foulds and Frank (2010) is applied when a label (class) is associated with a bag of instances (feature vectors) rather than a single instance, and it is not known in advance which subset of the vectors contributes to the class label. Similarly, in our regression formulation, a MIL setup is more

appropriate: several events appear shortly before the occurrence of the target event but only a small subset of them is related to it (i.e. predictors). That is, for the interval right before the occurrence of $e_{\mathcal{T}}$ where $f(i;t) \gg 0$, apart from possible predictors there are other events that are not associated with $e_{\mathcal{T}}$. Therefore, if we discard this information the training set will contain wrong y values for those timesteps when only unrelated events occur before $e_{\mathcal{T}}$. Moreover, when the target event is rare, which is usual to critical events Koh (2009), we have to deal with the imbalances in the labels.

In order to tackle both problems, we adopt a sliding window approach in order to oversample the intervals where $f(i;t) \gg 0$ and create more samples where possible predictors occur. Specifically, for the intervals where $f(i;t) \geq 0.1$, we apply a sliding window over the feature vectors, x^i, by averaging their values, $x' = \frac{1}{w}\sum_i^{i+w} x^i$ and retain as y value the sigmoid output of the last timestep within that window, $y' = f(i+w;t)$ (Figure 4.3).

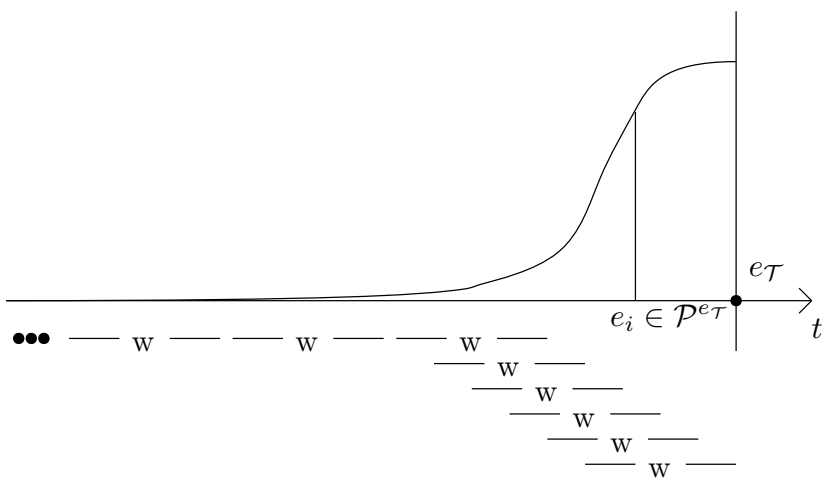


Figure 4.3: Sliding window approach: during the critical interval the window step decreases

The window length w should be large enough to create multiple instances where $\mathcal{P}^{e_\mathcal{T}}$ is present. Also, for our approach we set the step size equal to 1.

4.4.2 *Prediction*

The trained model's output, $\hat{r}(x^i)$ corresponds to risk of $e_\mathcal{T}$, given the events that occurred in x^i. Once the model is deployed and new sequences of events arrive, we obtain a new risk timeseries $[\hat{r}(x^1), \hat{r}(x^2), \dots]$ which we use to perform the prediction of $e_\mathcal{T}$. Figure 4.4 shows a real example of well fitted $\hat{r}(x^i)$ of a single episode from our test set, consisting of 7446 timesteps. The top part of the figure shows the whole episode and the bottom part presents the last timesteps.

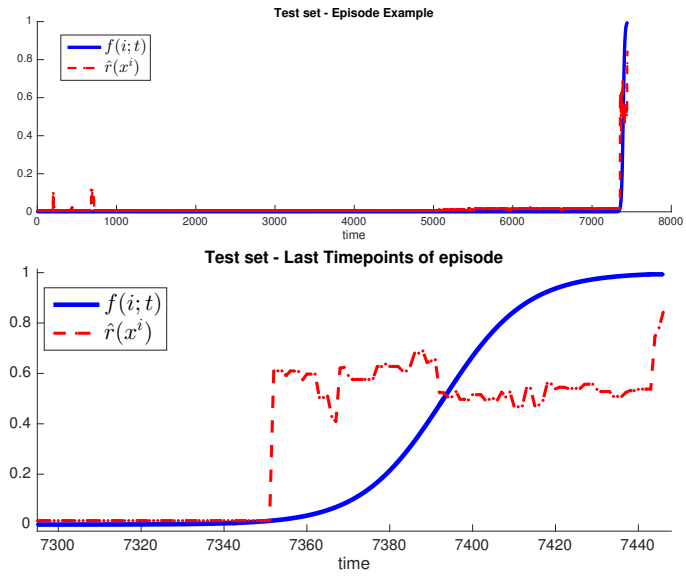


Figure 4.4: $\hat{r}(x^i)$ for a single episode. Top: the whole episode. Bottom: Last timesteps until the occurrence of $e_\mathcal{T}$

In order to decide wether $e_{\mathcal{T}}$ will occur we employ a simple thresholding approach. Specifically, we define a thresholding parameter, h and make a positive prediction if $r(x^i) \geq h$. In order to make a correct prediction in the example 4.4, the threshold could be set as $h \geq 0.2$. Unlike the probability thresholds in binary classification that are used to generate the ROC curves, increasing this threshold does not guarantee the reduction of false positives. However, we study the value of the threshold and its relationship to precision and recall in section 4.6.2.

4.4.3 *Method summary*

The steps of our methodology for creating and deploying the prediction model are the following:

1. **Preprocessing**
   - Partition the data into episodes.
   - Group the events of each episode into segments. The segment size should comply to the log collection characteristics.
   - Create matrices X^{ep_i} for all episodes.
   - Apply the preprocessing steps.
2. **Training/Validation**
   - Group segments into windows of size w.
   - Create the response vector $[f(i;t)]^T$ for each episode
   - Form the regression dataset $< X, Y >$
   - Train the desired regression model via cross validation

3. **Testing/Deployment**

   - As new segments arrive, keep the latest by applying a sliding window (w) and create the event matrix X^w
   - For each new segment update and preprocess X^w
   - Compute $\hat{r}$ and raise an alert if $\hat{r} \geq h$

4.4.4 *Parameters*

In Table 4.2 we present the list of parameters of our method. Most of the parameters can be selected via cross validation while training. Parameters w, γ and c should be defined by experts because they quantify the expected time (*in segments*) before the occurrence of $e_{\mathcal{T}}$ at which a prediction is possible. However, if no prior knowledge is available or it is hard to acquire, one can use the method presented in the previous chapter, that investigates the intervals between $e_{\mathcal{T}}$ and the other failures. The value of the decision threshold h can be configured according to desired confidence and the prediction interval (see section 4.6.3).

stage	parameter	description
preprocessing	w	window size
	c	sigmoid shift
	s	sigmoid steepness
	γ	feature selection
model training	#trees	number of trees
	#depth	max depth of each tree
model deployment	h	decision threshold

Table 4.2: parameters

4.5 EXPERIMENTAL SETUP

We apply our method on event logs generated by monitoring systems onboard, where the events correspond to failures of aircraft components. Our goal is to predict such failures that are critical to the function of the aircraft. When these failures occur, the aircraft must stay on the ground for control or repair and therefore, being able to predict these failures in advance and plan the corresponding maintenance increases the availability of the aircraft.

4.5.1 *Dataset*

Our data consists of event logs from a fleet of 60 aircraft and spans within several years of operation. The alphabet of events consists of more than 3400 distinct failures. Each failure corresponds to a system; the information about the system comes from each failure's Air Transport Association (ATA) [2] chapter.

The target event $e_{\mathcal{T}}$ is predefined and corresponds to a failure that affects the landing gear (ATA 32). Figure 4.5 shows the survival function Cox and Oakes (1984b) of $e_{\mathcal{T}}$. In 50% of the total cases this failure occurs around 2000 flights after its previous occurrence.

4.5.2 *Training, Validation and Test*

In order to create the training set for the target event $e_{\mathcal{T}}$ we initially select the subset of the aircraft at which the corresponding

[2] https://en.wikipedia.org/wiki/ATA_100

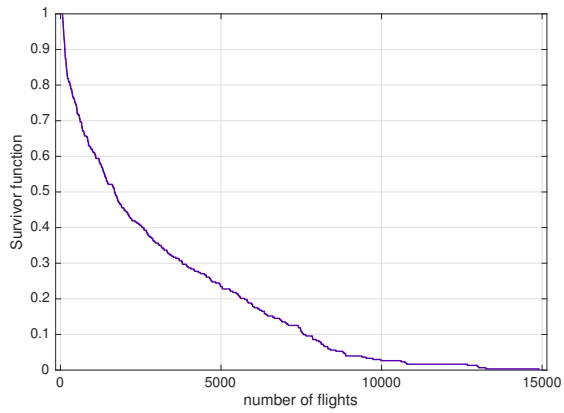


Figure 4.5: Survival function of the target event $e_{\mathcal{T}}$

failure occurred ($AC^{e_{\mathcal{T}}} \subseteq AC$). For the training set we select only aircraft from $AC^{e_{\mathcal{T}}}$.

We partition each sequence of events of every aircraft in $AC^{e_{\mathcal{T}}}$ into episodes and we discard the events that occurred after the last occurrence of $e_{\mathcal{T}}$ of every aircraft. Our approach is **leave k episodes out**, that is, we randomly select k episodes from $AC^{e_{\mathcal{T}}}$ for the training set. We select half of the aircraft in $AC^{e_{\mathcal{T}}}$ for training and half for testing. In order to train the random forest model and select the proper parameters we use 5-fold cross validation on the training set.

4.5.3 *Baseline Algorithm*

In order to assess the performance of our method we employ a simple, yet popular approach Murray et al. (2005); Sipos et al. (2014b) for comparison. In our baseline approach, the event prediction is performed by a binary classifier trained in the multiple instance learning context. The classifier takes as input an instance

(or a bag of instances) and predicts if the target event will occur in the near future. Figure 4.6 shows where the instances of the two classes lay with respect to the target event.

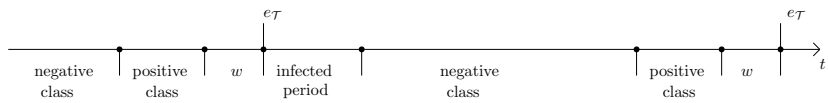


Figure 4.6: Classification for event prediction: time-position of classes

The depicted intervals in Figure 4.6 correspond to:

- *positive class*: This interval corresponds to the time before $e_{\mathcal{T}}$ when a prediction is correct and useful.
- w: The prediction margin. Corresponds to the minimum amount of time required by experts to respond to a failure.
- *negative class*: Instances within this interval are classified as negative since they are distant to $e_{\mathcal{T}}$
- *infected period*: This is the period that follows the occurrence of the target event.

The intervals above should be defined by field experts, so that they correspond to actual requirements. For a fair comparison, we set the positive class with respect to shift parameter c of our approach (Figure 4.7).

We employ linear Support Vector Machines (SVM) Fan et al. (2008) with $l1$ regularisation for the classification. Also, we perform the same train/validation/test scheme and we create the positive and negative instances similar to Sipos et al. (2014b).

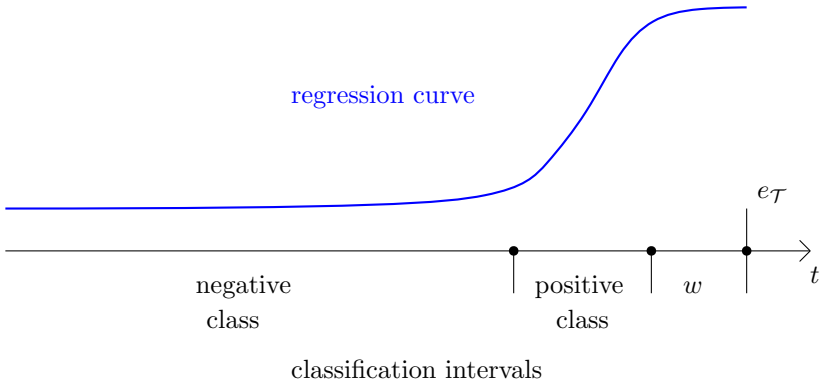


Figure 4.7: Regression and classification intevals

4.5.4 *Evaluation at the episode level*

In prediction/classification problems one can evaluate the performance of a method using standard metrics, such as F1-score, precision, recall and Area Under the Curve (AUC). In the literature, these metrics are evaluated at the instance level, (i.e. for each bag or for every feature vector). However, when dealing with event prediction in temporal environments these metrics should be evaluated at the episode level. In order to justify this approach consider the following artificial, extreme-case example: In Figure 4.8 we present the output of a hypothetical classification-based prediction method for three episodes. The blue lines correspond to the binary classifier's output and below each line the classification intervals are shown. The two blue spikes correspond to two instances, falsely marked as positives (false positives). It is clear that at the bag/instance level, the performance of the classifier is very high. However, at the episode level, only the first episode is a true positive, the following two are false positives.

In many real world applications and specifically in predictive maintenance, a false positive triggers unnecessary processes that

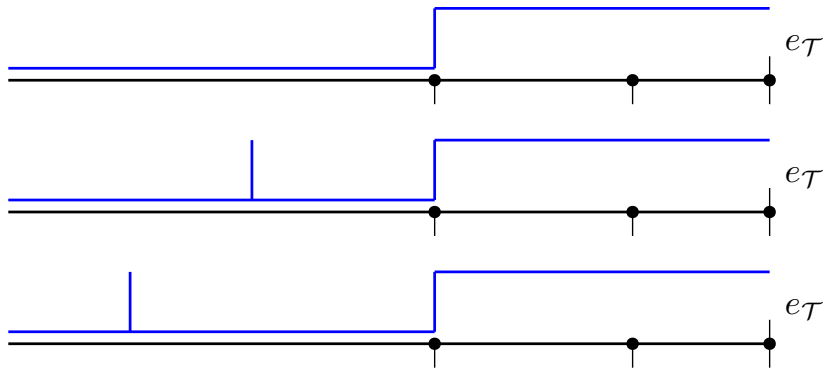


Figure 4.8: Artificial Example: A true positive episode (top) and two false positives (middle and bottom)

have a cost and it is critical to minimize their amount. Therefore, episode-level evaluation is more important in this context and we evaluate our method accordingly.

Regarding our regression approach, the evaluation at the instance level is performed by the mean squared error whereas for the episode level, we use the aforementioned thresholding approach.

Specifically, assuming that the threshold was crossed at i flights before the occurrence of $e_{\mathcal{T}}$ the episode is true positive if $v \leq i \leq 2c$, where c is the shift parameter of our sigmoid function and v is a constant that determines the minimum time before the occurrence of $e_{\mathcal{T}}$ when the prediction is useful[3]. For fair comparison with the baseline we set $v = w$, the prediction margin.

[3] A prediction is useful if it is performed within a significant time period before $e_{\mathcal{T}}$ in order to provide enough time to plan the corresponding maintenance.

4.6 RESULTS

4.6.1 *Bag-level Performance*

In Figure 4.9 we present the ROC curve for the baseline model at the instance level. We performed 5-fold cross validation to estimate the best parameters for the SVM classifier.

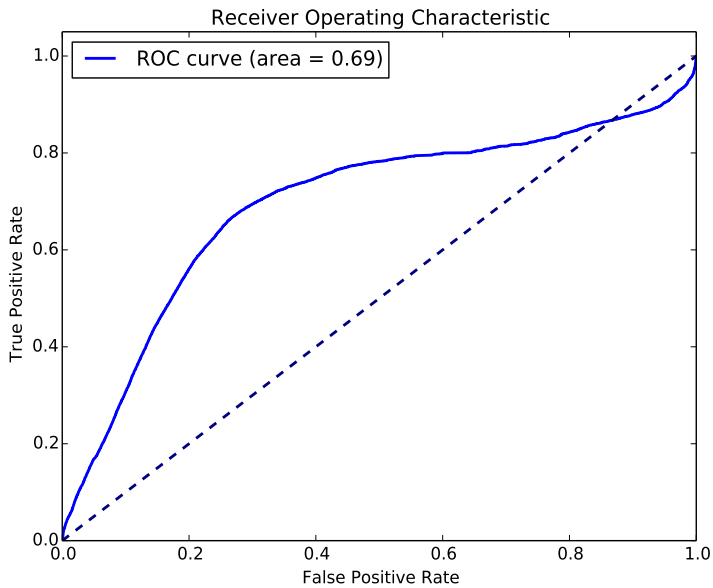


Figure 4.9: Baseline approach: SVM ROC Curve

Figure 4.10 presents the mean squared error of our random forest regression approach with respect to the number of trees.

The parameters that lead to the minimum MSE at the validation set were selected.

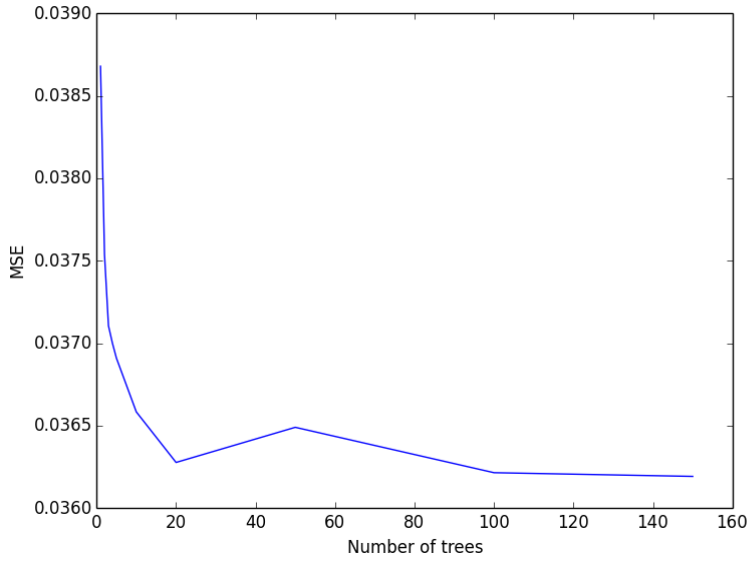


Figure 4.10: Random Forest Performance: Number of trees vs Mean Square Error (MSE)

4.6.2 *Episode-Level Performance*

Table 4.3 shows the prediction results at the episode level, considering the best parameters for both our approach (Random Forest Regression (RFR)) and the baseline (SVM). An episode is considered a true positive if $e_{\mathcal{T}}$ is predicted between 5 and 200 flights before its end. More specifically, we compute the number of flight before the target failure that the decision threshold, γ, was cross *for the first time*. Consequently, if the threshold was crossed more than 200 flights before the occurrence of $e_{\mathcal{T}}$, the episode is a false positive and finally, an episode is false negative if the threshold was never higher that γ.

Similar to Sipos et al. (2014b), we evaluate our experiments using standard metrics adapted in predictive maintenance:

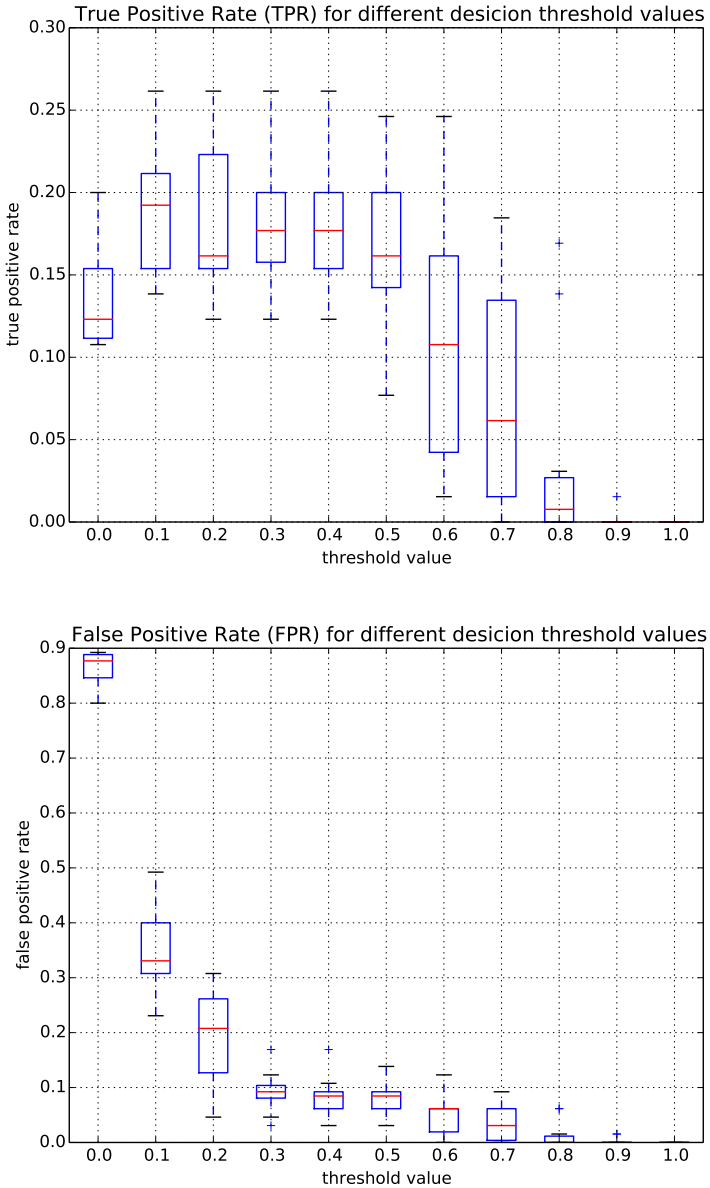


Figure 4.11: Theshold values impact on number of True Positives (left) and number of False Positives (right)

- precision: TPs / (TPs + FPs)

- recall: TPs / all failures
- f1-score: $2\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}$

Our experimental evaluation presented in Table 4.3 shows that our method, RFR, outperforms the selected baseline approach, which has very poor performance.

method	precision	recall	F1-score
SVM	0.21	0.02	0.03
RFR	0.64	0.23	0.34

Table 4.3: Prediction results at the episode level

The threshold value that lead to the best prediction results was chosen. In the next section we discuss the most important aspects for selecting a proper threshold value.

Finally, we have to note that several other approaches were tested but we omit the corresponding results because of the poor performance. Specifically, we performed similar experiment using Hidden Markov Models but we had to limit the amount of time-steps before the failure (or the episode length) because of the performance constraints. Thus, with this limitation the HMM was unable to give any significant results. Moreover, we tested association rule mining, aiming towards estimating rules of the form $P^{e_{\mathcal{T}}} \rightarrow e_{\mathcal{T}}$ in order to capture the aforementioned predictors. The main constraints in this approach were the lack of the cardinality information and the formation of large *buckets* to capture predictors that occurred many flights before the failures, which led to many high-confidence rules. Finally, sequence prediction approaches were not able to perform well, mainly due to the fact that most of the failures were spurious and not related to the target failure.

4.6.3 *Decision threshold selection*

The decision threshold h constitutes a very important parameter of our approach and thus, we investigate its impact on the results with respect to the evaluation metrics.

In Figure 4.11 we present the impact of different threshold values on the final prediction.

Setting a proper threshold value can be performed according to the desired TPR and FPR and also, to the time before the target event at which a prediction is desired. In figure 4.12 we present the impact of the different threshold values to time that a prediction is performed. On the x-axis we place the different threshold values and on the y-axis the **first time** (at every episode) at which the threshold was crossed.

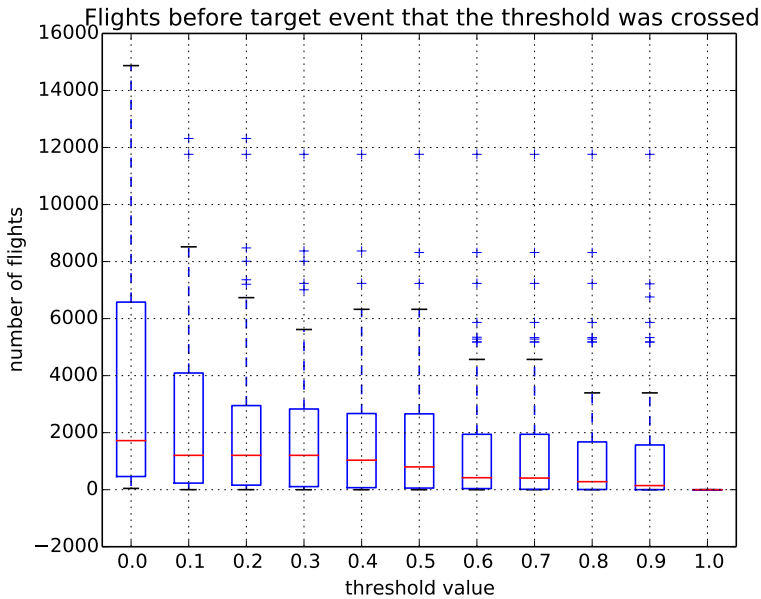


Figure 4.12: $\hat{r}$ vs distance (in flights) from target event

Figure 4.12 also shows that the risk function $\hat{r}$ crosses higher threshold values when approaching the target event. The information from this box plot can be used to compute a confidence value for the time to the target event after observing failures for several flights

4.6.4 *False Positives*

As stated previously, false positives' absence is very important in predictive maintenance and it is very critical to study their occurrence and their time of occurrence with respect to the target event. In Figure 4.13 we present the empirical cumulative distribution function (ecdf) of the false positives' distance from the target event.

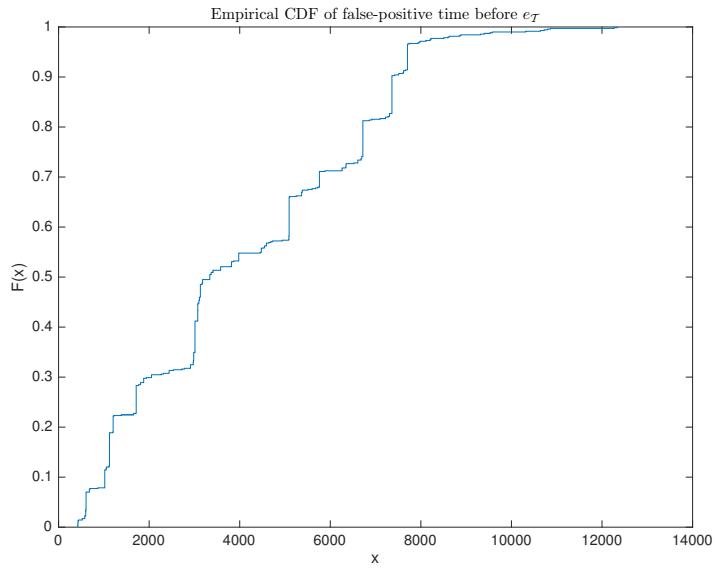


Figure 4.13: Empirical CDF of the time between false positives and $e_{\mathcal{T}}$

Unfortunately, a false positive may appear far before the occurrence of $e_{\mathcal{T}}$. However, a possible explanation of their presence is the intervention of engineers that affected the target event, by performing actions that prevented its occurrence.

4.6.5 *Model Interpretation*

In predictive maintenance it is important to understand and interpret the prediction model for two main reasons: first, it can provide very useful feedback to experts and second, it is a qualitative way of evaluating a model (i.e. the features make sense). In this work, our features consists only of other events/failures and therefore, we can identify candidate predictors $P^{e_{\mathcal{T}}}$ from the features weights assigned by the random forest.

Table 4.4 presents the top most important features/events as they were ranked by the random forest. The feature with the highest score corresponds to a failure that also belongs to the landing gear (ATA 32).

event id	score	ATA chapter
6226	0.260	32-31
1244	0.167	46-20
2813	0.090	77-23
4604	0.088	27-92
6444	0.041	73-21
348	0.026	30-31

Table 4.4: Top Ranked Features (events)

Figure 4.14 shows the Empirical Probability Density Function (PDF) for of the random variables described in Section 4.3 that

correspond to a low ranked feature e_{1353} and the third highest one, e_{6226}.

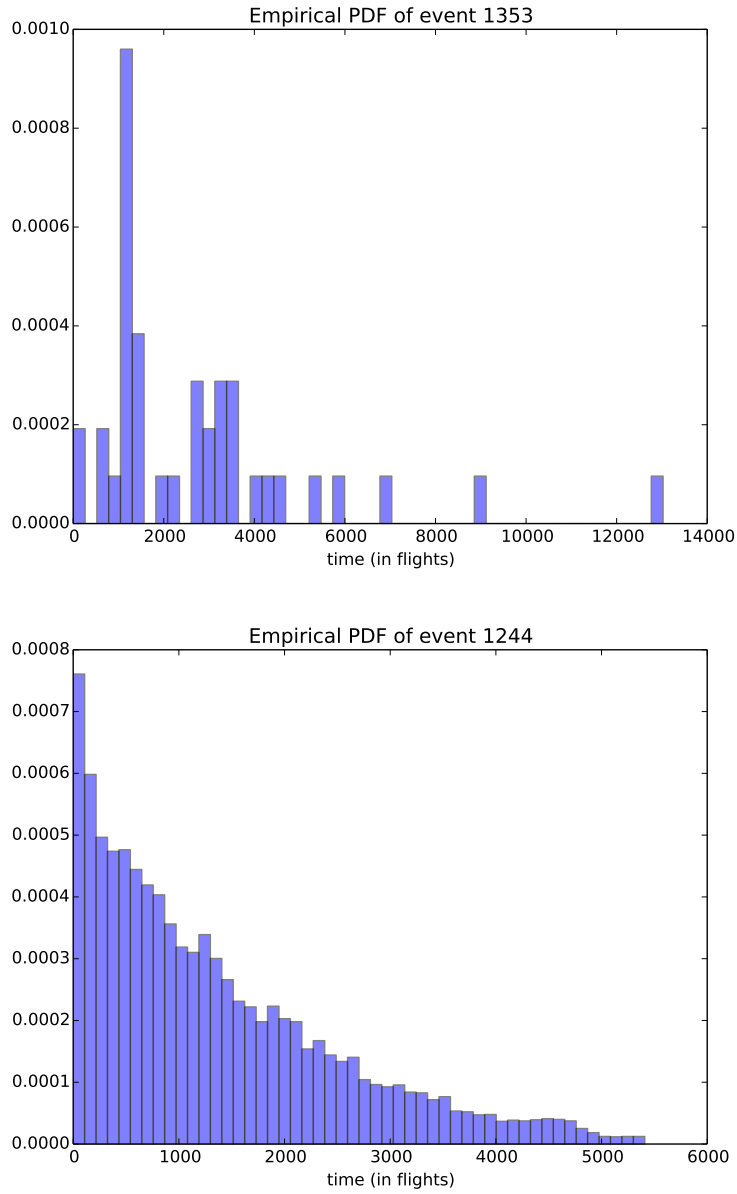


Figure 4.14: Empirical PDFs. Top: event with low score ($T^{e_\mathcal{T}}_{e_{1353}}$). Bottom: event with hight score ($T^{e_\mathcal{T}}_{e_{1244}}$)

In Figure 4.15 we show three examples of our model's output. If the decision threshold is set as $h = 0.6$, then the first one (left), shows a true positive episode where a prediction was correctly made 100 flights before the occurrence of $e_{\mathcal{T}}$, while during the first 8700 flights the output value of our risk function was low. The second episode (middle) is a false positive since our models output was grater than the threshold more than 3000 flights before the target's occurrence and finally, the last episode is a false negative.

4.7 CONCLUSIONS AND FUTURE WORK

In this chapter we presented a method for predicting future events from event logs in the context of predictive maintenance. It constitutes a novel combination of state of the art statistical and machine learning techniques and our experimental evaluation shows that it outperforms a common baseline approach. A major contribution of this work is the fact that it constitutes the first attempt to perform failure prediction given only post flight reports. Our future work will aim to increase the performance by exploiting information from a variety of sources, such as sensors, maintenance logs and environmental variables.

4.7.1 *Infusion and Impact*

The importance of failure prediction and the possible impact of our approach can be highlighted by studying its impact on airlines. Specifically, stats show that the unexpected occurrence

of our target event [4] can lead to either a flight delay that costs approximately 4000 USD to the airline or to a flight cancellation, costing 15.000 USD[5]. Within a year of observations on several airlines, its occurrence costed around 2.000.000 USD and therefore, being able to predict 20% of its occurrences on time, can lead to saving up to 400.000 USD per year for this particular failure.

[4] The target event was not selected based on the financial impact of its occurrence and thus, there are other failures that may have much bigger financial impact on the airlines and our approach could be of greater benefit.

[5] the numbers were calculated based on information from this document: https://www.eurocontrol.int/sites/default/files/content/documents/sesar/business-case/european_airline_delay_cost_reference_values_2011.pdf

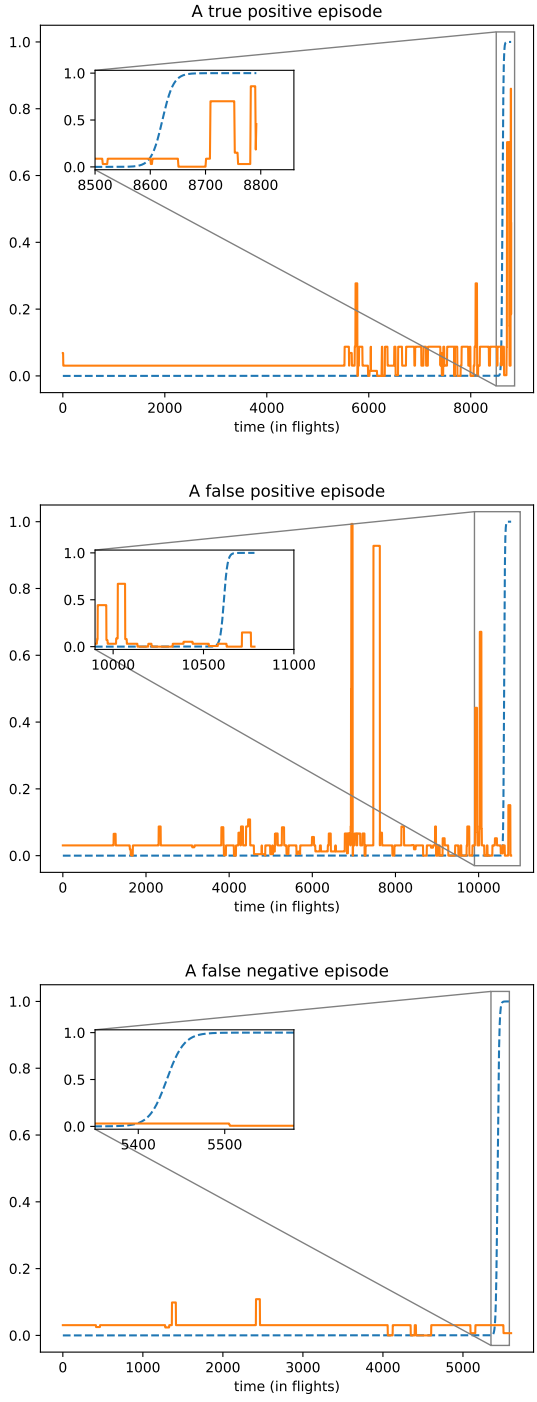


Figure 4.15: Examples of $\hat{r}(x^i)$ for three episodes. Top: true positive, middle: false positive, bottom: false negative

5

LOGBOOK DATA PREPROCESSING

In this chapter we present our approach for logbook data preprocessing and information extraction. Specifically, given a set of logbook entries, we propose an unsupervised method for cleaning and extracting information regarding aircraft maintenance activities. Our approach is based on recent advances in text mining using neural networks and traditional Natural Language Processing (NLP) techniques. Our results show the potential of using our method for unsupervised cleaning of such logs using zero domain knowledge.

5.1 RELATED WORK

Event reports or logbook entries are being collected in many domains and it is not until recently that researchers started analyzing these data, since they contain valuable information about the domain and are written by experts Ittoo et al. (2016). For examlple, the authors in Tixier et al. (2016a) and Tixier et al. (2016b) analyze injury reports in the construction domain and their goal is to extract keywords and predict accidents. In our field, aviation, the authors in Saeeda (2017) analyze safety reports in order to extract related onotologies and in Tanguy et al. (2016) they study the probelm of topic modelling in similar data. In Deleger et al. (2010) they use a simple, rule-based approach to

extract information from patient records containing information about medication, in Savova et al. (2010) the authors propose cTAKES, an NLP system for processing free-text clinical records and in Siklosi et al. (2013) the authors propose a method to correct spelling mistakes from clinical records.

It is evident that the logbook contains information that comes from domain expertise and thus, being able to extract it could be of great benefit for many applications. Towards this direction, we devoted most of our efforts in cleaning the free-text in our maintenance logs and we developed a *context-aware* approach for correcting errors in our corpus, which is based on word embeddings.

5.2 LOGBOOK DATA IN AVIATION

In aviation, the logbook is the place to store the information about any maintenance action performed on an aircraft, either scheduled or not. Its schema contains technical information about issues along with the corresponding maintenance actions. The most important pieces of information are stored in the following columns:

- Aircraft identifier: An alphanumeric that is unique for each aircraft, such as aircraft registration number or Manufacturer Serial Number (MSN).
- Date and time of issue and the corresponding action
- Issue text: The text that describes why the maintenance was requested, such as noise or system failure. Usually this is written by the crew. If the maintenance was scheduled

then this field contains an identifier that corresponds to the maintenance type.

- Action text: The action that was performed to resolve the issue. In case of scheduled maintenance with no issues found this is empty.

- Equipment replaced: If a component was replaced during the action then the serial numbers of both the new and the old one are written here.

- Related flight: If maintenance was requested by an issue that occurred during a flight, then the flight number is also included.

As indicated above, there are two types of maintenance actions that can take place: scheduled and requested. An entry for a scheduled maintenance activity, such as a daily inspection does not contain an action text unless something was discovered that required an action. On the other hand, a logbook entry for a requested maintenance will consist of the irregularities spotted by the crew and the actions performed by the maintenance personnel.

5.2.1 *Data Description*

Logbook entries contain usually short text that comes from a very specific vocabulary. The sentences are very compact and descriptive and usually, both the issue and the maintenance text consist of a single sentence. In Figure 5.1 we present the the length of the sentences in logarithmic scale for our dataset,

which shows that the vast majority of the entries have less than 20 words ($log_e(20) \approx 2.99$). The figure was generated using 440.000 issue text entries and 440.000 maintenance text entries ($440.000 + 440.000 = 880.000$).

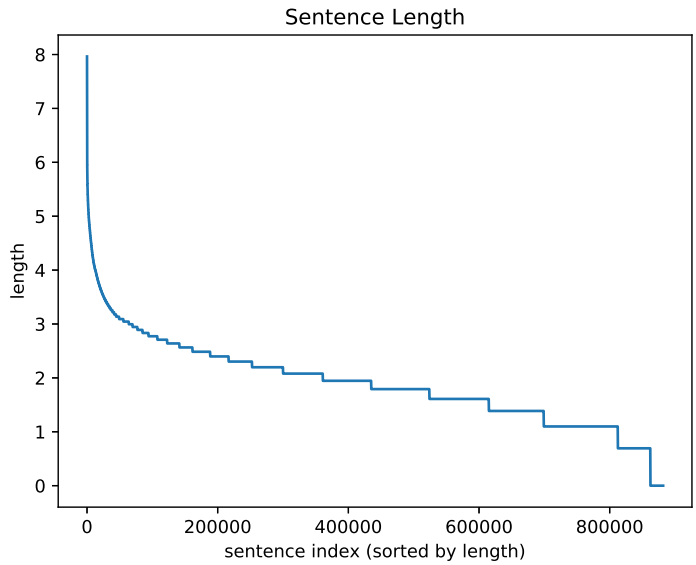


Figure 5.1: Sentence lengths in logbook entries

Furthermore Figure 5.2 shows the logarithm term-frequency of all the terms in our dataset. There exist 54000 unique terms and around half of them occur only once. Moreover, since more than 95% of the terms occur less than 55 times ($log_e(55) \approx 4$) in 880.000 entries, we can see that the vocabulary used is very limited. Moreover, as we will see in the next sections, a large percentage of the terms consist of typos and thus the actual number of terms is much lower.

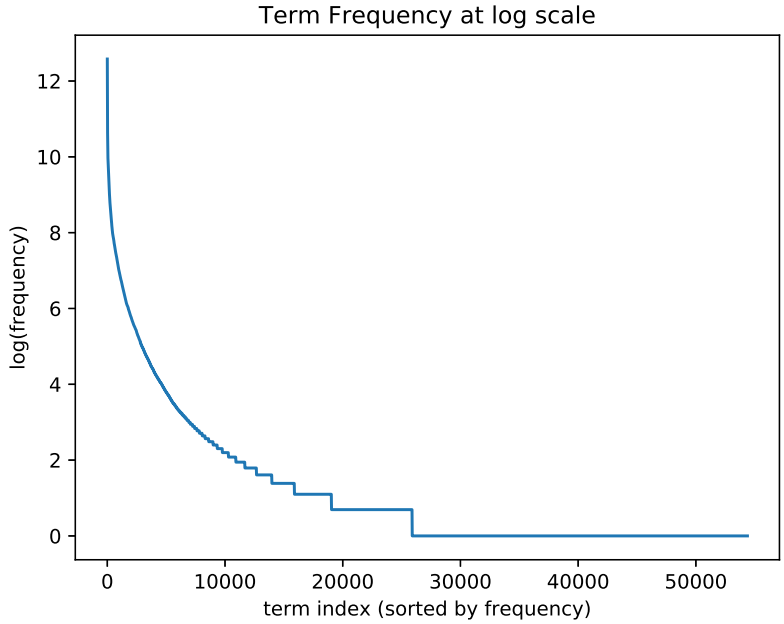


Figure 5.2: Term frequencies in logbook entries

5.2.2 *Cleaning the Logbook*

In every application where input is directly entered by users there is a lot of *noise* and when the input is free text, it usually contains typos and syntactical errors. Moreover, people tend to develop their own, domain specific *language* that consists of many abbreviations. Since our purpose is to use the logbook information for predictive maintenance, we have to process the raw data and make sure that we will be able to extract as much information as possible that is related to the maintenance actions that took place and the the reasons that triggered the actions (if not scheduled).

In Table 5.1 we show some examples of words that have been mistyped and the suggestions obtained by using the Enchant[1] spell checker (using the Python bindings pyenchant[2]).

Correct Suggestions		Wrong Suggestions	
mistyped word	suggestion	mistyped word	suggestion
windoe	window	windo	wino
prinat	printer	priner	prineer
wheeels	wheels	whls	whys
erplaced	replaced	rplcd	purpled
birdstrike	bird strike	birstroke	masterstroke
landinf	landing	langing	slanging
barke	brake	brakelt	brakemen
lihgt	light	ligt	gilt
tihgten	tighten	untighen	untiring

Table 5.1: Example of mistyped words and dictionary suggestions

Moreover, Table 5.1 highlights the need of using domain-aware methods for cleaning the text. For example, consider the suggestion $rplcd \rightarrow purpled$ that maps a common abbreviation to a wrong word. We treat this problem by proposing a context aware method that learns the context from the set of available logbook entries.

5.3 THE IMPORTANCE OF LOGBOOK DATA

It is evident that the logbook contains very important information about the components of the aircraft, however, we have to define specific objectives according to expected outcomes of processing such data. More precisely, have to answer the following questions:

[1] https://www.abisource.com/projects/enchant/
[2] https://pypi.python.org/pypi/pyenchant/

What do we expect to gain by including our logbook data in predictive maintenance? How should we integrate them into our predictive models?

As mentioned in the beginning of this chapter, logs written by experts have already attracted the attention of the data mining community because they contain invaluable information about the specific domain. Having the history of all maintenance actions gives the ability to perform very interesting experiments such as analyzing the lifetime of individual components, estimating the effectiveness of the actions performed and proposing maintenance actions based on occurring issues.

On the other hand, there are more reasons that are not so obvious and are highly important, especially for our ultimate objective, failure prediction in predictive maintenance. Aviation industry exists for many years now and precious knowledge has been *empirically* gathered by experts and has been exchanged withing their communities. Although there are many types of analysis that could benefit from such prior knowledge, in the context of our thesis we dealt with a very specific problem. Recall the predictor[3] events from Section 3 and consider the following example: When such predictors occurs, some maintenance engineers may know from experience that this will lead to the failure of our target failure and thus, he takes some actions that prohibit the target event's occurrence. Furthermore, the same effect could possibly occur when something is identified and fixed during a scheduled maintenance by a replacement or a repair. All this information appears in the logbook and the existence of such

[3] given a target failure that we aim to predict, a predictor is a failure whose occurrence increases the confidence that the target failure will occur in the near future

cases impact the formation of our machine learning dataset in the following manner:

- Since the target event does not occur after the engineer's intervention, our training set does not assign the proper risk for the flights when the predictors occur. In Figure 5.3 we illustrate this problem.
- Consecutively, this will affect the evaluation of the algorithm by increasing the number of false positives. The model outputs high risk by observing one or more predictors, however, since the target event will not occur due to the maintenance, the episode will be marked as false positive.

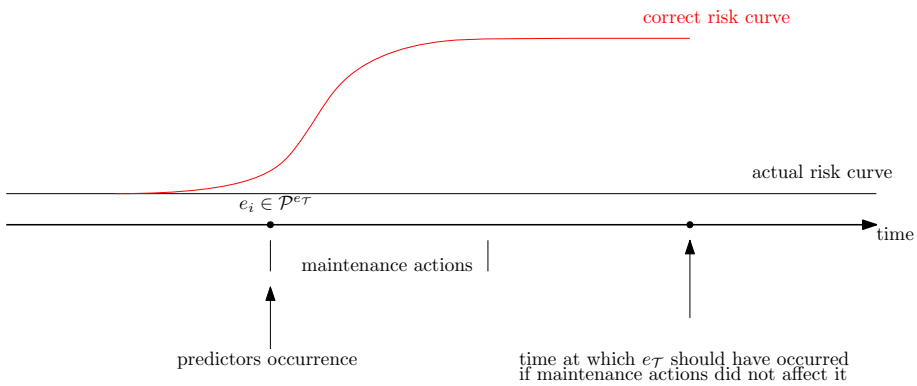


Figure 5.3: Maintenance actions' impact on failure prediction dataset

As a result, once the information from the logbook is extracted we can use it in order to solve the two problems accordingly:

- Form a different learning dataset which will also include maintenance actions related to the target failure as events, or

- Infuse this information directly on the regression problem by adding features extracted from the maintenance actions text.

Our efforts were initiated by observing the aforementioned problems and consequently, we devoted significant amount of time in cleaning the maintenance logs.

5.4 CONTEXT-AWARE SPELL CORRECTION VIA WORD EMBEDDINGS

Let $\mathcal{L} = \{lb_1, lb_2, \ldots, lb_m\}$ be the logbook containing m entries. For our purposes we only use the information in the issue and the maintenance text (i.e. we don't perform per aircraft or per component analysis so we discard this information) and thus, each logbook entry consists of two *documents*, $lb_j = \{lb_j^{issue}, lb_j^{maint}\}$, the issue and the maintenance text respectively. Furthermore, Let $\mathcal{W} = \{w_1, \ldots w_k\}$ be the set of unique words in the logbook, i.e. the vocabulary, which contains both correct words and typos (both intentional and intentional)[4].

Our main idea is to correct these typos using contextual information, that is, information that comes from the neighboring words in the sentence. For example, since the words "replaced","rplaced" and "rpld" will usually occur in similar sentences, we aim to use this information for better text cleaning. It is obvious that most traditional spellcheckers will fail to map the word "rpld" to the word "replaced".

[4] we consider abbreviations to be *intentional typos*. For example, the word "replaced" may appear in the logbook as "rplcd"

This type of similarity has been recently obseverd in word embeddings, or vector representation of words. Thus the approach presented in the next sections is based on such methodology.

5.4.1 *Word Embeddings & the skip-gram Model*

Vector representation of words (or word embeddings) Maas et al. (2011) are representations of words in d-dimensional spaces, i.e. mappings of a vocabulary $\mathcal{W}$ to a high dimentional space; $f : \mathcal{W} \longmapsto \mathbb{R}^d$. When an embedding is generated, it aims to capture some specific kind of information so as to be reflected in the projected space. For example Latent Semantic Indexing (LSI) Landauer et al. (1998) can be seen as an embedding of words which preserves information about *latent topics*: Words that belong in the same topic are close in the projected space.

More recently, these representations are being generated by neural networks Mikolov et al. (2013a,b) that are trained on a given corpus. The word embeddings come from the coefficients (or parameters) of the neural network (usually the hidden layer), similarly to other domains, such as image embeddings.

In our work we use the skip-gram Mikolov et al. (2013b) (Figure 5.4) model, a recent approach based on neural networks. Its objective is to predict a given word's *context*, i.e. its neighboring words in a sentence.

The objective function to be minimized by the skip-gram model is the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c}\log p(w_{t+j}|w_t) \tag{5.1}$$

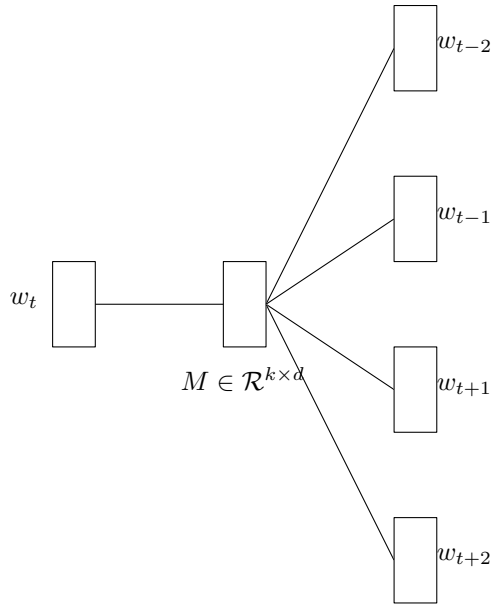


Figure 5.4: The skip-gram model

Given a document, the training set is generated by a sliding window over the words of the document following the direction of the text flow. Furthermore, the authors introduce the method of *negative sampling* in order to increase the efficiency of the training process. For more information on the skip-gram we encourage to read the original paper Mikolov et al. (2013a).

Currently, there are several out of the shelf embeddings that one can use. These embeddings have been trained on huge corpora and provide vector representations that can be used directly by researcher on several tasks, such as text categorization, topic discovery and keyword extraction. Word2Vec[5] and GloVe[6] are probably the most popular ones. However, for our purpose we have to build our own ebeddings for three main reasons:

[5] https://code.google.com/archive/p/word2vec/
[6] https://nlp.stanford.edu/projects/glove/

- We want to obtain the representations of words that are certainly not available in the available embedding collections, such as typos and abbreviations.

- The available embeddings are trained on huge corpora and the representations capture more *global* structure and similarities. In our case, we have a limited vocabulary and thus we need more specialized vector representations.

- Due to the particularities of the logbook entries we need to learn the representation in order to capture the contextual information of each word in the context of maintenance in aviation.

Therefore, in the following section we present our approach for creating the embeddings based on word2vec, trained on the available logbook entries.

5.4.2 *Creating word embeddings from logbook entries*

Our corpus consists of $2m = 880000$ documents (where m is the number of entries in the logbook) because we consider the issue and the maintenance text of each entry as two distinct documents,

$$\mathcal{D} = \{\underbrace{lb_1^{issue}}_{d_1}, \underbrace{lb_1^{maint}}_{d_2}, \ldots, \underbrace{lb_m^{issue}}_{d_{2m-1}}, \underbrace{lb_m^{maint}}_{d_{2m}}\}$$

For every word in each document we predict all the other words that exist in the document. We selected this approach because the logbook entries are short and the syntax is not important. Thus, our objective function is slightly modified:

$$\frac{1}{2m}\sum_{i=1}^{2m}\sum_{w\in d_i}\log p(\{w\in d_i-\{w\}\}|w) \tag{5.2}$$

After training the neural network, we obtain the emdedding matrix $M\in\mathbb{R}^{k\times d}$. Figure 5.5 shows an interesting example of the generated word embeddings that highlights the main idea of our approach, which is described in the following section. The graph was created by selecting 4 terms, *detached, printer, tightened and replaced*, and for each one, we selected the 10 most similar terms in the embedding space using the cosine similarity. Then, we created the graph whose edges correspond to pairs of words that have cosine similarity greater than 0.6 (i.e. their vector representation). Interestingly, the graph consists of 3 connected components (and not 4). *Birdstrike's and printer's* neighborhoods consist of words that are related to their context, such as impact, blood, feather and paper, printing along with many misstyped version of these words. *Tightened's and detached's* neighborhoods exhibit the same behaviour, however, they are connected by edges that connect nodes such as *fallen-resecured and tightened-lose* and thus, the corresponding context is captured by the graph and the underlying embeddings. Such graphs were the key to observe this information which was captured in the embedding space and lead to the development of our method.

Finally, althought work has been done to combine information about topics and word embeddings Liu et al. (2015), to our knowledge, our approach is the first one to perform text cleaning using contextual information that comes from the embeddings.

Let $M\in\mathbb{R}^{m\times d}$ the embedding vectors in $\mathbb{R}^d$ of each word. It has been recently shown Choi et al. (2017) Grbovic et al. (2015)

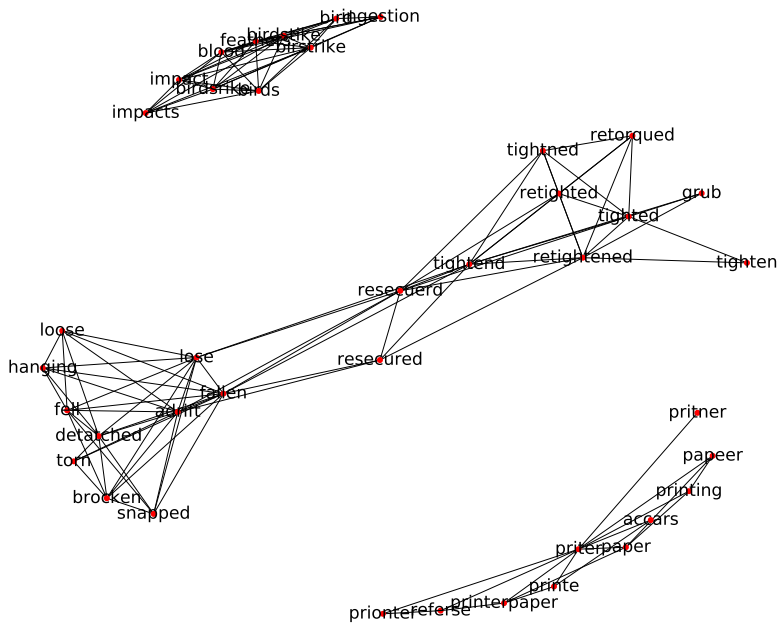


Figure 5.5: A graph created from the nearest neighbors in embedding space for 4 terms: 'detached','printer','tightened','birdstrike'

Sordoni et al. (2015) that the similarity (e.g. cosine) of words in the vector representation is related to semantic similarity. Motivated by this, we explored the euclidean neighborhood in this space of important words such as "*replaced*", "*tightened*", and we discovered that it contains three different kinds of neighboring words:

- Typos/abbreviations
- Words of the same root and synonyms
- Words that belong to the same topic/context

In figure 5.6, we present some examples of the neighborhood of 4 words in the embedding space. The neighborhood is calculated with the cosine similarity,

$$cos(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{5.3}$$

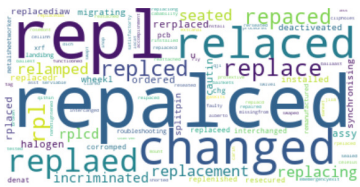
which is the most popular similarity metric used in comparing word embeddings.

5.5 LOGBOOK CLEANING USING WORD EMBEDDINGS

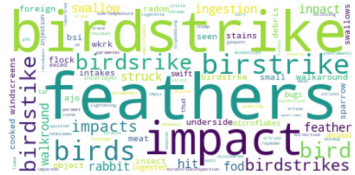
Based on the observations presented in the previous section, we start by splitting the vocabulary into three subsets:

- Proper words $\mathcal{W}_p$: Words that are correctly spelled.
- Serial numbers $\mathcal{W}_s$: Words consisting of characters and numbers and are unique for aircraft parts
- Misspelled words $\mathcal{W}_m$: Words that are typos

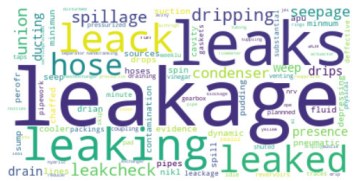
Note that $\mathcal{W}_p \cup \mathcal{W}_s \cup \mathcal{W}_m = \mathcal{W}$ and $\mathcal{W}_p \cap \mathcal{W}_p \cap \mathcal{W}_m = \emptyset$, that is, $\mathcal{W}_p$, $\mathcal{W}_s$ and $\mathcal{W}_m$ are proper subsets of $\mathcal{W}$. Algorithm 1 divides the vocabulary into these three subsets. The input consists of the vocabulary and a set $\mathcal{T}$, which is a thesaurus of the language in which the logbooks are written. The thesaurus is a set containing all the words of the corresponding language. Furthermore, the condition in line three, where we check is a serial number, can be re-written as a regular expression that matches the serial numbers or even more simply, by just checking if a word contains both numbers and letters (although in some cases this could be a spelling error).



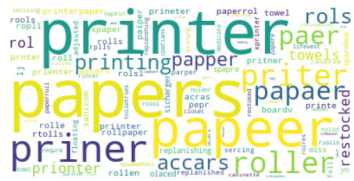
(a) term: *replaced*



(b) term: *blood*

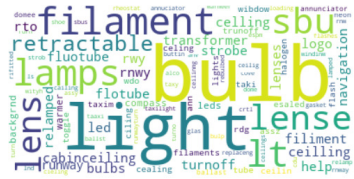


(c) term: *leak*



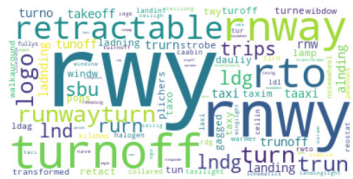
(d) term: *paper*

(e) term: *oil*



(f) term: *lamp*

(g) term: *coffee*



(h) term: *runway*

Figure 5.6: Wordclouds of the 40-neighborhood of terms in the embedding space

5.5.1 *Mapping spelling errors to correct words*

In our approach we create a function (or mapping) $f : \mathcal{W}_m \mapsto \mathcal{W}_p$ that maps each misspelled word or abbreviation to a proper one

Algorithm 1 VocabularySplit

Input: $\mathcal{W}, \mathcal{T}$
Returns: $\mathcal{W}_p, \mathcal{W}_s, \mathcal{W}_m$
$\mathcal{W}_p, \mathcal{W}_s, \mathcal{W}_m \leftarrow \emptyset$
for $w \in \mathcal{W}$ **do**
    if w is serial number **then**
        $\mathcal{W}_s \leftarrow \mathcal{W}_s \cup w$
    else if $w \cap \mathcal{T} \neq \emptyset$ **then**
        $\mathcal{W}_p \leftarrow \mathcal{W}_p \cup w$
    else
        $\mathcal{W}_m \leftarrow \mathcal{W}_m \cup w$
    end if
end for

using their vector representations. By using this approach we assume that every misspelled word has been correctly spelled in another part of the logbook but this assumption is correct for the vast majority of the cases.

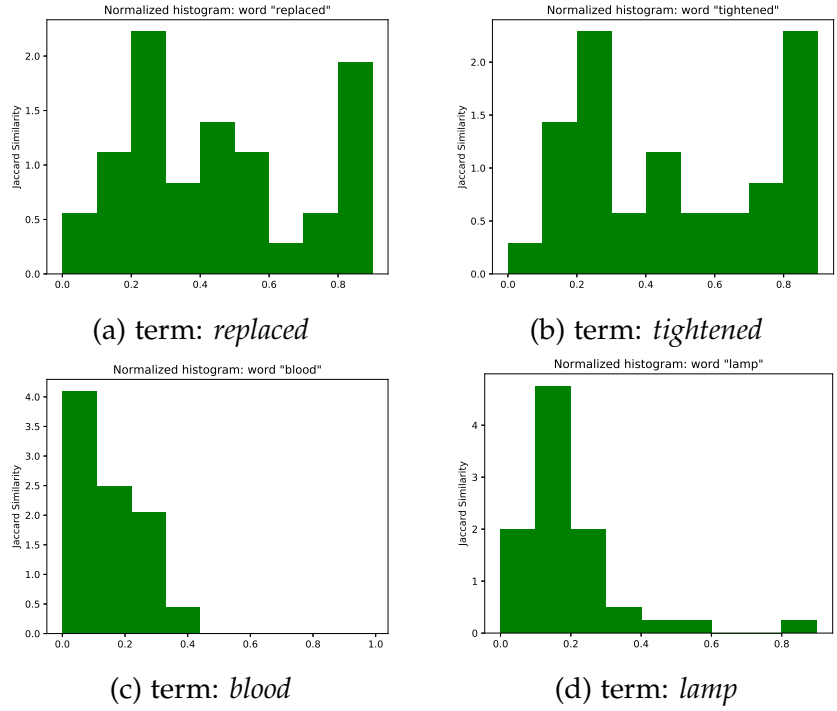
We define $\mathcal{G}(w_i, k)$, the k-neighborhood of a word w_i in the embedding space as the k most similar words given a distance/similarity measure. In the following sections, we will omit any details related to the creation of the vector representations and we will use the notation $\mathcal{G}^{\mathcal{L}}(w_i, k)$, to state that the neighborhood is calculated based on the vectors learned using the logbook $\mathcal{L}$.

The neighborhood of a word in the embedding space is not enough to provide accurate mappings, because its closest neighbors in the representation space are not only misspelled occurrences of itself. Thus, assigning all these neighbors to each word will lead to multiple false matches (see Figure 5.6 where the neighborhood of the words don't consist only of spelling mistakes). We therefore take into account the actual characters of which the

word is composed by using the Jaccard similarity Cohen et al. (2003),

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.4}$$

which is the number of common letters in strings A and B, divided by the total number of unique letters of both these strings.



(a) term: *replaced*

(b) term: *tightened*

(c) term: *blood*

(d) term: *lamp*

Figure 5.7: Empirical distribution of Jaccard similarities

In Figure 5.7 we present the empirical distribution of the Jaccard similarity of the neighborhood for 4 terms. The neighborhoods of the terms *replaced* and *tightened* contain several typos (see Figure 5.6) and thus, the distribution consists of two components, one with high expected value that corresponds to the misspelled words and one that correspond to contextually similar

ones. On the other hand, since the terms *blood* and *lamp* are not frequently misspelled, their neighborhood consists of mostly contextually similar terms and the distribution is composed by only a singe component.

Given a proper word $\mathbf{w} \in \mathcal{W}_p$, we retrieve the misspelled words in it's neighborhood, $w \in \mathcal{G}(\mathbf{w}, k) \cap \mathcal{W}_m$, using their Jaccard similarities. We employ a simple yet effective threshold-based approach to decide whether each non-proper word in the neighborhood is a typo by comparing the Jaccard similarity with the threshold. Given a proper word w, the set of misstyped words that will be match to it is

$$\{v | v \in \mathcal{G}(\mathbf{w}, k) \cap \mathcal{W}_m \; if \; J(\mathbf{w}, v) > \gamma\}. \tag{5.5}$$

Algorithm 2 presents *CorrectWords*, the process of mapping misspelled words to proper ones.

Algorithm 2 CorrectWords

Input: $\mathcal{L}$, γ, $\mathcal{G}$
Returns: $\mathcal{L}_c$

$\mathcal{W}_p, \mathcal{W}_m, \mathcal{W}_m \leftarrow VocabularySplit(\mathcal{W})$
$\mathcal{L}_c \leftarrow \mathcal{L}$
for $\boldsymbol{w}_p \in \mathcal{W}_p$ **do**
    $g \leftarrow \mathcal{G}(\boldsymbol{w}, k)$ # Find the nearest neighbors
    for $\boldsymbol{w}_m \in g \cap W_m$ **do**
        if $J(\boldsymbol{w}_m, \boldsymbol{w}_p) < \gamma$ **then**
            $\boldsymbol{w}_m \leftarrow \boldsymbol{w}_p$# replace the misspelled word $\boldsymbol{w}_m$
            $\mathcal{W}_m = \mathcal{W}_m - \{v\}$
            $\mathcal{L}_c \leftarrow \mathcal{L}_c(\boldsymbol{w}_m \leftarrow \boldsymbol{w}_p)$
        end if
    end for
end for

The operation $\mathcal{L}_c(\boldsymbol{w}_m \leftarrow \boldsymbol{w}_p)$ in line 9 of algorithm 2 means that all occurrences of $\boldsymbol{w}_m$ in $\mathcal{L}$ are replaced with the word $\boldsymbol{w}_p$.

Finally, in Figure 5.8 we present the new neighborhoods of the same words presented in 5.6. The new WordClouds contain more information about their context, since the mistyped terms have been successfully replaced.

5.5.2 *Method Summary*

Finally, the result of algorithm 2 is the cleaned logbook. However, we observed that not all misspelled words are being mapped to proper ones and therefore, we investigated the application of our method iteratively. At each iteration, we clean the logbook and we re-train the neural network, which in turn leads to new word embeddings. Algorithm 3 presents the full approach.

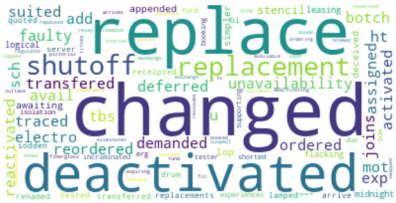
Algorithm 3 CleanLogbookIter

Input: $\mathcal{L}$, n_{iter}, γ
Returns: $mathcal L_{n_{iter}}$
1: **for** $i \in [1, \ldots, n_{iter}]$ **do**
2: $\quad \mathcal{G} \leftarrow TrainModel(\mathcal{L})$
3: $\quad \mathcal{L} \leftarrow CorrectWords(\mathcal{L}, \gamma, \mathcal{G})$
4: **end for**

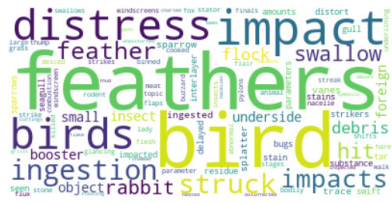
At every step of the iteration, the vocabulary size is reduced since the set $\mathcal{W}_m$ decreases. Intuitively, at every iteration the neighborhood of each word contains more *semantically* similar ones, since the mispelled words are removed. Finally, in Table 5.2 we show examples of words mapped at the first iteration steps.

	iteration 1	iteration 2	iteration 3
replaced	'replced', 'repled', 'relaced', 'rplaced', 'relpaced', 'replacedi', 'repalced', 'rplcd', 'replacded', 'replaed', 'replacecd', 'reolaced', 'replaqced', 'replacde', 'replacediaw'	'repleced', 'desplaced', 'replaceed', 'replpaced'	'replacedv', 'teplaced'
tightened	'retighned', 'tighteded', 'retightened', 'tightned', 'tightenedchecked', 'tigten', 'retighted', 'retighrnrd', 'tighttened', 'tighened', 'retightend', 'tighteened', 'tightend', 'tigtened', 'tightewed', 'tightnrd', 'tighted', 'retightned'	'retighed', 'thightened', 'thigthened'	'tighetened', 'tichtened'
replacement	'replaement', 'replacenent', 'replacemnt', 'remplacement', 'repalcement'	'repacement','replacemant', 'replaacement'	'recplacement', 'replamcement'
leakage	'leack', 'lek'		
paper	'paer', 'xpaper', 'paiper', 'paperv', 'papere', 'papper', 'ppaper', 'papeer', 'papaer'	'paperl', 'papar', 'pepr', 'parper'	'opaper'
repaired	'reapired', 'repaied', 'repaireref', 'reaired', 'reppaired', 'repaider', 'rpaired'	'repais', 'repait'	'repairede', 'repiared'

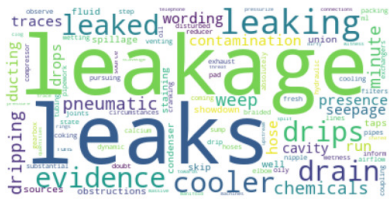
Table 5.2: Iterative correction of spelling mistakes



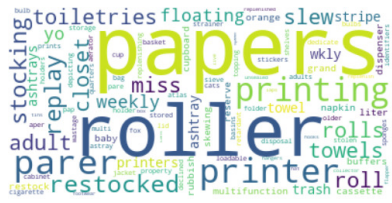
(a) term: *replaced*



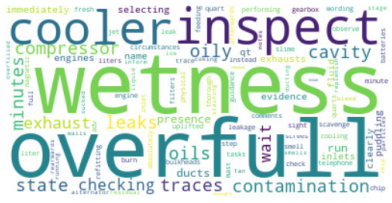
(b) term: *blood*



(c) term: *leak*



(d) term: *paper*



(e) term: *oil*

(f) term: *lamp*

(g) term: *coffee*

(h) term: *runway*

Figure 5.8: Wordclouds of the 40-neighborhood of terms in the embedding space created using the cleaned logbook

5.6 INFORMATION EXTRACTION

Part of speech tagging Brill (1992) Toutanova et al. (2003) is the task of identifying the part of speech (noun, verb, etc.) of each

word in a document. It is an important NLP task and it is used in keyword and information extraction Liu et al. (2009) Califf and Mooney (1999).

In our work, we are interested in extracting the actions which are expressed with verbs in the corresponding logbook entries, such as replace, fix, tight and clean. We use the standard POS tagger of the very well know library NLTK[7]. Figure 5.9 shows some extracted verbs.

Figure 5.9: Maintenance keywords extracted from the logbook - verbs via POS Tagging. The size of each word is proportional to its number of occurrences in the logbook

Our motivation for extracting the verbs is twofold: First, it is important to identify which actions were performed on specific components in order to use this information for failure prediction. Second, it would be of great benefit to study the impact of each action on the lifetime of the corresponding component.

[7] http://www.nltk.org

5.7 CONCLUSIONS AND FUTURE WORK

In this chapter we presented our approach for pre-processing and information extraction from logbook data. We used vector representations in order to capture contextual information and clean the logbook. Furthermore, we used standard Part-of-Speech tagging in order to extract the maintenance actions (verbs) and the objects (nouns) from the cleaned text.

6

COMPONENT CONDITION ASSESSMENT USING TIME SERIES DATA

Time series generated by sensors on-board constitute the core of the data generated by aircraft. The event logs presented in the previous chapters are generated by sampling and analyzing sensor values. One the other hand, since sensor measurements directly reflect the health of every component, they are the most complete source of information. Therefore, in predictive maintenance, if one cannot achieve the desired results using other sources of information, the analysis of sensor time series becomes indispensable.

In this Chapter we present a method for assessing the condition of equipment on-board, using time series generated by sensors that monitor their health. We formulate and solve a time series decomposition problem in order to extract the trend and the fluctuation of each time series, and along with statistical properties of the samples, we quantify the severity of the fault. Finally, our method doesn't require any prior knowledge and our results demonstrate the effectiveness of our approach.

6.1 DEGRADATION

Hardware is not imperishable and may fade through time and thus, monitoring its condition is very important. In this section

we present our work on faulty hardware detection and risk assignment. More specifically, given a collection of sensor time series that monitor the condition of several components from multiple aircraft, we aim to detect when a sensor starts emitting abnormal values and we aim to estimate a function that quantifies the degradation level.

The sensors under consideration have a specific pattern when the corresponding component is fading: the emitting values show gradually increasing behavior with increased variance. Figure 6.1 shows an example.

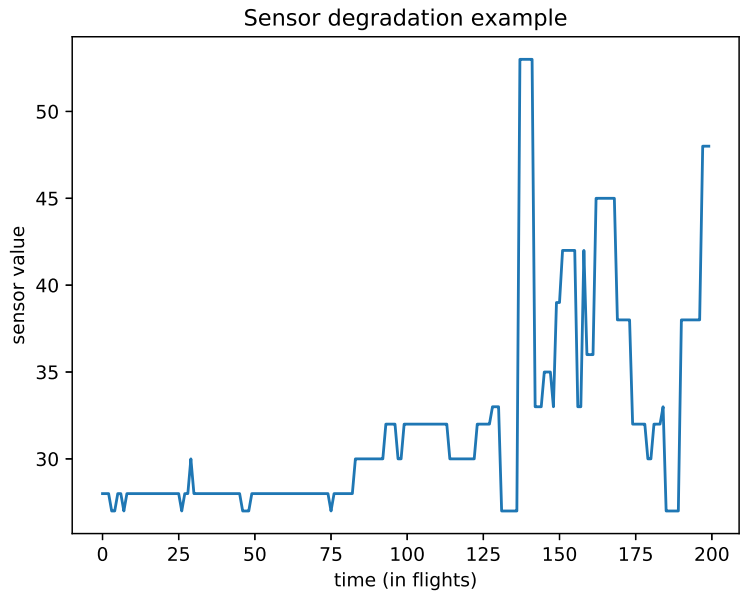


Figure 6.1: Example of degrading sensor

In this example the component functions normally for approximately the first 90 flights until it starts fluctuating. Separating the fluctuation from the *increasing* trend is the key concept of our approach, since the degradation level is a function of those two:

the sensors condition is reflected by the deviation of the trend from the normal value and the variance of the fluctuation.

6.2 RELATED WORK

Trend extraction Alexandrov et al. (2012) is a well studied problem in time series analysis. Aiming towards an effective solution, a plethora of methods have been proposed due to the fact that there is no standard definition for the trend Wu et al. (2007) and thus, each approach is driven by a specific application. However, most of the existing work does not tackle the problem under the monotonicity constraint that is critical for our application.

Two simple yet sometimes effective approaches are Low-pass filtering Harvey and Trimbur (2003) and linear regression Weisberg (2005). Using a low pass filter can remove the fluctuation however, it is not guaranteed that the trend will be monotonic. On the other hand, fitting a line on the data will lead to a monotonic but linear trend. Unfortunately, in most applications including ours, the trend is not linear.

Another two popular approaches are the Singular Spectrum Analysis (SSA) Elsner and Tsonis (2013) and the Empirical Mode Decomposition (EMD) Mhamdi et al. (2011) Flandrin et al. (2004). SSA is based on Singular Value Decomposition of the *traversal matrix*[1] of the time series and EMD is a method that decomposes the time series based on oscillatory components. However, both of them do not guarantee a monotonic trend. Similarly, since trend extraction is a special case of time series decomposition,

[1] it is the Hankel Iokhvidov (1982) matrix whose rows are segments created by a sliding window over the tme series

Independent Component Analysis (ICA) Comon (1994) can be also used when monotonicity is not a constraint.

More recently, Gaussian Process (GP) Regression using monotonicity or shape constraints has been proposed and could be employed for trend extraction (Wang and Berger (2016) Golchi et al. (2015)). In these approaches the monotonicity is achieved by introducing virtual derivative points (with positive values for increasing behavior) in order to enforce the desired behavior. Although they have not been studied for the problem of trend extraction, we investigated their capability and we experimented with the one presented in Riihimaki and Vehtari (2010) as a baseline for comparison. A major drawback of GP approach is that the values of the derivatives must be manually selected whereas automatic approaches have been proposed for the positions of the points. However although monotonicity (increasing) becomes likely by adding the virtual points iteratively in positions where the derivative is expected to be negative, it is not enforced.

Finally, we build our approach upon an existing one that was proposed for a similar problem, in the context of sensor degradation analysis. In this approach Ulanova et al. (2015a), the authors formulate an optimization problem to separate the monotonic trend from the fluctuation. However, we deviate from the original one by introducing a regularization term in the objective function that guarantees that the trend will be close to the original time series and by modifying the constraint in order to allow negative values for the fluctuation.

Our approach is the only one to fulfill all the constraints for the problem in hand with respect to the trend: a) guaranteed

monononically increasing, b) non-linear and c) *close*[2] to the original time series.

6.3 DATASET

Our dataset consists of measurements coming from 23 aircraft over a few hundreds of flights (the number of flights per aircraft varies). We targeted 32 sensors of interest that exhibit the aforementioned degrading behavior. Therefore, we form 23 matrices, $X^i, i = 1, \dots 23$, one for each aircraft. All matrices have 32 rows that correspond to the 32 sensors, and the number of columns vary according to the number of flights available for each aircraft. Note that each sensor is regularly sampled within each flight, however, here we analyze the mean value per flight because it is capable of capturing the type of degradation that we aim to study.

6.4 MODELING DEGRADATION WITH GMMS

We assume that each sample is drawn from a mixture of two distributions, that correspond to the condition of the sensor. Specifically, one mixture component that corresponds to the normal condition of the sensor and another one that corresponds to the degraded condition. Since our dataset is not annotated and we don't know in advance which samples correspond to normal and degraded condition of the component, we treat each

[2] closeness in enforced by penalizing the trend's distance from the original time series. This will be presented in section 6.5.2

sample as a random variable that comes from a Gaussian Mixture Model (GMM) of two components:

$$P(x) = \theta_1 p(x|k=1) + \theta_2 p(x|k=2) \tag{6.1}$$

Given a history of sensor measurements, we fit a GMM of two components, aiming to capture the normal and the abnormal condition of the sensor. In Figure 6.2 we present an example mixture that was estimated on a specific sensor. After learning the mixture using the standard Expectation Maximization (EM) algorithm, we label the mixture component having the lowest mean as the *normal* and the other as the *degraded*.

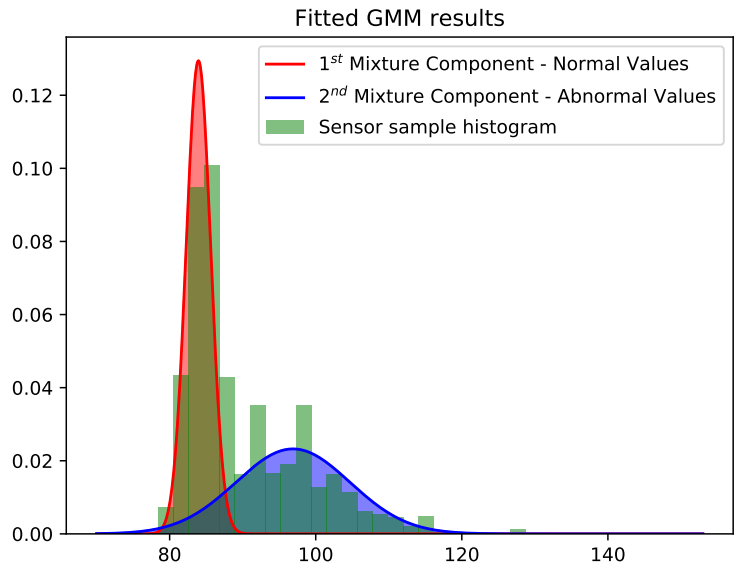


Figure 6.2: Example of a Gaussian Mixture Model fitted on a specific sensor's samples

We fit one GMM for each sensor because our goal is to be able to assess the condition of each sensor without having necessarily

access to the others. Figure 6.3 shows more examples of this approach.

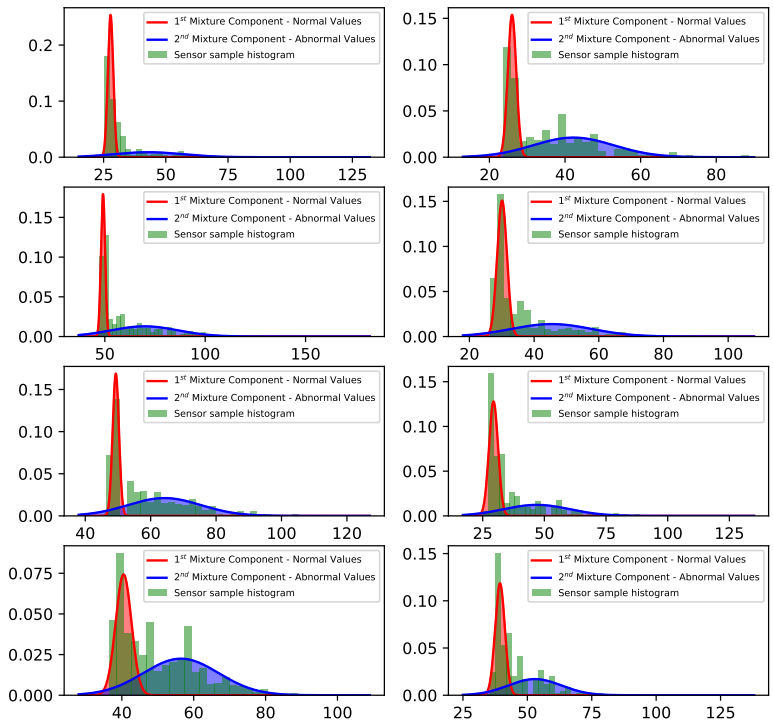


Figure 6.3: GMM fitted on 8 different sensors.

Furthermore, the mixture model can be used in order to label each sample according to the probability of it being generated by each one of the two components:

$$c(x_i) = \underset{j}{\operatorname{argmax}} \ \theta_j p(x_i|k=j). \tag{6.2}$$

The information of the learned distributions will be used at the final step of the risk assessment. In the next session we proceed with the description of the main component of our approach and finally, we put everything together in section 6.5.

6.5 TIME SERIES DECOMPOSITION

We propose an approach that is based on the decomposition of each time series into the two aforementioned components, the fluctuation and the trend Alexandrov et al. (2012) (Figure 6.4).

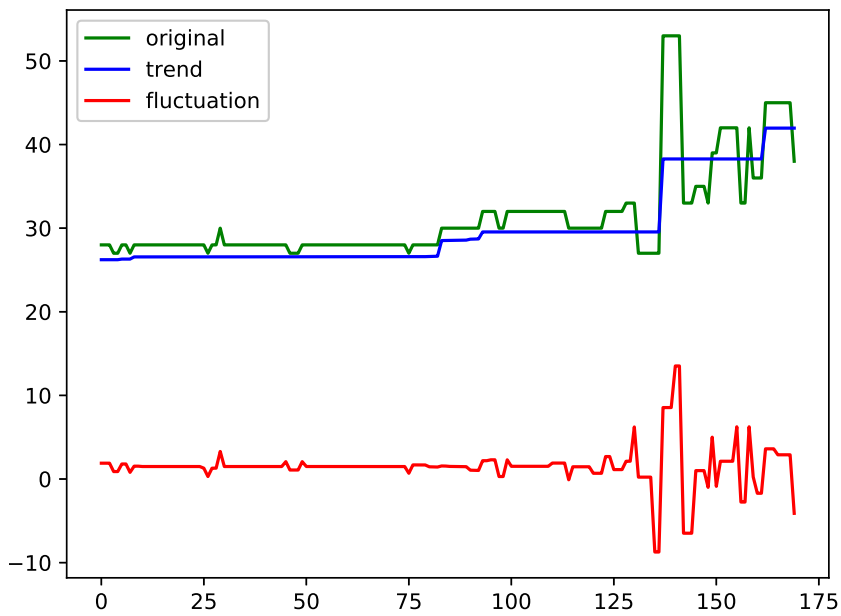


Figure 6.4: Decomposing the time series into the fluctuation and the trend

For a time series $s \in R^{n\times 1}$, finding the trend $u \in R^{n\times 1}$ and the fluctuation $v \in R^{n\times 1}$, so that $s = u + v$ (which is the common formulation Alexandrov et al. (2012)), can be translated into an optimization problem having the following structure:

$$\begin{aligned} \min_{u,v} \quad & \|s - u - v\|^2 \\ \text{s.t.} \quad & u \text{ is non-decreasing} \end{aligned} \tag{6.3}$$

A solution to this problem has already been recently provided by Ulanova et al. (2015b), in which the authors formulate an non-negative quadratic problem. However, we modified this algorithm in order to better match our requirements, by solving a convex, quadratic programming problem instead. Our main deviation from their work is the addition of a regularization term which, although deprives[3] the formulation of a non-negative quadratic problem, gives more accurate and robust estimation of the trend component.

6.5.1 *Quadratic Programming Formulation*

We start by presenting and discussing the solution proposed by the authors in Ulanova et al. (2015b). Introducing two specifications for the fluctuation, being wide sense stationary and non-negative, the optimization problem in 6.3 can be written as follows:

[3] The multiplicative updates for solving a non-negative convex quadratic problem lead to faster convergence Sha et al. (2003, 2007)

$$\begin{aligned} \min \quad & ||s-u-v||^2 + \sum (m_i - m_j) W_{ij} \\ \text{s.t.} \quad & Bu > 0 \\ & v > 0 \end{aligned} \tag{6.4}$$

where,

$$B = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \dots & \vdots \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix} \in \{0,1\}^{n-1 \times n}$$

Regarding the constraints, the product Bu produces the pairwise differences between every pair of points in u in a sliding window, and thus, the constraint $Bu > 0$ guarantees that the trend u will be increasing. Regarding the objective function, $||s-u-v||^2$ is the reconstruction error and $\sum_{i,j=1}^{K} (m_i - m_j) W_{ij}$, is the term that achieves the stationarity. More precisely, the time series is partitioned into K segments having mean values $m_1, m_2, \dots, m_K$ and thus, minimizing the weighted difference between them enforces stationarity. The weight matrix $W \in R^{K \times K}$ can be used to configure the importance of the differences between the segments. It can be simply set to all ones, or if one needs to make sure that the mean will not change as the distance between the segments increases, it can be set as $W_{ij} = |i-j|$.

However, the problem presented in 6.4 is not in a solvable form. In order to achieve it, we introduce:

- $x = [u; v] \in R^{2n \times 1}$, a vector that is the concatenation of the trend and the fluctuation.

- $E_2 = [I\ I] \in R^{n \times 2n}$, where $I \in R^{n \times n}$ is the identity matrix
- $E_1 = [\mathbf{0}_{n \times n}\ I]$, where $\mathbf{0}_{n \times n}$ is the $n \times n$ all zeros matrix
- $C = [B\ \mathbf{0}_{n-1 \times n-1}] \in R^{n-1 \times 2n-1}$
- $L = D - W$, where $D = diag(W \cdot \mathbf{1}_{K \times K})$, $\mathbf{1}_{K \times K}$ is the all 1s $K \times K$ matrix.

Additionally,

$$E = \begin{bmatrix} 1/l & \overbrace{\dots}^{l} & 1/l & 0 & \dots & 0 & 0 & \cdots & 0 \\ 0 & \dots & 0 & \underbrace{1/l & \cdots & 1/l}_{l} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \dots & \vdots & \overbrace{\vdots}^{l} & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \underbrace{1/l & \dots & 1/l}_{l} \end{bmatrix},$$

where l is the segment length after partitioning the time series into K segments ($l = n/K$). Having the definitions above, we can now start building the final formulation of the optimization problem.

The term $\sum_{i,j=1}^{K} (m_i - m_j) W_{ij}$ can be rewritten as $(E_1 x)^T \left(E^T L E\right) (E_1 x)$, because $E_1 x = v$ and $Ev = \mathbf{m}$. Next, the reconstruction error $||s - u - v||^2$ can be written as:

$$\begin{aligned}\|s-u-v\|^2 &= \left\| s - \overbrace{(E_2 x)}^{=u+v} \right\|^2 \\ &= (s-E_2x)^T(s-E_2x) \\ &= s^Ts - 2\left(E_2^Ts\right)x + x^TE_2^TE_2x\end{aligned} \tag{6.5}$$

From 6.5 and REF, the final form of the objective function is

$$\begin{aligned}&\min \|s-u-v\|^2 + \sum (m_i - m_j)W_{ij} \\ &\quad = \min\ s^Ts - 2\left(E_2^Ts\right)x + x^TE_2^TE_2x + x^TE_1^T\left(E^TLE\right)(E_1x) \\ &\quad = \min\ -2\left(E_2^Ts\right)x + x^TE_2^TE_2x + x^TE_1^T\left(E^TLE\right)(E_1x) \\ &\quad = \min\ x^T\left(E_2^TE_2 + E_1^T\left(E^TLE\right)E_1\right)x - 2\left(E_2^Ts\right)x\end{aligned} \tag{6.6}$$

Finally, the new constraints are $Cx > 0$ and $x > 0$ and therefore, the final formulation is

$$\begin{aligned}\min\quad & x^T\left(E_2^TE_2 + E_1^T\left(E^TLE\right)E_1\right)x - 2\left(E_2^Ts\right)x \\ \text{s.t.}\quad & Cx > 0 \\ & x > 0\end{aligned} \tag{6.7}$$

The optimization problem 6.7 is convex. A quadratic optimization problem of the form $\min x^TQx + w^Tx$ is convex if Q is positive semidefinite. This holds in 6.7 because $E_2^TE_2 + E_1^T\left(E^TLE\right)E_1$ is a sum of positive semidefinite matrices.

6.5.2 *Reformulating the Optimization Problem*

Our approach differs from the one presented in Ulanova et al. (2015b) in two ascpects: a) we introduce the extra penalization term and b) we allow negative values for the noise. Although our approach loses in efficiency compared to the non-negative quadratic programming, we achieve better fit of the trend on (Figure 6.5), which is critical for our application.

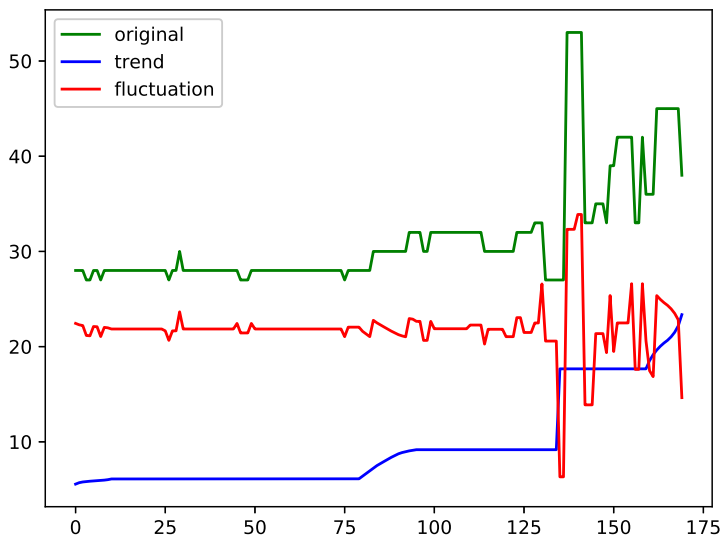


Figure 6.5: Trend captured by solving the original QP problem

We add a new term on the objective function, in order to deal with the issue depicted in 6.5. The term is $\|s - u\|^2$. In order to properly add the term in the objective function, we define a matrix $E_3 = [\mathbf{I}_{n\times n}\ \mathbf{0}_{n\times n}]$ so that we can express the regularization term as follows:

$$\begin{aligned} \|s-u\|^2 &= \left\| s - \overbrace{(E_3 x)}^{=u} \right\|^2 \\ &= (s-E_3x)^T(s-E_3x) \\ &= s^Ts - 2\left(E_3^Ts\right)x + x^TE_3^TE_3x \end{aligned} \tag{6.8}$$

Finally, by embedding 6.8 (by omitting s^Ts) and remove the constraint $x > 0$ we obtain our final form of the convex, quadratic problem:

$$\begin{aligned} \min \quad & x^T\left(E_2^TE_2 + E_1^T\left(E^TLE\right)E_1 + 2E_3E_3^T\right)x - 2\left(E_2^Ts + 2E_3^Ts\right)x \\ \text{s.t.} \quad & Cx > 0 \end{aligned} \tag{6.9}$$

Problem 6.9 is still convex since the matrix $E_3E_3^T$ is positive semidefinite.

6.6 CONDITION ASSESSMENT

Having decomposed the signal into the fluctuation and the trend, we now use this information to compute an indicator that reflects the condition of the component. This indicator should be interpreted as a risk function, bounded in the interval $[0, 1]$, as a scaled function of the trend[4]. Values of the function close to zero correspond to properly functioning components, whereas higher values quantify the level of degradation.

At timestep t, given a sample our formula is

[4]This decision was made according to the requirements of the application

$$r^i(t) = \begin{cases} 0 & if\ u_t \leq \mathbb{E}_{X|k=0;\mu_0,\sigma_0}[X] \\ \int_{-\infty}^{u_t} P(x|k=1;\mu_1,\sigma_1)dx & otherwise \end{cases} \tag{6.10}$$

where

- u_t is the trend extracted at timestep t
- $\mathbb{E}_{X|k=0;\mu_0,\sigma_0}[X]$ is the expected value of the samples in normal condition
- $\int_{-\infty}^{u_t} P(x|k=1;\mu_1,\sigma_1)dx$ is the cumulative density of the mixture that corresponds to the degrading state

The proposed function $r^i(t)$ is zero when the trend extracted at timestep t is lower that the mean value of the PDF that corresponds to the normal condition. Furthermore, using the cdf in order to quantify the risk fulfills the major requirements of the risk function: a) it is increasing, it is bounded within $[0-1]$ and c) it is compatible with the distribution of the abnormal values, i.e. it is maximized when the trend reaches values that are too high (rightmost part of the corresponding pdf).

In Figure 6.6 we demonstrate our risk function for several sensors.

6.7 EVALUATION

We evaluate our approach using synthetic data since for our sensor data there is no ground truth for the trend. Furthremore, we compare against the Gaussian Process (GP) approach presented

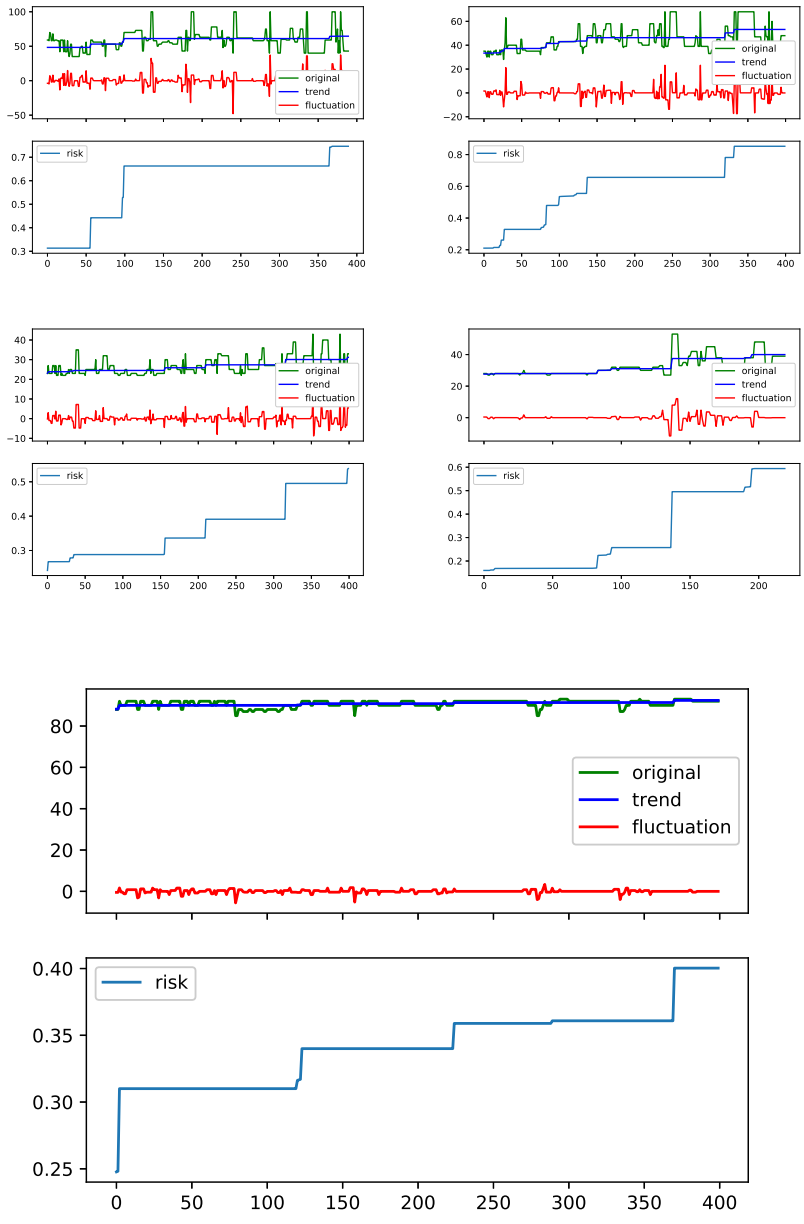
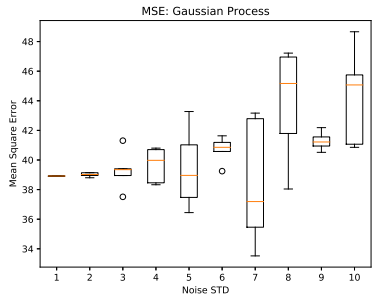
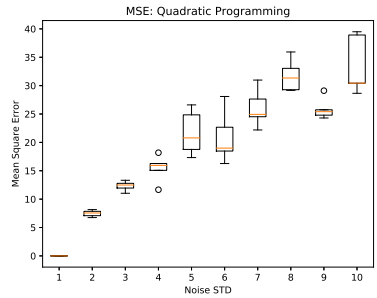


Figure 6.6: Sensor risk examples

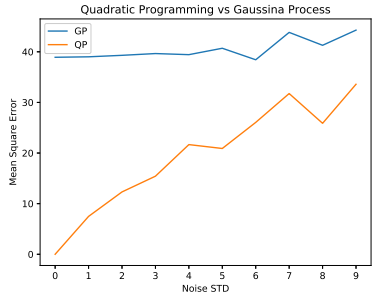
in Riihimaki and Vehtari (2010) both in the effectiveness and efficiency. We created three different types of synthetic data which consist of a monotonically increasing trend and additive random noise in order to emulate the fluctuation. Our experiments are performed on several values of σ in order to simulate different magnitudes of the fluctuation.



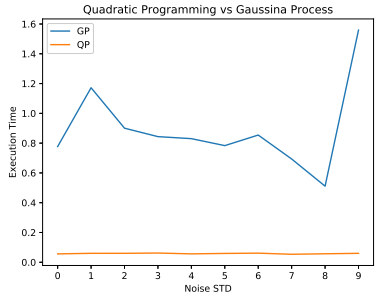
(a) Gaussian Process MSE



(b) Quadratic Programming MSE



(c) QP vs GP: MSE

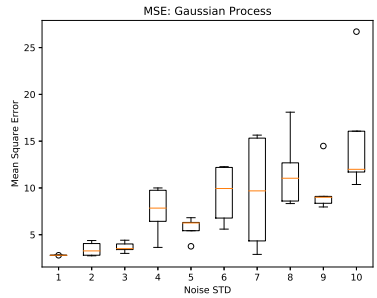


(d) QP vs GP: Execution Time

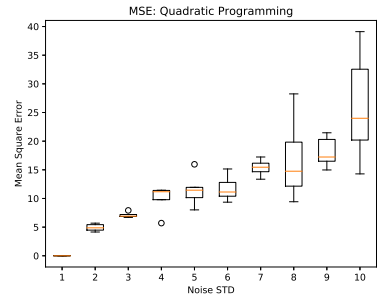
Figure 6.7: Exponential Trend

The first model is an exponential trend with additive noise, $y(t) = e^{\alpha t} + n_t$. In Figure 6.7 we present: a) on the top left the mean squared error of the trend estimation using the GP approximation, b) on the top right the mean squared error of the trend estimation using our approach, c) on the bottom right

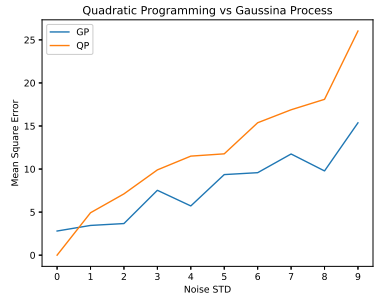
the average error for both approaches in order to make a clear comparison and d) the execution time of both approaches.



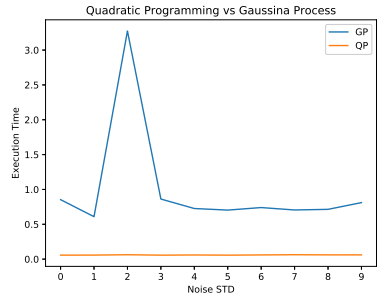
(a) Gaussian Process MSE



(b) Quadratic Programming MSE



(c) QP vs GP: MSE

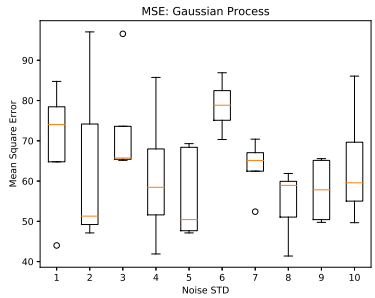


(d) QP vs GP: Execution Time

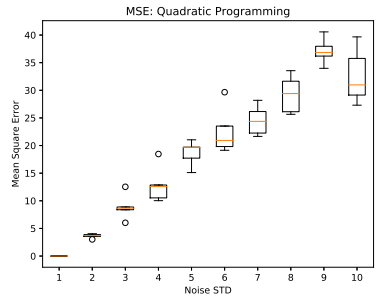
Figure 6.8: Linear Trend

The second model is a linear model $y(t) = \alpha t + n_t$. In Figure 6.8 we present the same types of results. This is the only case where the Gaussian Process outperforms our approach in our synthetic datasets.

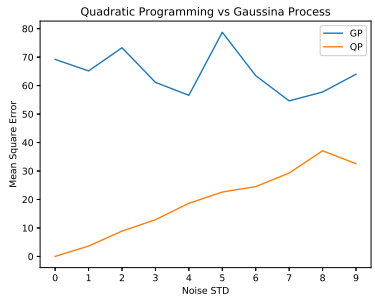
Finally, the third model consists of a step-wise increase with additive noise. In Figure 6.9 we present the results.



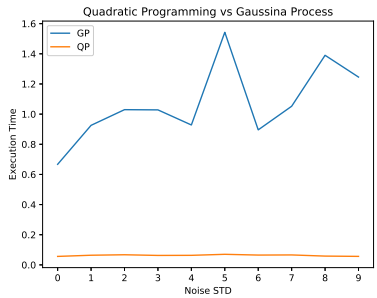
(a) Gaussian Process MSE



(b) Quadratic Programming MSE



(c) QP vs GP: MSE



(d) QP vs GP: Execution Time

Figure 6.9: Step-wise Trend

6.7.1 *Discussion*

In most of the cases, the Quadratic Programming approach outperforms the Gaussian Process both in efficiency and in effectiveness. Moreover, we should note that the execution time of the Gaussian Process is affected by the number of derivative points that have to be added in order to achieve monotonicity. In most cases, this is related to the amplitude of the fluctuation.

6.8 CONCLUSIONS

In this chapter we presented our approach for assessing the condition of components on-board using sensor time series. We used Gaussian Mixture Models in order to identify abnormal values and we formulated a quadratic programming approach in order to extract the trend of the time series and separate it from the fluctuation. This approach was driven by the specific, monotonically increasing degradation pattern of the time series in hand. Finally, our method outputs the risk which is computed as the trend's deviation from the expected normal values, using the information from the Mixture Models.

7

DISCUSSION

The goal of this thesis was to propose Machine Learning approaches for solving problems in the aviation industry, in the context of predictive maintenance. Our work was performed under the regime of a CIFRE Ph.D. and consequently, both our objectives and the evaluation methods were derived from an industrial perspective. We targeted problems that have high impact on the industry and our solutions are completely data driven, using minimum intervention and prior knowledge from maintenance experts.

7.1 SUMMARY OF CONTRIBUTIONS

The main contributions of the thesis can be summarized as follows.

- In Chapter 3 we presented some exploratory analysis outcomes using survival analysis techniques. The outcomes from such an analysis are useful when designing prediction experiments, by acquiring knowledge about the interval distribution of a target failure, by evaluating possible predictors and also, by studying the impact of variables on target failure's occurrence.
- In Chapter 4 we presented our approach on predicting upcoming failures, using only the history of failure logs.

To our knowledge, this is the first proposed solution that achieves results. The difficulty of this problem enforced a detailed design of all the methods steps. Finally, we showed the expected impact of our predictions on the industry.

- In Chapter 5 we studied the problem of cleaning and extracting information from logbook data. We proposed an approach which is based on recent advances in neural networks and we identified the potential of creating our own vector of the words in the logbook.

- In Chapter 6 we proposed an unsupervised method for analyzing the decaying behavior of sensors/equipment onboard and quantifying their level of degradation. Our approach is based on a time-series decomposition technique by formulating a Quadratic Programming problem.

A key point which should be highlighted is that this thesis constitutes an holistic approach, that investigated almost all the available data related to the operation of the aircraft in order to achieve the aforementioned results. Consecutively, we explored a wide range of methods related to all of the data types and the problems that we encountered and we obtained valuable knowledge regarding the potential of every approach, regardless of its effectiveness. In industry, this information is critical in order to a) avoid future (and potentially repeated) attempts with insignificant results and b) identify cases where methods will probably produce significant results.

7.2 FUTURE DIRECTIONS

During our work we identified several interesting outcomes which due to the limited duration of the thesis could not be further investigated.

- Inject data from multiple sources into our failure prediction approach. The design of our approach facilitates the utilization of any data in a per-flight manner. Such data could be measurements coming from sensors on-board or meteorological data related to the weather condition during the flight.
- Identify which variables contribute to the sensor degradation presented in Chapter 6. Having computed the risk, a regression problem could be formulated that aims to investigate the connection between several variables and the increase of risk.
- Identify the impact of maintenance actions on the survival function of target events. The keywords extracted from the logbook could be used as time-dependent covariates in survival analysis. This information can lead to assessing the effectiveness of maintenance actions.

Finally, the design of a data-centric architecture that supports Machine Learning applications using aircraft data is a potential future direction with great impact on the industry.

NOTATION

MACHINE LEARNING AND PROBABILITY

SYMBOL	MEANING
$p(\cdot)$	probability mass function
$S(t)$	survival function
$h(t)$	hazard function
$\mathbb{E}_P[X]$	expected value of X, w.r.t. distribution P
θ	model parameters
$\mathcal{L}()$	likelihood function
n	number of samples in the dataset
$X \in \mathbb{R}^{n \times d}$	feature matrix
$Y \in \mathbb{R}^{n \times 1}$	labels

SENSOR TIME SERIES

SYMBOL	MEANING
$s = [s_1, \ldots, s_t]$	sensor time series
$u = [u_1, \ldots, u_t]$	trend of time series
$v = [v_1, \ldots, v_t]$	fluctuation of time series
$r(\cdot)$	risk function

POST FLIGHT REPORT AND FAILURE PREDICTION

SYMBOL	MEANING
AC	set of aircraft
$\mathcal{E} = \{e_i, \ldots, e_k\}$	system failure ids
$e_{\mathcal{T}}$	critical (target) failure
$\mathcal{P}^{e_{\mathcal{T}}}$	predictors for $e_{\mathcal{T}}$
w	window of flights length
$T_{e_i}^{e_i}$	time interval between e_i and e_j
$r(\cdot)$	risk function
γ	risk decision threshold

LOGBOOK DATA ANALYSIS

SYMBOL	MEANING
$\mathcal{L} = \{lb_1, \ldots, lb_m\}$	set of logbook entries
$\mathcal{W} = \{w_1, \ldots, w_k\}$	set of unique terms in the logbook
$\mathcal{G}^L()$	skip-gram model trained in logbook $\mathcal{L}$
$\mathcal{G}^L(w, k)$	k-nearest neighbors in the embedding space created by $\mathcal{G}$ using $\mathcal{L}$
$J(w_i, w_j)$	Jaccard similarity of words w_i and w_j

ACRONYMS

MSE	Mean Square Error
MAE	Mean Absolute Error
GMM	Gaussian Mixture Model
EM	Expectation Maximization
GD	Gradient Descent
SGD	Stochastic Gradient Descent
PFR	Post Flight Report
QP	Quadratic Programming
KM	Kaplan-Meier
CPH	Cox Proportional Hazard
HMM	Hidden Markov Model
miNB	Multiple Instance Naive Bayes
SVM	Support Vector Machines
LSI	Latent Semantic Indexing
SVR	Support Vector Regression
MSN	Manufacturer Serial Number
MIL	Multiple Instance Learning

ROC	Receiver Operating Characteristic
ATA	Air Transport Association
AUC	Area Under the Curve
RFR	Random Forest Regression
PDF	Probability Density Function
CDF	Cumulative Distribution Function
NLP	Natural Language Processing
SSA	Singular Spectrum Analysis
EMD	Empirical Mode Decomposition
ICA	Independent Component Analysis
GP	Gaussian Process

BIBLIOGRAPHY

Adams, P. C., Speechley, M., and Kertesz, A. E. (1991). Long-term survival analysis in hereditary hemochromatosis. *Gastroenterology*, 101(2):368–372.

Alexandrov, T., Bianconcini, S., Dagum, E. B., Maass, P., and McElroy, T. S. (2012). A Review of Some Modern Approaches to the Problem of Trend Extraction. *Econometric Reviews*, 31(6):593–624.

Archer, K. J. and Kimes, R. V. (2008). Empirical characterization of random forest variable importance measures. *Computational Statistics & Data Analysis*, 52(4):2249–2260.

Barber, D. (2012). *Bayesian Reasoning and Machine Learning*. Cambridge University Press. Google-Books-ID: yxZtddB_OboC.

Binet, J. L., Auquier, A., Dighiero, G., Chastang, C., Piguet, H., Goasguen, J., Vaugier, G., Potron, G., Colona, P., Oberling, F., Thomas, M., Tchernia, G., Jacquillat, C., Boivin, P., Lesty, C., Duault, M. T., Monconduit, M., Belabbes, S., and Gremy, F. (1981). A new prognostic classification of chronic lymphocytic leukemia derived from a multivariate survival analysis. *Cancer*, 48(1):198–206.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer. Google-Books-ID: kTNoQgAACAAJ.

Bottou, L. (2010). Large-Scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Physica-Verlag HD. DOI: 10.1007/978-3-7908-2604-3_16.

Boyd, S. P. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press. Google-Books-ID: mYm0bLd3fcoC.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and Regression Trees*. Taylor & Francis.

Brill, E. (1992). A Simple Rule-based Part of Speech Tagger. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, pages 112–116, Stroudsburg, PA, USA. Association for Computational Linguistics.

Califf, M. E. and Mooney, R. J. (1999). Relational Learning of Pattern-match Rules for Information Extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, pages 328–334, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Caruana, R. and Niculescu-Mizil, A. (2006). An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 161–168, New York, NY, USA. ACM.

Chen, J., Li, K., Tang, Z., Bilal, K., Yu, S., Weng, C., and Li, K. (2017). A Parallel Random Forest Algorithm for Big Data in

a Spark Cloud Computing Environment. *IEEE Transactions on Parallel and Distributed Systems*, 28(4):919–933.

Choi, H., Cho, K., and Bengio, Y. (2017). Context-dependent word representation for neural machine translation. *Computer Speech & Language*, 45:149–160.

Cohen, W. W., Ravikumar, P., and Fienberg, S. E. (2003). A Comparison of String Distance Metrics for Name-matching Tasks. In *Proceedings of the 2003 International Conference on Information Integration on the Web*, IIWEB'03, pages 73–78, Acapulco, Mexico. AAAI Press.

Comon, P. (1994). Independent Component Analysis, a New Concept? *Signal Process.*, 36(3):287–314.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553.

Cox, D. R. and Oakes, D. (1984a). *Analysis of survival data*, volume 21. CRC Press.

Cox, D. R. and Oakes, D. (1984b). *Analysis of survival data*, volume 21. CRC Press.

Dahl, J. and Vandenberghe, L. (2006). Cvxopt: A python package for convex optimization. *Proc. eur. conf. op. res*.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Deleger, L., Grouin, C., and Zweigenbaum, P. (2010). Extracting medical information from narrative patient records: the case of medication-related information. *Journal of the American Medical Informatics Association : JAMIA*, 17(5):555–558.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Duan, K., Keerthi, S. S., and Poo, A. N. (2003). Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing*, 51:41–59.

Elsner, J. B. and Tsonis, A. A. (2013). *Singular Spectrum Analysis: A New Tool in Time Series Analysis*. Springer Science & Business Media. Google-Books-ID: 5sfSBwAAQBAJ.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Flandrin, P., Gonçalvès, P., and Rilling, G. (2004). Detrending and denoising with empirical mode decompositions. In *2004 12th European Signal Processing Conference*, pages 1581–1584.

Foulds, J. and Frank, E. (2010). A review of multi-instance learning assumptions. *The Knowledge Engineering Review*, 25(01):1–25.

Fu, Z., Robles-Kelly, A., and Zhou, J. (2011). Milis: Multiple instance learning with instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):958–977.

Golchi, S., Bingham, D., Chipman, H., and Campbell, D. (2015). Monotone Emulation of Computer Experiments. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):370–392.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. Google-Books-ID: Np9SDQAAQBAJ.

Grbovic, M., Djuric, N., Radosavljevic, V., Silvestri, F., and Bhamidipati, N. (2015). Context- and Content-aware Embeddings for Query Rewriting in Sponsored Search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 383–392, New York, NY, USA. ACM.

Gross, A. J. and Clark, V. (1975). *Survival Distributions: Reliability Applications in the Biomedical Sciences*. Wiley. Google-Books-ID: VFhrAAAAMAAJ.

Gu, X., Papadimitriou, S., Yu, P. S., and Chang, S. P. (2008). Online failure forecast for fault-tolerant data stream processing. In *2008 IEEE 24th International Conference on Data Engineering*, pages 1388–1390.

Harvey, A. C. and Trimbur, T. M. (2003). General Model-Based Filters for Extracting Cycles and Trends in Economic Time Series. *The Review of Economics and Statistics*, 85(2):244–255.

Huet, J., Besseau, S., Maillard, B., and Michaud, F. (2015). Method and computer program for the maintenance aid of aircraft equipment. US Patent App. 14/341,272.

Hunter, J. D. (2007). Matplotlib: A 2d Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95.

Iokhvidov, I. S. (1982). *Hankel and Toeplitz matrices and forms: algebraic theory*. Birkhäuser. Google-Books-ID: jQPvAAAAMAAJ.

Ittoo, A., Nguyen, L. M., and van den Bosch, A. (2016). Text analytics in industry: Challenges, desiderata and trends. *Computers in Industry*, 78:96–107.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Kaiser, K. A. and Gebraeel, N. Z. (2009). Predictive maintenance management using sensor-based degradation models. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(4):840–849.

Kaplan, E. L. and Meier, P. (1958). Nonparametric Estimation from Incomplete Observations. *Journal of the American Statistical Association*, 53(282):457–481.

Kauschke, S., Fürnkranz, J., and Janssen, F. (2016). Predicting cargo train failures: A machine learning approach for a lightweight prototype. In *International Conference on Discovery Science*, pages 151–166. Springer.

Kelly, P. J. and Lim, L. L.-Y. (2000). Survival analysis for recurrent event data: an application to childhood infectious diseases. *Statistics in Medicine*, 19(1):13–33.

Klein, J. P. and Moeschberger, M. L. (2005). *Survival Analysis: Techniques for Censored and Truncated Data*. Springer Science & Business Media.

Koh, Y. S. (2009). *Rare Association Rule Mining and Knowledge Discovery: Technologies for Infrequent and Critical Event Detection: Technologies for Infrequent and Critical Event Detection*, volume 3. IGI Global.

Kohavi, R. (1995). A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press. Google-Books-ID: 7dzpHCHzNQ4C.

Landauer, T. K., Foltz, P. W., and Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes*, 25(2-3):259–284.

Laxman, S., Tankasali, V., and White, R. W. (2008). Stream prediction using a generative model based on frequent episodes in event sequences. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 453–461, New York, NY, USA. ACM.

Lee, E. T. and Wang, J. W. (2013). *Statistical Methods for Survival Data Analysis*. John Wiley & Sons. Google-Books-ID: _fD9AAAAQBAJ.

Liao, H., Zhao, W., and Guo, H. (2006). Predicting remaining useful life of an individual unit using proportional hazards model and logistic regression model. In *RAMS '06. Annual Reliability and Maintainability Symposium, 2006.*, pages 127–132.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R news*, 2(3):18–22.

Liu, F., Pennell, D., Liu, F., and Liu, Y. (2009). Unsupervised Approaches for Automatic Keyword Extraction Using Meeting Transcripts. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 620–628, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liu, Y., Liu, Z., Chua, T.-S., and Sun, M. (2015). Topical Word Embeddings.

Ma, Z. and Krings, A. W. (2008). Survival Analysis Approach to Reliability, Survivability and Prognostics and Health Management (PHM). In *2008 IEEE Aerospace Conference*, pages 1–20.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, Stroudsburg, PA, USA. Association for Computational Linguistics.

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., Xin, D., Xin, R., Franklin, M. J., Zadeh, R., Zaharia, M., and Talwalkar, A. (2016). MLlib: Machine Learning in Apache Spark. *J. Mach. Learn. Res.*, 17(1):1235–1241.

Mhamdi, F., Poggi, J.-M., and Jaïdane, M. (2011). Trend extraction for seasonal time series using ensemble empirical mode decomposition. *Advances in Adaptive Data Analysis*, 03(03):363–383.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*. arXiv: 1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed Representations of Words and Phrases and their Compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

Miller, R. G. (2011). *Survival Analysis*. John Wiley & Sons. Google-Books-ID: MvO9I8g3zxAC.

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.

Nakagawa, T. and Osaki, S. (1975). The discrete weibull distribution. *IEEE Transactions on Reliability*, 24(5):300–301.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot,

M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Riihimaki, J. and Vehtari, A. (2010). Gaussian processes with monotonicity information. In *PMLR*, pages 645–652.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv:1609.04747 [cs]*. arXiv: 1609.04747.

Saeeda, L. (2017). Iterative Approach for Information Extraction and Ontology Learning from Textual Aviation Safety Reports. In *The Semantic Web*, Lecture Notes in Computer Science, pages 236–245. Springer, Cham.

Salfner, F., Lenk, M., and Malek, M. (2010). A survey of online failure prediction methods. *ACM Computing Surveys (CSUR)*, 42(3):10.

Savova, G. K., Masanz, J. J., Ogren, P. V., Zheng, J., Sohn, S., Kipper-Schuler, K. C., and Chute, C. G. (2010). Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513.

Schwarzkopf, O. (1995). The Extensible Drawing Editor Ipe. In *Proceedings of the Eleventh Annual Symposium on Computational Geometry*, SCG '95, pages 410–411, New York, NY, USA. ACM.

Sha, F., Lin, Y., Saul, L. K., and Lee, D. D. (2007). Multiplicative Updates for Nonnegative Quadratic Programming. *Neural Computation*, 19(8):2004–2031.

Sha, F., Saul, L. K., and Lee, D. D. (2003). Multiplicative Updates for Nonnegative Quadratic Programming in Support Vector

Machines. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 1065–1072. MIT Press.

Siklosi, B., Novák, A., and Prószéky, G. (2013). Context-Aware Correction of Spelling Errors in Hungarian Medical Documents. In *Statistical Language and Speech Processing*, Lecture Notes in Computer Science, pages 248–259. Springer, Berlin, Heidelberg.

Sipos, R., Fradkin, D., Moerchen, F., and Wang, Z. (2014a). Log-based predictive maintenance. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1867–1876. ACM.

Sipos, R., Fradkin, D., Moerchen, F., and Wang, Z. (2014b). Log-based predictive maintenance. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1867–1876. ACM.

Son, J., Zhou, Q., Zhou, S., Mao, X., and Salman, M. (2013). Evaluation and comparison of mixed effects model based prognosis for hard failure. *IEEE Transactions on Reliability*, 62(2):379–394.

Sordoni, A., Bengio, Y., Vahabi, H., Lioma, C., Grue Simonsen, J., and Nie, J.-Y. (2015). A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 553–562, New York, NY, USA. ACM.

Susto, G. A. and Beghi, A. (2016). Dealing with time-series data in predictive maintenance problems. In *Emerging Technologies*

and Factory Automation (ETFA), 2016 IEEE 21st International Conference on, pages 1–4. IEEE.

Tanguy, L., Tulechki, N., Urieli, A., Hermann, E., and Raynal, C. (2016). Natural language processing for aviation safety reports: From classification to interactive analysis. *Computers in Industry*, 78:80–95.

Theodoridis, S. (2015). *Machine Learning: A Bayesian and Optimization Perspective*. Academic Press. Google-Books-ID: NHODBAAAQBAJ.

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Tixier, A. J. P., Hallowell, M. R., Rajagopalan, B., and Bowman, D. (2016a). Automated content analysis for construction safety: A natural language processing system to extract precursors and outcomes from unstructured injury reports. *Automation in Construction*, 62:45–56.

Tixier, A. J.-P., Vazirgiannis, M., and Hallowell, M. R. (2016b). Word Embeddings for the Construction Domain. *arXiv:1610.09333 [cs]*. arXiv: 1610.09333.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ulanova, L., Yan, T., Chen, H., Jiang, G., Keogh, E., and Zhang, K. (2015a). Efficient Long-Term Degradation Profiling in Time Series for Complex Physical Systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 2167–2176, New York, NY, USA. ACM.

Ulanova, L., Yan, T., Chen, H., Jiang, G., Keogh, E., and Zhang, K. (2015b). Efficient long-term degradation profiling in time series for complex physical systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 2167–2176, New York, NY, USA. ACM.

Wang, P., Li, Y., and Reddy, C. K. (2017). Machine Learning for Survival Analysis: A Survey. *arXiv:1708.04649 [cs]*. arXiv: 1708.04649.

Wang, X. and Berger, J. (2016). Estimating Shape Constrained Functions Using Gaussian Processes. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):1–25.

Watanabe, Y., Otsuka, H., Sonoda, M., Kikuchi, S., and Matsumoto, Y. (2012). Online failure prediction in cloud datacenters by real-time message pattern learning. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 504–511. IEEE.

Weisberg, S. (2005). *Applied Linear Regression*. John Wiley & Sons. Google-Books-ID: xdotNdFOOjcC.

Weiss, G. M. and Hirsh, H. (1998). Learning to predict rare events in event sequences. In *KDD*, pages 359–363.

Wu, Z., Huang, N. E., Long, S. R., and Peng, C.-K. (2007). On the trend, detrending, and variability of nonlinear and nonstationary time series. *Proceedings of the National Academy of Sciences*, 104(38):14889–14894.

Yu, L., Zheng, Z., Lan, Z., and Coghlan, S. (2011). Practical online failure prediction for blue gene/p: Period-based vs event-driven. In *Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on*, pages 259–264. IEEE.

Yuan, Y., Zhou, S., Sievenpiper, C., Mannar, K., and Zheng, Y. (2011). Event log modeling and analysis for system failure prediction. *IIE Transactions*, 43(9):647–660.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., and Stoica, I. (2016). Apache Spark: A Unified Engine for Big Data Processing. *Commun. ACM*, 59(11):56–65.

Zhang, C. and Ma, Y. (2012). *Ensemble Machine Learning: Methods and Applications*. Springer Science & Business Media. Google-Books-ID: CjAs4stLXhAC.

Zhang, K., Xu, J., Min, M. R., Jiang, G., Pelechrinis, K., and Zhang, H. (2016). Automated it system failure prediction: A deep learning approach. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 1291–1300. IEEE.

Zheng, Z., Lan, Z., Park, B. H., and Geist, A. (2009). System log pre-processing to improve failure prediction. In *Depend-*

able Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on, pages 572–577. IEEE.

Zhou, Q., Son, J., Zhou, S., Mao, X., and Salman, M. (2014). Remaining useful life prediction of individual units subject to hard failure. *IIE Transactions*, 46(10):1017–1030.

Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms*. CRC Press. Google-Books-ID: BDB50Ev2ur4C.

COLOPHON

This document was typeset in LaTeX using the typographical look-and-feel `classicthesis`. The bibliography is typeset using `biblatex`.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Apprentissage Automatique pour la Maintenance Predictive dans le Domaine de l'Aviation

Mots clefs : maintenance predictive, aviation, apprentissage automatique

Résumé : L'augmentation des données disponibles dans presque tous les domaines soulève la nécessité d'utiliser des algorithmes pour l'analyse automatisée des données. Cette nécessité est mise en évidence dans la maintenance prédictive, où l'objectif est de prédire les pannes des systèmes en observant continuellement leur état, afin de planifier les actions de maintenance à l'avance. Ces observations sont générées par des systèmes de surveillance habituellement sous la forme de séries temporelles et de journaux d'événements et couvrent la durée de vie des composants correspondants. Le principal défi de la maintenance prédictive est l'analyse de l'historique d'observation afin de développer des modèles prédictifs.

Dans ce sens, l'apprentissage automatique est devenu omniprésent puisqu'il fournit les moyens d'extraire les connaissances d'une grande variété de sources de données avec une intervention humaine minimale. L'objectif de cette thèse est d'étudier et de résoudre les problèmes dans l'aviation liés à la prévision des pannes de composants à bord. La quantité de données liées à l'exploitation des avions est énorme et, par conséquent, l'évolutivité est une condition essentielle dans chaque approche proposée.

Cette thèse est divisée en trois parties qui correspondent aux différentes sources de données que nous avons rencontrées au cours de notre travail. Dans la première partie, nous avons ciblé le problème de la prédiction des pannes des systèmes, compte tenu de l'historique des Post Flight Reports. Nous avons proposé une approche statistique basée sur la régression précédée d'une formulation méticuleuse et d'un prétraitement / transformation de données. Notre méthode estime le risque d'échec avec une solution évolutive, déployée dans un environnement de cluster en apprentissage et en déploiement. À notre connaissance, il n'y a pas de méthode disponible pour résoudre ce problème jusqu'au moment où cette thèse a été écrite.

La deuxième partie consiste à analyser les données du livre de bord, qui consistent en un texte décrivant les problèmes d'avions et les actions de maintenance correspondantes. Le livre de bord contient des informations qui ne sont pas présentes dans les Post Flight Reports bien qu'elles soient essentielles dans plusieurs applications, comme la prédiction de l'échec. Cependant, le journal de bord contient du texte écrit par des humains, il contient beaucoup de bruit qui doit être supprimé afin d'extraire les informations utiles. Nous avons abordé ce problème en proposant une approche basée sur des représentations vectorielles de mots. Notre approche exploite des similitudes sémantiques, apprises par des neural networks qui ont généré les représentations vectorielles, afin d'identifier et de corriger les fautes d'orthographe et les abréviations. Enfin, des mots-clés importants sont extraits à l'aide du Part of Speech Tagging.

Dans la troisième partie, nous avons abordé le problème de l'évaluation de l'état des composants à bord en utilisant les mesures des capteurs. Dans les cas considérés, l'état du composant est évalué par l'ampleur de la fluctuation du capteur et une tendance à l'augmentation monotone. Dans notre approche, nous avons formulé un problème de décomposition des séries temporelles afin de séparer les fluctuations de la tendance en résolvant un problème convexe. Pour quantifier l'état du composant, nous calculons à l'aide de Gaussian Mixture Models une fonction de risque qui mesure l'écart du capteur par rapport à son comportement normal.

Title : Machine Learning for Predictive Maintenance in Aviation

Keywords : predictive maintenance, aviation, machine learning

Abstract : The increase of available data in almost every domain raises the necessity of employing algorithms for automated data analysis. This necessity is highlighted in predictive maintenance, where the ultimate objective is to predict failures of hardware components by continuously observing their status, in order to plan maintenance actions well in advance. These observations are generated by monitoring systems usually in the form of time series and event logs and cover the lifespan of the corresponding components. Analyzing this history of observation in order to develop predictive models is the main challenge of data driven predictive maintenance.

Towards this direction, Machine Learning has become ubiquitous since it provides the means of extracting knowledge from a variety of data sources with the minimum human intervention. The goal of this dissertation is to study and address challenging problems in aviation related to predicting failures of components on-board. The amount of data related to the operation of aircraft is enormous and therefore, scalability is a key requirement in every proposed approach.

This dissertation is divided in three main parts that correspond to the different data sources that we encountered during our work. In the first part, we targeted the problem of predicting system failures, given the history of Post Flight Reports. We proposed a regression-based approach preceded by a meticulous formulation and data preprocessing/transformation. Our method approximates the risk of failure with a scalable solution, deployed in a cluster environment both in training and testing. To our knowledge, there is no available method for tackling this problem until the time this thesis was written.

The second part consists analyzing logbook data, which consist of text describing aircraft issues and the corresponding maintenance actions and it is written by maintenance engineers. The logbook contains information that is not reflected in the post-flight reports and it is very essential in several applications, including failure prediction. However, since the logbook contains text written by humans, it contains a lot of noise that needs to be removed in order to extract useful information. We tackled this problem by proposing an approach based on vector representations of words (or word embeddings). Our approach exploits semantic similarities of words, learned by neural networks that generated the vector representations, in order to identify and correct spelling mistakes and abbreviations. Finally, important keywords are extracted using Part of Speech Tagging.

In the third part, we tackled the problem of assessing the health of components on-board using sensor measurements. In the cases under consideration, the condition of the component is assessed by the magnitude of the sensor's fluctuation and a monotonically increasing trend. In our approach, we formulated a time series decomposition problem in order to separate the fluctuation from the trend by solving a convex program. To quantify the condition of the component, we compute a risk function which measures the sensor's deviation from it's normal behavior, which is learned using Gaussian Mixture Models.

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