Méthodes d'apprentissage statistique pour l'analyse de données de production et de performances des moteurs d'avion

Sara Rejeb

THÈSE

en vue de l'obtention du grade de

Docteur de Sorbonne Université

Discipline : **Mathématiques Appliquées**

Spécialité : **Statistique**

Laboratoire de Probabilités, Statistique et Modélisation

École Doctorale Sciences Mathématiques de Paris Centre

par **Sara Rejeb**

Méthodes d'apprentissage statistique pour l'analyse de données de production et de performances des moteurs d'avion

---

Sous la direction de : Tabea Rebafka et Catherine Duveau

Devant un jury composé de :

M.	Mustapha Lebbah	*PR, Université Paris-Saclay*	Rapporteur
Mme.	Madalina Olteanu	*PR, Université Paris Dauphine*	Rapporteuse
Mme.	Nataliya Sokolovska	*PR, Sorbonne Université*	Examinatrice
Mme.	Catherine Matias	*DR, Sorbonne Université/CNRS*	Présidente
M.	Etienne Côme	*CR, IFSTTAR/Université Gustave Eiffel*	Examinateur
M.	Mohamed Achibi	*Ingénieur Safran Aircraft Engines*	Examinateur
Mme.	Tabea Rebafka	*MCF, Sorbonne Université*	Directrice
Mme.	Catherine Duveau	*Ingénieure Safran Aircraft Engines*	Co-encadrante

Abstract:
The manufacturing process of turbofan engine parts generates deviations due to overruns in foundry, weaving or machining. It is therefore necessary to carry out costly and slow metrological measurements. The main objective of the thesis is the statistical analysis of production data and aircraft engine performance in order to detect and anticipate risky production drifts as soon as possible. In particular, we seek to understand the impact of different geometric characteristics of engine parts on the performance. Different statistical learning methods are proposed in order to model the engine performance taking into account the specificities of the data. The first part of the thesis is dedicated to the exploration and visualization of partially observed data. We propose an adaptation of self-organizing maps for incomplete data that aims to simultaneously optimize the map and the imputation of missing values. The proposed missSOM algorithm improves the quality of the self-organizing map and the representation of partially observed data. The second part of the thesis focuses on the statistical modeling of the engine performance based on data concerning the parts of engine. Several statistical learning models are studied and compared. This study reveals strong dependencies related to test conditions and the production of the engine parts. The third part of the thesis deals with the presence of missing data in an influence analysis tool designed to identify the main causes of a production drift. This involves studying the impact of missing data on the results of the influence analysis. Different approaches are considered in order to adapt the tool to incomplete data, and are evaluated according to several scenarios.

Résumé :
Le processus de fabrication de pièces de moteurs turbofan génère des dérogations dues à des dépassements de côtes en fonderie, tissage ou en usinage. Ces dépassements entraînent la nécessité de pratiquer quasi systématiquement des mesures de métrologie lentes et coûteuses. L'objectif principal de la thèse est l'analyse statistique des données de production et des performances des moteurs d'avion afin de détecter et anticiper au plus tôt des dérives de production à risque. On cherche notamment à comprendre l'impact des différentes caractéristiques géométriques des pièces moteurs sur les performances. Différentes méthodes d'apprentissage statistique sont proposées afin de modéliser au mieux les performances moteurs tenant en compte des spécificités des données. La première partie de la thèse est consacrée à l'exploration et à la visualisation des données partiellement observées. Nous proposons une adaptation des cartes auto-organisatrices pour des données incomplètes qui vise à optimiser simultanément la carte et l'imputation des valeurs manquantes. L'algorithme missSOM proposé améliore la qualité de la carte auto-organisatrice et la représentation des données partiellement observées. La deuxième partie de la thèse est dédiée à la modélisation statistique des performances moteurs à partir de la production des pièces. Une étude et une comparaison de plusieurs modèles d'apprentissage statistiques sont réalisées et révèlent de fortes dépendances liées aux conditions d'essais et à la production des pièces moteurs. La troisième partie de la

thèse traite la présence de données manquantes dans un outil d'analyse d'influence qui vise à identifier les principales causes d'une dérive de production. Il s'agit d'étudier l'impact des données manquantes sur les résultats d'analyse d'influence. Différentes approches sont considérées afin d'adapter l'outil aux données incomplètes et sont évaluées selon plusieurs scénarios.

Contents

Chapter 1

Introduction

Contents

Cette thèse porte sur l'analyse statistique des données de production et des performances des moteurs d'avion afin de détecter et anticiper au plus tôt des dérives de production à risque. On cherche à comprendre l'impact des différentes caractéristiques géométriques des pièces moteurs sur les performances. Différentes méthodes d'apprentissage statistique sont proposées afin de mieux étudier la base de données et de modéliser au mieux les performances moteurs tenant compte des spécificités des données. Nous attachons une importance particulière à la gestion des données manquantes qui est abordée selon différents aspects, d'abord dans un cadre non-supervisé pour des cartes auto-organisatrices puis dans un cadre supervisé pour l'analyse d'influence.

Plus précisément, cette thèse a été réalisée dans le cadre d'un contrat CIFRE entre le Laboratoire de Probabilités, Statistique et Modélisation (LPSM) de Sorbonne Université et l'entreprise Safran Aircraft Engines du groupe Safran. Plus précisément, elle s'inscrit dans la continuité de mes travaux réalisés dans le cadre de mon stage de fin d'études de Master 2 en 2019. La thèse a débuté au sein des bureaux d'études de la Direction Technique de Safran puis au sein du DataLab.

Le LPSM est une unité mixte de recherche (UMR 8001) dépendant du CNRS, de Sorbonne Université et de l'Université Paris Cité. Les activités de recherche au sein du LPSM couvrent un large spectre en Probabilités et Statistique, depuis les aspects les plus fondamentaux (qui incluent notamment l'Analyse Stochastique, la Géométrie Aléatoire, les Probabilités Numériques et les Systèmes Dynamiques) jusqu'aux applications à la Modélisation dans diverses disciplines (Physique, Biologie, Sciences des Données, Finance, Actuariat, etc), applications qui incluent des partenariats en dehors du monde académique.

Safran est un groupe international de haute technologie dans les secteurs de l'aéronautique, la défense et l'espace. Le groupe est composé de plusieurs sociétés dont Safran Aircraft Engines qui conçoit, développe, produit et commercialise des moteurs pour satellites, lanceurs spatiaux et pour avions civils et militaires. Safran Aircraft Engines propose à ses clients civils et militaires une gamme complète de supports et services, afin d'optimiser la disponibilité des appareils.
Le Datalab de Safran Aircraft Engines a pour mission d'améliorer la conception, optimiser la fabrication et surveiller en continu les données issues des nombreux capteurs. Son équipe possède des compétences en matière de statistique, d'analyse de données ou de génie logiciel et arrive en support pour de nombreux projets de l'entreprise, toujours en coopération avec les bureaux d'étude qui sont responsables de la conception des moteurs d'avions civils et militaires.
Créé en 1947, le site à Villaroche était principalement dédié aux essais en vol et au sol des moteurs. Aujourd'hui, les activités du site se sont fortement développées autour de différents projets de recherche et de développement, de la conception, des opérations de montages ainsi que des moyens d'essais.

Dans cette thèse, on s'intéresse à la production d'un certain type de moteur et à l'étude de ses performances.

Dans ce chapitre, nous décrivons d'abord dans la section 1.1 le fonctionnement d'un turboréacteur et le rôle principal d'un compresseur. Ensuite, dans la section 1.2, nous présentons le contexte et les objectifs de la thèse. De plus, une description des données de production et des indicateurs de performance des moteurs analysées dans la thèse est fournie dans la section 1.3. Les principales méthodes statistiques utilisées dans la thèse sont introduites dans la section 1.4. Enfin, la section 1.5 présente les principales contributions réalisées dans ce travail de thèse.

1.1 Fonctionnement d'un moteur

Pour une meilleure compréhension de la base de données qui est au cœur de cette thèse, nous présentons d'abord le fonctionnement d'un moteur d'avion.

Un moteur fournit une **poussée** à l'avion qui lui permet d'avancer et de voler.

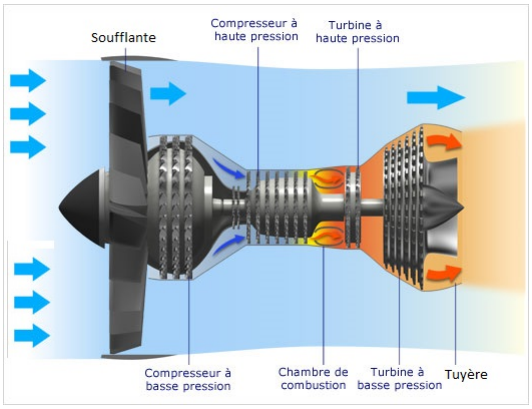


Figure 1.1: Schéma d'un turboréacteur

Les principaux composants d'un turboréacteur double flux sont les suivants : une soufflante (ou fan), des compresseurs basse et haute pression, une chambre de combustion, des turbines et une tuyère de sortie.
Un ensemble de capotages entoure le moteur, appelé nacelle : en particulier la manche d'entrée d'air et le capot. La manche à air permet de capter l'air pour rendre régulière l'entrée d'air dans la soufflante.

Schématiquement, le turboréacteur absorbe de l'air par une manche d'entrée d'air, le comprime, le chauffe puis l'éjecte à l'extérieur par l'intermédiaire d'une tuyère. Pour fournir une poussée, cette vitesse d'éjection doit être supérieure à celle de l'admission.
A l'avant du moteur, la soufflante est constituée d'un ensemble d'**aubes** (ou pales) entouré d'un **carter**. La distance entre le carter et le sommet d'une aube, appelée **jeu du fan**, quantifie l'interaction aube-carter. Pour éviter une altération des pièces due aux frottements, le jeu ne doit pas être trop petit. En tournant, les aubes du fan aspirent un grand volume d'air. Ce flux d'air traverse le moteur en passant d'abord par le compresseur à basse pression qui le comprime. Il entre ensuite dans le compresseur à haute pression où les injecteurs mélangent les particules de kérosène et de l'air. Ce mélange, devenu homogène, est emmené dans la chambre de combustion qui va augmenter la température et la pression des gaz. Elle apportera les gaz chauds à la turbine pour la faire tourner. L'énergie est créée par la combustion du carburant et de l'air comprimé. La turbine et la tuyère de sortie transforment l'énergie de la chambre de combustion en énergie mécanique. La poussée, la température et la consommation de carburant font partie des performances essentielles d'un moteur.
Les bureaux d'études modules sont responsables de la conception de chaque module: soufflantes, compresseurs, turbines et chambres de combustion.

Le rôle du compresseur, et en particulier du fan

Un compresseur résulte d'un empilage d'étages composés chacun d'un aubage mobile et d'un aubage fixe. L'aubage mobile est constitué d'un disque sur lequel sont fixées des aubes. Cette roue tourne devant l'aubage fixe qui est également constitué d'aubes qui elles sont fixes. La compression de l'air se passe en deux phases. Lors de la première phase, l'aubage mobile procure une accélération aux particules d'air en les déviant par rapport à l'axe du moteur. Pendant la deuxième phase, l'aubage fixe qui le suit, ralentit ces particules et transforme une partie de leur vitesse en pression. Cet aubage ramène l'écoulement de l'air, accéléré par l'aubage mobile, dans l'axe du moteur. Les performances d'un étage de compresseur sont caractérisées par trois grandeurs : son débit d'air, son taux de compression et son rendement.
Le fan est le premier étage du compresseur. Il a la particularité d'avoir des aubes mobiles de grande dimension, et que le flux d'air se sépare en deux à la sortie entre le flux primaire et le flux secondaire. Il assure une grande partie de la poussée.
Le rôle du fan est d'abord de fournir un débit d'air pour assurer la poussée du turboréacteur, avec certaines conditions de rendement et d'opérabilité à respecter. En particulier, à partir d'un certain angle le profil d'une aube n'assure plus sa portance. Pour que le fan réalise sa fonction, la déviation, et donc les angles des aubes fan sont importants.

1.2 Contexte et objectif de la thèse

La production d'un moteur est soumise à une réglementation stricte définie dans un cahier des charges. Les moteurs produits conformément aux exigences de qualité souhaitées par le client obtiennent une certification qui autorise leur livraison et leur mise en service. Après avoir contrôlé et validé un par un chaque pièce et chaque module, les moteurs effectuent des tests en essais de réception afin d'évaluer leur performance et valider ainsi leur conformité.
Le contrôle de toute pièce qui doit sortir d'une usine vérifie au préalable les caractéristiques et les limites à ne pas dépasser. Cependant, le processus de fabrication de pièces de moteurs turbofan génère des dérogations dues à des dépassements de côtes en fonderie, tissage ou en usinage. En effet, les exigences sur les intervalles de tolérance spécifiés sont élevées, et ces intervalles sont difficiles à respecter lors du processus de production. Ces dépassements entraînent la nécessité de pratiquer quasi systématiquement des mesures de métrologie lentes et coûteuses. Pourtant, l'expérience et les tests en essais de réception des moteurs montrent que malgré ces écarts au modèle les moteurs produits présentent des marges de performance attendues. Ce fait est principalement dû au panachage de grands nombres de pièces produites toutes avec des côtes légèrement différentes qui en moyenne, par exemple sur un disque de compresseur, ont tendance à se compenser.

La thèse se situe dans le contexte de la production d'un type moteur et concerne l'étude de ses performances. L'objectif principal de la thèse est l'analyse statistique de ces données de production et des performances des moteurs d'avion afin de détecter et anticiper au plus tôt des dérives de production à risque pour les performances et ainsi d'améliorer la production. Cela consiste à déterminer les conditions de production qui provoquent des performances à risque afin de les éviter le plus tôt possible pendant le processus de fabrication. Cela permettrait d'alerter plus rapidement la production pour entamer une première recherche des causes racines d'une dérive de performance. Plus précisément, cette recherche consiste à identifier et comprendre les facteurs principaux du procédé de fabrication qui ont déclenchés une telle dérive. On cherche notamment à comprendre l'impact des différentes caractéristiques géométriques des pièces moteurs sur les performances afin d'améliorer la production et de réduire le coût d'acquisition des données. De plus, une bonne compréhension permettrait de justifier et valider certains élargissements des tolérances géométriques souvent très strictes et difficiles à respecter. Dans un tel contexte industriel, l'interprétation et l'explicabilité des résultats statistiques sont primordiales afin d'assister les experts métiers dans leur prise de décision.

Dans le cadre de la thèse, on se concentre sur la production des pièces du fan et on s'intéresse essentiellement au débit d'air des moteurs, une des performances fournie par le fan. Le travail de thèse pourra être étendu aux autres mesures de performances du moteur.

1.3 Présentation de la base de données

Dans ce travail, nous avons à disposition des données de contexte et de traçabilité ainsi que des données de mesures sur les pièces, les montages et les essais de réception des moteurs. Les données de fabrication des pièces du fan, d'assemblage et de performances d'un type de moteur sont collectées automatiquement chaque jour. Une base de données a été mise en place à Safran pour collecter l'ensemble de ces données. Cette base de données qui est assez récente est très volumineuse et riche, ce qui nécessite une réfexion sur l'utilisation et l'analyse de ces données. Les données analysées pendant la thèse couvrent les essais d'une période donnée. Au total, 1550 moteurs ont été testés. Chacun d'entre eux possède 19 pièces de son fan, dont 18 aubes et un carter, fabriquées dans deux usines différentes. Environ 57% des pièces proviennent de la première usine A et le reste de la seconde usine B.

Commençons par décrire les données de fabrication des pièces fan des moteurs d'avion.

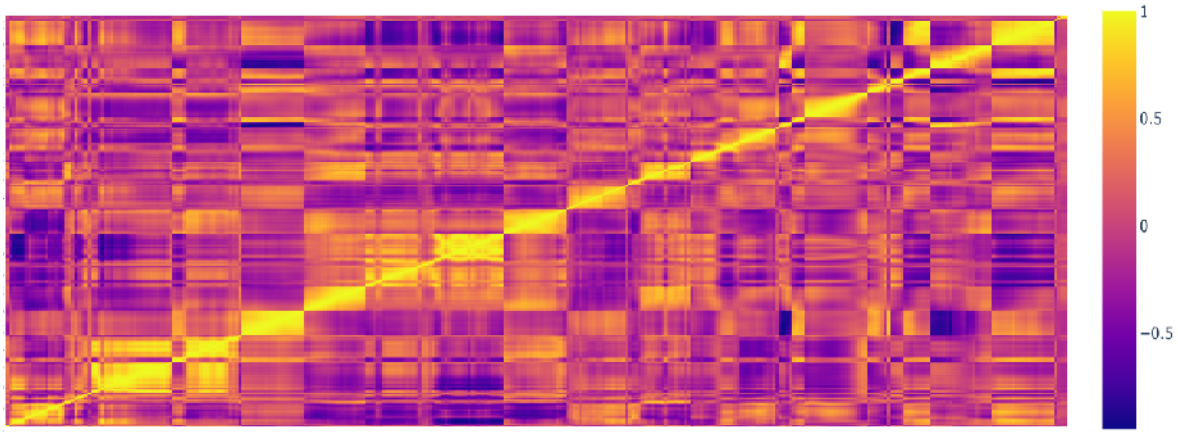


Figure 1.2: Matrice de corrélation

1.3.1 Données de fabrication

Les données de fabrications sont celles des aubes et des carters fan de chaque moteur. Pour chaque pièce du fan, on possède des informations de traçabilité comme le fournisseur, la date de la mesure, la valeur nominale de la mesure et l'intervalle de tolérance associé (valeur minimale et maximale tolérées).

Pour chaque carter fan, on mesure le rayon de la veine abradable en six stations axiales. Pour chacune de ces six stations, 24 stations angulaires sont mesurées. Cette caractéristique permet de mesurer l'usure de la pièce due à l'interaction aube-carter. Pour chaque carter, on prendra la moyenne des mesures sur chaque station axiale.

Par ailleurs, comme nous l'avons précisé précédemment, le profil des aubes impacte fortement la quantité d'air qu'elles absorbent. Afin de contrôler et vérifier la conformité des aubes, des mesures de différentes caractéristiques sont effectuées sur plusieurs sections, du pied jusqu'au sommet de l'aube. Pour chacune d'entre elles, les mesures sont de deux types : **géométrique** et **état de surface**. Chacune des caractéristiques est représentée sous forme vectorielle, ce qui représente un total de 323 paramètres à mesurer pour chaque aube. D'autres caractéristiques du fan considérées comme non impactant sur la performance moteur n'ont pas été étudiées dans notre cadre de travail. Une première sélection des caractéristiques basée sur des notions en aérodynamique avait en effet été faite par les experts métiers. Les données de métrologie des aubes fan sont ainsi en grande dimension et très volumineuses contenant 14180 pièces mesurées sur trois ans : environ 29 % la première année, 14 % la deuxième année et 57 % la troisième année.

La figure 1.2 montre la matrice de corrélation linéaire entre toutes les variables, révélant de nombreuses relations linéaires entre les paramètres. On observe en particulier de nombreux groupes de variables fortement corrélées. En effet, les blocs

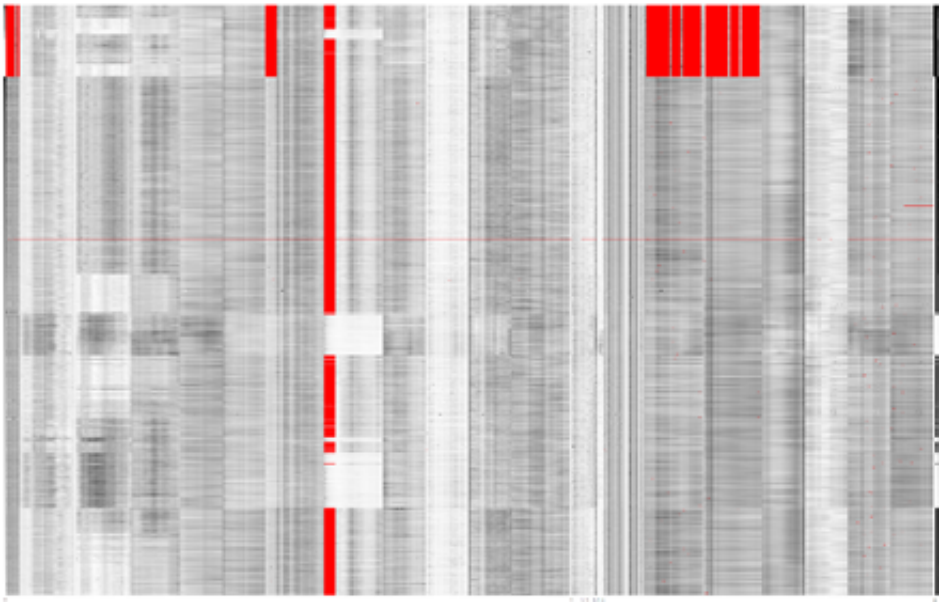


Figure 1.3: Localisation des données manquantes en rouge dans la matrice des mesures des aubes fan

sur la diagonale correspondent à des mesures d'une même caractéristique mesurée sur toute la hauteur de l'aube. Des dépendances entre certaines caractéristiques géométriques sont également observées. Nous sommes donc confrontés à un problème de multicolinéarité élevée entre les variables. Cette multicolinéraité qui peut rendre l'interprétation des résultats difficile nécessite alors une sélection de variables en déterminant un nombre limité de variables influentes et pertinentes pour la modélisation des données de performance. De plus, dans notre contexte, il existe également d'autres dépendances liées au fournisseur des pièces.
Par ailleurs, dans les données enregistrées, on trouve de nombreuses données manquantes, c'est-à-dire que la métrologie des pièces est incomplète. Cela arrive pour différentes raisons comme des changements de spécification ou des problèmes d'acquisition. Des protocoles de mesure différents dans les deux usines peuvent également être à l'origine de variables manquantes. De plus, les mesures aberrantes, qui sont assez fréquentes en métrologie, sont considérées comme des données manquantes. La figure 1.3 visualise la localisation des données manquantes représentées en rouge dans la matrice des mesures des aubes fan composée de 323 colonnes et 14180 lignes. On observe en particulier des paquets de variables et d'aubes fan pour lesquelles la mesure est manquante. On peut également observer la présence de données manquantes à d'autres endroits un peu moins visible sur la figure.

Il est important de tenir compte de tous ces éléments dans la thèse.

Après la production des pièces, on assemble les différents modules pour constituer

le moteur.

1.3.2 Données de montage

Les données de montage sont également composées d'informations de traçabilité, de définition et de mesure. Cela concerne la composition des pièces fan montées sur un moteur, les mesures de jeux effectuées lors du montage et les intervalles de tolérance des mesures.
A l'extrémité d'une aube, le jeu fan est mesurée sur trois positions axiales : au bord d'attaque, au centre et au bord de fuite. Sur chacune de ces positions axiales, le jeu fan est mesurée de la façon suivante : on détermine d'abord l'aube la plus longue puis on mesure le jeu sur huit positons horaires différentes de cette aube sur la roue fan.
Pour chaque moteur, on prendra la moyenne du jeu sur chaque position axiale.

En revanche, cette façon de mesurer les jeux fan n'est pas assez précise provoquant ainsi de nombreuses erreurs de mesure et des mesures aberrantes. Cette incertitude de la mesure des jeux fan est un problème majeur dans notre travail de thèse car la mesure qui impacte fortement le débit d'air des moteurs est une information essentielle pour la modélisation de la performance. Cette imprécision de la mesure qui peut fortement dégrader la qualité de la modélisation est prise en compte dans notre travail de thèse.

Après le montage des modules fan, les performances des moteurs sont testées dans des bancs en essais de réception.

1.3.3 Données d'essais de réception

Pour réaliser les essais, des accessoires sont utilisés pour être installés sur un moteur: une nacelle, une tuyère de sortie et une manche d'entrée d'air. Les données d'essais contiennent ces informations ainsi que le banc d'essai utilisé lors du test et la date de l'essai. Puisqu'il existe plusieurs bancs d'essais et accessoires d'installation, les moteurs ne sont pas tous testés dans les mêmes conditions d'essais. De plus, les performances des moteurs sont mesurées à différents régimes de rotation et dans des conditions ambiantes différentes telles que la pression atmosphérique, la température et l'humidité de l'air. Ces conditions peuvent rendre la mesure imprécise ne reflétant pas la réelle performance du moteur lorsque celui-ci est par exemple en vol. Les mesures d'essais sont alors corrigées et normalisées par rapport à un moteur de référence et dans des conditions ambiantes standard. En revanche, ces corrections effectuées par des ingénieurs d'essais ne suffisent pas. En effet, malgré ces corrections, les équipements d'installation ont toujours un impact sur les mesures de performances des moteurs. Cela provient principalement des conditions d'essais qui

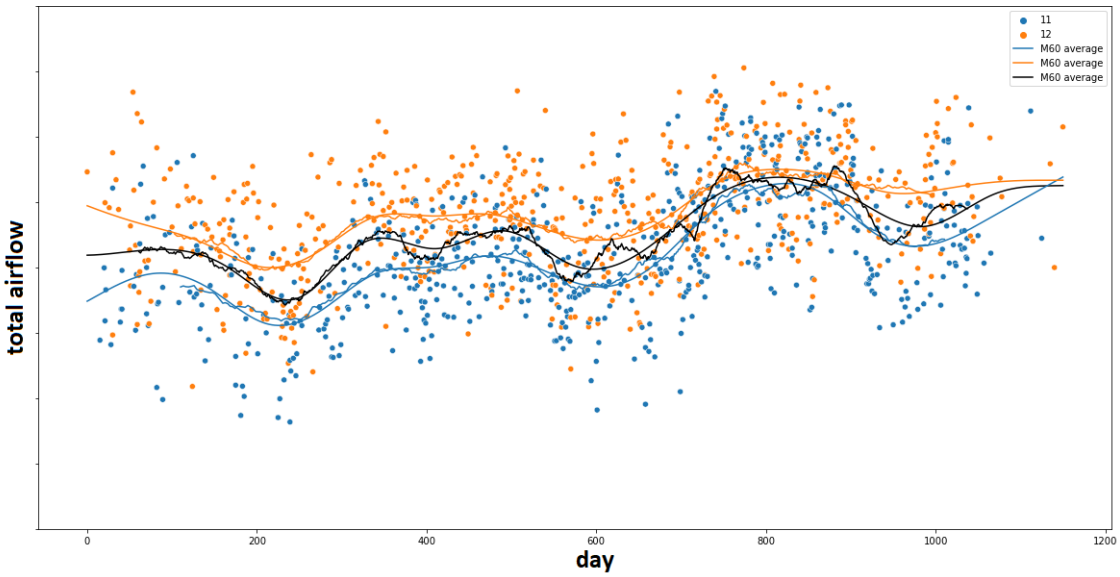


Figure 1.4: Débit d'air total des moteurs d'avion en fonction de la date d'essai

peuvent provoquer une usure des équipements nécessitant souvent des opérations de maintenance. Cette usure est ainsi à l'origine des dérives observées sur les accessoires d'installation. Des biais de mesure de performance sont alors constatés, entraînant une augmentation ou une diminution de la valeur réelle attendue. La figure 1.4 représente selon la date de l'essai le débit d'air des moteurs d'avions catégorisés par le banc d'essai utilisé pendant les tests. Les courbes bleue et orange représentent respectivement la tendance moyenne globale du débit d'air des moteurs testés sur le banc 11 et 12. De plus, la courbe en couleur noire représente la tendance globale du débit d'air de tous les moteurs testés, tous bancs confondus. On observe que les moteurs testés avec le banc 12 ont globalement un débit d'air plus élevé que ceux testés avec le banc 11. Les équipements utilisés pendant les essais ont clairement une influence sur les mesures des performances des moteurs.

1.4 Méthodes statistiques

Une première étape d'exploration et d'analyse des données de production des pièces du fan est essentielle pour une meilleure vue globale de la structure interne des données de façon à en détecter des tendances de production, des effets d'usine et éventuellement des pièces atypiques. Cette étape préliminaire à la modélisation des performances nous permet de mieux comprendre l'influence des différents paramètres sur les performances des moteurs et ainsi de mieux les modéliser.
Les tâches courantes de l'exploration des données sont la visualisation et le clustering des données. Bien qu'il existe de nombreuses méthodes permettant d'accomplir

l'une ou l'autre de ces tâches, nous nous intéressons principalement aux cartes auto-organisatrices qui fournissent simultanément une représentation visuelle des données en faible dimension sous forme de carte et un partitionnement des observations. Introduite par Kohonen [49], cette approche consiste à cartographier les données et effectuer une discrétisation de l'espace d'entrée tout en préservant les propriétés topologiques des données. Cette technique qui est devenue très populaire dans de nombreuses applications du monde réel fournit une vue globale des données avec des résultats facilement interprétables. Les cartes auto-organisatrices sont ainsi largement utilisées en économie, en biologie [67], en sciences humaines [66] et en industrie, comme pour la surveillance de la santé des moteurs d'avions [18]. Au sein du groupe Safran, cette méthode est aujourd'hui beaucoup utilisée pour de la détection d'anomalie ou plus généralement pour l'exploration des données de vol.

Cependant, les cartes auto-organisatrices ne peuvent s'employer uniquement sur des données complètement observées sans valeurs manquantes. L'impact des données manquantes sur les résultats statistiques peut être conséquent, entraînant des estimations biaisées, une perte d'information et une moindre généralisation des résultats. Dans notre travail de thèse, nous nous intéressons aux cartes auto-organisatrices en présence de données manquantes. Nous avons analysé plus amplement la structure de nos données manquantes et développé des stratégies pour leur traitement dans l'analyse statistique. Plus précisément, une extension des cartes auto-organisatrices pour les données incomplètes a été proposée.

Par ailleurs, il est essentiel de réduire dans les données d'essais les variations importantes liées aux équipements du banc qui peuvent avoir un impact négatif sur la modélisation du débit d'air. De plus, le processus de mesure est également soumis à d'autres variations. En effet, les données d'essais qui sont issues de capteurs génèrent souvent des données bruitées sous différentes formes. Ces bruits peuvent provenir de biais de mesure, d'imprécisions de mesure liées aux conditions d'essais ou encore des données aberrantes causées par des dysfonctionnements de capteurs. Cette imprécision de la mesure, qui ne reflète pas complètement la réalité, altère la qualité des données, ce qui peut rendre la modélisation difficile. En effet, les méthodes d'apprentissage statistique sont souvent sensibles au bruit. Si ce dernier est trop élevé, aucune des méthodes ne peut fonctionner et la modélisation devient donc impossible. En particulier, certains algorithmes d'apprentissage supervisé sont sensibles au sur-apprentissage avec un impact dramatique sur leur performance et leur vitesse de convergence en présence de données bruitées. Dans la thèse, nous prenons en compte ces différents éléments ainsi que l'aspect temporel qui montre des tendances liées aux conditions d'essais, à l'usine de fabrication et à la production des pièces en constante évolution. En effet, sur la figure 1.4, on remarque que le biais associé à l'équipement banc dépend de la date d'essai et ne reste pas constant sur toute la période d'essai. Il est alors important d'en tenir compte en adaptant

les méthodes d'apprentissage statistique courantes au contexte et en ajustant les modèles à ce type de données. De plus, une bonne performance prédictive des modèles est nécessaire pour garantir aux experts métiers la précision des résultats et la capacité à détecter des phénomènes importants dans les données. Cependant, les modèles d'apprentissage statistique connus pour leur remarquable performance prédictive sont souvent complexes notamment pour modéliser des relations non-linéaires et sont ainsi difficilement interprétables.

Par ailleurs, lorsqu'une forte dérive de production ou de performance est détectée, il est important de rechercher rapidement les facteurs qui ont déclenchés une telle dérive, autrement dit d'identifier les causes d'une dégradation ou amélioration de la mesure de performance. Dans cette thèse, nous nous intéressons ainsi à un outil d'analyse d'influence ayant pour objectif de déterminer les causes de variation des mesures de performance de manière simple et interprétable. L'outil permet en plus de classer par ordre d'influence les causes qu'il a identifiées. C'est une méthodologie qui a été développée par Lacaille [53] comme un outil d'aide à l'analyse d'un problème survenu lors d'un processus industriel comme des déviations ou anomalies détectées pendant le processus de fabrication d'une pièce. Il peut s'employer comme une étape préliminaire à une recherche profonde des causes. Dans la thèse, on étudie l'outil en présence de données manquantes, et nous proposons et comparons différentes approches pour adapter l'outil aux données partiellement observées. Ces travaux ont un intérêt plus général qui dépasse le contexte et le cadre de la thèse car l'outil peut être utilisé plus largement dans des cadres opérationnels différents.

1.5 Contributions de la thèse

Les différentes contributions de la thèse sont présentées en détail dans les chapitres 2, 3 et 4.

Plus précisément, le chapitre 2 présente une extension des cartes auto-organisatrices aux données contenant des valeurs manquantes dans un objectif d'exploration de données partiellement observées et d'imputation des données manquantes. Nous introduisons un nouveau critère à optimiser, qui vise à définir simultanément la meilleure carte auto-organisatrice et la meilleure imputation des valeurs manquantes. Ainsi, la méthode peut également être utilisée comme une méthode d'imputation des données manquantes. Un algorithme itératif proposé alterne l'apprentissage d'une carte et l'imputation des données manquantes de façon à améliorer la qualité de représentation des observations incomplètes sur la carte. De plus, nous développons une version accélérée de cet algorithme afin de gagner en temps de calcul tout en gardant les mêmes propriétés. Une étude numérique approfondie est réalisée pour évaluer la performance de notre méthode dans divers contextes et en comparaison avec des méthodes alternatives de la littérature. Par ailleurs, ce chapitre montre

tout l'intérêt de cette contribution dans le contexte industriel avec une application à la métrologie des aubes fan pour visualiser et comprendre la structure interne des données et gérer les mesures incomplètes. Ces travaux ont également un intérêt qui dépasse le contexte des moteurs d'avion, car de nombreux jeux de données affichent des problématiques similaires.

Cette méthodologie a fait l'objet d'un article publié dans *Chemometrics and Intelligent Laboratory Systems* [RDR22b] et a été également présentée aux *53èmes Journées de Statistique de la SFdS* en juin 2022 à Lyon [RDR22a]. Ces travaux ont donné lieu au développement d'un package R `missSOM`[1] qui implémente la méthode et est disponible sur le CRAN [RDR22].

Le chapitre 3 porte sur la modélisation statistique du débit d'air des moteurs à partir de la production des aubes fan. Il s'agit de prédire la performance du moteur avant leur arrivée en essai de réception afin de détecter au plus tôt des dérives de performances à risque et déterminer les caractéristiques géométriques des pièces les plus influentes. Nous effectuons d'abord un pré-traitement des données pour réduire les biais de mesure de performance causés par les différents équipements utilisés pour les essais. Dans ce chapitre, nous explorons, d'une part, des modèles classiques d'apprentissage statistique pour déterminer la relation entre la production des aubes fan et le débit d'air des moteurs. D'autre part, nous appliquons différents modèles de séries temporelles pour mieux modéliser les dépendances. Une comparaison des performances des modèles est également fournie. Cette contribution a permis de soulever de nombreuses questions liées aux conditions des essais des moteurs et aux données.
Ce travail a fait l'objet d'un article présenté à la conférence *Turbomachinery Technical Conference & Exposition* qui a eu lieu à Boston en juin 2023 [RDR23].

Le chapitre 4 porte sur l'adaptation de l'outil d'analyse d'influence à la présence des valeurs manquantes dans les jeux de données. Plus précisément, il est question d'intégrer l'imputation des données manquantes dans l'outil sans perte de performance. Dans ce chapitre, nous étudions l'impact des données manquantes sur les résultats d'analyse d'influence. De nombreuses expériences numériques sont proposées afin de comparer et évaluer différentes approches possibles selon plusieurs scénarios. Ce chapitre fournit une application industrielle qui montre tout l'intérêt de ce travail dans le contexte industriel de la thèse, mais aussi dans d'autres contextes à Safran. Une amélioration de l'outil est également fournie. Ce chapitre fait l'objet d'un article en cours de finalisation.

[1] https://cran.r-project.org/web/packages/missSOM/index.html

Chapter 2

Self-Organizing Maps for the Exploration of Partially Observed Data and Imputation of Missing Values

Contents

Abstract To monitor the production process of turbofan aircraft engines, multiple measurements of various geometrical parameters are systematically recorded on manufactured parts. Engine parts are subject to extremely high standards as they can impact the performance of the engine. Therefore, it is essential to analyze these databases to better understand the influence of the different parameters on the engine's performance. The self-organizing map is an unsupervised neural network which is widely used for data visualization and clustering in the field of aircraft engine condition monitoring. The classical Kohonen algorithm that computes self-organizing maps is suitable only for complete data without any missing values. However, in many applications, partially observed data are the norm. In this chapter, we propose an extension of self-organizing maps to incomplete data via a new criterion that also defines estimators of the missing values. In addition, an adaptation of the Kohonen algorithm, named missSOM, is provided to compute these self-organizing maps and impute missing values. This method is efficiently implemented in R and has been released on CRAN as the package `missSOM` [RDR22]. Numerical experiments on simulated data and a real dataset illustrate the short computing time of missSOM and assess its performance regarding various criteria and in comparison to the state of the art. An application to fan blade measurements is also provided and shows the practical interest of missSOM. This work has been the subject of an article published in a journal [RDR22b] and was also presented at the *53èmes Journées de Statistique de la SFdS* in June 2022 in Lyon [RDR22a].

2.1 Introduction

In data analysis, we can encounter different types of problems that require a more important and considerable data pre-processing step. One of the difficulties in data analysis that we are often confronted with, is that we have no prior knowledge of the form and structure of the data. Moreover, with the explosion of data volume in many application domains, this internal structure is often difficult to detect because it is hidden behind the massive data, very rich in information. Data mining, which extracts more of this information and possibly summarizes redundant information, is essential and is usually the first part of the analysis of any data set. The initial understanding of the data gained through exploratory data analysis is particularly useful for data modeling. Common tasks of data exploration are visualization and

clustering of the data. While there is plethora of methods addressing one of the tasks, self-organizing maps simultaneously provide both a low-dimensional visual data representation in form of a map and a clustering of the observations. Introduced by Kohonen [49], this approach consists in mapping the data and performing vector quantization of the input space while preserving topological properties of the data even when these data are high-dimensional. These are properties that other data analysis methods do not have. For instance, the k-means method only categorises data, but the topology gets lost and it does not allow data visualisation. A dimensionality reduction method is often performed to visualize the data.

Several examples of applications in different areas are provided in [20] illustrating the practical interest of self-organizing maps. In [77] it is shown that self-organizing maps are a powerful tool for visualisation of high-dimensional data. A fraud detection method based on the SOM visualization and classification is proposed in [74]. In [76], self-organizing maps are used to visualize and classify complex geologic data. Moreover, self-organizing maps have proven to be of considerable value in finance, where they are able to structure, analyze, and visualize large amounts of multidimensional financial data in a significant manner [22, 27].

While self-organizing maps provide interpretable results with a global view of multidimensional data, they have become very popular in many application areas to easily detect phenomena in data. Furthermore, self-organizing maps are widely used in chemometrics [34, 60, 105, 52, 5, 21], but also in biology [67], humanities [66], industry, such as health monitoring of aircraft engines [18]. Many variants of the standard self-organizing map have been developed, such as Generative Topographic Mapping [6], which is a probabilistic version of the self-organizing map, or extensions to more complex data types (mixed, textual, etc.) [50, 57] demonstrating the relevance of self-organizing maps until today.

Modern data acquisition based on high-throughput technology is often confronted with the problem of missing data. The absence of certain information poses a significant problem for analysts, as the information is incomplete and therefore less reliable. It is necessary to properly process missing data before performing statistical analyses.

Data may be incomplete for a large variety of reasons. In surveys, for instance, they occur due to non-responses to questions that affect privacy [112, 7, 69]. In industrial applications and chemometrics, measuring instruments may have malfunctions or detection limits yielding erroneous and missing entries [54, 26]. In medical research, missing data can occur in clinical trials when patients abandon or stop taking the treatment for a certain period of time [88, 61, 13]. Missing data also frequently occur in chemical [56, 14] and environmental [89] studies. Moreover, concerning huge databases, merging several datasets from different sources can also result in missing data, as some entries may not be recorded at all for some of the sources.

The impact of missing data on statistical results can be serious, leading to biased

estimates, loss of information and weakened generalizability of findings. However, for a long time, in statistics, missing data have been treated in a very simple and inappropriate way, either by deletion of incomplete measurements or by basic data completion by mean or median values. This has changed during the last decades, by the development of many statistical methods that account for missing data in a meaningful way. There are two general approaches to deal with missing data: either a statistical method is directly adapted to the partially observed data, or first an appropriate imputation method is applied to complete the data such that the statistical method of interest can be used on the completed or augmented data.

In this chapter, we are interested in self-organizing maps in the presence of missing data. This problem has been considered among others by Cottrell and P. [19] by simply restricting all vector calculations to the observed entries. As such, all observed data entries are taken into account in the algorithm. However, the method performs rather poorly when the number of incomplete observations with multiple missing entries is large. Moreover, the imputation of missing data is done afterwards by replacing missing entries by the closest features on the learned map. Related approaches are presented in [28, 84, 73, 1, 47] and more recently in [46, 48].

Missing data imputation can be useful to avoid incomplete data, which is crucial in data mining when methods cannot handle any missing entries. We have the ambition to combine the tasks of imputation and learning the map by a principled approach. Our motivation is the fact that any non trivial imputation method is based on some data model, and so it is natural to use the self-organizing map for imputation. Conversely, a better map may be learned when data are complete. Thus, treating both tasks simultaneously may be beneficial for the two of them.

Our approach can be viewed as an extension of the standard Kohonen algorithm for self-organizing maps.

A mathematical presentation of the method, a new loss function that encodes our double goal of imputation and learning a self-organizing map and two algorithmic solutions are given in Section 2.2. Moreover, Section 2.3 provides an extensive numerical study assessing the robust performance of our method in various settings and in comparison to alternative methods from the literature. An application to fan blade measurements is also provided in Section 2.4 and shows the practical interest of missSOM.

2.2 Self-organizing maps with incomplete data

In this section we first formally state the classical self-organizing map, before introducing the new loss function and two algorithms for the computation of self-organizing maps with partially observed data and missing data imputation.

Algorithm 1: Standard Kohonen algorithm

Input: Data matrix X, size and topology of the map, neighborhood function V_{λ_t}, sequence of radii $(\lambda_t)_{0\leq t\leq T}$ and learning steps $(\varepsilon_t)_{0\leq t\leq T}$.

Initialize code vectors $W^{(0)}$;

Initialize the counter of iterations: $t = 0$;

while *not converged* **do**

  Increment t: Set $t = t + 1$;

  Choose an observation $i \in \{1, \ldots, n\}$ randomly ;

  Assignment: Compute winning neuron $\ell = h(x_i, W^{(t-1)})$;

  Update code vectors:

  for $k = 1, \ldots, K$ **do**

    $w_k^{(t)} = w_k^{(t-1)} + \varepsilon_t V_{\lambda_t}(k, \ell)\left(x_i - w_k^{(t-1)}\right)$.

  end

end

Output: Code vectors $W^{(t)}$.

2.2.1 Classical Kohonen algorithm

The data matrix containing the measurements is denoted by $X = [x_1, \ldots, x_n] \in \mathbb{R}^{n\times p}$. A self-organizing map discretizes the input space by nonlinearly projecting the high-dimensional data X onto a low-dimensional subspace in a way that preserves topological properties of the data. This subspace is usually a two-dimensional map represented as a regular grid composed of K fixed neurons. The arrangement of the neurons on the map is given by some neighborhood function $V_\lambda : \{1, \ldots, K\}^2 \mapsto \mathbb{R}_+$. The neighborhood radius $\lambda > 0$ describes the zone of influence around a neuron and decreases during the Kohonen algorithm. The best prototype vectors of the self-organizing map are defined as the minimum of the loss function F defined by

$$F(W) = \frac{1}{2n}\sum_{i=1}^{n}\sum_{k=1}^{K} V_\lambda(k, h(x_i, W))\|x_i - w_k\|_2^2, \tag{2.1}$$

where $W = [w_1, \ldots, w_K] \in \mathbb{R}^{p\times K}$ is the matrix of K prototype vectors and $h : \mathbb{R}^p \times \mathbb{R}^{p\times K} \mapsto \{1, \ldots, K\}$ denotes the allocation function, attributing the closest prototype to a data point x w.r.t. the Euclidean distance, defined as

$$h(x, W) = \arg\min_{1\leq k\leq K} \|x - w_k\|_2. \tag{2.2}$$

The loss F takes into account all distances between every measurement and all code vectors, weighted by the neighborhood function evaluated on the corresponding neurons. As a result, code vectors that minimize the loss are similar if they are close on the map. In the specific case where the neighborhood function satisfies

$V_\lambda(k,\ell) = 0$ for all $k \neq \ell$, the loss F is the criterion minimized by the k-means algorithm. That is, clusters obtained by k-means are independent, while prototypes of a self-organizing map are organized in a topological way.

To compute the minimum of loss F, Ritter et al. [79] showed that in the given framework a gradient descent algorithm can be used, referred to as the Kohonen stochastic algorithm. For a randomly picked observation x_i with winning neuron $\ell = h(x_i, W^{(t)})$, the updates of the code vectors are given by

$$w_k^{(t+1)} = w_k^{(t)} + \varepsilon_t V_\lambda(k,\ell)(x_i - w_k^{(t)}), \tag{2.3}$$

where $(\varepsilon_t)_{t\geq 0}$ is a sequence of decreasing learning steps. This update attracts all prototypes towards observation x_i. The attraction is the strongest at the beginning of the algorithm, when ε_t is large, and when the winning neuron ℓ is close to prototype k on the map. Those updates eventually result in an ordered map, where neighboring neurons have similar prototype vectors. It is also common to shrink the neighborhood by using a decreasing sequence of radii $(\lambda_t)_{t\geq 0}$ in the neighborhood function V_λ. The algorithm is summarized in Algorithm 1.

2.2.2 Notation for incomplete data

Now we consider an incomplete $n \times p$ data matrix containing missing values. Let the matrix $M = (m_{i,j})_{i,j} \in \{0,1\}^{n\times p}$ be the missing-data pattern which indicates where the entries are missing or masked, and that is defined by

$$m_{i,j} = \begin{cases} 1 & \text{if } x_{i,j} \text{ is observed} \\ 0 & \text{if } x_{i,j} \text{ is missing} \end{cases}$$

We denote X^{obs} the set of observed data values and X^{miss} the set of non-observed data entries hidden by the missing-data pattern M. The complete data are denoted by $X^{\text{compl}} = (X^{\text{obs}}, X^{\text{miss}})$. Likewise, for the observation vector x_i we denote by x_i^{obs} and x_i^{miss} the observed and unobserved entries, respectively, and, with some abuse of notation, $x_i^{\text{compl}} = (x_i^{\text{obs}}, x_i^{\text{miss}})$ is the complete vector, which also corresponds to the i-th row of X^{compl}.

Our goal is to adapt the model of self-organizing maps to partially observed data, and moreover, learn the values of the missing data. The motivation to treat these tasks simultaneously is that learning missing values requires a data model, and as we are interested in self-organizing maps it is natural to use this model for data imputation. At the same time, learning a map with completed data may give better results compared to using only the observed part X^{obs} of the data.

2.2.3 New loss function

We introduce a new loss function that considers both problems : finding the best self-organizing map and the best values for imputation of the missing data. In other

words, by minimizing the new loss function F_{missom} we search for both the best code vectors $W \in \mathbb{R}^{p\times K}$ for the map and the best values for the missing data denoted by X^* chosen in the set of all possible values for the missing entries $\mathcal{X}^{\text{miss}}$.

To define the new criterion, an adaptation of the definition of the winning neuron is in order. In the presence of missing values, it is natural to restrict the Euclidean distance in (2.2) only to the observed entries. More precisely, for any vectors $x^{\text{obs}} \in \mathbb{R}^{p'}$ $(p' \leq p)$, $m \in \{0,1\}^p$ with $\sum_{j=1}^{p} m_j = p'$ and code vectors $W \in \mathbb{R}^{p\times K}$, we set

$$h^{\text{miss}}(x^{\text{obs}}, m, W) = \arg\min_{1\leq k\leq K} \|x^{\text{obs}} - w_k \odot m\|_2,$$

where $w_k \odot m$ denotes the p'-vector made of the elements $w_{k,j}$ of w_k such that $m_j = 1$. Now, we define the new loss as

$$F_{\text{missom}}(W, X^*) = \frac{1}{2n}\sum_{i=1}^{n}\sum_{k=1}^{K} V_\lambda\left(k, h^{\text{miss}}(x_i^{\text{obs}}, m_i, W)\right)\left\|(x_i^{\text{obs}}, x_i^*) - w_k\right\|_2^2,$$

where $m_i \in \mathbb{R}^p$ is the i-th row of the matrix M and $(x_i^{\text{obs}}, x_i^*)$ denotes the i-th measurement vector completed with x_i^*. Since

$$\left\|(x_i^{\text{obs}}, x_i^*) - w_k\right\|_2^2 = \left\|x_i^{\text{obs}} - w_k \odot m_i\right\|_2^2 + \|x_i^* - w_k \odot (\mathbf{1}_p - m_i)\|_2^2,$$

where $\mathbf{1}_p = (1, \ldots, 1)^T \in \mathbb{R}^p$, the criterion F_{missom} can be decomposed into two parts according to the observed and the missing entries as

$$F_{\text{missom}}(W, X^*) = F_{\text{obs}}(W) + F_{\text{miss}}(W, X^*),$$

where $F_{\text{obs}}(W)$ is the part of the loss over the observed entries given by

$$F_{\text{obs}}(W) = \frac{1}{2n}\sum_{i=1}^{n}\sum_{k=1}^{K} V_\lambda\left(k, h^{\text{miss}}(x_i^{\text{obs}}, m_i, W)\right)\left\|x_i^{\text{obs}} - w_k \odot m_i\right\|_2^2,$$

and $F_{\text{miss}}(W, X^*)$ is the contribution of the imputed values X^* to the loss, defined as

$$F_{\text{miss}}(W, X^*) = \frac{1}{2n}\sum_{i=1}^{n}\sum_{k=1}^{K} V_\lambda\left(k, h^{\text{miss}}(x_i^{\text{obs}}, m_i, W)\right)\|x_i^* - w_k \odot (\mathbf{1}_p - m_i)\|^2.$$

Note that in the complete-data case, where the missing-data pattern is $M = \mathbf{1}_{n\times p}$, $F_{\text{miss}}(W, X^*) = 0$ for any W and any X^*, so that the criterion F_{missom} is equal to the one of the classical self-organizing map, that is, $F_{\text{missom}}(W, X^*) = F(W)$.

Algorithm 2: missSOM algorithm

Input: Incomplete data X^{obs}, missing-data pattern M, size and topology of the map, neighborhood function V_λ, sequence of radii $(\lambda_t)_{0\leq t\leq T}$ and learning steps $(\varepsilon_t)_{0\leq t\leq T}$.

Initialize imputed values $X^{*(0)}$ and code vectors $W^{(0)}$;

Initialize the counter of iterations: $s = 0$;

while *not converged* **do**

Increment s: Set $s = s + 1$;

Update code vectors by Kohonen Algorithm 1 on the augmented data with winning neurons obtained by h^{miss} instead of h:

$W^{(s)} \leftarrow \text{Kohonen}(X^{\text{aug}} = (X^{\text{obs}}, X^{*(s-1)}))$;

Update imputed values: for i, j such that $m_{i,j} = 0$,

$$x_{i,j}^{*(s)} = \frac{\sum_{k=1}^{K} V_{\lambda_T}(k, h^{\text{miss}}(x_i^{\text{obs}}, m_i, W^{(s)}))w_{k,j}^{(s)}}{\sum_{k=1}^{K} V_{\lambda_T}(k, h^{\text{miss}}(x_i^{\text{obs}}, m_i, W^{(s)}))}.$$

end

Output: Code vectors $W^{(s)}$ and imputed data $X^{*(s)}$.

2.2.4 Minimization algorithm

For the minimization of $(W, X^*) \mapsto F_{\text{miss}}(W, X^*)$ on $\mathbb{R}^{K\times p} \times \mathcal{X}^{\text{miss}}$ we propose to alternate the minimization in W and X^* while keeping the other argument fixed.

For fixed X^*, the function $W \mapsto F_{\text{missom}}(W, X^*)$ is similar to the objective function F in (2.1) in the complete-data case applied to the augmented data $X^{\text{aug}} = (X^{\text{obs}}, X^*)$. The only difference lies in the definition of the winning neurons by h^{miss} that appear in the neighbourhood function V_λ. Thus, a Kohonen algorithm applied to X^{aug} can be used to find the best code vectors W.

In turn, when W is fixed, the minimization of $X^* \mapsto F_{\text{missom}}(W, X^*)$ boils down to minimize $X^* \mapsto F_{\text{miss}}(W, X^*)$. This problem has a unique explicit solution given for all $1 \leq i \leq n$ and j such that $m_{i,j} = 0$ by

$$x_{i,j}^* = \frac{\sum_{k=1}^{K} V_\lambda(k, h^{\text{miss}}(x_i^{\text{obs}}, m_i, W))w_{k,j}}{\sum_{k=1}^{K} V_\lambda(k, h^{\text{miss}}(x_i^{\text{obs}}, m_i, W))}. \tag{2.4}$$

That is, the imputed values are a weighted mean of the prototype vectors weighted according to the neighborhood function.

To summarize, the algorithm updates imputed values for the missing data and applies the classical Kohonen algorithm with adjusted winning neuron function h^{miss} to learn the map. This is repeated until convergence or until a maximum number of iterations chosen by the user is attained. The algorithm is described in Algorithm 2.

As initial values for the imputed values X^*, one can simply impute the sample mean or median of the variables obtained over the observed entries.

Algorithm 3: Accelerated missSOM algorithm

Input: Incomplete data matrix X^{obs}, missing-data pattern M, size and topology of the map, neighborhood function V_λ, sequence of radii $(\lambda_t)_{0\leq t\leq T}$ and learning steps $(\varepsilon_t)_{0\leq t\leq T}$.

Initialize imputed values $X^{*(0)}$ and code vectors $W^{(0)}$;

Initialize the number of epochs: $t = 0$;

while *not converged* **do**

  Increment t: Set $t = t + 1$;

  Set $\tilde{W}^{(0)} = W^{(t-1)}$;

  for $i = 1, \ldots, n$ **do**

    Assignment: Compute winning neuron $\ell = h^{\text{miss}}(x_i^{\text{obs}}, m_i, \tilde{W}^{(i-1)})$;

    Update code vectors:

    for $k = 1, \ldots, K$ **do**

      $w_k^{(i)} = \tilde{w}_k^{(i-1)} + \varepsilon_t V_{\lambda_t}(k, \ell)\left((x_i^{\text{obs}}, x_i^{*(t-1)}) - \tilde{w}_k^{(i-1)}\right)$.

    end

  end

  Set $W^{(t)} = \tilde{W}^{(n)}$;

  Update imputed values: for i, j such that $m_{i,j} = 0$,

$$x_{i,j}^{*(t)} = \frac{\sum_{k=1}^{K} V_{\lambda_t}(k, h^{\text{miss}}(x_i^{\text{obs}}, m_i, W^{(t)}))w_{k,j}^{(t)}}{\sum_{k=1}^{K} V_{\lambda_t}(k, h^{\text{miss}}(x_i^{\text{obs}}, m_i, W^{(t)}))}.$$

end

Output: Code vectors $W^{(t)}$ and imputed data $X^{*(t)}$.

2.2.5 Accelerated version

Algorithm 2 happens to be expensive in terms of computing time, in particular when the number of iterations is large, since the entire standard Kohonen algorithm is carried out during each iteration. A speed up is obtained by interwining updates of the missing data and the iterations of the Kohonen Algorithm 1. More precisely, we propose to update the missing data at every epoch, that is, after every pass through the data. This procedure gives rise to Algorithm 3. As such, the Kohonen algorithm is carried out only once, while in the initial Algorithm 2, the entire Kohonen algorithm is applied repeatedly. Thus the computing time of the accelerated version of missSOM is comparable to the computing time of the standard Kohonen Algorithm 1, since the update of the imputed values is fast.

While the first version of the missSOM algorithm has some theoretical justification, the accelerated version lacks this foundation. A numerical study given in Section 2.2.6 shows that the Algorithms 2 and 3 provide very similar maps and hence justifies the utilization of the accelerated version, which achieves a significant

	5%			20%			40%		
	MCAR	MAR	MNAR	MCAR	MAR	MNAR	MCAR	MAR	MNAR
Topographic error									
basic missSOM	0.337 (0.047)	0.345 (0.035)	0.336 (0.037)	0.313 (0.032)	0.355 (0.063)	0.339 (0.046)	0.294 (0.058)	0.303 (0.057)	0.306 (0.083)
accelerated missSOM	0.352 (0.039)	0.345 (0.049)	0.338 (0.030)	0.301 (0.035)	0.299 (0.039)	0.322 (0.042)	0.282 (0.035)	0.280 (0.059)	0.285 (0.040)
Quantization error									
basic missSOM	0.405 (0.115)	0.394 (0.042)	0.407 (0.145)	0.327 (0.115)	0.326 (0.116)	0.330 (0.112)	0.225 (0.074)	0.233 (0.080)	0.233 (0.077)
accelerated missSOM	0.413 (0.149)	0.408 (0.139)	0.407 (0.138)	0.351 (0.117)	0.340 (0.116)	0.341 (0.115)	0.261 (0.081)	0.245 (0.071)	0.260 (0.074)
Imputation error									
basic missSOM	0.601 (0.140)	0.599 (0.154)	0.613 (0.171)	0.659 (0.125)	0.686 (0.141)	0.759 (0.108)	0.758 (0.093)	0.795 (0.095)	0.871 (0.091)
accelerated missSOM	0.615 (0.097)	0.624 (0.096)	0.654 (0.103)	0.665 (0.092)	0.694 (0.087)	0.750 (0.075)	0.742 (0.069)	0.793 (0.069)	0.848 (0.072)
ARI									
basic missSOM	0.922 (0.092)	0.929 (0.089)	0.937 (0.071)	0.849 (0.088)	0.846 (0.108)	0.823 (0.089)	0.686 (0.108)	0.640 (0.101)	0.623 (0.086)
accelerated missSOM	0.923 (0.090)	0.923 (0.086)	0.922 (0.082)	0.841 (0.092)	0.845 (0.081)	0.805 (0.090)	0.647 (0.094)	0.629 (0.109)	0.591 (0.089)
Computing time									
basic missSOM	33.169 (0.546)	32.940 (0.737)	31.860 (0.396)	33.076 (0.517)	32.988 (0.776)	32.948 (0.530)	33.031 (0.583)	31.920 (0.061)	32.846 (0.0644)
accelerated missSOM	0.394 (0.010)	0.387 (0.006)	0.376 (0.009)	0.557 (0.008)	0.561 (0.017)	0.556 (0.006)	0.790 (0.016)	0.753 (0.011)	0.784 (0.022)

Table 2.1: Comparison of the basic missSOM Algorithm 2 and its accelerated version Algorithm 3 in terms of different errors and computing time (in seconds) on the gaussian mixture data.

gain in computing time.

Note that while the selection of the observations x_i in Algorithm 3 is deterministic, it is possible to use a random selection scheme.

2.2.6 Validation of accelerated missSOM algorithm

Table 2.1 compares the results of the basic missSOM Algorithm 2 and its accelerated version Algorithm 3 on the simulated gaussian mixture data in various conditions. On the one hand, for all settings the errors are totally equivalent. This indicates that the accelerated version provides the same self-organizing maps and very similar imputations as the basic missSOM algorithm. On the other hand, we see that in terms of computing time we gain two orders of magnitude. Hence, the use of the accelerated algorithm instead of the basic version is completely justified.

2.3 Numerical experiments

In this section, the performance of the proposed method missSOM is evaluated and compared to alternative methods. A simulation study is conducted to assess the quality of the representation of the data via the map and the accuracy of imputed values under various conditions.

2.3.1 Performance criteria

Let W^* be the code vectors of the final self-organizing map, X^* the imputed values and $\hat{x}_i = (x_i^{\text{obs}}, x_i^*)$ the completed observation vectors.

The quality of the map as a representation of the data can be evaluated by two criteria. First, the *quantization error* defined as the average of the squared distances between the observations and their nearest prototype vector given by

$$E = \frac{1}{n_{\text{obs}}} \sum_{i=1}^{n} \|x_i^{\text{obs}} - w_{h^{\text{miss}}(x_i^{\text{obs}}, m_i, W^*)}\|_2^2,$$

where $n_{\text{obs}} = \sum_{i,j} m_{i,j}$ is the number of observed entries in the data, informs on whether the prototype vectors are good representations of the data. Second, the *topographic error* evaluates the preservation of the topology of the data in the map by the proportion of observations for which the winning neuron and the second closest neuron are not neighbors, i.e. not connected on the grid. It is defined as

$$T = \frac{1}{n} \sum_{i=1}^{n} e(\hat{x}_i),$$

where

$$e(\hat{x}_i) = \begin{cases} 0 & \text{if the neurons } h^{\text{miss}}(x_i^{\text{obs}}, m_i, W^*) \text{ and } v^{\text{miss}}(x_i^{\text{obs}}, m_i, W^*) \text{ are adjacent} \\ 1 & \text{otherwise} \end{cases}$$

with $v^{\text{miss}}(x_i^{\text{obs}}, m_i, W^*) = \arg\min_{j \neq h^{\text{miss}}(x_i^{\text{obs}}, m_i, W^*)} \|x_i^{\text{obs}} - w_k^* \odot m_i\|_2^2$ the second closest neuron to measurement x_i^{obs}. The topographic error is expected to be small when the map is well organized and ordered. As both errors are computed only on the observed entries, the focus is put on the quality of the map with respect to the observed part of the data. The accuracy of the imputed values is assessed separately by the *imputation error*, which is defined as the root mean square error and quantifies the quality of the imputed values compared to the true missing values.

2.3.2 Datasets

For the numerical experiments, two settings are considered. First, the dataset *wines* from the UCI machine learning repository [25] is used, which contains the results of a chemical analysis of 178 wines on 13 quantitative variables. We generate 100 perturbed datasets by adding gaussian noise with mean 0 and standard deviation equal to one tenth of the mean value in each variable.

In the second setting, 100 datasets are simulated from a multivariate gaussian mixture with dimension $p = 5$, four equal-sized groups and $n = 2000$ observations. The correlation among all pairs of components is equal to 0.5 and for every dataset, the 4 gaussian means are drawn independently from a centered normal distribution with standard deviation equal to 5.

In both settings, missing values are generated using the `ampute` function from the R package `mice` [101]. Different proportions of missing values, namely 5%, 20% and 40%, and different mechanisms of missingness are considered. The literature on missing data traditionally distinguishes the reasons why data are missing. Rubin [83] introduces three mechanisms that lead to missingness and it is well known that the performance of imputation methods may be sensitive to the mechansim at

work. The mechanism is said to be *missing completely at random* (MCAR), when the causes of missing values are independent from the data and the probability of being absent is the same for all items. That is, a subset of observations is chosen at random using independent Bernoulli variables with fixed success probability for all entries. In contrast, when the probability that a value is missing depends on the values of the observed variables, the mechanism is called *missing at random* (MAR). In our simulations, to obtain MAR, for each variable, missing values are obtained using a logistic regression model depending only on the other variables. Finally, when the probability of being absent also depends on the unobserved value, the mechanism is called *missing not at random* (MNAR) and we can consider missing data simulated by using a logistic regression model depending on all variables.

2.3.3 Alternative methods

Our study is twofold. On the one hand, we compare *missSOM* to the state of the art on self-organizing maps. The simplest method (referred to as *deletion* in the figures) consists in deleting the observations containing missing values and applying the standard Kohonen algorithm to the remaining data. Incomplete observations are classified once the map is built by assigning them to their closest prototypes and missing entries are imputed by the corresponding values of the winning prototypes. *Cottrell*'s approach is a variant of the classical self-organizing maps appropriate to deal with incomplete observations [19]. *Cottrell* adapts the Kohonen algorithm by simply restricting all vector calculations to the observed entries. The main difference with *missSOM* is that data imputation is performed only after learning the map by imputation with the values of the closest prototypes.

On the other hand, in our experiments, we compare *missSOM* to the state of the art on missing data imputation. The literature provides numerous general imputation methods. A basic approach (here referred to as *mean*) is the imputation by the mean value of the observed variables. A non-parametric approach called *missForest* predicts missing values using a random forest trained on the observed part of the dataset. Moreover, the method referred to as *knn* is a k-nearest neighbors approach, which is implemented in the `VIM` package [51]. Missing values are imputed iteratively by a weighted average of the k closest observations. Finally, assuming a gaussian mixture model for the data, the `Amelia` package [38] performs imputation using the expectation-maximization algorithm and a bootstrap approach to iteratively estimate missing values. In our numerical study, the imputation error of missSOM is compared to the one obtained by all these methods. In addition, we also compare the map obtained by missSOM with the maps obtained by the classical Kohonen algorithm applied to the complete dataset, where missing values are imputed by the aforementioned imputation methods.

As a benchmark we consider a self-organizing map trained on the complete data set (corresponding to a missing rate of 0%) and compute the error rates only on the

observed values that are provided to the other methods. This method is referred to as *complete-data SOM*.

2.3.4 Parameters of the methods

The missSOM algorithm is applied with the default parameters in the package `missSOM`. That is, the neighborhood function is a gaussian and the sequence $(\lambda_t)_{t\leq T}$ decreases from $\lambda_0 = 4.58$ to $\lambda_T = 0.5$ for the wine dataset and from $\lambda_0 = 8.89$ to $\lambda_T = 0.5$ for the gaussian mixture data. The maximum number of epochs is $T = 100$. Concerning the map, it has a hexagonal topology and the grid is composed of $K = 9 \times 7$ neurons for the wines data and $K = 16 \times 14$ neurons for the gaussian mixture data. The choice of the size of the map depends on the sample size and on the objectif of the analysis. In general, missSOM has the same parameters as classical SOM and they can be chosen in the same way as for SOM.

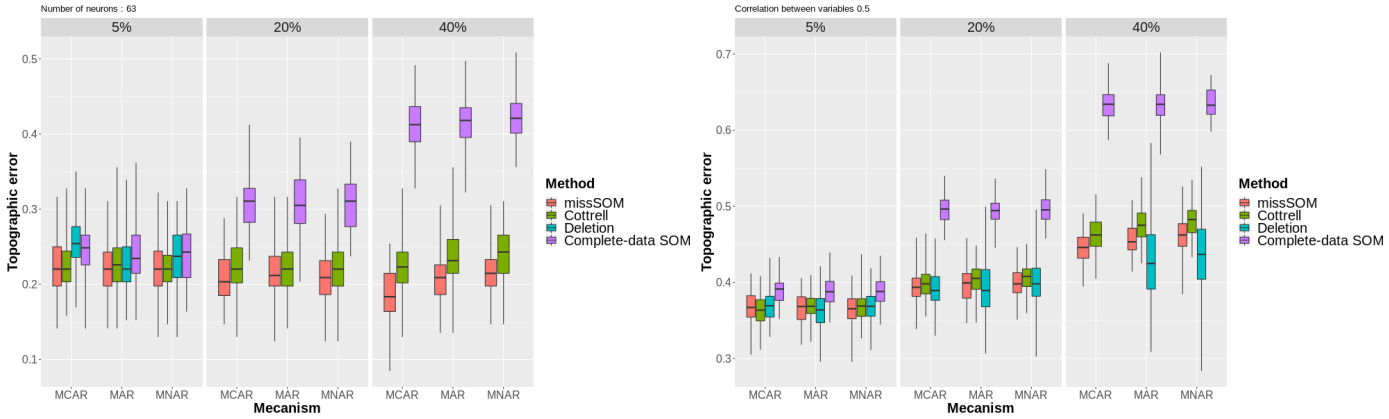
For the alternative methods, the maximum number of iterations is set to 100 and for the other parameters the default values are used.

2.3.5 Comparison to the state of the art on self-organizing maps

Figure 2.1 and 2.2 present the boxplots of the topographic and the quantization error for all SOM-type methods in both settings. They serve to evaluate the quality of the representation of the data using a self-organizing map computed with the different approaches. The corresponding imputation errors in Figure 2.3 allow to judge their performances as imputation methods.

First, we observe that when the percentage of missing data is low, most methods have very similar performance. With increasing percentage of missingness, the problem becomes harder and differences among the methods appear. But as errors are evaluated only on the observed part of the data, errors are not necessarily increasing. Namely quantization errors for the wines data decrease with increasing missingness. This may be a consequence of the higher variance and less structure of the wines data compared to the Gaussian mixture data. So when deleting entries from the wines data, the data variance decreases and the resulting map is a better representation of the observed part of the data. Concerning the mechanisms of missingness, it appears that the quality of the map does not depend on it for any SOM-type method. However, imputation is impacted by the mechanism. Imputation is the easiest under MCAR and the hardest under MNAR.

Next, we see that the *deletion* method is very unstable. In some scenarios its error rates are among the worst, and, more importantly, when too many values are missing, the method breaks down and does not produce any result. Indeed, on small datasets as wines with 20% of missingness, the number of complete observations is smaller than the size of the map and thus classical SOM is just not applicable. On the



(a) Wines data

(b) Gaussian mixture data

Figure 2.1: Topographic error of SOM-type methods for various amounts and mechanisms of missingness on the wines (a) and the gaussian mixture data (b).

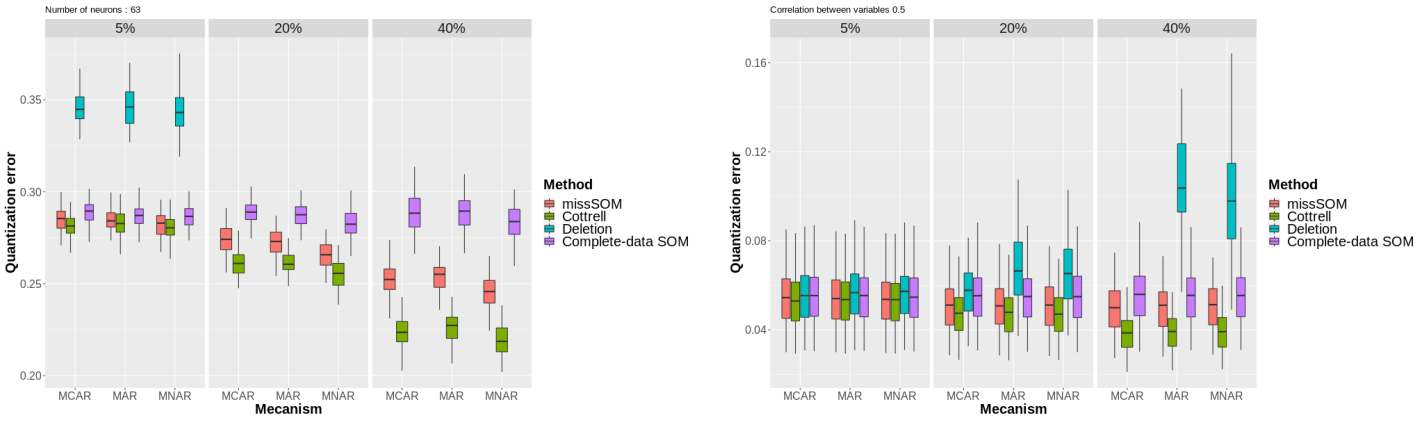
large gaussian mixture dataset, the *deletion* method produces the best topographic error, but the associated quantization error is disastrous, disqualifying the approach.

The quantization error of the *complete-data SOM* method is constant when varying the amount of missingness, because the underlying map remains the same. For the topographic error, an increase is observed which is due to the definition of the error. The error determines the closest and second closest prototypes only with respect to the observed entries, while the map was optimized by taking into account the complete data. This explains why *missSOM* and *Cottrell* have lower topographic errors, as their maps are learned with a notion of closeness restricted on the observed entries.

Finally, we observe that *Cottrell*'s method always achieves the best quantization errors, directly followed by *missSOM*. Concerning the topographic error, *missSOM* is consistently doing better than *Cottrell*. Thus, in terms of quality of the map and representation of the data, none of the methods outperforms all others, and *Cottrell* and *missSOM* have both a good global performance. Now, considering the imputation error, there is a clear winner. When the proportion of missingness is important, *missSOM* outperforms all other methods in every scenario. This confirms us in the use of *missSOM* with respect to *Cottrell*, especially when accurate imputation is desired.

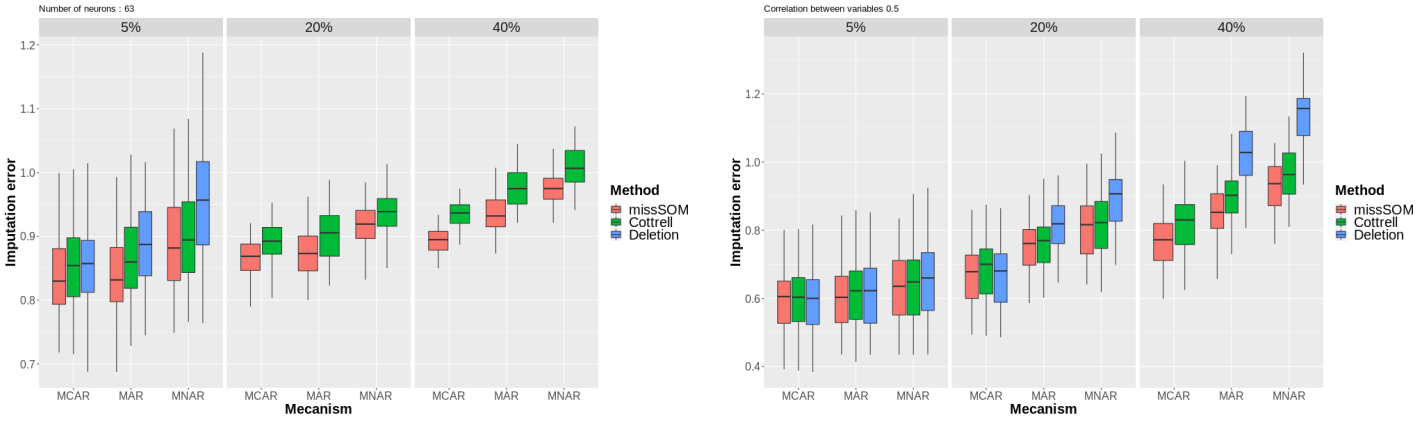
2.3.6 Comparison to the state of the art on missing-data imputation

In the second part of the simulation study, *missSOM* is compared to general imputation methods. Figure 4.1 shows that as an imputation method *missForest* is unbeaten regardless of the missingness mechanism and the percentage of missing



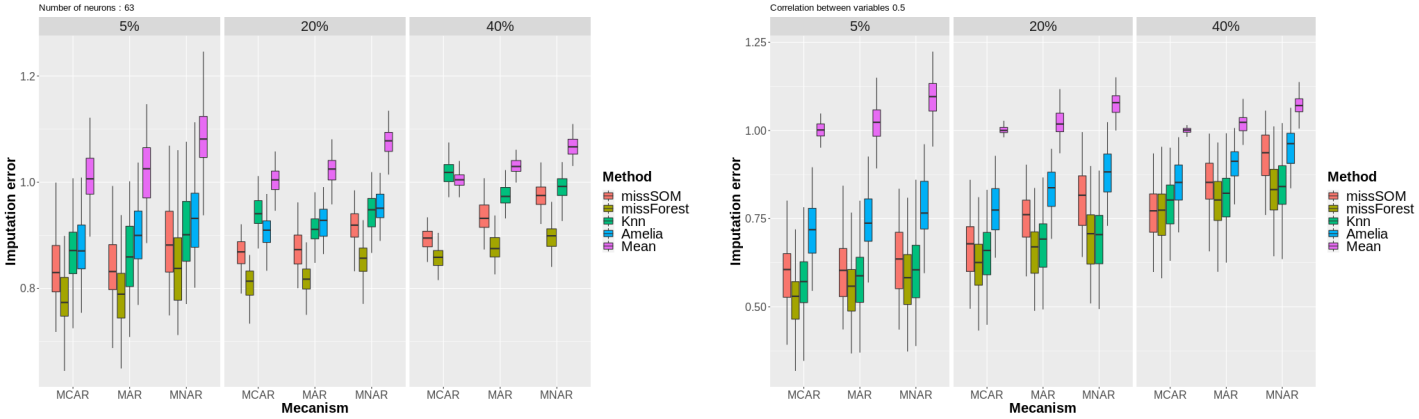
(a) Wines data (b) Gaussian mixture data

Figure 2.2: Quantization error of SOM-type methods for various amounts and mechanisms of missingness on the wines (a) and the gaussian mixture data (b).



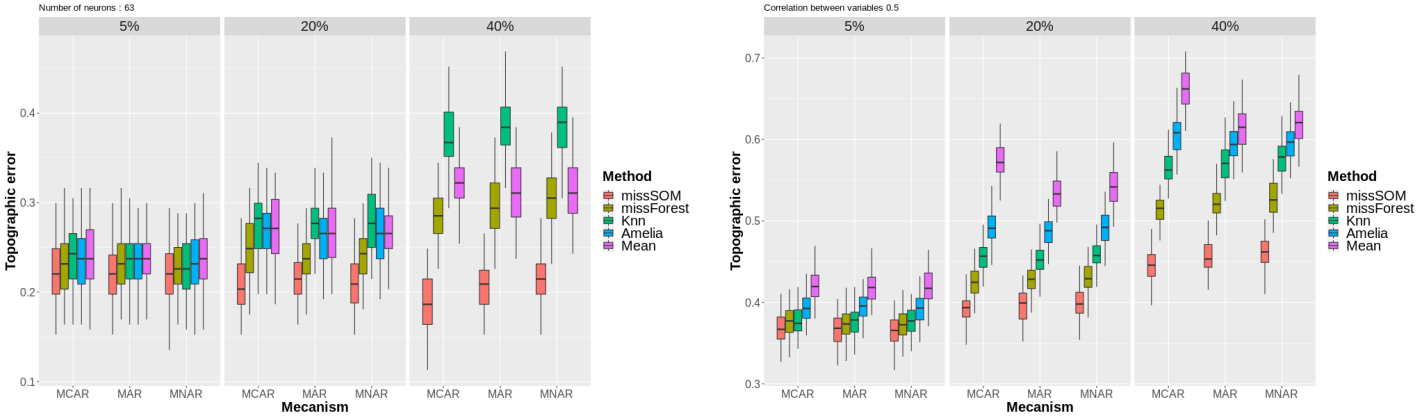
(a) Wines data (b) Gaussian mixture data

Figure 2.3: Imputation error of SOM-type methods for various amounts and mechanisms of missingness on the wines (a) and the gaussian mixture data (b).



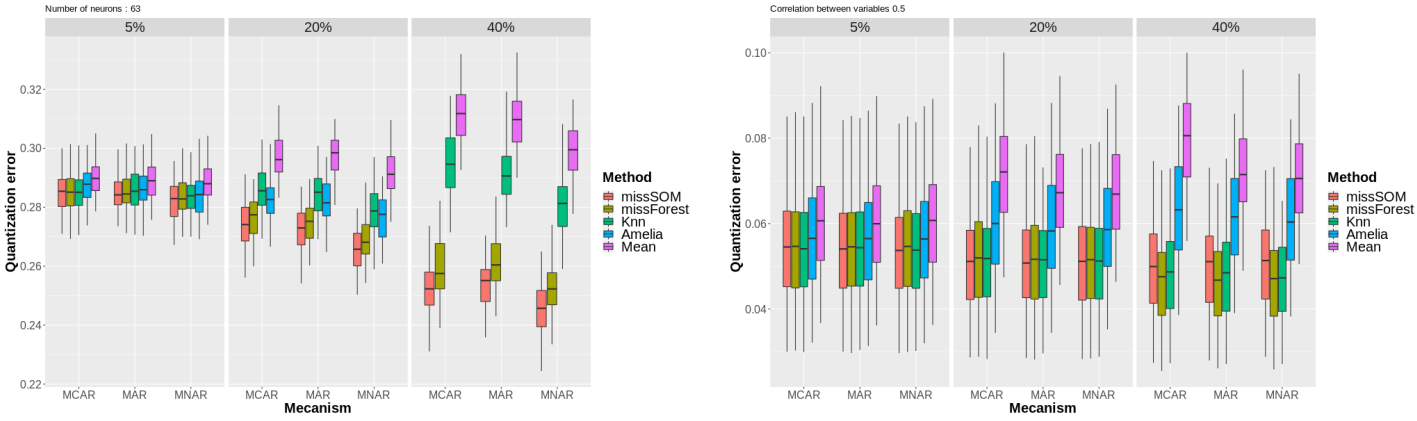
(a) Wines data (b) Gaussian mixture data

Figure 2.4: Imputation error of *missSOM* and classical imputation methods for various amounts and mechanisms of missingness on the wines (a) and the gaussian mixture data (b).



(a) Wines data (b) Gaussian mixture data

Figure 2.5: Topographic error of *missSOM* and classical imputation methods for various amounts and mechanisms of missingness on the wines (a) and the gaussian mixture data (b).



(a) Wines data

(b) Gaussian mixture data

Figure 2.6: Quantization error of *missSOM* and classical imputation methods for various amounts and mechanisms of missingness on the wines (a) and the gaussian mixture data (b).

data. On gaussian mixture data, *knn* also provides accurate imputations. The imputation accuracy achieved by *missSOM* is high and *missSOM* is mostly among the best imputation methods.

Finally, Figure 2.5 and Figure 2.6 provide insights on the quality of the data representation, when classical SOM is applied to the data with missing values imputed by the different imputation methods. In this respect, *missSOM* clearly outperforms all the other methods. In particular, *missForest* provides self-organizing maps of poorer quality, especially in terms of preservation of the topology of the input data. This confirms the initial motivation of *missSOM*: in the presence of missing values and when a self-organizing map is required, it is better to learn the map simultaneously with the imputed values than to treat the two tasks separately.

A further advantage of *missSOM* is its computing time, which is about 10 times shorter than the one of *missForest*.

To summarize, we have seen that *missSOM* is the method of choice when both tasks data imputation and data representation are desired. However, when the focus is only on data imputation, then better alternatives as *missForest* exist.

2.4 Application to fan blade metrology

In this section, we illustrate our method by analyzing and exploring the metrology of fan blades, a necessary and useful step to better model engine performance.

2.4.1 Fan blade measurements

To control the production of parts and validate their conformity, different geometrical characteristics are measured. For fan blades, this represents a total of 323 parameters to be measured at different points over the entire height of the part. The databasis, which contains 14180 fan blades, is then quite voluminous and massive, but also in large dimension. Remember that the databasis contains missing data, so fan blades are only partially measured. This can be due to changes in specifications or acquisition problems. Obviously, erroneous measurements, which are frequent in such databases, are deleted and lead to further missing values.

Analyzing this database is essential to better understand the influence of the different parameters on the engine's performance. We propose to explore the data in order to visualise the fan blades measurements, even partially observed ones, and eventually compare the two suppliers. The self-organizing map model will therefore be used. Here we use missSOM as the method is adapted to our data and our objectives. Moreover, this approach performs well on correlated variables. In particular, imputing missing values is interesting for metrology engineers.

We explore each characteristic separately by applying missSOM to each of the characteristics. For illustration, we will show the results obtained for the geometrical characteristic measuring the intrados shape on 25 sections of the fan blades. For this characteristic, about 20% of the fan blades have missing entries, in total there are about 10% of missing values. The two qualitative variables indicating the factory and the year of measurement will not be used to compute the self-organizing map but only to visualise the data clustering. The parameters of the missSOM algorithm are as follows: the number of iterations is set to $T = 1000$, we use a grid with hexagonal topology composed of $K = 10 \times 15$ neurons, a gaussian neighborhood function with a sequence of neighborhood radii $(\lambda_t)_{t \leq T}$ decreasing from $\lambda_0 = 7.21$ to $\lambda_T = 0.1$ and a sequence of learning steps $(\varepsilon_t)_{t \leq T}$ with $\varepsilon_0 = 0.05$ and $\varepsilon_T = 0.01$.

2.4.2 Results

The quality of the map can be evaluated according to two criteria. First, the quantization error defined as the average of the squared distance between the observations and their nearest prototype informs on whether the prototype vectors are good representations of the data. Second, the topographic error evaluates the preservation of the topology of the data by the proportion of observations for which the winning neuron and the second closest neuron are not neighbors, i.e. not connected on the grid. The topographic error is small if the map is well organized and ordered. The map has a topographic error of 0.327 and a quantization error of 4.022. Self-organizing maps define a clustering of the data, where each neuron represents one cluster. Neighboring neurons can be grouped together to form larger clusters and so a clustering into a small number of groups can be derived. In our study we use hierarchical ascending classification (HAC) based on Ward's criterion [45] applied

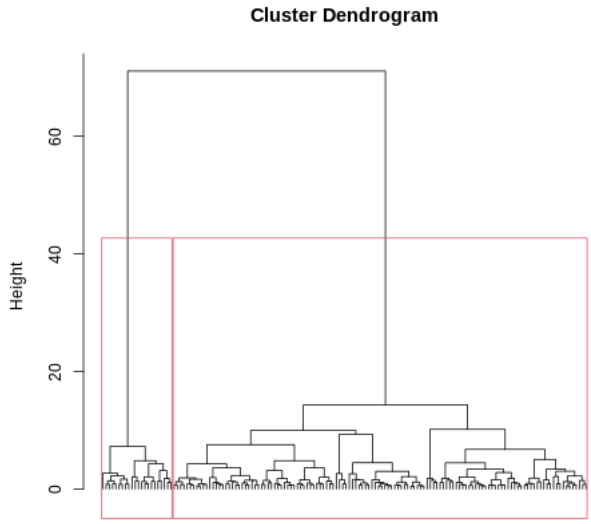


Figure 2.7: Dendrogram produced by HAC on the distances between the vector codes

to the matrix of distances between code vectors. Figure 2.7 represents the obtained dendrogram and the separation into two clusters which seems the most relevant. In Figure 2.8, the map at the top illustrates the classification results of the fan blades categorized by their manufacturing factory using different colors. The orange cluster is composed of a small number of fan blades from both the first A and the second factory B. However, the blue cluster is more dense and contains mostly fan blades manufactured by the first supplier. The fan blade factories therefore do not produce completely different blades.

Moreover, Figure 2.9 shows the same map where colors now represent the measurement date. We remark that the orange cluster contains as a whole fan blades of year 1. However, the blue cluster is more heterogeneous but is mainly composed of parts that were manufactured in year 3. Thus, production trends are present. For more precision, the month in which the fan blades were measured can also be visualized in the same way as for the year. Additional detailed analysis, particularly with the help of business experts, is required to determine the exact origin of this difference between the fan blades measurements in the year 1 and other years. The same work was done on the other geometrical characteristics of the fan blades where the conclusions are similar.

2.5 Conclusion

In this chapter, we have proposed an extension of the self-organizing map for partially observed data, referred to as missSOM. The proposed method addresses si-

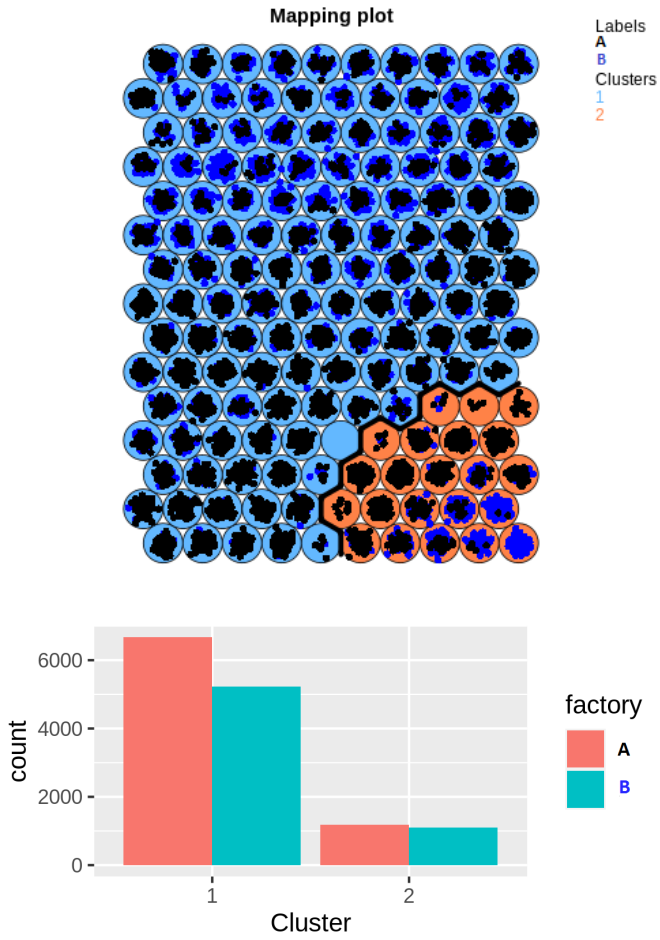


Figure 2.8: Classification of the fan blades represented by their supplier on the SOM map (at the top) and the distribution of their supplier within the clusters (at the bottom)

multaneously the two problems of computing a self-organizing map and imputing missing values. A numerical study assesses the good performance of missSOM regarding various criteria and in comparison to the state of the art. An application to measurements on fan blades aircraft engines has also been provided, which shows the practical interest of missSOM, in particular in terms of data visualisation and clustering. While this chapter focuses on the standard Kohonen algorithm, in future work we may address the transfer of our approach to other existing variants of self-organizing maps on more complex data types such as mixed data to enable them to deal with missing data.

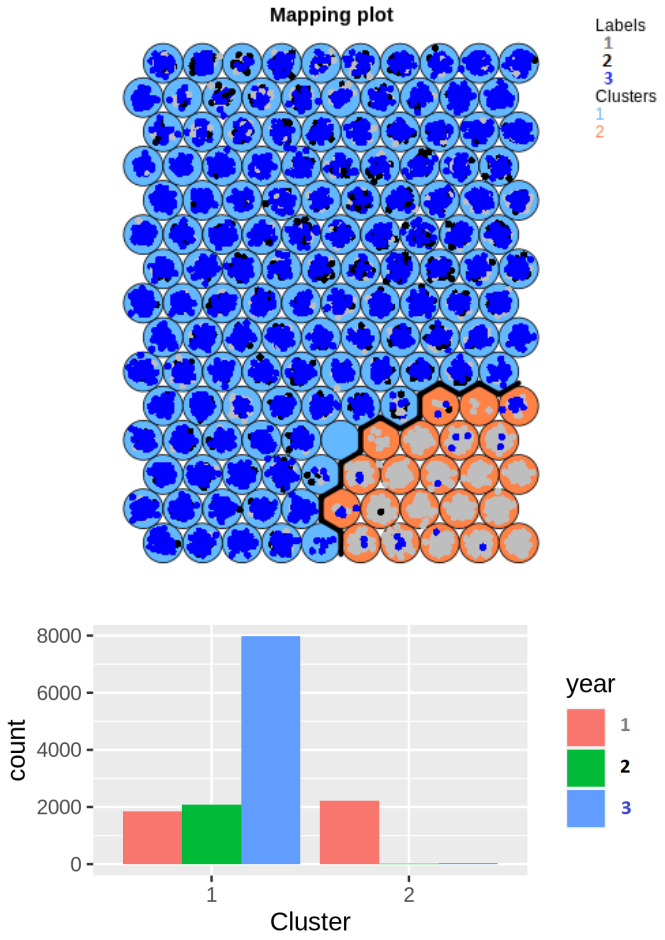


Figure 2.9: Classification of the fan blades represented by the year of their measurement on the SOM map (at the top) and the distribution of the year within the clusters (at the bottom)

Chapter 3

Modeling of turbofan's airflow

Contents

This chapter presents work carried out to model aircraft engine performance using part production. It has been the subject of a paper accepted at the *Turbomachinery Technical Conference & Exposition* with an oral presentation to be held in June 2023 in Boston [RDR23].

3.1 Introduction

After assembly of the compliant parts within the engines, the latter are tested on the acceptance test bench to validate their performance, particularly the performance of their fan. There is a delay of couple of weeks between the assembly and the acceptance tests. Forecasting the engines total airflow before they pass the acceptance test bench is very important and useful because it allows to detect and anticipate as soon as possible the production drifts that are risky for performance and thus improve production. This would make it possible to alert production more quickly to begin an initial search for the cause of a performance drift linked to the fan. With the increase in the production rate of aircraft engines, an automatic approach based on the collected data is necessary. Machine learning methods are able to automatically make prediction and forecasting, detect trends and capture even complex relationships in data. Furthermore, machine learning is widely used in the aeronautical industry for prediction [30, 94, 35] or anomaly detection [8, 41]. In other words, the goal of this work is to model the airflow in order to perform forecasting using machine learning methods. The geometric parameters will be considered as explanatory variables for the performance and it is valuable to learn the impact of each of them. In addition, we would like to justify and validate certain extensions of geometric tolerances that are too strict and often very difficult to respect. In fact, experience and acceptance tests show that despite these deviations from the model, the engines have the expected performance margins. This is mainly due to the large number of produced parts, all with slightly different dimensions, which on average, on a compressor disk, tend to compensate each other.

However, the modelisation is a challenging task due to the complexity of the data. Indeed, the geometric parameters of fan blades are numerous with the presence of strong dependencies difficult to take into account. Moreover, the engine data show trends and large variations depending on the test conditions, the manufacturing factory and the production of the parts which are constantly evolving. The modeling must therefore be adapted to the context and correctly fit to this type of data.

This chapter is organized as follows. In Section 3.2, the data pre-processing and an analysis of the engine data will first be presented that aim to reduce the measurement biases induced by the test bench equipment. In Section 3.3, we introduce notations and present some classical supervised learning algorithms for regression tasks. In Section 3.4, we consider an i.i.d. setting and apply classical machine learning regression models to build a model that relates the total airflow to the fan blade production and attempt to determine what relationships exist. In Section 3.5, we consider data over larger time intervals and thus model by times series. In particular, we explore the offline learning based on static machine learning models and the sliding window learning which consists to regularly re-train a machine learning model over time. We also analyze and model the total airflow of the engines using autoregressive and recurrent time series models where the geometric parameters of

the fan blades will be considered as exogenous variables.

3.2 Data analysis

In this section, we first present the data on aircraft engines. In particular, we show that the total airflow is biased by equipments used during the assessment test.

3.2.1 Measurement of fan performance

On aircraft engines, acceptance tests are performed to validate their performance. The performance data are recorded, in particular those of the fan characterized by three quantities: its airflow, its compression ratio and its efficiency. In the following, we are mainly interested in the total airflow measured when the engine reaches its highest speed and normalized under standard ambient conditions (pressure and temperature).

3.2.2 Bias reduction

The tests are performed on different test benches using specific auxiliary equipments such as a slave cowl and an air nozzle. These pieces have an impact on the performance measurements, as we have seen in Chapter 1 in Section 1.3.3 (particularly in Figure 1.4). There is for example an equipment drift or the maintenance operations carried out on the equipment. Biases are noticed, resulting in an increase or decrease of the expected true measurement. It is essential to reduce these large variations related to bench equipments that can have a negative impact on the airflow modeling. We use the approach proposed by Mourer and Lacaille [70] which has been tested and validated on the same data set to detect and reduce the biases mentioned above. While in the previous chapter we observed certain trends related to the parts manufacturing factory, we propose to carry out the bias correction by factory. The method makes the assumptions that the biases of the different equipments are additive, that they depend non-linearly on the test date of the engine and that the test bench equipments are independent (i.e. there is no interaction between the bench equipments). The method consists in first estimating the global production trend. This amounts to estimating the total airflow as a function of the test date using the cubic spline functions. The production trend is then removed from the data and the bias is estimated and successively removed for each equipment, starting with the slave cowls because their bias is the most significant. The estimation of the biases as a function of time is done using cubic spline functions. The measurements are debiased according to the slave cowls by subtracting from each biased measurement its estimate, which amounts to recentering the data by group, taking into account the production date. In the same way, we remove the biases of the test benches and then of the air nozzles. Thus, the biases are corrected successively. Finally,

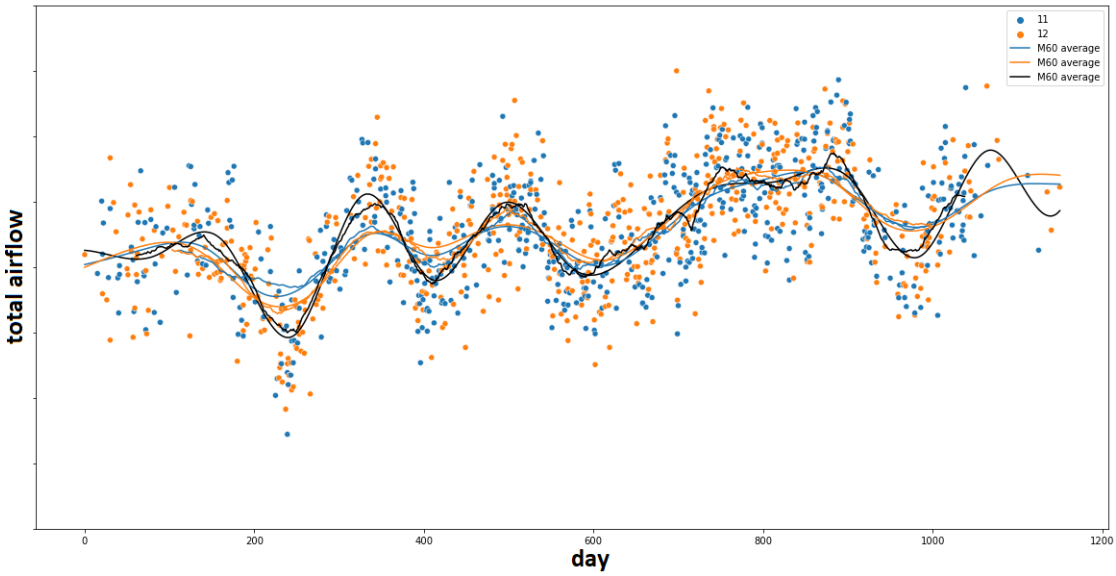


Figure 3.1: Unbiased total airflow of the tested engines as a function of the test date in days

we reintroduce the production trend by adding it to the debiased measurements. Figure 3.1 shows the debiased version of the total engine airflow categorized by the test bench. We notice in particular that the unbiased version is less noisy.

3.3 Supervised learning

3.3.1 Notation

Let Y be the target variable that represents the total airflow of the tested engine. Each fan module is represented by the state of its fan blade, evaluated by the geometrical measurements. In this study, we consider for each geometrical parameter, the average of the measurements of all the fan blades located on an engine. The input variable $\mathbf{X} = (\mathbf{X}^1, ..., \mathbf{X}^p) \in \mathbb{R}^p$ is therefore the state of the fan blades of an engine, where $p = 323$ is the dimension of the input space. Thus, $\mathbf{X}^j \in \mathbb{R}$ represents the average of the fan blade measurements for characteristic j of the engine.

We place ourselves in a regression problem where the objective is to predict the total debiased airflow of engines Y based on the fan blade measurements $\mathbf{X}$. Regression aims at estimating the function $g(\mathbf{x}) = \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$. We calculate the expected prediction error of an estimator $\hat{g}$ by $L(\hat{g}) := \mathbb{E}[(Y - \hat{g}(\mathbf{X}))^2]$ which measures the quality and relevance of the estimator $\hat{g}$. It is also the criteria to be minimized for choosing the best estimator.

In our study, we consider a sample $\mathcal{D}_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ composed of $n = 1277$ observations drawn from $(\mathbf{X}, Y)$ whose probability density f is unknown.

The empirical error using $\mathcal{D}_n$ is given by $L_n(\hat{g}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{g}(x_i))^2$. It is also known as the Mean Squared Error (MSE) which measures the average loss on the sample $\mathcal{D}_n$. There are other evaluation metrics such as the Symmetric Mean Absolute Percentage Error (SMAPE) based on relative errors and the adjusted R-squared (R2). They are defined by

$$\text{SMAPE} := \frac{100\%}{n}\sum_{i=1}^{n} 2\frac{|y_i - \hat{g}(x_i)|}{|y_i| + |\hat{g}(x_i)|}.$$

$$\text{R2} := 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{g}(x_i))^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2},$$

where $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ is the sample mean of the observations $\{y_1, \ldots, y_n\}$.

The following section presents some classical machine learning models to estimate the regression function.

3.3.2 Regression models

In the literature, several families of regression models are distinguished, such as linear and non-linear models. The simplest linear model is the multiple linear regression model which assumes a linear relationship between the explanatory input variables and the output variable. However, in high dimension or in the presence of multicollinearity between the input variables such as engine data, the estimation of the parameters of the model can become critical: larger variances of the parameter estimates and predictions. There are therefore some methods to circumvent this problem. For example, the penalized model approach incorporates an additional assumption on the model parameters to remove unnecessary redundant variables. The idea is to force a certain number of coefficients to be zero or close to zero in the model. The Lasso (Least absolute shrinkage and selection operator) regression uses for example a penalty in $\mathbf{L}^1$ norm and allows to eliminate some variables by setting to zero the small coefficients [96]. This penalty is often used for variable selection. However, in the presence of a group of highly correlated variables, the Lasso usually selects only one variable in the group and ignores the others even if some of them are important for the prediction. The Rigde regression uses a $\mathbf{L}^2$ norm penalty, the values of the coefficients decrease towards 0 but are not totally zero [37]. Ridge is useful to do group selection of redundant correlated variables. However, since highly correlated variables will have the same importance, the Ridge regression does not allow selecting from a group of redundant variables. The ElasticNet penalty combines the penalties of Ridge and Lasso to solve the problems of Lasso [113].

There are other linear models such as Partial Least Squares (PLS) regression which is an extension to multiple linear regression that aims to construct a small number of orthogonal components in such a way that they best explain the target variable [107]. This approach is used as an alternative to principal component

regression (PCR) where the construction of the principal components is done independently of the variable to be predicted, which can lead to worse predictions. The first principal components are not necessarily those that best explain the target variable. Thus, the PLS method often requires fewer components than PCR. The PLS regression is useful when the explanatory variables are numerous and very strongly correlated. However, this method which has a predictive purpose is not suitable for variable selection.

These linear models have the ability to be easily explained and interpreted. However, they are by definition not suitable for modeling nonlinear behavior in complex data such as aircraft engine data. Support vector regression (SVR) [102] is a supervised learning algorithm that is used to predict continuous values. The idea of SVR is to find the best fitting line: the hyperplane that has the maximum number of points while considering a margin. The algorithm chooses an error tolerance from a satisfactory error margin and an acceptance fit that exceeds the acceptable error rate. To solve non-linear problems, SVR usually uses kernel functions in order to project the data into a higher dimensional space in which the problem becomes linear. SVR is robust to outliers, as the algorithm can allow some errors (soft margin concept). Moreover, it works quite well on small datasets and generally has a good prediction accuracy. This is why this model is often used in many applications in the aeronautical industry [111, 40, 106], but the training time can be long on quite large datasets. SVR can be less effective in the presence of significant noise and outliers.

Another regression method are random forests [11] which combine a large number of independently trained decision trees. Random trees generally reduce the variance of predictions and the risk of overfitting compared to a single tree. Trees are learned on a different sets of variables, with each node split into a subset of explanatory variables that are randomly drawn. Random forests are known to perform well in prediction but can be quite slow in training. Several examples of applications in the aeronautical industry illustrate the practical interest of random forests [63, 62]. Random forests are so-called black-box models because of the large number of operations required to obtain a prediction. They are therefore difficult to interpret. Finally, XGBoost (eXtreme Gradient Boosting) [16] is a supervised learning model based on sequential ensemble learning and decision trees. A serie of decision trees is trained. At each iteration, each decision tree in the series is trained to correct the errors of the previous tree. The predictions of all decision trees in the series are combined to give a final prediction. This algorithm, which is mainly used to improve weak models, is fast, accurate and efficient. It works well on small data and on complicated data. Therefore, this model is widely used in the aeronautical industry [59]. The biggest limitation of this algorithm is the black box nature which makes it difficult to interpret. Recent approaches have been proposed to solve this problem. For example, Bénard et al. [4] propose a method based on random forests that extract a small set of relevant and simple rules.

Classical machine learning models suppose independance of the observations and thus fail to model dependencies and temporal drifts. They are not suited and appropriate for time series data. The total engine airflow can also be considered as a time series, since the engine performance evolves over time. Here, we propose to explore the fitting of two time series models to the data, namely ARIMA [9] and LSTM [36] models which are used in many fields of application, especially in aeronautical industry [35, 109, 55, 99, 110, 65].

3.3.3 Time series models

The aim of time series is to forecast the future on the basis of past values. Time series models can be used to analyze temporal data in order to understand past trends and predict future ones.

The basic idea of the ARIMA model is to decompose the observations in obvious trends of the series (such as growth and decrease phenomena) and the residuals which are modelled as random fluctuations. The residuals are ideally modelled by white noise. Let $p, d, q \in \mathbb{N}$. The model ARIMA(p, d, q) can be written as

$$\nabla^d Y_t = (\mathbb{I} - B)^d Y_t = \alpha + \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{k=1}^{q} \theta_k \epsilon_{t-k} + \epsilon_t,$$

where B is the backward shift operator such that $BY_t = Y_{t-1}$, $\alpha, \phi_i, \theta_k \in \mathbb{R}$ and ϵ_t is the Gaussian white noise of standard deviation $\sigma > 0$. Exogenous variables $\mathbf{X}_t$ can be added and are assumed to be stationary time series. Such linear regression models work best when the predictors are poorly correlated and independent of each other.

The LSTM model is a recurrent neural network mainly used to forecast time series far enough into the future to detect non linear patterns in the data. An LSTM network cell is composed of an input gate, an output gate and a forget gate. It also has two states: the cell state c_t and the hidden state h_t.
The mechanism of these 3 gates is modeled by a sigmoid function σ as follows

$$F_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \in [0,1]^h \text{ for forget gate,}$$

$$I_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \in [0,1]^h \text{ for input gate,}$$

$$O_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \in [0,1]^h \text{ for output gate,}$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c),$$

where $x_t \in \mathbb{R}^d$ is the input vector for the LSTM unit, h is the number of the current hidden units in the model, $c_t \in \mathbb{R}^h$ is the cell state vector and $h_t \in [-1,1]^h$ is the hidden state vector. Moreover, $W_f, W_o, W_i \in \mathbb{R}^{h \times d}, U_f, U_o, U_i \in \mathbb{R}^{h \times h}$ and $b_f, b_o, b_i \in \mathbb{R}^h$ are weight matrices and bias parameters to be learned from the data.

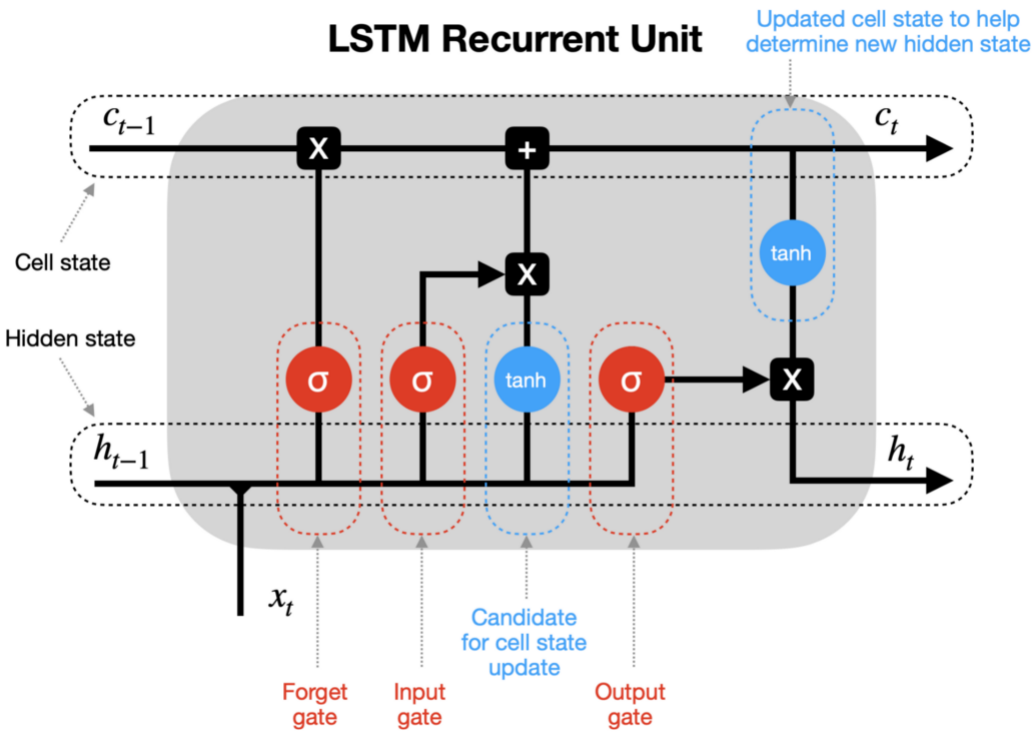


Figure 3.2: Neural network LSTM

The forget gates performs a filter of the information contained in the memory cell at previous instants $t-1$. They determine which irrelevant information should be discarded from a previous state by assigning it a value between 0 and 1. A value close to 1 means that the information should be retained and a value close to 0 means that it should be discarded. Input gates decide which relevant new information should be stored in the current state, using the same mechanism as the forget gates. Output gates control the information to be returned by the current cell by assigning a value from 0 to 1 to the information, taking into account the previous state and the current state. The selective output of relevant current-state information by the network enables useful dependencies to be maintained over a long period of time for predictions far into the future.
Finally, the cell state and the hidden state are calculated as follows

$$c_t = F_t \odot c_{t-1} + I_t \odot \tilde{c}_t,$$

$$h_t = O_t \odot \tanh(c_t),$$

where $\odot$ is the Hadamard product (element-wise product). Figure 3.2 summarizes the main stages of the LSTM model.

3.4 Impact of geometric characteristics of fan blades on performance in the i.i.d. case

A good knowledge and understanding of relationships between performance and production parts allow to better understand and improve the production of aircraft engines.

Therefore, this part consists in determining and estimating the effective impact of the geometry of fan blades on engine's total airflow in the independent and identically distributed (i.i.d.) case. We propose to develop a regression model that links the performance and the geometrical parameters. We will adapt classical statistical learning methods to our model to adjust the parameters to the data and to select the most relevant variables for the prediction. Indeed, as the dimension of data is large, it is important to perform variable selection to have an interpretable model and to decrease the cost of data acquisition.

In real applications, it is difficult to verify the assumption of i.i.d. observations. A short time interval is considered here, so that temporal trends are negligible and the i.i.d. hypothesis is justified and acceptable. We take a time interval with around $\tilde{n} = 700$ engines where the production trend seems to be stable. Here, all the variables are stationary. We assume therefore that the data are identically distributed. Moreover, the business experts assume that the engines are produced independently, which suggests independent data.

First, we start by using classical machine learning models. The usual machine learning models, which have become very popular and efficient to answer industrial problems, are able to handle the case of high dimensional data and to choose relevant variables. In addition, they are often able to build models that generalize well on unseen data. The results obtained will allow us at the end to verify the previously mentioned hypotheses. We divide randomly the sub-sample $D_{\tilde{n}}$ in two samples $D^1_{\tilde{n}}$ and $D^2_{\tilde{n}}$, the first one is made of 70% of the learning data on which the model is learned and the rest is the validation sample on which the model will be validated.

We summarize the results of the performance comparisons of the regression models cited. The idea is to identify the model that best fits the data and gives better results in terms of prediction. The models will be evaluated using the MSE, SMAPE and R2 scores. The results obtained on the training and validation data are respectively presented in the Tables 3.1 and 3.2.

We observe that the best model is obtained with a SVR model but the performance remains quite low in terms of prediction. However, it does not overfit the data, unlike the two other models, the XGBoost and the random forest, which predict very well only on the training data. However, the SVR model with a gaussian kernel is quite difficult to interpret.

The performance of the linear models Lasso, Ridge, ElasticNet and PLS are quite close but are quite poor in terms of prediction. We decide not to go further in

Table 3.1: Scores of the different models on the training data

	MSE	SMAPE	R2
Lasso	2.697	0.105	0.443
Ridge	2.641	0.104	0.455
ElasticNet	2.748	0.106	0.433
PLS	2.741	0.106	0.434
SVR	2.325	0.097	0.520
XGBoost	2.256	**0.073**	**0.741**
Random Forest	**1.433**	0.078	0.704

Table 3.2: Scores of the different models on the validation data

	MSE	SMAPE	R2
Lasso	3.162	0.114	0.383
Ridge	3.121	0.114	0.391
ElasticNet	3.148	0.115	0.385
PLS	3.132	0.115	0.389
SVR	**3.002**	**0.114**	**0.414**
XGBoost	3.257	0.118	0.364
Random Forest	3.238	0.117	0.368

the selection of the most relevant variables because of the low scores which remain insufficient to give conclusions.

Classical machine learning models based on supposedly independent distributions do not model dependencies, their approach is unsuitable and invalid for time series data where the time information is important.

The independence assumption is violated. The measurements taken by the sensors depend on the state of the measuring machine, which in turn depends on the operations carried out by this machine, which can be known through the previous measurements.

3.5 Forecasting the turbofan's flow

Here, we assume that the $\mathcal{D}_n$ is sorted by chronological order (according to the date of the engine tests). Figure 3.3 represents the total airflow of the tested engines as a function of the test date in days over a test period of approximately two years. We notice that the time series is not stationary. The trends are probably due to the production of the engine parts and factory effects. Moreover, we notice that the time series is quite noisy.
The objective is to predict the total debiased airflow of tomorrow's engines and thus detect at the earliest possible time the performance drifts at risk.

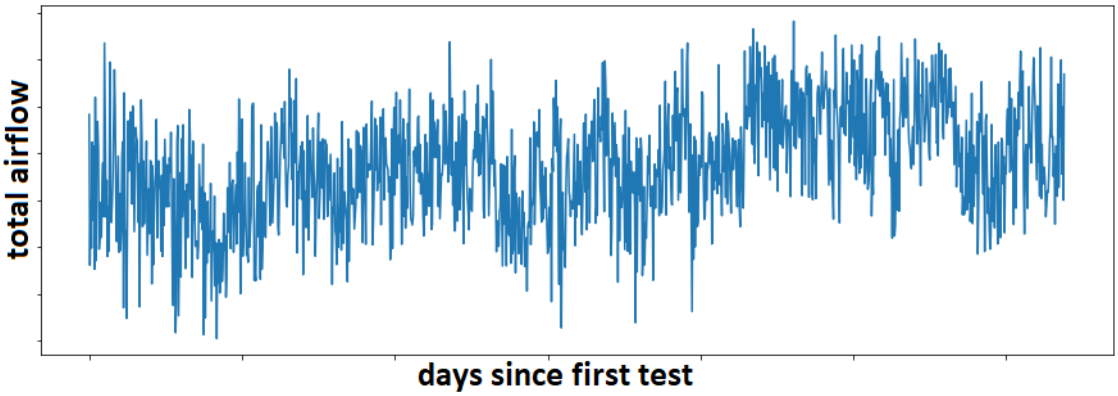


Figure 3.3: Total airflow of the tested engines as a function of the test date in days

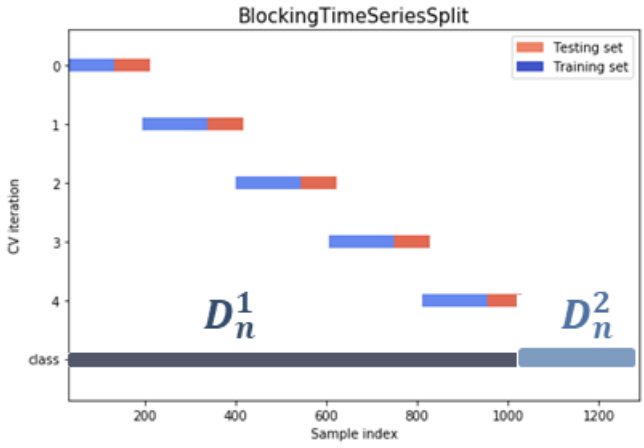


Figure 3.4: 5-fold cross-validation of the training set $\mathcal{D}_n^1$

3.5.1 The offline learning

First, we considered an offline approach based on static Machine Learning models to explain the total engine airflow from the fan blade geometry. A K-fold cross validation is performed on the training data $\mathcal{D}_n^1$ consisting of dividing it K times into several training and test data. In forecasting, the division is not done randomly to keep the time dependency, so that on each fold, the test data are arranged after the training data. On each fold, a predictive model is fitted on the training data to correctly calibrate the model hyper-parameters. The fitted model is then evaluated on the test data. The final model selected is the one that on average has a better performance on the K-fold. This model is then evaluated on the validation data by the MSE and the R2 score. Figure 3.4 shows a 5-fold cross validation split. Other divisions were also tested.

Several classical machine learning models have been evaluated, including those mentioned in the section 3.3.2.

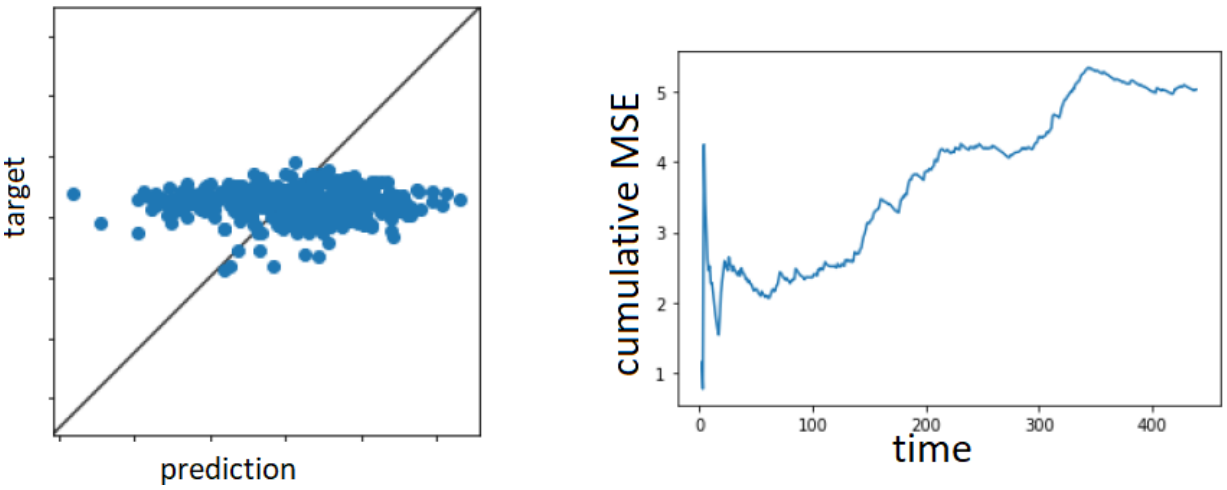


Figure 3.5: On the left, graph representing the total airflow against the predictions obtained by the random forests. On the right, graph representing the cumulative MSE versus time computed on the validation data.

One of the limitations of this approach is the degradation over time of the predictive quality of the models used on the validation data. The models have indeed the difficulty to generalize on new future data. The approach is not adapted to time series that present new trends that could not be learned during the learning phase. Alternatives would be to regularly update a model by assigning weights to the observations (with lower weights to the oldest) and eventually delete the oldest.

The results obtained on the validation data with the random forests are presented in figure 3.5. The R2 score obtained is -0.07 (against 0.77 on the training data). We observe an overfitting of the model which does not generalize on new data which seem very different from the training data. The predictive quality of the model increases over time.

3.5.2 Sliding window learning

The idea of sliding window learning is to regularly re-train a machine learning model to explain the engine airflow from the fan blade geometry. At each iteration, a shift of the training data $\mathcal{D}_n^1$ of fixed size is performed by adding the same number of the most recent engines and removing the same number of the oldest engines at each shift. At each iteration, a shift of the fixed-size validation data $\mathcal{D}_n^2$ is also performed with a storage of the model predictions fitted to the training data. The approach stops until the end of the available dataset is reached. Figure 3.6 shows the shifts in the training and validation data at each iteration. The MSE and R2 scores are computed by considering the totality of the predictions obtained at each shift.

Figure 3.7 represents in orange the predictions obtained with SVR models. At each iteration, an SVM model is fitted to a subset of the data consisting of 100

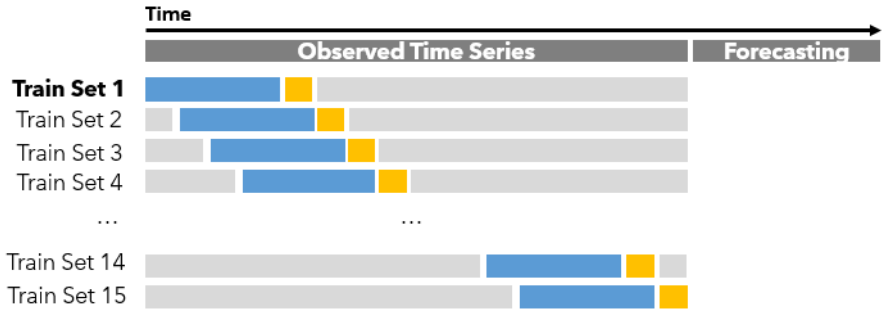


Figure 3.6: Sliding window learning

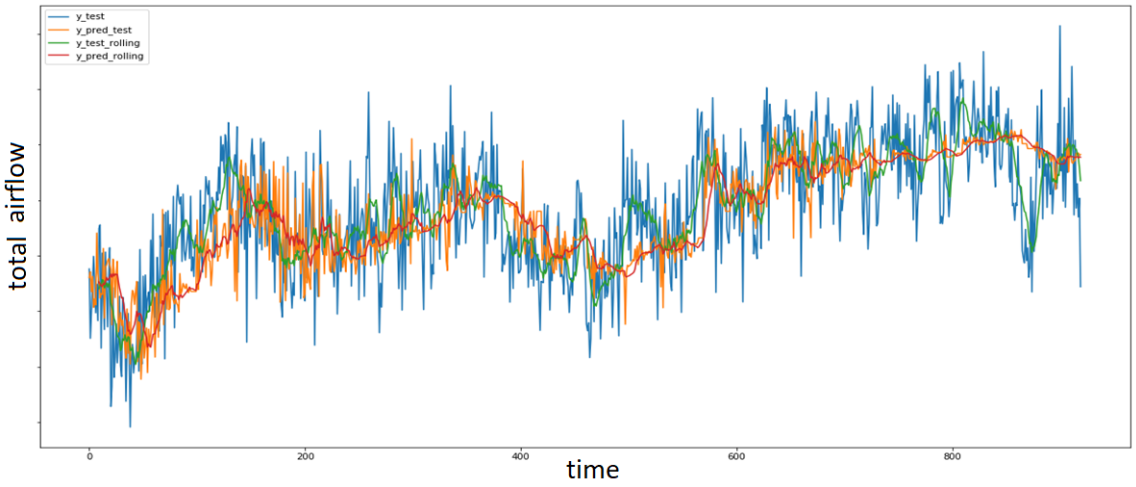


Figure 3.7: Predictions obtained with an SVR model on the validation data set represented in orange color. The total airflow measurements of the engines are shown in blue.

engines and then predicts the total airflow of the next 5 engines. In the next iteration, the first 5 engines are removed from the training data and the last 5 engines predicted in the previous step are added. Figure 3.8 representing the residuals shows that they are not completely de-correlated by the presence of trends. Figure 3.9 represents the histogram of the residuals as well as a QQ-Plot and confirms in particular that the residues are not completely decorrelated.

One of the limitations of this approach is the rather heavy computation time due to the optimization of the hyper-parameters of the models at each shift. Moreover, this approach is not able to obtain forecasts on a too long time horizon (approximately until the 10th next engine, beyond that the forecasts are bad).

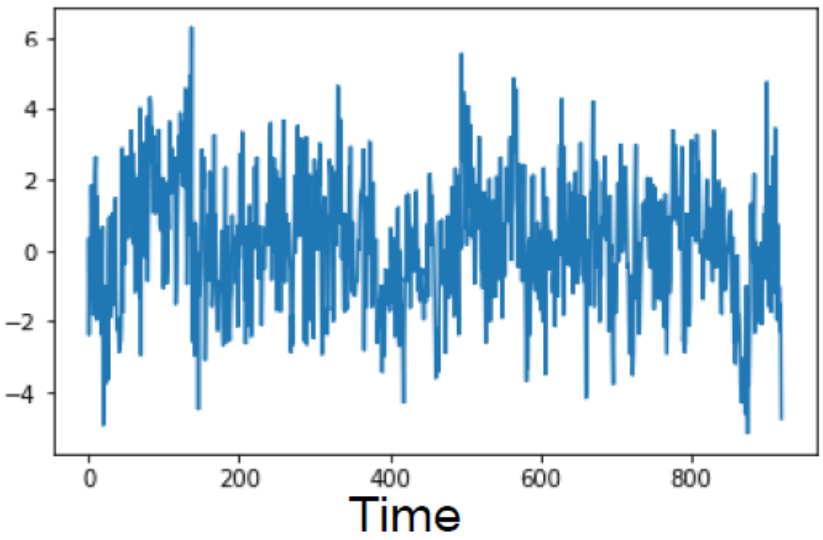


Figure 3.8: Residuals of predictions obtained by SVR models as a function of time.

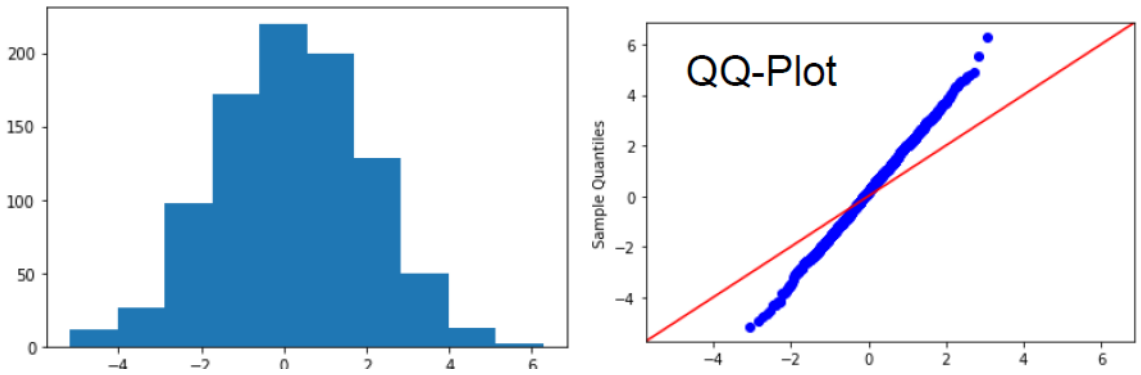


Figure 3.9: On the left, histogram of residuals and on the left QQ-Plot of residuals with respect to a standard normal distribution.

3.5.3 Time series models

We divide $\mathcal{D}_n$ in two samples $\mathcal{D}_n^1$ and $\mathcal{D}_n^2$, the first one is made of 70% of the learning data on which the model is learned and the rest is the validation sample on which the model will be validated. We now fix these two samples.

ARIMA model

Let's start with the simplest linear model ARIMA.

Autocorrelation and partial autocorrelation

The figure 3.10 shows the empirical autocorrelation and partial autocorrelation of the series. We observe that the speed of decay towards 0 of this function is rather slow. This confirms in particular that we can suspect a non-stationarity. Moreover, we can deduce some parameters of the model. We also notice that the series does not contain any seasonality, which remains in agreement with the engine tests.

Results

The parameters of the model are optimized on the training set D_n^1. The better model obtained is ARIMA(1, 1, 2). The different scores obtained by this model on training and validation data are presented in table 3.3.

Table 3.3: Scores of ARIMA(1, 1, 2) model on the training and validation data

	MSE	SMAPE	R2
Training	2.321	0.114	0.437
Validation	2.219	0.098	0.421

Figures 3.11 and 3.12 show respectively the forecasts obtained by the model on the training and validation data which are represented by the orange curve. We can see that the model has captured the global trend of the data.

We perform a residual analysis to verify and evaluate this model. It is necessary to verify that the chosen model is suitable by testing that the estimated noise is white. The normality of the residuals means a relatively homogeneous pattern over time without any trend. When the residuals of the model are white noise, it means that all the information in the time series has been used by the model to make predictions and that it is no longer possible to improve the forecasting model. All that remains are the random fluctuations that cannot be modeled. The results on the training data D_n^1 are shown in the figure 3.13. The model errors have been represented in the form of a QQ-Plot, an auto-correlation and partial auto-correlation. From these results, we observed that the error distribution is gaussian and similar to white noise. Moreover, the results obtained on the validation data are the

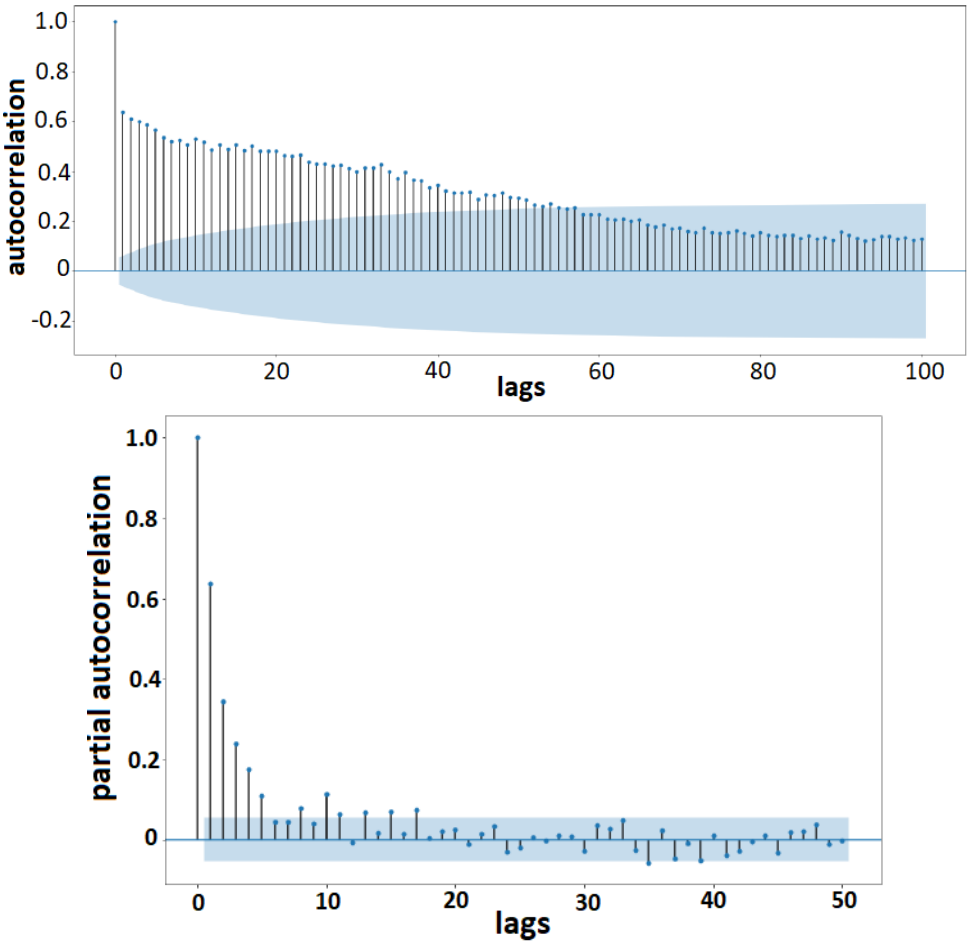


Figure 3.10: Autocorrelation and partial autocorrelation plots

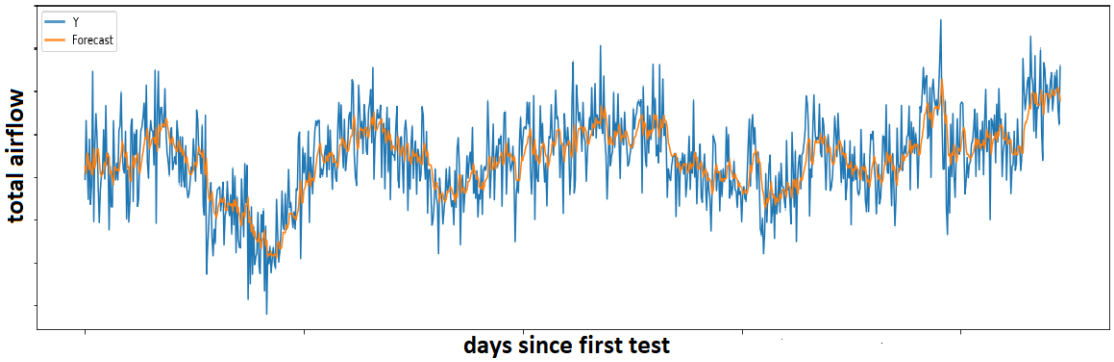


Figure 3.11: Forecasting results obtained by the ARIMA(1, 1, 2) model on the training data

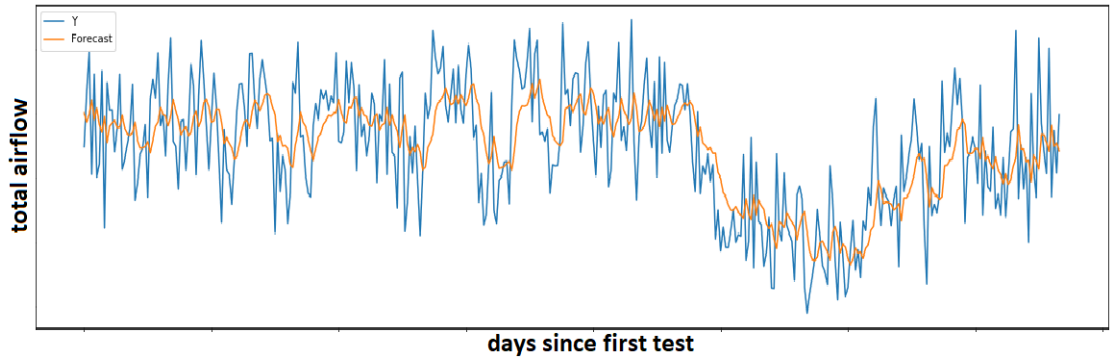


Figure 3.12: Forecasting results obtained by the ARIMA(1, 1, 2) model on the validation data

same. This implies that it is no longer possible to model anything. Indeed, if a time series is white noise, then by definition it is random. It is no longer possible to model it and make predictions. In particular the geometrical parameters of the fan blades will not provide any additional information. It was found that there were no statistically significant relationships. The ARIMA(1, 1, 2) model means that the prediction of total airflow in the future depends on measurements and errors from two previous days of engine's total airflow. This highlights that the dependencies related to test conditions (noise related to test bench equipments, environmental conditions, machines and sensors, ...) and engine part production are strong. On the other hand, this model is not able to predict far enough into the future. This is one of the main limitations of the ARIMA model. But our need is to anticipate engine performance drifts much earlier, several days before they leave the factory. Predicting the airflow of engines two days before the acceptance test is not enough.

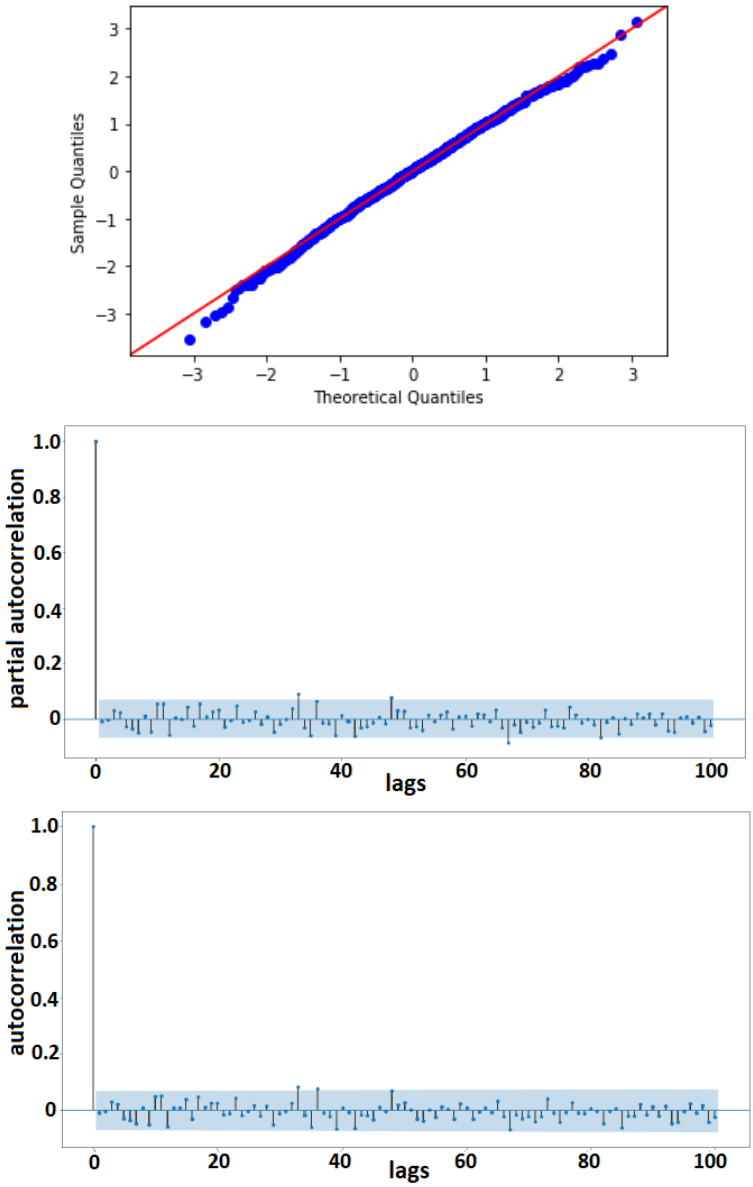


Figure 3.13: Residual analysis of ARIMA(1, 1, 2) model on the training set

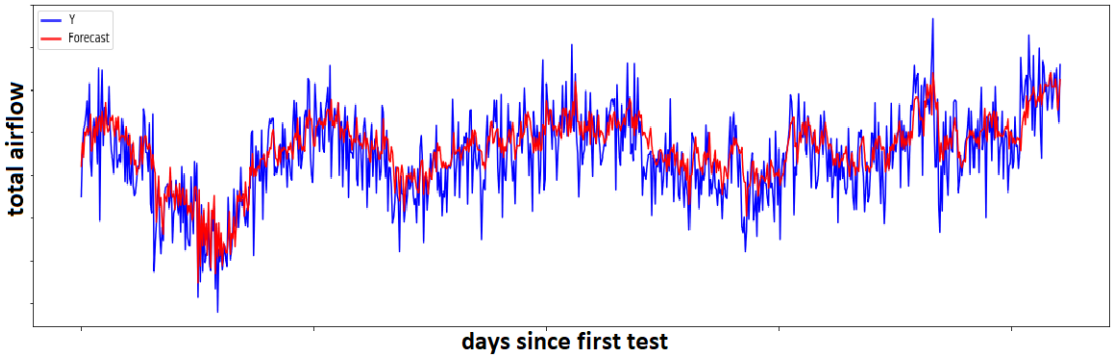


Figure 3.14: Forecasting results obtained by the LSTM model on the training data

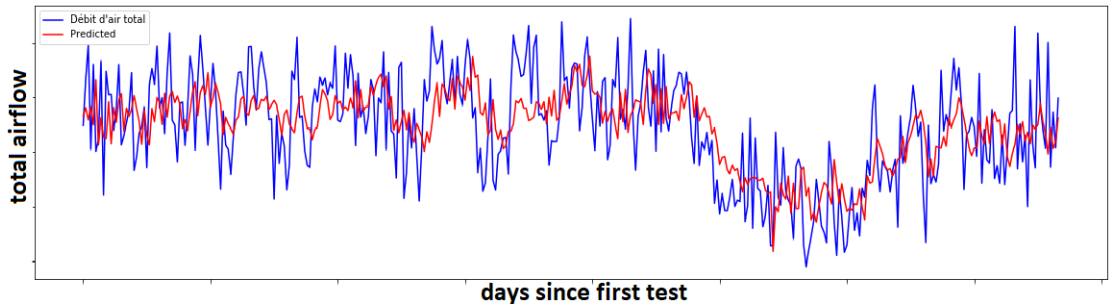


Figure 3.15: Forecasting results obtained by the LSTM model on the validation data

LSTM model

The LSTM model is a recurrent neural network mainly used to forecast time series far enough into the future able to detect non linear patterns in the data. However, as for most neural networks, the number of training data must be sufficient to avoid overlearning. Here, this is the main difficulty because we do not have enough data in D_n^1. Moreover, the number of exogenous variables being high, it is essential to reduce the dimension of the problem beforehand in order to avoid overfitting. In view of the results obtained previously with the ARIMA model, we first present the results without the exogenous variables. This will allow us to determine if the conclusions will be identical to those with the ARIMA model, i.e. if it is still possible to model the residuals of the LSTM model. For the optimization of the model hyperparameters (in particular, the lag size, the number of hidden layers, the dropout rate, the number of neurons in each layer, the optimization algorithm, the learning rate, the epoch size, and the batch size), we applied the random hyperparameter search methodology.

Figures 3.14 and 3.15 show respectively the forecasts obtained by the model on the training and validation data which are represented by the red curve. Since the LSTM model is a fairly complex neural network, we observe that it suffers from overfitting. Indeed, the figure 3.14 shows that the model fits well on the training

data but does not fit correctly on the validation data as we can see on the figure 3.15. The model nevertheless captures the global trend of the validation data.

Moreover, the table 3.4, which presents the different scores obtained by the LSTM model on the training and validation data, also shows this overfitting effect. One reason is that the number of observations of training data is insufficient.

Table 3.4: Scores of LSTM model on the training and validation data

	MSE	SMAPE	R2
Training	1.936	0.087	0.537
Validation	2.441	0.102	0.363

As before, we perform a residual analysis to evaluate this model. The results on the training data D_n^1 are shown in the figure 3.16. The model errors have been represented in the form of a QQ-Plot, an auto-correlation and partial auto-correlation. From these results, we observed that the error distribution is gaussian and similar to white noise. Moreover, the results obtained on the validation data are the same. The conclusions are the same than those obtained with ARIMA model: it is no longer possible to model anything. It is no longer necessary to integrate geometrical parameters of the fan blades in the modeling of the time series because it will not provide any additional information.

To conclude, it is better to choose an ARIMA model rather than an LSTM model because it performs better than a LSTM model on the validation data and it does not suffer from overfitting. Indeed, on the validation data, the forecast quality of the ARIMA model is better. Thus, ARIMA generalizes better on new data than LSTM. On the other hand, in the long term, with more data available, the results and conclusions can be different. In particular, the risk of overfitting may decrease and the quality of the LSTM model may improve. An ARIMA model seems to be more adapted to our time series which does not have enough observations. Moreover, one of the advantages of an ARIMA model is its simple interpretation. However, this model does not allow us to completely answer our objective which consists in predicting as soon as possible the performance drifts before the passage in acceptance tests and to determine the influence of the geometrical parameters of the fan blades.

3.6 Conclusion

The objective of this study is to predict engines's total airflow using fan blade production through machine learning methods in order to anticipate as soon as possible the production drifts that are risky for performance.

First, we presented and analyzed the engines data and identified some of their characteristics. We detected and reduced the biases induced by the different equipments used during the tests.

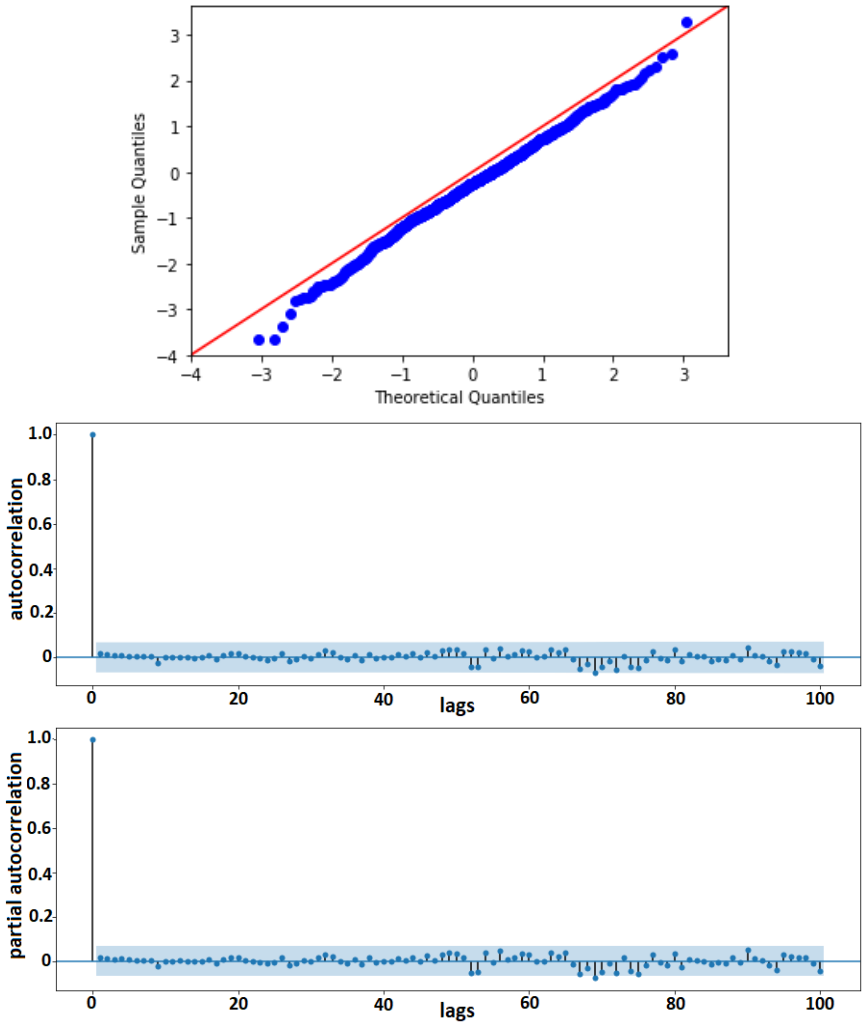


Figure 3.16: Residual analysis of LSTM model on the training set

In a second step, we were interested in modeling the total airflow by applying first classical machine learning models and then time series models. We explored different approaches in order to identify the one that would allow us to pursue our research towards a scientific contribution. The general trend of the total airflow measurements is fairly well represented and described by the models. In contrast, the fan blade measurements alone do not allow us to further model the performance and study their influence. The data dependency related to the test conditions and the production of aircraft engine parts turns out to be important and takes over the geometrical characteristics of the fan blades.

We cannot capture the influence of secondary effects with the data we have. The geometric measurements of fan blades do not predict the airflow measurements of future aircraft engines. We deduce the following result: the engine is robust to the variations of production that we could have in the past. This study has allowed us to raise several issues. The first major issue concerns the temporal uncertainties related to the test benches, the environmental conditions (temperature, pressure, etc.) and the cumulative effects of all the elements disturbing the flow downstream of the fan. The second question is rather about the consideration of some important elements such as the interactions between the geometrical parameters, the effect of heterogeneity and the arrangement of the fan blades within the engine. Finally, a third question about the need to review the method of debiasing total airflow.

These results do not allow us to pursue our research further. We therefore decided to work on an influence analysis tool for incomplete mixed data.

Chapter 4

Influence analysis and missing data

Contents

4.1 Introduction

The production of aircraft engines is subject to numerous requirements, particularly in terms of quality. Deviations or anomalies can occur during the manufacturing

process, such as a defective part. The production process is typically monitored via some performance measure representing the quality of the production. In addition, in general a set of operational measures of the manufacturing process is available. When the performance meausure drops down, it is important to rapidly identify the cause of degradation, that is, to identify the responsible operational variables. This step is essential for the production quality because it allows us to find solutions to the problem and also to avoid recurrent disfunctions.

A natural way to adress this problem by a statistical approach consists in explaining the performance measure using the operational measurements by supervised learning models. Then finding the causes of a degradation of the performance corresponds to evaluating the importance of the variables.

4.1.1 Variable importance measures

In the literature, there are different ways to determine influential variables in a supervised learning framework. The idea of variable importance is to quantify the extent to which variables impact the output in a supervised learning model. There are several measures of variable importance defined from machine learning black box algorithms that are very effective in terms of prediction in many applications, but are difficult to interpret.

A global sensitivity analysis can be used to determine how the output of the model reacts to variations of its inputs. Its objective is to explain the model in its entirety (its trends), i.e. to determine which variables have the greatest impact on average in the model. Introduced by Sobol [90], the Sobol indices study the importance of the input variable on the variance of the target variable. They measure the reduction of the variance of the output of a model when one or more variables are fixed. The more important a variable is, the greater the reduction will be. However, these indices are well defined only in the case where the input variables are independent. Otherwise, when inputs are correlated, the indices become insignificant. The Shapley indices, well known in game theory [87], allow to solve this problem by taking into account the correlations and interactions between variables [75]. Futhermore, these indices can also be adapted to locally measure the importance of variables. The SHAP (SHapley Additive exPlanations) values allow to explain each individual prediction by determining the contribution of each variable on the prediction of a given individual [64]. The prediction of an observation is written in a simple form of a sum of the contributions of each variable as in the case of a linear regression. This allows us to know whether the feature has a positive or negative impact on the predictions.

Another approach consists in perturbing data and measuring the decrease of prediction accuracy in the model. For example, Lei et al. [58] propose a variable importance by Leave-one-covariate-out (LOCO) which consists in evaluating the decrease in accuracy by removing a variable from the model. However, this simple

approach can be expensive in terms of computing time, in particular when the number of variables is large. Moreover, this approach is not suitable when input variables are highly correlated. Indeed, dropping one of the variables may not affect the result, because the estimator still has access to the same information from the other input variables. However, groups of two or more correlated variables may all be important in explaining the output, their combination may be the cause behind a degradation in manufacturing quality. We can also remove a whole group of correlated variables to evaluate their impact.

Some measures are specific to supervised learning algorithms. Decision trees recursively partition the training data into several nodes. In a decision tree, prediction consists of a majority vote in a node in classification and an average in regression. When constructing the decision tree, for each node, one selects a variable which is used to divide the training data in the node in two. This is done such that a measure of impurity is optimized, which is the entropy or the Gini index in classification and the variance in regression. For decision trees, an influence measure for a given variable is the Mean Decrease Impurity (MDI) defined as the total decrease in loss or impurity contributed by all splits involving the given variable [10]. However, this variable importance measure is known to be biased when the inputs are dependent [71, 85]. In a decision tree, a selection bias exists especially when the explanatory variables are highly correlated. Indeed, to divide a node, one has to choose a single variable and this reduces the importance of the other influential correlated variables, which are not selected. Futhermore, the MDI can have a very high bias by overestimating the importance of some non-discriminating variables, especially when the noise has a larger variance than the regression function. Moreover, the MDI tends to select continuous variable with large variance and categorical input variables with a large number of modalities [92].

The model for which variable importance has been studied the most are random forests. Random forests [11] combine a large number of independent decision trees to reduce the variance of predictions and the risk of overfitting that could be obtained with a single tree. The trees are learned on different sets of observations and variables, with each node split into a subset of explanatory variables that are randomly drawn. In random forests, prediction is obtained simply by aggregating the predictions of the trees. This involves averaging the predictions of all the trees in the forest in regression, and retaining the most frequent prediction in classification. Random forests are known to perform well in prediction but they are so-called black-box models because of the large number of operations required to obtain a prediction. They are therefore difficult to interpret. The most popular importance measures specific to random forests are the permutation importance measure and the MDI averaged over all trees in the forest. The permutation importance measure, which is called the Mean Decrease Accuracy (MDA) by Breiman [11], evaluates the increase in the prediction error of the model after randomly permuting feature values of a given variable. The greater the loss, the more influential the variable. The

approach is based on the idea that permutation breaks the relationship between the feature and the target variable. The importance measure of variables allows to make prediction results a little more interpretable. However, these variable importance measures are known to be biased when the inputs are dependent [92, 2, 72, 32, 39]. The MDA overestimates the importance of variables that are correlated with influential variables. Furthermore, when the degree of correlation and the number of correlated variables increase, the importance of correlated influential variables decreases. Thus, independent and less informative variables can be estimated as more influential than correlated and more relevant variables [97, 32].

A strong dependency that exists between two variables is destroyed after permutation, which often forces the model to extrapolate into unseen regions of very low density where predictions are poor. The importance of correlated variables is therefore biased.

Several measure alternatives have been proposed in order to reduce and remove these different biases. Strobl et al. [93] propose a conditional permutation importance for random forests, the permutation is made conditionally to other correlated variables. Gregorutti et al. [32] propose an approach that recursively eliminates variables using the MDA as a ranking criterion. Bénard et al. [12] estimate the total Sobol index for random forest models based on the MDA.

However, the interpretability of the results in such an industrial context is essential. Indeed, it is useful to provide business experts with easily interpretable results that allow them to better understand and determine the root causes of a non-conformity or a failure identified on parts. This allows them to validate and make certain modifications or adjustments to improve the manufacturing process of a product. For this reason simple decision trees are preferable to random forests in our context and a popular tool, based on decision trees, is presented in the next section.

4.1.2 Influence analysis tool at Safran

Influence analysis is an exploratory data analysis tool developed by Lacaille [53] for an industrial context. It aims at explaining and determining in a simple and interpretable way the causes of variations of a performance measure of an industrial process. The influence analysis tool is based on a decision tree model for the sake of explicability and interpretability. Indeed, influence analysis tries to explain what caused the deviations in a performance measure in a simple and interpretable way to help business experts in their decision making. A measure of influence based on entropy and mutual information is used to evaluate the relevance of a group of potential causes. It measures the amount of information provided by a group of factors on the performance measure. The tool allows to classify the causes in order of influence and thus identify for instance the variable which is the main cause. A visualization tool also allows to easily summarize the results of the influence

analysis through interactive figures. Today, this tool is widely used within the Safran group because of its practicality, interpretability and convenience. Lacaille [53] gives several examples of applications of the influence analysis in the field of the aeronautical and automobile industry. One of the applications illustrates the tool's methodology for investigating the set of parameters that can explain the increase in ignition time of aircraft engines. The analysis identifies atmospheric conditions and fuel density as the main causes of ignition delay variations. It can also be applied to other fields of application such as finance, medecine, etc.

To manage the strong dependencies between variables, the tool has recently been extended at Safran by an additional data compression step. The idea, which had already been proposed by Chavent et al. [15] in the case of random forests, consists in creating groups of correlated variables and summarizing the information of each group by a synthetic variable. Then the classical influence analysis is performed on these synthetic variables.

4.1.3 Missing data in decision trees

Influence analysis is only developed for complete data. However, data may be incomplete for a large variety of reasons. In industrial applications, measuring instruments may have malfunctions or detection limits yielding erroneous and missing entries. For a long time, in statistics, missing data have been treated in a very simple and inappropriate way, either by deletion of incomplete observations or by basic data completion by mean or median values. The currently implemented tool deletes any incomplete data. The impact of missing data on statistical results can be serious, leading to biased estimates and loss of information. In fact, the influence measures of incomplete variables can be biased, as they are lower than those of fully observed variables. An incomplete variable whose missing values have been imputed by a constant value such as the mean, contains little information and variability, especially if the number of missing data is high.

In the literature on missing data, several authors have been interested in this problem in supervised learning, in particular with decision trees. In supervised learning, the objective is not estimation but prediction where the goal is to minimize the prediction risk by estimating a regression function. Decision trees can easily deal with missing data intrinsically without imputation. In Josse et al. [44], it is shown that the risk of tree-based models can be naturally optimized for inputs in semi-discrete spaces.

In the construction of the decision tree, a first problem arises when the node separation is performed on an incomplete variable because then it is unclear how to evaluate the separation criterion with missing data. The most natural way to deal with this problem is to first separate the node by considering only the complete observations with no missing entries on the split variable, and then place the incomplete observations in the child nodes. We could, for example send all incomplete

observations to the child node that contains the most observations. The default method in C4.5 [78] uses another more judicious startegy based on the probabilistic splits where each incomplete observation is sent to a child node with a weight proportional to the number of instances already present in the child node.

Moreover, the algorithm CART (Classification And Regression Trees) handles missing data in decision trees by using surrogate splits and variables to place incomplete observations [10, 95]. A surrogate variable is an alternative variable that best agrees with the original splitting variable and tries to predict the current split. Surrogate variables are ranked according to the misclassification risk for predicting the child node of all complete observations. The best surrogate splits the data in exactly the same way as the primary split. Any missing observation in the splitting variable is then classified using the first surrogate variable, or if it is missing, the second is used, and so on. This approach is particularly effective when the variables are correlated but cannot be used if all variables are incomplete.

Alternatively, Twala et al. [98] propose a simple and efficient method for handling missing data in decision trees for classification. This approach, called 'Missingness Incorporated in Attributes' (MIA), consists in sending all the missing values to the left node or all to the right node at each split. The split criterion uses also the missing data and the best choice is the one with the lowest error. It is very much linked to the approach that treats missing values as a category. This approach is easy to implement and allows a minimization of the empirical risk which takes into account the missing data. Moreover, Josse et al. [44] recommend using this approach, as it shows good performance in terms of prediction even in the case of non-random missing data.

Futhermore, imputation methods can also be used to handle missing data in supervised learning, in particular with decision tree models. The literature distinguishes two types of imputation: single imputation and multiple imputation. The first one fills in each missing value by a single value to create a complete dataset; the simplest practice is to impute by the mean of the observed values. For estimation, mean imputation is known to be poor because it distorts the relationships between variables and induces bias in estimators when the mechanism is not random. However, in supervised learning, the objective is not estimation but prediction where the goal is to minimize the prediction risk by estimating a regression function. Josse et al. [44] show that asymptotically, good imputation is not needed for good prediction. Average imputation can be quite appropriate and leads to a consistent estimation of the prediction function, provided that one has a lot of data and a super powerful learner. This asymptotic result is extremely useful in practice. More sophisticated single imputations are based on non-parametric models such as random-forest models built on the observed values [91]. Other popular imputation methods include the iterative Principal Component Analysis (PCA) which uses correlations between variables and similarities between individuals to iteratively impute missing data by a PCA-type model [42]. The second approach imputes more than

one value for each missing entry based on an imputation model, such that several completed datasets are created and that can be analyzed separetely by a statistical analysis before combining the results [81, 82]. In prediction, the results obtained in each imputed dataset are combined by using the mean of predictions in regression and the majority vote in classification. Contrary to single imputation, MI allows to take into account the uncertainty due to missing data in the statistical analysis. As an example, the *mice* package (Multiple Imputation by Chained Equations) implements the popular MICE algorithm [100]. The method is based on Fully Conditional Specification, where each incomplete variable is imputed iteratively by a specific model using the complete variables in the data. The MICE algorithm can be used for mixed-type data but it works with the assumption that the data are Missing At Random (MAR). The imputation method should be in accordance with the analysis model (the learning algorithm) to enssure the concept of congeniality introduced by Meng [68].

The proposed methods have been studied and analyzed only in terms of prediction. Hapfelmeier et al. [33] propose a new variable importance measure for random forest models when data are missing. For random forests that use the surrogate split approach, interpretation of the MDA becomes difficult for incomplete variables, especially if they contain many missing values. To measure the importance of an incomplete variable, they propose another strategy instead: for each tree and each node involving the incomplete variable, missing observations are randomly sent either to the left or right child nodes according to a probability proportional to the number of observations already in the node. We can evaluate the decrease in accuracy of the model after modifying the forest. Numerical simulations show that this new measure preserves variable ranking rather well only when incomplete variables contain little missing data. Moreover, multiple imputation performs well even in the MNAR case, the results are close to those obtained in the original full data.

In this chapter, we propose to adapt the influence analysis tool of Lacaille [53] to incomplete measurements. The objective of this work is to test and compare different methods of handling missing data in the context of influence analysis. The methods will be evaluated in terms of influence of variables.

4.2 Classical influence analysis

The influence analysis tool provides interpretable results that easily explain and represent the relationships between the features (operational measures, phenomena) and the target variables (a performance measure of an industrial process). This tool may also help in building an explanatory model. The tool can be used as a first step in data analysis to check the statistical content of a dataset.

4.2.1 Mutual information

First, we introduce some notations and recall the definitions of the entropy and the mutual information. Let Z be a discrete random variable taking $k \in \mathbb{N}$ different values $\{z_1, \ldots, z_k\}$ with probability $p_l^Z = \mathbb{P}(Z = z_l)$ for $l = 1, \ldots, k$. The Shannon entropy of Z [86] measures the average amount of information provided by the outcomes of an experiment modeled by the variable Z and defined as

$$H(Z) := -\sum_{l=1}^{k} p_l^Z \log(p_l^Z).$$

The entropy is maximal when all outcomes are equiprobable because then the uncertainty is maximal (if Z is equidistributed then $H(Z) = \log(k)$). Moreover, the entropy increases with the number of possible states k. Indeed, the more possible choices there are, the greater the uncertainty. The entropy is null when $k = 1$ since a certain event does not bring any information.

Now, let $Z_1, \ldots, Z_n$ be a sample of n independent copies of Z and denote by $\hat{p}_l^Z$ the estimate of p_l^Z corresponding to the proportion of occurences of label z_l over all observations, that is $\hat{p}_l^Z = \frac{\#\{1 \leq i \leq n : Z_i = z_l\}}{n}$. Then the plug-in estimate of the Shannon entropy $H(Z)$, denoted by $\widehat{H(Z)}$, is given by

$$\widehat{H(Z)} := -\sum_{l=1}^{k} \hat{p}_l^Z \log(\hat{p}_l^Z). \tag{4.1}$$

Moreover, the variance of the entropy estimate $\widehat{H(Z)}$ can be estimated as follows

$$\mathrm{Var}[\widehat{H(Z)}] \approx \frac{1}{n} \sum_{l=1}^{k} \hat{p}_l^Z (1 - \hat{p}_l^Z) \log^2(\hat{p}_l^Z) + \frac{k-1}{2n^2}. \tag{4.2}$$

For more details on this approximation, we refer the reader to Lacaille [53].

The joint entropy of a pair of discrete random variables (Z_1, Z_2) is defined by

$$H(Z_1, Z_2) := -\sum_{z_1, z_2} p^{(Z_1, Z_2)}(z_1, z_2) \log(p^{(Z_1, Z_2)}(z_1, z_2)), \tag{4.3}$$

where $p^{(Z_1, Z_2)}(z_1, z_2) = \mathbb{P}(Z_1 = z_1, Z_2 = z_2)$ is the joint probability mass function of Z_1 and Z_2. In the same way as before, we can construct an estimator of $H(Z_1, Z_2)$, which we denote $\widehat{H(Z_1, Z_2)}$, by using the empirical distribution of (Z_1, Z_2).

To measure the stochastic relationship between the variables Z_1 and Z_2, we can consider mutual information defined as

$$I(Z_1, Z_2) := \sum_{z_1, z_2} p^{(Z_1, Z_2)}(z_1, z_2) \log \frac{p^{(Z_1, Z_2)}(z_1, z_2)}{p^{Z_1}(z_1) p^{Z_2}(z_2)},$$

where $p^{Z_1}(z_1)$ and $p^{Z_2}(z_2)$ are the marginal probability mass functions of Z_1 and Z_2, respectively. The mutual information is null if and only if the variables are independent, and increases when the dependence increases. Moreover, mutual information is related to the entropies by

$$I(Z_1, Z_2) = H(Z_1) + H(Z_2) - H(Z_1, Z_2).$$

Thus, a natural estimate $\widehat{I(Z_1, Z_2)}$ of the mutual information between Z_1 and Z_2 is given by

$$\widehat{I(Z_1, Z_2)} = \widehat{H(Z_1)} + \widehat{H(Z_2)} - \widehat{H(Z_1, Z_2)}. \tag{4.4}$$

We remark that entropy and mutual information can also be defined in the case of continuous real-valued random variables, where all sums become integrals. However, the practical computation in the continuous case is more difficult because it requires estimates of the densities of the continuous random variables and the numerical approximation of integrals. For this reason, it is preferable to consider the discrete case.

4.2.2 Methodology for the influence criterion

In the context of the influence analysis tool, which is a support tool for the analysis of a manufacturing process, we dispose of the following variables: a performance measure $\tilde{\mathbf{Y}}$ (e.g. a measure of the production quality) and operational variables of the manufacturing process $\mathbf{X}$. Here $\mathbf{X}$ can be a mixed random vector composed of $d \in \mathbb{N}$ quantitative and/or qualitative variables and the output $\tilde{\mathbf{Y}}$ can be categorical or quantitative. We suppose to observe an i.i.d. sample $\mathcal{D}_n := \{(x_1, \tilde{y_1}), \ldots, (x_n, \tilde{y_n})\}$, where each $(x_i, \tilde{y_i})$ is a realisation of $(\mathbf{X}, \tilde{\mathbf{Y}})$. Let $X = [x^1, \ldots, x^d] = [x_1, \ldots, x_n]^T \in \mathbb{R}^{n \times d}$ be the input matrix of d variables, where x^j denotes the j-th variable and $\tilde{Y} = [\tilde{y_1}, \ldots, \tilde{y_n}]$ be the vector of the n observations of the response variable.

Recall that we aim at identifying the factors that cause a degradation or an improvement in the performance measure. In other words, the goal is to find the causes that have led to unusual measurements.

To this end we first define a discrete performance indicator denoted by $\mathbf{Y}$ which represents a score of the production quality $\tilde{\mathbf{Y}}$. For example, a performance indicator can be considered as a binary output that either specifies an improvement or a deterioration of production quality. We may assign the label 1 for an anomaly and 0 otherwise. This helps to identify and understand why a process or system has failed or caused a problem. In addition to having a simpler interpretation, this categorization may simplify certain calculations. The categorization is done by defining thresholds and creating classes defined by intervals for the performance measure $\tilde{\mathbf{Y}}$. In particular, in the case where $\tilde{\mathbf{Y}}$ is continuous, the classes can be obtained using the associated empirical quantiles. In the discrete case, $\mathbf{Y}$ can be chosen identical to $\tilde{\mathbf{Y}}$. Now, let $Y = [y_1, \ldots, y_n]$ be the discrete vector of n observations.

The purpose of the influence analysis tool is to rank the potential causes, that is the d variables $x^1, \ldots, x^d$, in decreasing order of their influence on the performance. The potential causes are extracted from operational measurements. Business experts can sometimes easily identify obvious and predictable first causes. For example, the temperature can be known to be a major cause of deviations in the performance measure. However, the situation is generally more complex and often it is a combination of a number of parameters that are jointly responsible for the variation in performance (e.g. both temperature and pressure). The influence analysis tool aims to automatically identify the different causes, whether they are simple or complex, and to quantify the influence of each of them.

4.2.3 Influence of a group of variables

The general idea is as follows. First a subset $\mathcal{I} \subset \{1, \ldots, d\}$ of variables is chosen, whose influence on the performance we wish to determine. Then we build a classification model that explains the performance Y as a function of the subset of variables using the submatrix $X_{\mathcal{I}}$ of X with $|\mathcal{I}|$ columns. We denote by $\hat{Y}_{\mathcal{I}}$ the predictions obtained by the classification model and evaluate a criterion of influence of the model defined as the normalized mutual information between the predicted values $\hat{Y}_{\mathcal{I}}$ and the true values Y. More precisely, the influence criterion $\lambda(\hat{Y}_{\mathcal{I}})$ is given by

$$\lambda(\hat{Y}_{\mathcal{I}}) := \frac{\widehat{I(\hat{Y}_{\mathcal{I}}, Y)}}{\widehat{H(\hat{Y}_{\mathcal{I}})}} \in [0, 1]. \tag{4.5}$$

The influence criterion indicates the proportion of performance explained by the model. The normalized mutual information can be used as a measure to compare two clusterings of the same dataset [104]. The mutual information is widely used for variable selection [3, 103, 17]. A model trained on potential causes $X_{\mathcal{I}}$ with a value of $\lambda(\hat{Y}_{\mathcal{I}})$ close to 1, means that this model is relevant and that $X_{\mathcal{I}}$ contains the main causes that explain the variations in the performance measure.

More precisely, the indicator $\lambda(\hat{Y}_{\mathcal{I}})$ allows us to evaluate the relevance and the influence of a group of potential cause variables $X_{\mathcal{I}}$. It thus makes it possible to compare different models that use different cause variables, that is different sets $\mathcal{I}$ of variables. We can thus consider this indicator as a model importance measure. If few operational variables are available, an exhaustive study can be performed by comparing models trained on all possible subsets $\mathcal{I}$ of variables. However, in high dimension with a high number of operational variables, this approach can be very costly in terms of computation time or even impossible. We adresse this issue in Section 4.2.5 and propose a feasible strategy to explore relevant relevant subsets $\mathcal{I}$ of variables.

4.2.4 Influence of a variable within a given model

To determine the relevance and the relative influence of each variable x^j in a given model with variable $X_{\mathcal{I}}$, we can use any existing variable importance measure. Here, we propose to evaluate the loss in terms of influence of deleting variable x^j from the model. This consists in training a new classification model without this variable while keeping the others, that is $\mathcal{I}\backslash\{j\}$, and calculating its influence criterion. The difference obtained between the full initial model and the sub-model defines the influence of the variable x^j. The larger the difference, the more influential the variable is considered to be. We denote by $\alpha_{\mathcal{I}}^{x^j}$ the normalized influence of the variable x^j which is defined as

$$\alpha_{\mathcal{I}}^{x^j} := \frac{\lambda(\hat{Y}_{\mathcal{I}}) - \lambda(\hat{Y}_{\mathcal{I}\backslash\{j\}})}{\sum_k \lambda(\hat{Y}_{\mathcal{I}}) - \lambda(\hat{Y}_{\mathcal{I}\backslash\{k\}})}, \tag{4.6}$$

where $\hat{Y}_{\mathcal{I}\backslash\{j\}}$ is the predicted performance indicator based on the subset $\mathcal{I}\backslash\{j\}$ after deleting the variable x^j from the potential causes $X_{\mathcal{I}}$.

4.2.5 The influence tool in practice

In the implemented influence analysis tool used at Safran, the chosen classification model are based on decision trees. Since the causes of the variations in the performance are physical phenomena that need to be easily explained to experts. Decision trees give interpretable results (especially on the partition of data) and indicate the best parameters (intervals that classify the potential causes X_j) to have an optimal performance (degradation or amelioration of production quality). This tool is therefore a help and support for business experts. It can easily help the operator to make modifications, improvements (adjustments) on the system. A representation of the indicators in the form of logical rules is simple, interpretable and easier to convince (such as "if the temperature is high and the pressure is high, then the performance is low"), especially if the indicator is multivariate (mixture of several parameters/ measurements). This makes it easy to identify the potential causes of performance degradation and thus to improve the manufacturing process. The data is quantified in order to improve the mutual information with the performance indicator. Moreover, the construction of the decision tree is based on the entropy to discretize the input data into a moderate number of classes and a good distribution between the different classes.

If we want to compare all possible models, we need to consider all possible combinations of variables which can be laborious and time-consuming, especially when the d dimension is large. Instead of exploring a large number of subsets of variables explicitly, we can learn several decision trees with varying complexity (that is the depth of tree) on the entire set of variables and compare them via the influence criterion λ. That is, we let the decision tree identify and select the main causes

itself in an automatic way. The decision tree model is thus very practical in large dimension. More precisely, a decision tree trained on all d variables with maximum complexity M will select and use at most M tuples of variables among X to classify the data. To find the main cause, one simply sets the maximum complexity equal to 1. In general, the maximum size of the tree is gradually increased to combine and cross several causes and thus make them more complex. This can be particularly useful in cases where it is difficult to define them ourselves, as causes can often be complex and multivariate. This allows to avoid comparing several models with different subsets of possible variables.

However, correlations between variables can have a selection bias effect in the decision tree model. The handling of correlations discussed in the next section has recently been integrated into the tool at Safran.

4.2.6 Clustering of correlated variables

The variable importance measure in classification trees is sensitive to the presence of correlations between input variables. That is, correlations between variables can bias the relative influence of the variables. Indeed, if two measures are highly correlated, one or the other can be chosen as a cause of variation in the performance measure. The criterion of influence will be roughly the same by keeping one or the other. Moreover, in the presence of strong correlations between variables, interpretation can become difficult.

Now, it is well known that data compression through variable clustering reduces dimensionality and improves interpretability. Based on this, Chavent et al. [15] propose a way to handle correlated features in influence analysis based on random forests that consists in first performing hierarchical clustering of variables in order to remove redundant information and create independent synthetic variables on which the model will be trained. The hierarchical clustering of variables into say K clusters $C_1, \ldots, C_K \subset \{1, \ldots, d\}$ is based on correlations such that correlated variables are grouped together.

Then a synthetic variable $f^k \in \mathbb{R}^n$ is computed for every cluster C_k which extracts and summarizes the important information shared in the set of variables of cluster C_k. Chavent et al. [15] choose as synthetic variable f^k the first principal component associated with cluster C_k defined as

$$f^k := \underset{z \in \mathbf{R^n}}{\operatorname{argmax}} \left\{ \sum_{x^j \in C_k} r^2_{x^j,z} \right\},$$

where $r^2_{x^j,z}$ is the squared Pearson correlation between z and x^j if x^j is a quantitative variable and the correlation ratio if x^j is the categorical variable. The variable f^k extracts and summarizes the important information present in the set of variables of cluster C_k. It is an important advantage of the approach that the hierarchical clustering can be realized on mixed type data.

In this work, we apply the same strategy for our influence analysis tool based on decision trees. A clustering of the variables is thus performed before training the decision trees on the corresponding synthetic variables.

The user can choose the number of clusters. Business experts can provide their help and knowledge in partitioning variables and choosing the number of clusters. Otherwise, it can be chosen by cross-validation to assist the user. More precisely, first we build the tree of variables obtained by hierarchical clustering. Then we divide the dataset into two samples. In our implementation, the first one is made of 70% of the training data on which the decision tree model is learned and the rest is the validation sample on which the decision tree model will be validated. We now fix these two samples. For each $K = 2, \dots, d$, we train on the training data a decision tree model based on the K synthetic variables $f^1, \dots, f^K$ as predictors and the quantized performance indicator $\hat{Y}$ as output. We choose the optimal number of clusters $\hat{K}$ that minimizes the model error on the test set.

The influence criterion is finally computed on the $\hat{K}$ synthetic variables to measure the importances of the synthetic variables. We can also obtain their normalized influence by using (4.6). To return to the original variables and determine their influence, we choose to use for cluster C_k the correlations between the synthetic variable f^k and the original variable x_j in C_k. The influence of the variable $x^j \in C_k$ is defined as

$$\tilde{\alpha}^{x^j} = \frac{\alpha^{f^k}}{|C_k|} \times r^2_{x^j, f^k}.$$

We propose to normalize the influence of each synthetic variable by the size of its cluster. Larger clusters tend to have a greater influence, as they contain more information than smaller ones. Gregorutti et al. [31] propose a permutation importance measure for groups of variables in random forests, and also suggests rescaling importance measures by taking into account the group size. Furthermore, when groups of different sizes have the same importance, normalization will favor smaller groups, which are easier to interpret.

We normalize the influences of the variables in order to classify the variables within a cluster in order of influence as follows

$$\alpha^{x^j} = \frac{\tilde{\alpha}^{x^j}}{\sum_{j' \in C_k} \tilde{\alpha}^{x^{j'}}} \quad \forall\, j \in C_k. \tag{4.7}$$

4.2.7 Application of influence analysis to identify manufacturing process conditions related to part anomaly

In this section, we illustrate the influence analysis tool with a real-life application on blade manufacturing data. Measurements of 17 parameters on the state of the manufacturing process for 300 parts are available. The manufacturing quality of a part,

Algorithm 4: Influence analysis algorithm with correlated variables

Input: Data matrix $X = [x^1, \ldots, x^d] \in \mathbb{R}^{n \times d}$, output variable $Y \in \mathbb{R}^n$, number of clusters K and maximum complexity M.

Apply clustering to X with K clusters and compute the corresponding synthetic variables $(f^1, \ldots, f^K)$;

Train a decision tree of maximum complexity M on $(f^1, \ldots, f^K)$ to obtain the prediction $\hat{Y}$. Denote by m the number of selected synthetic variables by the tree;

Compute the criterion of influence of the model $\lambda(\hat{Y})$ using (4.5) ;

for $j = 1, \ldots, m$ **do**

  Train a new decision tree model without f^j keeping the other $m-1$ variables with maximum complexity M;

  Compute the normalized influence α^{f^j} using (4.6);

end

Set $\alpha^{f^j} = 0$ for $j = m+1, \ldots, K$;

for $j = 1, \ldots, d$ **do**

  Compute the normalized influence α^{x^j} using (4.7) ;

end

Rank the d variables $x^1, \ldots, x^d$ in decreasing order of their influence α^{x^j};

Output: The influence criterion λ of the model, ranking of the variables and their influence measure.

characterized by a level of anomaly at a given operating time, is transformed into a binary variable which is 0 if the part is anormal and 1 otherwise. There are 190 parts that are anormal. The aim is to determine the manufacturing conditions that caused these anomalies. However, the dataset contains missing data, particularly on 10 parts which the influence analysis tool removes by default. To highlight the interest of handling correlation in the tool, we will present the results of influence analysis with and without clustering of correlated variables.

First, the influence analysis is performed without clustering of correlated variables with a maximum complexity of 5, i.e. a maximum of 5 factors that would have caused an anormal part are investigated. Of the 17 process parameters, the best model retains only the following 5 in order: X_{17}, X_{12}, X_8, X_2 and X_5. Their ranking and their normalized influence are presented in Table 4.1. The influence of this group of variables on part anomaly is 74.01%. The influence of the other unselected variables is 0.

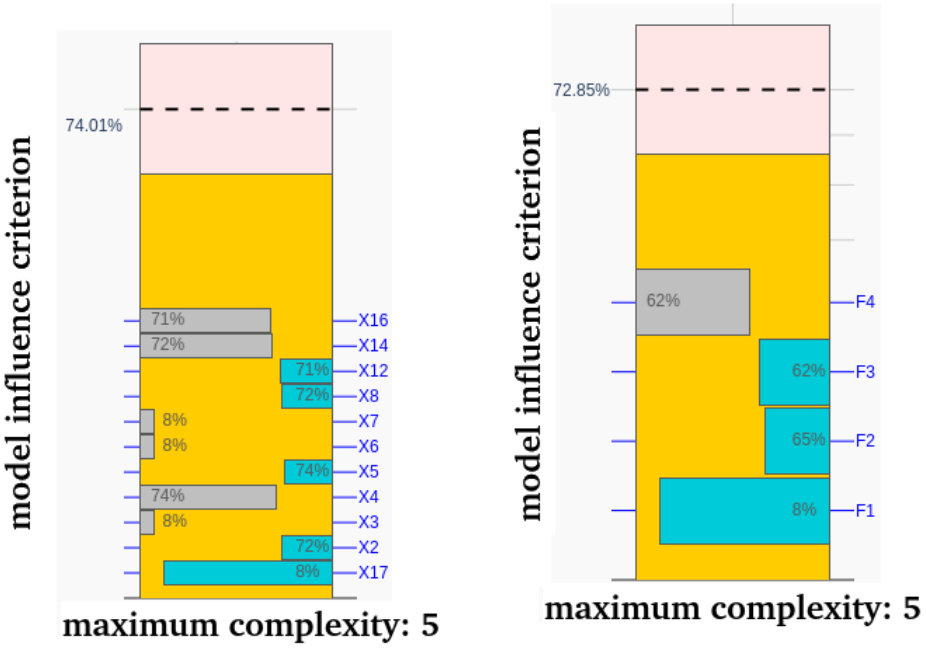
Figure 4.1a summarizes the results of the influence analysis. The dotted black line on the top of the bar indicates the criterion of influence and the light pink part is the confidence interval. For each selected variable, a small blue horizontal bar indicates the influence criterion of the sub-model trained without the variable

Table 4.1: Ranking of selected variables and their normalized influence

	α^x
X17	0.504
X12	0.130
X8	0.130
X2	0.130
X5	0.106

Table 4.2: Ranking and normalized influence of the 3 selected syntetic variables and their associated origin variables

Synthetic variables		Initial variables	
F1	0.735	X17	0.77
F2	0.147	X12	0.0228
		X1	0.0228
		X6	0.022
		X7	0.0206
		X8	0.0202
		X10	0.018
F3	0.118	X13	0.0234
		X14	0.023
		X16	0.0227
		X15	0.0226
		X2	0.010



(a) Without clustering of correlated variables

(b) With clustering of correlated variables

Figure 4.1: Results of influence analysis with a maximum complexity of 5 without clustering of correlated variables (a) and on the 5 synthetic variables (b).

with a maximum complexity equal to 5. For example, the influence criterion of the model trained without variable X_{17} is only 8%. This makes it possible to identify the relative influence of each of the selected parameters and to rank them in order of influence. Variables with small gray horizontal bars are those selected by one of the 5 sub-models. The value on the bar indicates the criterion of influence of the best sub-model that selected the variable. This helps to easily identify variables that could be replaced and included as potential causes.

Figure 4.2 shows the correlation matrix between the 17 variables. We observe some groups of highly correlated variables that can distort the results of the influence analysis. Then a hierarchical clustering of variables may be necessary. Figure 4.3 shows the dendrogram of variable partitions separated into 5 clusters. For ease of interpretation, it is preferable that the number of clusters is not too large. Each group which is related to a specific manufacturing process condition is represented by a synthetic variable. We can now perform influence analysis on the 5 synthetic variables denoted by $f^1, \ldots, f^5$.

Figure 4.1b summarizes the results of the influence analysis on these 5 synthetic variables with a maximum complexity of 5. The best model retains only the first 3 variables. Ranking and normalized influence of these variables and their associated

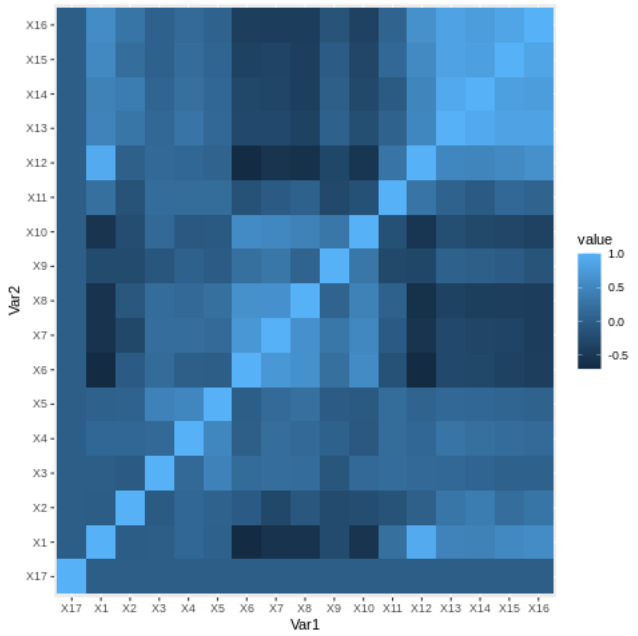


Figure 4.2: Correlation matrix between between the 17 manufacturing process parameters

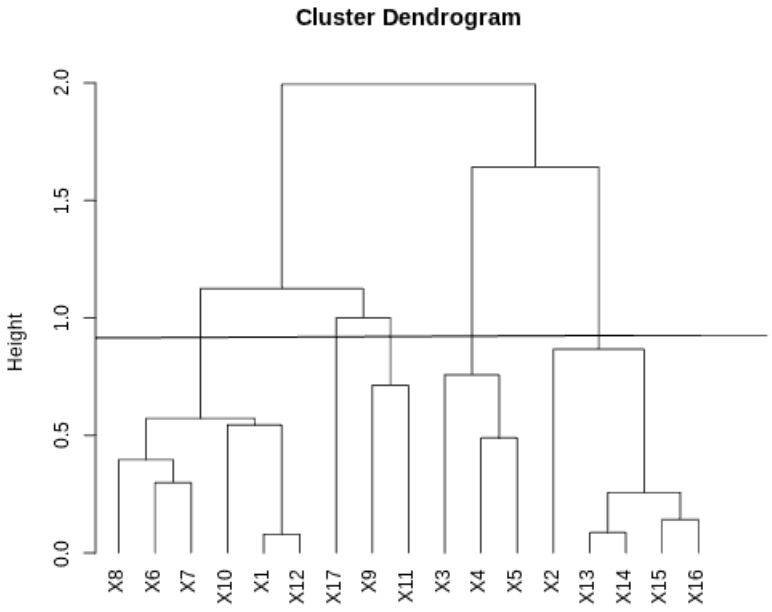


Figure 4.3: Dendrogram of variable partitions into 5 clusters

origin variables are presented in Table 4.2. The influence of this group of variables on part anomaly is 72.85%. The influence of the other unselected variables is 0.

By partitioning the variables into 5 groups, we were able to reduce the number of variables to be analyzed by the tool, while retaining the essential information in the data. In fact, the rate of explanation of part anomaly by the synthetic variables remains high and very close to that obtained on the original variables. What's more, partioning makes it possible both to select a whole group of relevant correlated variables and to establish a ranking of these variables within the group itself. Without partioning, only a few variables within a group are selected, resulting in selection bias. For example, in Table 4.1, X_2 was initially chosen as influential to the detriment of the other correlated variables, whereas X_{13} appears to be more relevant as we can see in Table 4.2.
On the basis of this data, variable X_{17} appears to be the main source of anomalies to the parts. This parameter should be corrected to avoid future anomalies.

4.3 Influence analysis with incomplete data

In this section, we focus on the problem of missing data in the influence analysis tool.

4.3.1 The state of the art

Now we consider the problem of handling missing data in the influence analysis tool. Essentially, the question is how to deal with missing entries in the first stage of the tool, that is when variables are clustered to form synthetic variables. The statistical literature provides a number of approaches to handle incomplete data. A first simple approach (referred to as *deletion* in the figures) consists in deleting incomplete observations and then applying the influence analysis tool to the remaining data. However, this approach is inefficient in the presence of a large number of incomplete observations, and impossible if all observations are incomplete.

A more conservative approach called *cov.pairwise.complete.obs* is to keep all observations, even those that are partially observed. This approach performs calculations of the covariances only on the observed components of the variables without any imputation. The synthetic variable of a cluster of variables $X_{\text{clust}} \in \mathbb{R}^{n\times p}$ can be obtained from the singular value decomposition of the correlation matrix $\text{corr}(X_{\text{clust}})$ of the matrix X_{clust}, which is a positive symmetric matrix. Thus, there exists an orthogonal matrix $V \in \mathbb{R}^{p\times p}$ and a diagonal matrix $D \in \mathbb{R}^{p\times p}$ composed of singular values such that

$$\text{corr}(X_{\text{clust}}) = VDV^T.$$

The first principal component of X_{clust} is a linear combination of the original variables: $f_1 := X_{\text{clust}}V_1$ where V_1 is the first column of the matrix V representing the

first eigenvector. In the case of incomplete data, correlations between each pair of variables are computed using all complete pairs of observations on the involved variables. Doquire and Verleysen [23] use the same strategy to measure dependence between variables based on the mutual information.

In our experiments, we will consider different imputation methods that can be used to first create a complete data set on which the influence analysis tool is applied as usual. A basic approach (here referred to as *mean*) is the imputation by the mean value of the observed variables. A non-parametric approach called *missForest* predicts missing values using a random forest trained on the observed part of the dataset. The imputation model can be used for mixed type data with complex interactions and non-linear dependencies. Other popular imputation methods include the iterative Principal Component Analysis (PCA) multiple imputation method referred to as *PCA*, which uses correlations between variables and similarities between individuals to iteratively impute missing data by a PCA-type model [43]. This is particularly interesting for the variable grouping stage, which is based on correlations between variables. Finally, we consider a multiple imputation method based on the popular MICE algorithm (here referred to as *mice*). However, in multiple imputation, the aggregation of influence analysis results obtained on each of the imputed datasets can be difficult, particularly in terms of variable selection. In addition, interpretation of the results is lost, as it becomes more difficult with several decision trees fitted to different imputed datasets. Wood et al. [108] highlight the risk of selecting irrelevant variables when selection is carried out separately on each imputed dataset. A possible and simple alternative is to perform the analysis on a single stacked data set [108, 24]. An observation weight by $\frac{1}{D}$ is added where D is the number of imputed datasets. On the other hand, optimizing hyperparameters by cross-validating the merged matrix may increase the risk of overlearning. Rodgers et al. [80] propose a modified multiple imputation approach for decision tree models when data are missing. A decision tree model is fitted to the stacked dataset, taking as hyperparameters the average of the optimized hyperparameters obtained on each imputed dataset.

4.3.2 Numerical experiments

Our aim is to study the impact of imputation methods in terms of influence analysis, in particular on the influence criterion of the model, ranking of the explanatory variables and their influence measures. We are mainly interested in the case where the number of missing data is high, as this is fairly common in real applications. We are also interested in the complex case of data with interaction effects between variables. In this section, the performance of the methods is evaluated and compared to the results in the case of complete data. A simulation study is carried out to assess the quality of influence analysis results on incomplete data under different conditions: varying amount of missing data, different mechanisms of missingness, various

imputation techniques and sample sizes. The simulation study also illustrates the advantages and drawbacks of each method.

Synthetic datasets

For the numerical experiments, two regression models are considered to simulate datasets. First, let $X = (x_1, \ldots, x_p)$ be a random vector of dimension $p = 19$ composed of 10 independent groups. We generate 50 datasets from a model developed by Friedman [29] with complex non-linear relationships between the output and input variables, which is often used in the literature [44, 23] and defined by

$$Y = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 5x_4 + 10x_5 + \epsilon,$$

where $Y \in \mathbb{R}^n$ is the output variable and $\epsilon \in \mathbb{R}^n$ is an independent centered gaussian random variable with standard deviation equal to 0.1. The first five variables that are relevant $[x_1, \ldots, x_5] \in \mathbb{R}^{n \times 5}$ have multivariate Gaussian distribution with independent components, mean 1 and standard deviation 1. The last five non-informative variables $[x_{15}, \ldots, x_{19}] \in \mathbb{R}^{n \times 5}$ added to the data are multivariate Gaussian distribution of independent components with mean 1 and standard deviation 1. The group of correlated variables $[x_5, \ldots, x_{14}] \in \mathbb{R}^{n \times 10}$ is a gaussian random vector of components of mean 1 and variance 1, where all pairs of components have a correlation of 0.7.

In the second setting, we consider a random vector X of dimension $p = 28$ composed of 11 independent groups and we simulate 50 datasets using the following linear regression model

$$Y = 2x_1 + 1.5x_2 + x_3 + x_4 + 0.5x_5 + \epsilon,$$

where $Y \in \mathbb{R}^n$ is the output variable and $\epsilon \in \mathbb{R}^n$ is an independent centered gaussian with standard deviation 0.1. The first five variables that are relevant $[x_1, \ldots, x_5] \in \mathbb{R}^{n \times 5}$ are multivariate Gaussian distribution of independent components with mean 1 and standard deviation 1. The group of correlated variables $[x_5, \ldots, x_{14}] \in \mathbb{R}^{n \times 10}$ is a gaussian random vector of components of mean 1 and variance 1 where two distinct components have a correlation of 0.7. Moreover, a group composed of five correlated non-informative variables $[x_{19}, \ldots, x_{23}] \in \mathbb{R}^{n \times 5}$ is a gaussian random vector of components of mean 1 and variance 1 where two distinct components have a correlation of 0.7. The last five non-informative variables $[x_{24}, \ldots, x_{28}] \in \mathbb{R}^{n \times 5}$ added to the data are multivariate Gaussian distribution of independent components with mean 1 and standard deviation 1. We introduce non-linear and linear interactions between the first variables in the following way

- $x_{15} = \sin(x_1) + \epsilon_1$,
- $x_{16} = x_2^2 + \epsilon_2$,

- $x_{17} = x_3 - 5 + \epsilon_3$,
- $x_{18} = 2x_4 + 5 + \epsilon_4$,

where the $\epsilon_i \in \mathbb{R}^n$ are independent centered gaussian vectors with independent components and standard deviation 0.05.

In both settings, we consider different sample sizes, namely 100, 300, 600 and 1000. For each sample size, we simulate missing data on all variables using the ampute function from the R package `mice` [100]. Different proportions of missing values, namely 10%, 20%, 40% and 60%, and different mechanisms of missingness are considered. The literature on missing data traditionally distinguishes three mechanisms that lead to missingness [83] and it is well known that the performance of imputation methods may be sensitive to the mechansim at work. The mechanism is said to be *missing completely at random* (MCAR), when the causes of missing values are independent from the data and the probability of being absent is the same for all items. That is, a subset of observations is chosen at random using independent Bernoulli variables with fixed success probability for all entries. In contrast, when the probability that a value is missing depends on the values of the observed variables, the mechanism is called *missing at random* (MAR). In our simulations, to obtain MAR, for each variable, missing values are obtained using a logistic regression model depending only on the other variables. Finally, when the probability of being absent also depends on the unobserved value, the mechanism is called *missing not at random* (MNAR) and we can consider missing data simulated by using a logistic regression model depending on all variables.

Performance criteria

Different criteria are considered to evaluate the methods. The influence analysis tool is decomposed into two main operations : the variable clustering and the classical influence analysis. As we are mainly interested in the impact of missing data on the results of the influence analysis, the true number of clusters K and the clustering of correlated variables (variable partitioning) are specified and fixed in the simulations for each dataset. This makes it possible to correctly evaluate and compare methods for handling missing data in terms of influence analysis, without taking into account any errors linked to variable clustering. Note that in the Friedman model, K is set to 10, and in the linear regression model, K is set to 11.
The methods will be evaluated in terms of influence analysis. First, for each method, we compare the model influence criterion obtained in the case of complete data with that obtained in the case of incomplete data. Now let $\Delta\lambda$ be the obtained difference defined by

$$\Delta\lambda := \frac{\lambda(\hat{Y}^{\text{full}}) - \lambda(\hat{Y}^{\text{miss}})}{\lambda(\hat{Y}^{\text{full}})},$$

where $\lambda(\hat{Y}^{\text{full}})$ and $\lambda(\hat{Y}^{\text{miss}})$ are respectively the model influence criterion in the complete and incomplete case.

Then, we will assess the impact of methods on the distinction between relevant and irrelevant variables. First, we will compare the influence of variables in the complete and incomplete cases. Let $\Delta\alpha^{x^j}$ be the relative difference in influence of variable x^j defined by

$$\Delta\alpha^{x^j} := \frac{\alpha^{x^j}_{\text{full}} - \alpha^{x^j}_{\text{miss}}}{\alpha^{x^j}_{\text{full}}},$$

where $\alpha^{x^j}_{\text{full}}$ and $\alpha^{x^j}_{\text{miss}}$ are respectively the influence of variable x^j in the complete and incomplete case. In particular, we will observe the relative difference in the measure of influence obtained on the most influential variable in the model, namely x_3 in the Friedman model and x_1 in the linear regression model.

Finally, the variables are ranked in order of influence. For each method, we compare the ranking obtained in the complete case with that obtained in the incomplete case. Spearman's correlation between variable rankings will be used. We will evaluate whether the most influent variables in the missing data setting are ranked in the same order as in the complete data setting.

Results in terms of influence criterion

Figures 4.4 and 4.5 show respectively the relative difference of the model influence criterion between the complete and incomplete data on the Friedman and linear regression models for $N = 100$. It can be seen that as the percentage of missing data increases, the influence criterion of the model in the incomplete case deteriorates. It decreases and becomes progressively weaker than that obtained in the complete case. This is particularly true for the *mice* method. The maximum complexity of the decision tree trained on the single stacked data set containing more observations remains fairly low. This results in an underfitting effect. Conversely, we observe an overfitting effect for the *deletion* method, which is applied to smaller datasets than in the complete case. This effect explains the negative relative difference observed in Figure. The other studied methods are quite similar, although the *cov.pairwise.complete.obs* method performs slightly better.
Figure 4.6 shows the relative difference of the model influence criterion between the complete and incomplete data on the Friedman model by varying N from 100 to 1000 with 20% missing data. The relative difference of the model influence criterion increases slightly as the sample size of the simulated data sets increases, and seems to stabilize from $N = 300$.

Results in terms of influence measures

We are particularly interested in the measure of influence of the most informative variable in the model, namely x_3 in the Friedman model and x_1 in the linear model.

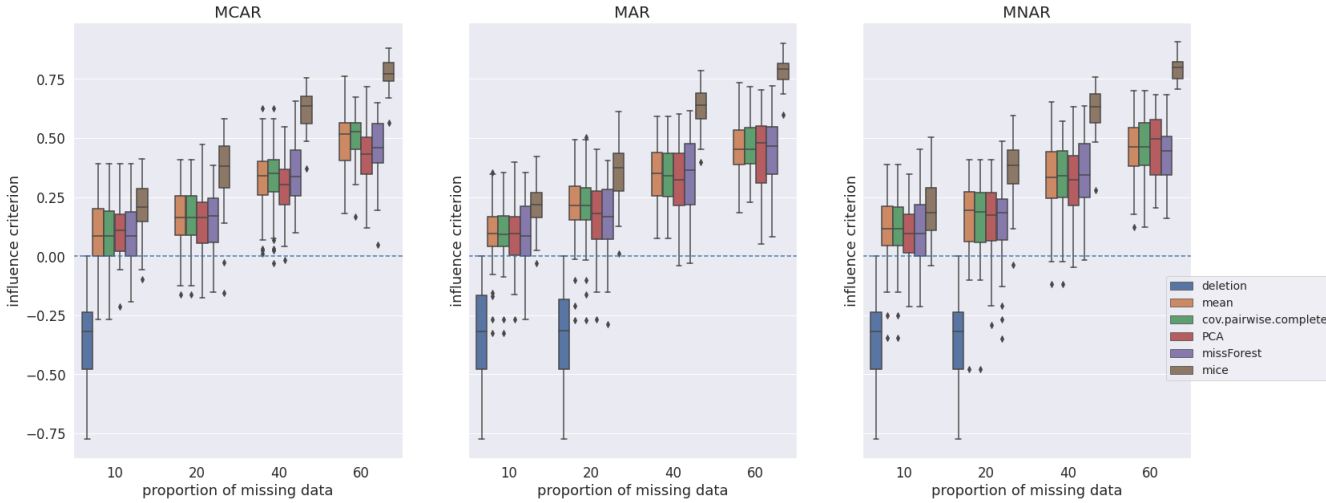


Figure 4.4: Relative difference $\Delta\lambda$ of the model influence criterion between the complete and incomplete data on the Friedman model.

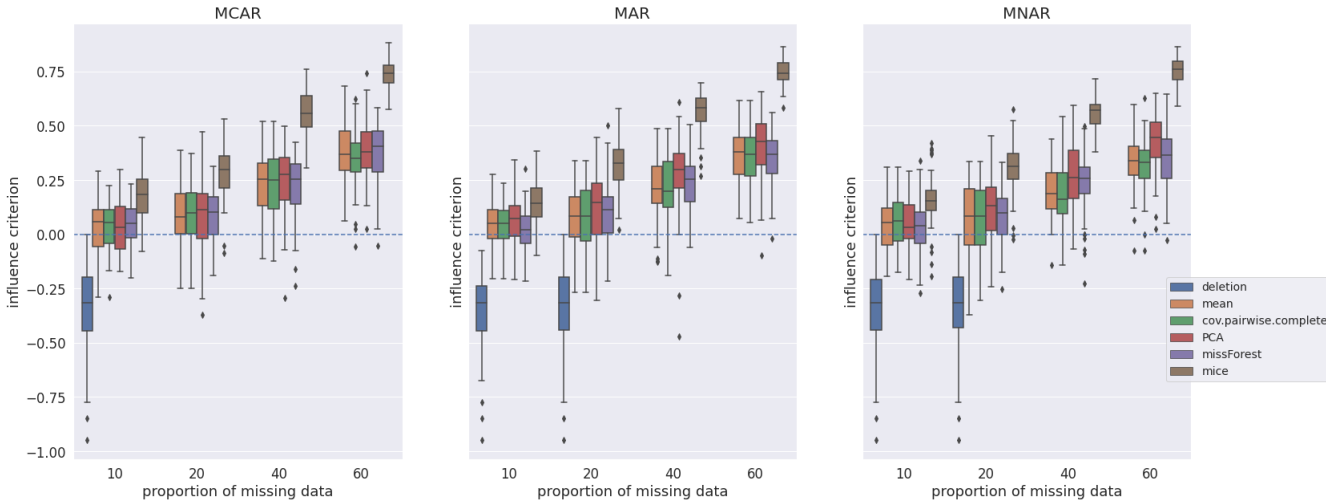


Figure 4.5: Relative difference $\Delta\lambda$ of the model influence criterion between the complete and incomplete data on the linear regression model.

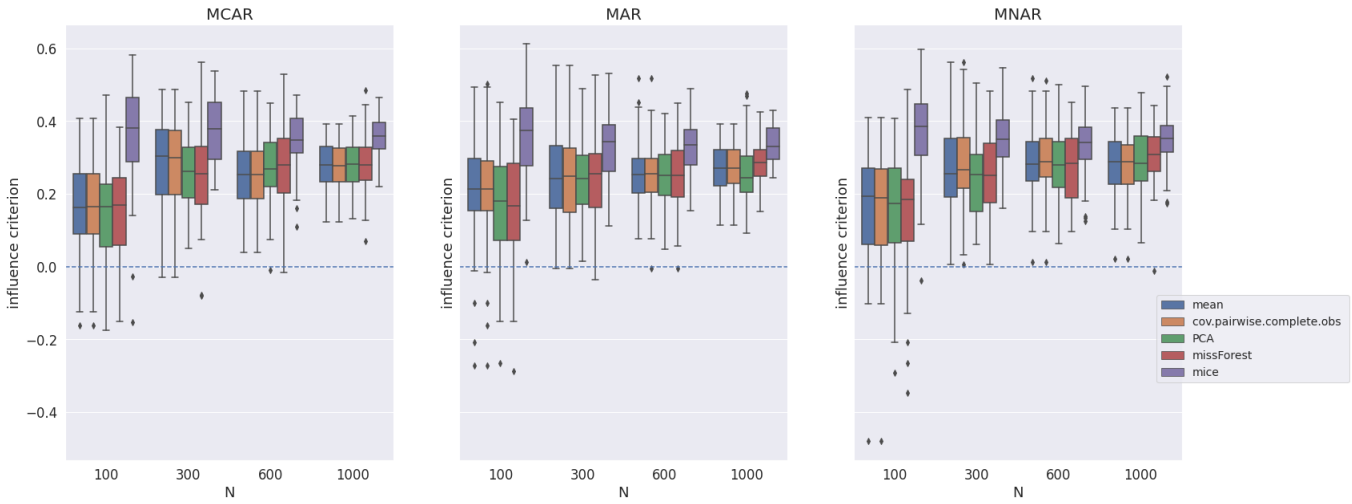


Figure 4.6: Relative difference $\Delta\lambda$ of the model influence criterion between the complete and incomplete data on the Friedman model by varying N with 20% missing data.

Figures 4.7 and 4.8 represent respectively the relative difference in influence of the most informative variable in the Friedman and linear model for $N = 100$. We can see that the methods are globally similar to each other, with the exception of the *deletion* method, which performs least well and is least robust. In the Friedman model, the *mice* method performs best on datasets containing a large number of missing data. The *cov.pairwise.complete.obs* method also performs well on such scenarios, with much lower computation times.
The same behavior is observed when increasing the sample size of the simulated data sets. Figure 4.9 shows the difference in the measure of influence of the first relevant variable between complete and incomplete data on the Friedman model by varying N from 100 to 1000 with 20% missing data. Moreover, we observe that the *deletion* method doesn't improve even when increasing N. It tends to underestimate the influence of the most important variable in the model, always with significant instability.

Ranking of variables according to their influence

The first objective of the influence analysis tool is to identify, in terms of importance, the first main causes of a deviation of the performance indicator. In the following, variables are ranked in descending order of importance according to their measure of influence.

Our aim is to assess the impact of missing data on the selection of influential variables. In this way, we will compare to the results in the case of complete data used as a benchmark.

For each of the two models, we will compare the ranking of variables obtained on

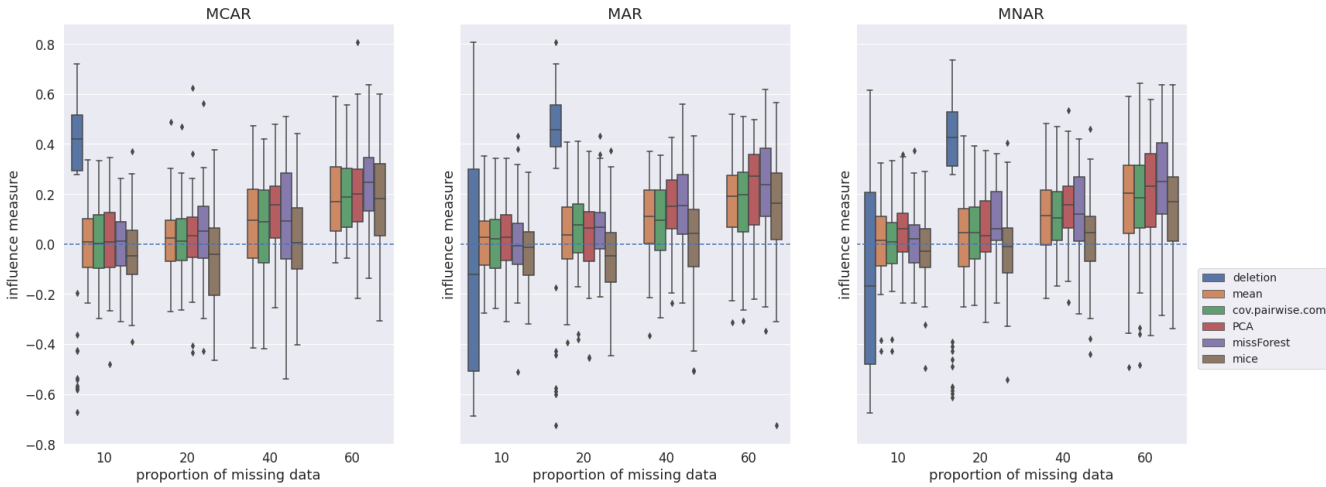


Figure 4.7: Difference in the measure of influence $\Delta\alpha^{x^j}$ of the first relevant variable between complete and incomplete data on the Friedman model.

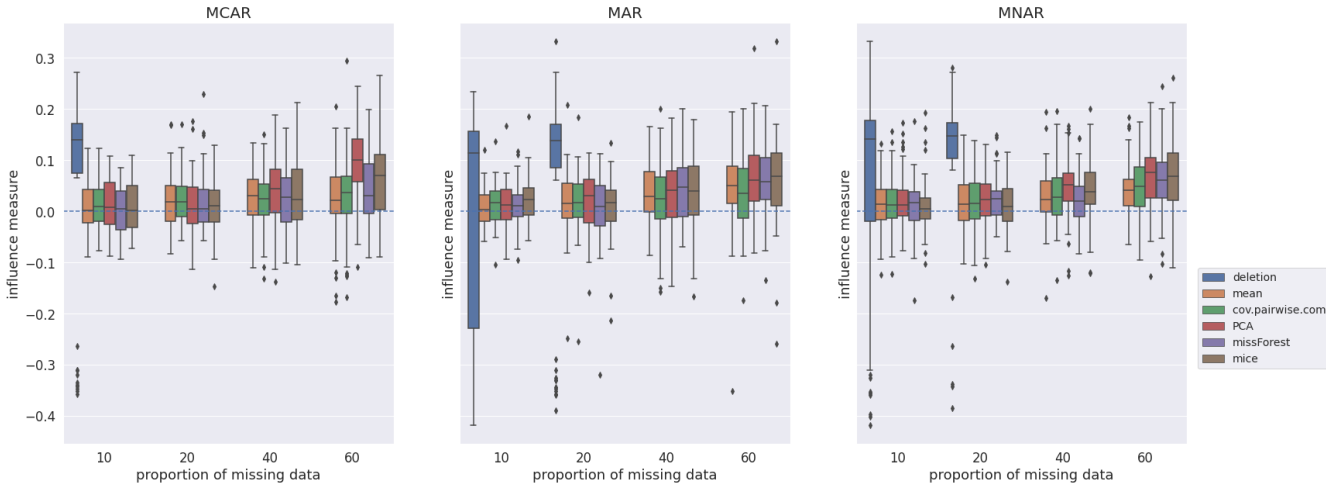


Figure 4.8: Difference in the measure of influence $\Delta\alpha^{x^j}$ of the first relevant variable between complete and incomplete data on the linear regression model.

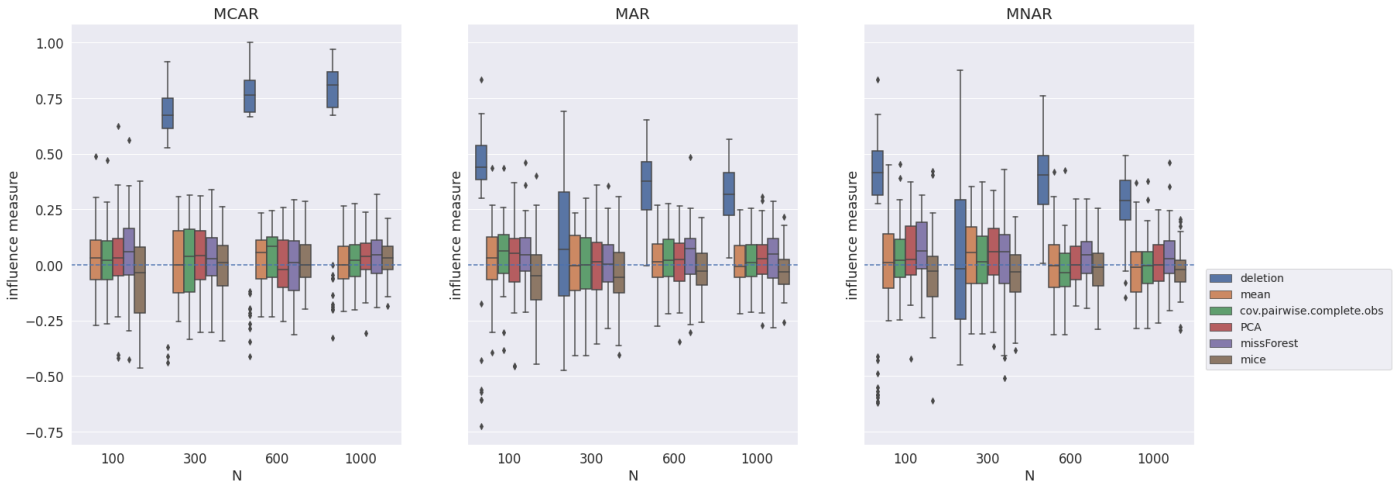


Figure 4.9: Difference in the measure of influence $\Delta\alpha^{x^j}$ of the first relevant variable between complete and incomplete data on the Friedman model by varying N with 20% missing data.

complete data with that obtained on incomplete data. The Spearman correlation of the two rankings will enable us to assess whether the relevant variables have been ranked in the same order of influence as in the case of complete data. The closer the Spearman correlation is to 1, the closer and more identical are the rankings of the variables in the complete and incomplete cases. However, the closer the spearman correlation is to -1, the more the rankings of the variables in the complete and incomplete cases are opposite and different.
Figure 4.10 summarizes the results obtained in the Friedman model according to the different scenarios proposed for $N = 100$. Even for small proportions of missing data, *deletion* is the least efficient of all the methods. *deletion* is also highly unstable, particularly for 20% missing data, where we observe considerable variability in the results. The method proves to be ineffective, and is generally no longer applicable beyond 20% missing data.
The other studied methods are fairly similar. However, *missForest* method appears to be less effective as the percentage of missing data increases. For such a scenario, *mice*, which takes into account the uncertainty due to missing data, performs better. However, the computation time for this method and for *PCA* increases sharply on datasets with a large number of missing data.
In addition, *cov.pairwise.complete.obs* method is fairly fast and, overall, the most efficient. This is achieved by retaining incomplete observations and performing calculations only on the observed components. Unlike other imputation methods, *cov.pairwise.complete.obs* is less biased.

Figure 4.11 shows the results obtained in the linear model for $N = 100$. We obtained the same conclusions as in the Friedman model, except for *PCA* method,

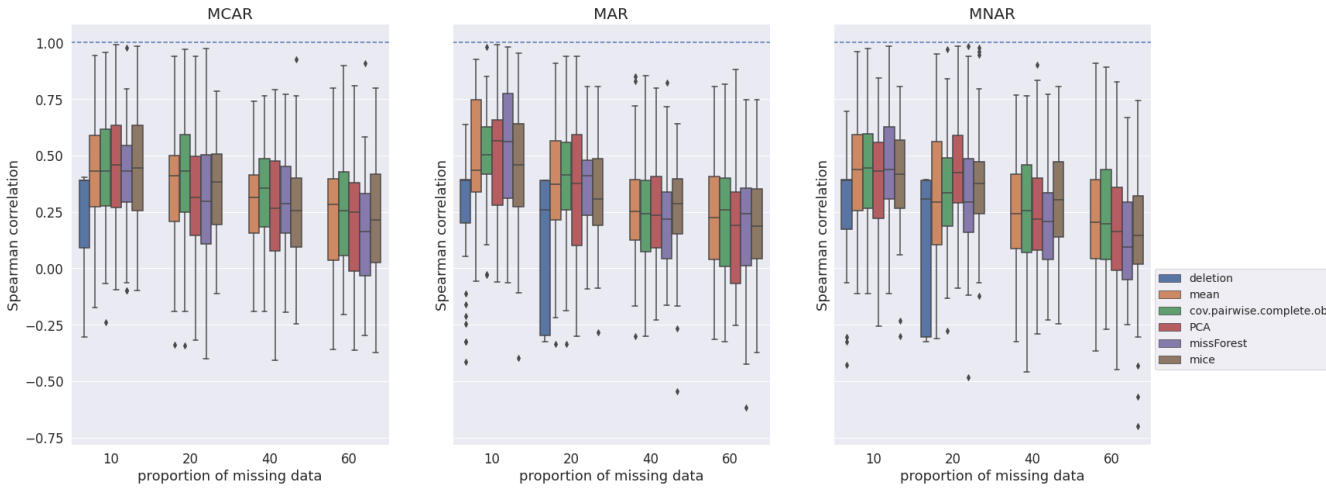


Figure 4.10: Spearman correlation between rankings of variables by order of influence obtained on complete and incomplete data in the Friedman model.

which seems to be the most inefficient and least robust, with sometimes unstable results.

Figure 4.12 summarizes the results obtained in the Friedman model by varying N from 100 to 1000 with 20% missing data. All methods improve with increasing dataset sample sizes. However, the results obtained with *deletion* method, which is always very unstable, improve very slightly. *Deletion* is still the worst-performing method of all. The other methods give fairly similar results. In particular, *missForest* and *mice* are very close, but *mice* performs slightly better, especially in the case of MNAR. The performance of the *mean* and *cov.pairwise.complete.obs* methods are very close, but *cov.pairwise.complete.obs* performs slightly better on data with many observations. However, *PCA* outperforms both methods in the MAR and MNAR cases.

4.4 Conclusion

In this work, we focused on difficult settings, in particular on data with complex relationships between variables and a large number of missing data, in order to mimic as closely as possible data from real applications.

Other simpler settings were also studied, where data were generated from linear regression models with no complex relationships between variables, and where only some variables were incomplete. In such simple settings, the methods used were quite similar. We sometimes observed better results for the *deletion* method, particularly on larger sample sizes.

In the scenarios considered in our work, the performance of the other studied methods is gloabally similar. However, some imputation methods require a large

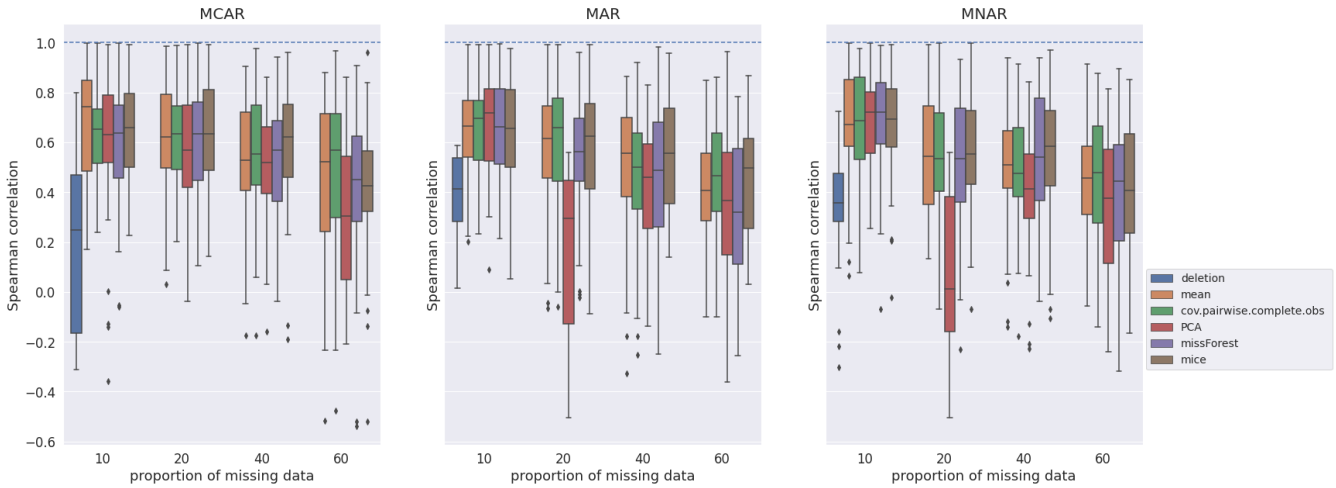


Figure 4.11: Spearman correlation between rankings of variables by order of influence obtained on complete and incomplete data in the linear regression model.

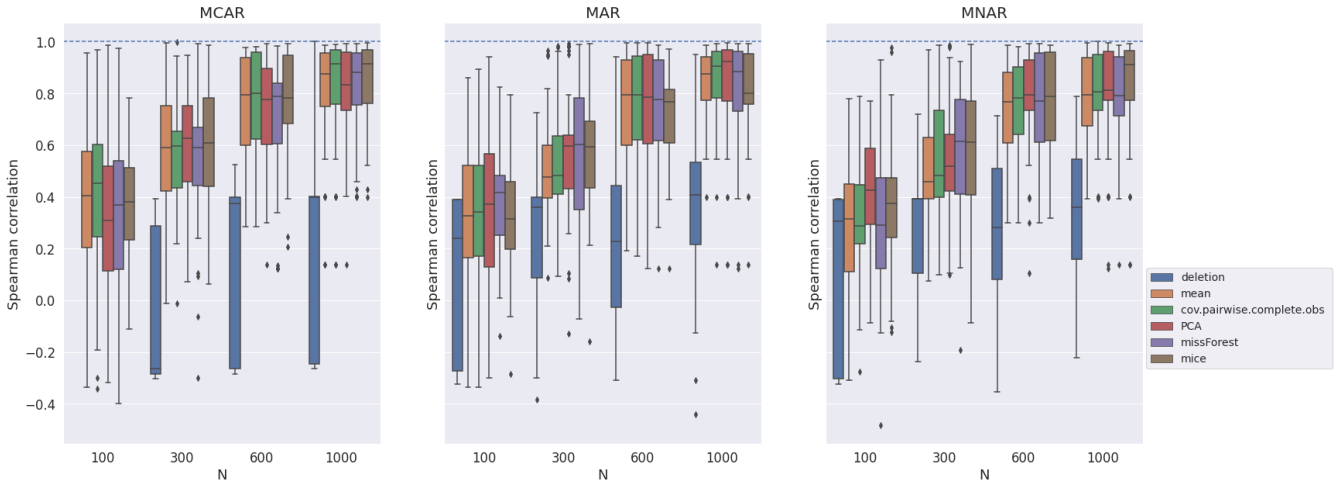


Figure 4.12: Spearman correlation between rankings of variables by order of influence obtained on complete and incomplete data in the Friedman model by varying N with 20% missing data.

computation time. If a fast method is prefered that gives results close to those obtained in the complete case, it's best to opt for the simple and fast *cov.pairwise.complete.obs* method which minimizes the risk of bias due to imputation of missing data.

Chapter 5

Conclusion générale et perspectives

Cette thèse a porté sur l'analyse statistique des données de production et de performances des moteurs d'avions. Plusieurs méthodes d'apprentissage statistiques ont été proposées afin de répondre au mieux à la problématique en s'adaptant au contexte des données utilisées dans cette thèse. Les méthodes proposées dépassent parfois le contexte et le cadre de la thèse car elles peuvent avoir un intérêt dans d'autres applications réelles.

L'exploration et la visualisation des données massives sont essentielles pour mieux comprendre la structure interne des données, capter l'information essentielle dans les données et ainsi de mieux les modéliser. Nous nous sommes principalement intéressés aux cartes auto-organisatrices pour leur interprétabilité et leur capacité à visualiser facilement des données de production multidimensionnelles et complexes. Nous avons proposé une extension de ces cartes aux données incomplètes qui permet d'imputer les données manquantes de façon à mieux représenter les données sur la carte. Une version accélérée de l'algorithme a été développé pour réduire significativement le temps de calcul tout en conservant les mêmes performances. Un package R `missSOM` a été développé et est disponible sur le CRAN. Ce travail a donné lieu à une publication d'un article dans une revue scientifique et d'une présentation orale aux JDS en 2022.
Bien que ce travail s'est concentré sur l'algorithme standard de Kohonen, nous pourrons, dans des travaux futurs, aborder le problème à d'autres variantes existantes des cartes auto-organisatrices sur des types de données plus complexes, tels que les données mixtes, afin de leur permettre de traiter les données manquantes.

La modélisation des performances moteurs à partir des données de production des pièces s'est avérée difficile en raison des fortes dépendances liées aux conditions d'essais et à la production des autres pièces moteurs en constante évolution. En effet, des biais de mesures de performances sont présents et provoqués par les différents équipements utilisés pendant les essais. Une méthode a été utilisée pour

tenter de corriger et de réduire ces biais mais les résultats montrent encore certaines imperfections à améliorer.
Différentes approches ont été étudiés pour tenter de modéliser les performances moteurs et étudier l'impact des paramètres géométriques des pièces. Des modèles de Machine Learning, de Deep Learning et de séries temporelles ont été comparés. La prédiction des performances moteurs en fonction de la géométrie des pièces avec les données qu'on dispose n'a pas été possible. De plus, la tendance globale des mesures de performances des moteurs a été assez bien représentée et modélisée par les modèles de séries temporelles. Ce travail révèle le résultat suivant : le moteur est robuste aux variations de production qu'il y a pu avoir dans le passé. Les résultats obtenus dans ce travail ont donné lieu à une publication d'un article et à une présentation orale au sein d'une conférence internationale en juin 2023.
Les modèles de séries temporelles utilisés dans ce travail, en particulier le modèle LSTM, doivent être prochainement réentrainés et réévalués sur des nouvelles données pour rechercher d'éventuelles améliorations en termes de prédiction.
Dans ce travail, nous avons constaté que la prédiction de la mesure de performance d'un moteur s'est avérée très complexe, en particulier en raison des nombreux bruits liés aux essais. Dans des travaux futurs, on pourra estimer la probabilité que la mesure de performance ne soit pas dans les normes. Par conséquent, un grand effort de modélisation sera nécessaire afin de développer un modèle probabiliste qui intègre toutes les particularités des données moteurs.
Enfin, alors que nous nous sommes principalement intéressés au débit d'air des moteurs, on pourra étudier et analyser les autres performances en considérant différents régimes de rotation du moteur.

L'identification des principales causes responsables d'une dérive de production est essentielle pour entamer rapidement des modifications et des réparations nécessaires pendant le processus de fabrication. L'outil d'analyse d'influence a été développé pour rechercher rapidement et identifier facilement les premières causes qui ont déclenchées une telle dérive sous la forme de résultats interprétables. Nous avons particulièrement considéré le cas des données corrélées. Dans la thèse, nous nous sommes intéressés à la présence de données manquantes dans l'outil et à leur impact sur les résultats d'analyse d'influence. Nous avons étudié et comparé différentes méthodes qui gèrent les données manquantes selon différents scénarios. La suppression des observations incomplètes donne des résultats très instables qui s'écartent de ceux obtenus dans le cas des données complètes. De plus, elle ne peut pas s'appliquer aux jeux de données avec peu d'observations ou avec des proportions de données manquantes élévées. Les autres méthodes considérées présentent des résultats similaires mais les simulations montrent généralement une meilleure performance de la méthode qui conserve les observations incomplètes mais en effectuant les calculs uniquement sur les composantes observées. Cette méthode qui ne requiert pas d'imputation de données manquantes permet de minimiser le risque de biais dûs à

l'imputation.
Ici nous avons effectué un très grand travail de simulations numériques pour comparer des méthodes de gestion de données manquantes dans l'outil d'analyse d'influence, cependant des futures recherches pourront être plus axées sur l'adaptation de la mesure d'influence des variables qui tiennent compte des valeurs manquantes.

Bibliography

[1] Adebayo J. Adeloye, Rabee Rustum, and Ibrahim D. Kariyama. Neural computing modeling of the reference crop evapotranspiration. *Environmental Modelling & Software*, 29(1):61–73, 2012. ISSN 1364-8152.

[2] Kellie J. Archer and Ryan V. Kimes. Empirical characterization of random forest variable importance measures. *Computational Statistics & Data Analysis*, 52(4):2249–2260, 2008. doi: https://doi.org/10.1016/j.csda.2007.08.015.

[3] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on neural networks*, 5(4):537–550, 1994.

[4] Clément Bénard, Gérard Biau, Sébastien Da Veiga, and Erwan Scornet. Interpretable Random Forests via Rule Extraction. April 2020.

[5] V. Besada, C. Quelle, J. M. Andrade, N. Gutiérrez, M. P. Gómez-Carracedo, and F. Schultze. A 10-year survey of trace metals in sediments using self-organizing maps. *Journal of Chemometrics*, 28(7), 2014.

[6] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. GTM: The Generative Topographic Mapping. *Neural Computation*, 10(1): 215–234, 01 1998. ISSN 0899-7667.

[7] Annelies G. Blom, Edith D. de Leeuw, and Joop J. Hox. Interviewer effects on nonresponse in the European Social Survey. ISER Working Paper Series 2010-25, Institute for Social and Economic Research, 2010.

[8] Feriel Boulfani, Xavier Gendre, Anne Ruiz-Gazen, and Martina Salvignol. Anomaly detection for aircraft electrical generator using machine learning in a functional data framework. In *2020 Global Congress on Electrical Engineering (GC-ElecEng)*, pages 27–32, 2020.

[9] G.E.P. Box, G.N. Jenkins, G.M. Jenkins, and G.C. Reinsel. *Time Series Analysis: Forecasting and Control*. Forecasting and Control Series. Prentice Hall, 1994. ISBN 9780130607744.

[10] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[11] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[12] Clément Bénard, Sébastien da Veiga, and Erwan Scornet. Mda for random forests: inconsistency, and a practical solution via the sobol-mda, 2022.

[13] Gregory Campbell, Gene Pennello, and Lilly Yue. Missing data in the regulation of medical devices. *Journal of Biopharmaceutical Statistics*, 21(2): 180–195, 2011.

[14] J. N. Cape, R. I. Smith, and D. Leaver. Missing data in spatiotemporal datasets: the UK rainfall chemistry network. *Geoscience Data Journal*, 2(1): 25–30, 2015.

[15] Marie Chavent, Jerome Lacaille, Alex Mourer, and Madalina Olteanu. Handling Correlations in Random Forests: which Impacts on Variable Importance and Model Interpretability? In *ESANN*, Bruges, Belgium, October 2021.

[16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[17] Jiehong Cheng, Jun Sun, Kunshan Yao, Min Xu, and Yan Cao. A variable selection method based on mutual information and variance inflation factor. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 268:120652, 2022. ISSN 1386-1425. doi: https://doi.org/10.1016/j.saa.2021.120652.

[18] E. Côme, M. Cottrell, M. Verleysen, and J. Lacaille. Aircraft engine health monitoring using Self-Organizing Maps. In *10th Industrial Conference ICDM*, pages 405–417, 2010.

[19] M. Cottrell and Letrémy P. Missing values : processing with the Kohonen algorithm. In *ASMDA*, pages 489–496, 2005.

[20] Marie Cottrell, Madalina Olteanu, Fabrice Rossi, and Nathalie N. Villa-Vialaneix. Self-Organizing Maps, theory and applications. *Revista de Investigacion Operacional*, 39(1):1–22, January 2018.

[21] Hágata Cremasco, Dionísio Borsato, Karina Gomes Angilelli, Olívio Fernandes Galão, Evandro Bona, and Marcos Eduardo Valle. Application of self-organising maps towards segmentation of soybean samples by determination of inorganic compounds content. *Journal of the Science of Food and Agriculture*, 96(1):306–310, 2016.

[22] Guido Deboeck and Teuvo Kohonen. *Visual explorations in finance: with self-organizing maps*. Springer Science & Business Media, 2013.

[23] Gauthier Doquire and Michel Verleysen. Feature selection with missing data using mutual information estimators. *Neurocomputing*, 90:3–11, 2012. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2012.02.031. Advances in artificial neural networks, machine learning, and computational intelligence (ESANN 2011).

[24] Jiacong Du, Jonathan Boss, Peisong Han, Lauren J. Beesley, Michael Kleinsasser, Stephen A. Goutman, Stuart Batterman, Eva L. Feldman, and Bhramar Mukherjee. Variable selection with multiply-imputed datasets: Choosing between stacked and grouped methods. *Journal of Computational and Graphical Statistics*, 31(4):1063–1075, 2022. doi: 10.1080/10618600.2022.2035739.

[25] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[26] Lisa Ehrlinger, Thomas Grubinger, Bence Varga, Mario Pichler, Thomas Natschläger, and Jürgen Zeindl. Treating missing data in industrial data analytics. In *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, pages 148–155, 2018.

[27] Tomas Eklund, Barbro Back, Hannu Vanharanta, and Ari Visa. Using the self-organizing map as a visualization tool in financial benchmarking. *Information Visualization*, 2(3):171–181, 2003.

[28] Laura Folguera, Jure Zupan, Daniel Cicerone, and Jorge F. Magallanes. Self-organizing maps for imputation of missing data in incomplete data matrices. *Chemometrics and Intelligent Laboratory Systems*, 143:146–151, 2015. ISSN 0169-7439.

[29] Jerome H. Friedman. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1 – 67, 1991. doi: 10.1214/aos/1176347963.

[30] Vipul Goyal, Mengyu Xu, Jayanta Kapat, and Ladislav Vesely. Prediction of Gas Turbine Performance Using Machine Learning Methods. volume Volume 6: Education; Electric Power of *Turbo Expo: Power for Land, Sea, and Air*, 09 2020.

[31] Baptiste Gregorutti, Bertrand Michel, and Philippe Saint-Pierre. Grouped variable importance with random forests and application to multiple functional data analysis. *Computational Statistics & Data Analysis*, 90:15–35, 2015. doi: https://doi.org/10.1016/j.csda.2015.04.002.

[32] Baptiste Gregorutti, Bertrand Michel, and Philippe Saint-Pierre. Correlation and variable importance in random forests. *Statistics and Computing*, 27, 05 2017. doi: 10.1007/s11222-016-9646-1.

[33] Alexander Hapfelmeier, Torsten Hothorn, Kurt Ulm, and Carolin Strobl. A new variable importance measure for random forests with missing data. *Statistics and Computing*, 24:21–34, 2014.

[34] Ryan High, Graham T. Eyres, Phil Bremer, and Biniam Kebede. Characterization of blue cheese volatiles using fingerprinting, self-organizing maps, and entropy-based feature selection. *Food Chemistry*, 347:128955, 2021. ISSN 0308-8146.

[35] Samuel M. Hipple, Zachary T. Reinhart, Harry Bonilla-Alvarado, Paolo Pezzini, and Kenneth Mark Bryden. Using machine learning to increase model performance for a gas turbine system. *ASME 2020 Power Conference*, 2020.

[36] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[37] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[38] James Honaker, Gary King, and Matthew Blackwell. Amelia II: A program for missing data. *Journal of Statistical Software*, 45(7):1–47, 2011.

[39] Giles Hooker, Lucas Mentch, and Siyu Zhou. Unrestricted permutation forces extrapolation: variable importance requires at least one more model, or there is no free variable importance. *Statistics and Computing*, 31, 11 2021. doi: 10.1007/s11222-021-10057-z.

[40] Handuo Hu, Jianyang Yu, Yanping Song, and Fu Chen. The application of support vector regression and mesh deformation technique in the optimization of transonic compressor design. *Aerospace Science and Technology*, 112:106589, 2021. ISSN 1270-9638.

[41] Sameer Kumar Jasra, Gianluca Valentino, Alan Muscat, and Robert Camilleri. Hybrid machine learning-statistical method for anomaly detection in flight data. *Applied Sciences*, 12(20), 2022. ISSN 2076-3417.

[42] J. Josse and F. Husson. Handling missing values in exploratory multivariate data analysis methods. *Journal de la SFdS*, 153(2):79–99, 2013.

[43] Julie Josse and François Husson. Handling missing values in exploratory multivariate data analysis methods. *Journal de la Societe Française de Statistique*, 153(2):79–99, 2012.

[44] Julie Josse, Nicolas Prost, Erwan Scornet, and Gaël Varoquaux. On the consistency of supervised learning with missing values. 2019.

[45] Joe H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[46] Niina Junno, Emilia Koivisto, Ilmo Kukkonen, Alireza Malehmir, and Markku Montonen. Predicting missing seismic velocity values using self-organizing maps to aid the interpretation of seismic reflection data from the Kevitsa Ni-Cu-PGE deposit in northern Finland. *Minerals*, 9(9):529, 2019.

[47] Aman Mohammad Kalteh and Peder Hjorth. Imputation of missing values in a precipitation–runoff process database. *Hydrology Research*, 40(4):420–432, 08 2009. ISSN 0029-1277.

[48] Bain Khusnul Khotimah, Miswanto, and Herry Suprajitno. A hybrid self organizing map imputation (SOMI) with naïve bayes for imputation missing data classification. *International Journal of GEOMATE*, 17(62):195–202, 2019. ISSN 2186-2982.

[49] Teuvo Kohonen. *Self-organizing maps*. Springer, Berlin, 1995.

[50] Teuvo Kohonen and Panu Somervuo. How to make large self-organizing maps for nonvectorial data. *Neural Networks*, 15(8):945–952, 2002. ISSN 0893-6080.

[51] Alexander Kowarik and Matthias Templ. Imputation with the R package VIM. *Journal of Statistical Software*, 74(7):1–16, 2016.

[52] Chanida Krongchai, Sujitra Funsueb, Jaroon Jakmunee, and Sila Kittiwachana. Application of multiple self-organizing maps for classification of soil samples in thailand according to their geographic origins. *Journal of Chemometrics*, 31(2):e2871, 2017.

[53] Jérôme Lacaille. An influence gauge to detect and explain relations between measurements and a performance indicator. 10 2015.

[54] K. Lakshminarayan, S. Harp, and T. Samad. Imputation of missing data in industrial databases. *Applied Intelligence*, 11:259–275, 2004.

[55] Guoxing Lan, Qing Li, and Nong Cheng. Remaining useful life estimation of turbofan engine using lstm neural networks. In *2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*, pages 1–5, 2018.

[56] Cosmin Lazar, Laurent Gatto, Myriam Ferro, Christophe Bruley, and Thomas Burger. Accounting for the multiple natures of missing values in label-free quantitative proteomics data sets to compare imputation strategies. *Journal of Proteome Research*, 15(4):1116–1125, 2016.

[57] Mustapha Lebbah, Aymeric Chazottes, Fouad Badran, and Sylvie Thiria. Mixed Topological Map. European Symposium on Artificial Neural Networks (ESANN), 2005.

[58] Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111, 2018. doi: 10.1080/01621459.2017.1307116.

[59] Sibo Li, Jingkun Qin, Miao He, and Roberto Paoli. Fast evaluation of aircraft icing severity using machine learning based on xgboost. *Aerospace*, 7(4), 2020. ISSN 2226-4310.

[60] S. Licen, S. Cozzutto, G. Barbieri, M. Crosera, G. Adami, and P. Barbieri. Characterization of variability of air particulate matter size profiles recorded by optical particle counters near a complex emissive source by use of self-organizing map algorithm. *Chemometrics and Intelligent Laboratory Systems*, 190:48–54, 2019. ISSN 0169-7439.

[61] Roderick J. Little, Ralph D'Agostino, Michael L. Cohen, Kay Dickersin, Scott S. Emerson, John T. Farrar, Constantine Frangakis, Joseph W. Hogan, Geert Molenberghs, Susan A. Murphy, James D. Neaton, Andrea Rotnitzky, Daniel Scharfstein, Weichung J. Shih, Jay P. Siegel, and Hal Stern. The prevention and treatment of missing data in clinical trials. *New England Journal of Medicine*, 367(14):1355–1360, 2012.

[62] Jinlong Liu, Christopher Ulishney, and Cosmin E. Dumitrescu. Application of Random Forest Machine Learning Models to Forecast Combustion Profile Parameters of a Natural Gas Spark Ignition Engine. volume Volume 6: Design, Systems, and Complexity of *ASME International Mechanical Engineering Congress and Exposition*, 11 2020.

[63] Enzo Losi, Mauro Venturini, Lucrezia Manservigi, Giuseppe Fabio Ceschini, Giovanni Bechini, Giuseppe Cota, and Fabrizio Riguzzi. Prediction of Gas Turbine Trip: A Novel Methodology Based on Random Forest Models. *Journal of Engineering for Gas Turbines and Power*, 144(3), 01 2022. ISSN 0742-4795. 031025.

[64] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[65] Huageng Luo, George Ghanime, and Liping Wang. ARMA Model for Turbine and Compressor Clearance Forecasting. volume Volume 3: Controls, Diagnostics and Instrumentation; Cycle Innovations; Marine of *Turbo Expo: Power for Land, Sea, and Air*, pages 107–114, 06 2010.

[66] Sébastien Massoni, Madalina Olteanu, and Patrick Rousset. Career-Path Analysis Using Optimal Matching and Self-Organizing Maps. In Risto Miikkulainen José C. Principe, editor, *Advances in Self-Organizing Maps*, Lecture Notes in Computer Science n°5629, pages 154–162. Springer, 2009.

[67] Patricia Melin, Julio Cesar Monica, Daniela Sanchez, and Oscar Castillo. Analysis of spatial spread relationships of coronavirus (covid-19) pandemic in the world using self organizing maps. *Chaos, Solitons & Fractals*, 138: 109917, 2020. ISSN 0960-0779.

[68] Xiao-Li Meng. Multiple-Imputation Inferences with Uncongenial Sources of Input. *Statistical Science*, 9(4):538 – 558, 1994. doi: 10.1214/ss/1177010269.

[69] Ardalan Mirzaei, Stephen R. Carter, Asad E. Patanwala, and Carl R. Schneider. Missing data in surveys: Key concepts, approaches, and applications. *Research in Social and Administrative Pharmacy*, 18:2308–2316, 2021.

[70] A Mourer and J Lacaille. Detection and correction of equipment biases during engine tests. *Insight - Non-Destructive Testing and Condition Monitoring*, 64 (8):459–464, 2022. ISSN 1354-2575.

[71] Kristin K. Nicodemus and James D. Malley. Predictor correlation impacts machine learning algorithms: implications for genomic studies. *Bioinformatics*, 25(15):1884–1890, 05 2009. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp331.

[72] Kristin K. Nicodemus, James D. Malley, Carolin Strobl, and Andreas Ziegler. The behaviour of random forest permutation-based variable importance measures under predictor correlation. *BMC Bioinformatics*, 11:110 – 110, 2010.

[73] Elias Nkiaka, Nida Nawaz, and Jon Lovett. Using Self-Organizing Maps to infill missing data in hydro-meteorological time series from the Logone catchment, Lake Chad basin. *Environmental Monitoring and Assessment*, 188(7): 400, 2016.

[74] Dominik Olszewski. Fraud detection using self-organizing map visualizing the user profiles. *Knowledge-Based Systems*, 70:324–334, 2014. ISSN 0950-7051.

[75] Art B. Owen. Sobol' indices and shapley value. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):245–251, 2014.

[76] Brian S. Penn. Using self-organizing maps to visualize high-dimensional data. *Computers & Geosciences*, 31(5):531–544, 2005. ISSN 0098-3004.

[77] Jimin Qian, Nam Phuong Nguyen, Yutaka Oya, Gota Kikugawa, Tomonaga Okabe, Yue Huang, and Fumio S. Ohuchi. Introducing self-organized maps (som) as a visualization tool for materials research and education. *Results in Materials*, 4:100020, 2019. ISSN 2590-048X.

[78] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[79] Helge Ritter, Thomas Martinetz, and Klaus Schulten. *Neural Computation and Self-Organizing Maps: An Introduction*. Addison-Wesley Longman Publishing Co., Inc., USA, 1992.

[80] Danielle M. Rodgers, Ross Jacobucci, and Kevin J. Grimm. A multiple imputation approach for handling missing data in classification and regression trees. *Journal of Behavioral Data Science*, 1(1):127–153, May 2021. doi: 10.35566/jbds/v1n1/p6.

[81] D.B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. Wiley Series in Probability and Statistics. Wiley, 1987. ISBN 9780471087052.

[82] D.B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. Wiley Classics Library. Wiley, 2004. ISBN 9780471655749.

[83] Donald B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976. ISSN 00063444.

[84] Rabee Rustum and Adebayo J. Adeloye. Replacing outliers and missing values from activated sludge data using Kohonen Self-Organizing Map. *Journal of Environmental Engineering*, 133(9):909–916, 2007.

[85] Erwan Scornet. Trees, forests, and impurity-based variable importance in regression. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 59(1):21 – 52, 2023. doi: 10.1214/21-AIHP1240.

[86] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[87] Lloyd S Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.

[88] W. Shih. Problems in dealing with missing data and informative censoring in clinical trials. *Current Controlled Trials in Cardiovascular Medicine*, 3:4, 2002.

[89] A. Smolinski and S. Hlawiczka. Chemometric treatment of missing elements in air quality data sets. *Polish Journal of Environmental Studies*, 16(4):613–622, 2007.

[90] Ilya M Sobol. Sensitivity analysis for non-linear mathematical models. *Math. Modeling Comput. Exp.*, 1:407–414, 1993.

[91] D.J. Stekhoven and P. Buhlmann. Missforest - non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2011.

[92] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8:25 – 25, 2007.

[93] Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. Conditional variable importance for random forests. *BMC bioinformatics*, 9:307, 08 2008. doi: 10.1186/1471-2105-9-307.

[94] J. V. Taylor, B. Conduit, A. Dickens, C. Hall, M. Hillel, and R. J. Miller. Predicting the Operability of Damaged Compressors Using Machine Learning. *Journal of Turbomachinery*, 142(5), 04 2020.

[95] Terry M Therneau, Elizabeth J Atkinson, et al. An introduction to recursive partitioning using the rpart routines. Technical report, Technical report Mayo Foundation, 1997.

[96] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[97] Laura Toloşi and Thomas Lengauer. Classification with correlated features: unreliability of feature ranking and solutions. *Bioinformatics*, 27(14):1986–1994, 05 2011. ISSN 1367-4803. doi: 10.1093/bioinformatics/btr300.

[98] B. E. T. H. Twala, M. C. Jones, and D. J. Hand. Good methods for coping with missing data in decision trees. *Pattern Recognition Letters*, 29(7):950–956, May 2008.

[99] Shafi Ullah, Shuguang Li, Khalid Khan, Shahbaz Khan, Ilyas Khan, and Sayed M. Eldin. An investigation of exhaust gas temperature of aircraft engine using lstm. *IEEE Access*, 11:5168–5177, 2023.

[100] Stef van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3):1–67, 2011. doi: 10.18637/jss.v045.i03.

[101] Stef van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software, Articles*, 45(3):1–67, 2011.

[102] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995. ISBN 0-387-94559-8.

[103] Jorge R Vergara and Pablo A Estévez. A review of feature selection methods based on mutual information. *Neural computing and applications*, 24:175–186, 2014.

[104] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(95):2837–2854, 2010.

[105] Tsvetomil Voyslavov, Stefan Tsakovski, and Vasil Simeonov. Surface water quality assessment using self-organizing maps and hasse diagram technique. *Chemometrics and Intelligent Laboratory Systems*, 118:280–286, 2012. ISSN 0169-7439.

[106] Zhitao Wang, Ningbo Zhao, Weiying Wang, Rui Tang, and Shuying Li. A fault diagnosis approach for gas turbine exhaust gas temperature based on fuzzy c-means clustering and support vector machine. *Mathematical Problems in Engineering*, 2015:1–11, 2015.

[107] H. Wold. Estimation of principal components and related models by iterative least squares. In P.R. Krishnaiah, editor, *In Multivariate Analysis*, volume 59, pages 391–420. Academic Press, NY, 1966.

[108] Angela M Wood, Ian R White, and Patrick Royston. How should variable selection be performed with multiply imputed data? *Statistics in medicine*, 27(17):3227—3246, July 2008. ISSN 0277-6715. doi: 10.1002/sim.3177.

[109] Yanming Yang, Wanchun Gao, and Chaoran Guo. Aero-engine lubricating oil metal content prediction using non-stationary time series arima model. In *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, volume 2, pages 51–54, 2017.

[110] Hang Yi, Guan Wang, Hui Geng, Hao Xu, Wei Wang, and Dengji Zhou. Telemetry data prediction of launch vehicle attitude control engine using lstm. volume Volume 13: Safety Engineering, Risk, and Reliability Analysis of *ASME International Mechanical Engineering Congress and Exposition*, 11 2019.

[111] Yulong Ying, Siyu Xu, Jingchao Li, and Bin Zhang. Compressor performance modelling method based on support vector machine nonlinear regression algorithm. *Royal Society Open Science*, 7(1):191596, 2020.

[112] Eric R. Ziegel. Book review: Nonresponse in household interview surveys by Robert M. Groves; Mick P. Couper. *Technometrics*, 41(4):381–381, 1999.

[113] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Publications by Sara Rejeb

[RDR22a] Sara Rejeb, Catherine Duveau, and Tabea Rebafka. Cartes auto-organisatrices pour l'exploration des données partiellement observées et l'imputation des données manquantes. In *53es Journées de statistique de la SFdS*, Lyon, France, Juin 2022.

[RDR22b] Sara Rejeb, Catherine Duveau, and Tabea Rebafka. Self-Organizing Maps for Exploration of Partially Observed Data and Imputation of Missing Values. *Chemometrics and Intelligent Laboratory Systems*, December 2022.

[RDR23] Sara Rejeb, Catherine Duveau, and Tabea Rebafka. Application of Machine Learning for Prediction of Turbofan's Airflow. In *Turbo Expo: Power for Land, Sea, and Air*, Boston, United States, June 2023.

Software by Sara Rejeb

[RDR22] Sara Rejeb, Catherine Duveau, and Tabea Rebafka. *missSOM: Self-Organizing Maps with Built-in Missing Data Imputation*, 2022. R package version 1.0.1.