Adaptation de méthodes d'apprentissage automatique dans l'industrie du semiconducteur : détection d'excursions, analyse de root cause et intégration de connaissances

Mathias Chastan

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : Mathématiques et Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Mathias CHASTAN

Thèse dirigée par **Jerome MALICK**
et codirigée par **Franck IUTZELER**,Maitre de Conférences, Université Grenoble Alpes

préparée au sein du **Laboratoire Jean Kuntzmann** dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Adaptation de méthodes d'apprentissage automatique dans l'industrie du semiconducteur : détection d'excursions, analyse de root cause et intégration de connaissances

Adaptation of machine learning methods in the semiconductor industry: excursion detection, root cause analysis and knowledge integration

Thèse soutenue publiquement le **29 mars 2022**,
devant le jury composé de :

Madame MARIANNE CLAUSEL
Professeur des Universités, UNIVERSITE DE LORRAINE, Rapporteure
Monsieur AMAURY HABRARD
Professeur des Universités, UNIVERSITE DE SAINT-ETIENNE -JEAN MONNET, Rapporteur
Monsieur PIERRE LEMAIRE
Maître de conférences HDR, GRENOBLE INP, Examinateur
Monsieur JERÔME LELONG
Professeur des Universités, GRENOBLE INP, Président

Adaptation de méthodes d'apprentissage automatique dans l'industrie du semiconducteur : détection d'excursions, analyse de root cause et intégration de connaissances

Mathias Chastan

Janvier 2022

Résumé

La complexification du processus de fabrication des puces électroniques rend nécessaire un contrôle de qualité toujours plus performant. Pour ce faire, des modèles d'apprentissage automatique (machine learning) sont désormais mis en oeuvre afin d'obtenir une réponse performante et peu coûteuse. L'intégration de ces nouvelles méthodes est compliquée et requiert des recherches pour les différents cas d'études qui possèdent des problématiques spécifiques. Trois projets d'intégration de méthodes de machine learning dans l'industrie du semiconducteur sont présentés dans ce manuscrit. Le premier porte sur la détection des signaux faibles. Le deuxième projet fait partie du domaine de l'analyse de la cause racine (root cause). Le troisième projet est un projet de recherche plus théorique sur l'intégration de connaissances dans un algorithme d'apprentissage automatique.

Les machines de lithographie sont équipés d'équipements de mesures appelés leveling sensors qui mesurent la topographie (leveling) de toutes les plaquettes (wafer) de silicium à chaque étape du processus. Pour chaque plaquette, environ 35 000 points sont mesurés et utilisés pour régler la machine, mais ils n'étaient auparavant pas sauvegardés. Une base de données a été créée pour stocker ces mesures qui peuvent être traitées pour détecter des signaux faibles en leveling sans aucun coût d'équipement de mesure (métrologie) supplémentaire. La solution proposée doit pouvoir traiter en temps réel un nombre conséquent de mesures (35 000 points par plaque / 25 plaques par lot / un lot par minute) bruitées qui proviennent de différents contextes (équipement / produit / couche). Un algorithme d'apprentissage non supervisé (DBSCAN) a été choisi pour répondre aux contraintes de vitesse d'exécution et de contextes multiples, car ce type d'algorithme ne nécessite pas d'apprentissage et il n'y a donc pas besoin de disposer d'un historique de données pour chaque contexte (ce qui est impossible dans le milieu du semiconducteur). Une solution de nettoyage de données a aussi été développée dans le cadre de cette thèse pour supprimer le bruit des données. Cette méthode complète a permis la création d'une application de détection automatique des signaux faibles avec système d'alerte par mail qui est utilisée actuellement chez STMicroelectronics Crolles.

L'analyse de la root cause est un sujet critique dans l'industrie du semiconducteur. En effet, détecter la cause d'une excursion est difficile dans cet environnement de production complexe ; une plaquette passe par environ 300 étapes process et 500 étapes de mesures, sur différents équipements et avec différents paramètres qui sont tous des causes potentielles d'un événement de non-qualité. La cause peut aussi être une combinaison de différents éléments du processus (un enchaînement d'opération par exemple) et la solution doit pouvoir détecter cette situation automatiquement. Il est aussi nécessaire de s'assurer de la pertinence des résultats de la méthode de détection de la root cause. Pour respecter ces contraintes une méthode utilisant l'algorithme random forest a été développée (Random Forest Discriminant Analysis). L'algorithme random forest est capable de traiter de grands volumes de données et les résultats sont facilement interprétable grâce aux métriques d'importance des features et de précision. Cette solution a été testée sur quelques cas avec de bons résultats et sera prochainement intégrée dans une application d'analyse de données.

L'intégration de connaissance est l'objet de recherches récentes et doit permettre de créer des modèles plus robustes en ajoutant des contraintes qui correspondent à une réalité métier. Ce type de modèle pourrait être utile chez STMicroelectronics et dans l'industrie du semiconducteur en général. Une méthode simple pour contraindre la monotonie des données avec random forest a été imaginée, développée et testée sur des jeux de données générés, jeux de données libres et jeux de données STMicroelectronics. Les résultats obtenus montrent que de plus amples recherches, notamment pour contraindre d'autres types de connaissances, pourraient permettent de créer des modèles qui seraient moins affectés par des petits échantillons de données bruitées courant dans l'industrie du semiconducteur.

Remerciements

Je remercie Auguste Lam pour m'avoir proposé ce sujet de thèse, m'avoir accompagné tout au long de mes travaux, avoir été disponible et pédagogue et m'avoir aidé dans l'écriture de mon premier article et de mon manuscrit.

Je remercie Franck Iutzeler pour m'avoir accompagné tout au long de mes travaux, avoir été disponible et pédagogue, m'avoir aidé pour les tâches administratives et m'avoir aidé pour l'écriture de mon deuxième article et de mon manuscrit.

Je remercie Jérôme Malick pour la confiance qu'il m'a accordé et pour m'avoir aidé dans la rédaction de mon manuscrit.

Je remercie Bertrand Le-Gratiet pour m'avoir encouragé à poursuivre vers une thèse après mon alternance, m'avoir conseillé auprès d'Auguste et de m'avoir aidé plusieurs fois au cours de mes 3 ans de thèse.

S/O à tous mes collègues et amis de STMicroelectronics et du laboratoire Jean Kuntzmann.

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Contexte de la thèse

L'industrie a connu trois révolutions, la production mécanique, la production de masse et la production automatique. L'industrie connaît maintenant sa quatrième révolution avec l'introduction de nouvelles technologies comme l'intelligence artificielle. Cette introduction de nouvelles technologies est imposée par des demandes de produits de plus en plus personnalisés, de plus en plus qualitatifs et des cycles d'innovation de plus en plus courts. Dans le secteur industriel et plus précisément dans l'industrie du semi-conducteur, ce changement est un processus complexe qui nécessite de la recherche.

STMicroelectronics est une multinationale fabricant de semi-conducteurs, secteur de pointe très concurrentiel, est concernée de près par cette révolution. L'innovation est au coeur de la stratégie de STMicroelectronics. L'entreprise essaie de répondre aux enjeux futurs de l'industrie en proposant un catalogue de produits de plus en plus variés, personnalisables et qualitatifs. Un levier pour atteindre ces objectifs est l'apprentissage automatique dont l'utilisation dans l'industrie permet de réduire les temps d'immobilisations non prévus, le temps de mise en place sur le marché et d'améliorer la qualité des produits. Les méthodes de contrôle de qualité doivent aussi être de plus en plus performantes. C'est l'objectif de cette thèse qui a pour but de rechercher des méthodes innovantes et performantes pour une partie spécifique du contrôle de qualité.

L'apprentissage automatique permet d'effectuer ce contrôle et d'aider à la prise de décision pour différentes problématiques, comme la détection d'excursions topographiques (leveling) sur les plaquettes (wafers) et l'analyse de la cause d'une excursion (root cause analysis). La détection d'excursions topographiques, le sujet du chapitre 2, est nécessaire car une topographie anormale peut rendre les puces non fonctionnelles. Le chapitre 3 traite des recherches sur le développement d'une méthode de détection de root cause qui permettra d'identifier la cause d'une excursion pour que l'utilisateur puisse trouver une solution à celle-ci. Les modèles utilisés en industrie doivent être robustes même avec de petits échantillons bruités, utiliser les connaissances métiers sur les données pour améliorer un algorithme d'apprentissage automatique est le sujet du chapitre 4.

Les travaux décrits dans ce manuscrit sont industrialisables à court terme et pour permettre ceci, des contraintes doivent être respectées dans le cadre de cette thèse. Les produits changent rapidement à STMicroelectronics, il est préférable de ne pas construire de modèle utilisable sur un seul contexte de données. Ainsi, il ne faut pas utiliser des méthodes de type "black-box" mais de favoriser des algorithmes simples et compréhensibles, d'où le choix de random forest pour les méthodes Random Forest Discriminant Analysis (chapitre 3) et Random Forest symétrique/monotone (chapitre 4). Les résultats doivent aussi être facilement interprétables. En cela des algorithmes ayant un comportement anthropomorphique comme DBSCAN (chapitre 2) sont intéressants.

La même méthodologie de recherche a été utilisée pour développer les différentes méthodes

présentées dans ce manuscrit. La première étape est la recherche bibliographique qui permet de trouver des idées qui sont ensuite assemblées pour construire une méthode originale. La méthode est ensuite testée sur un premier jeu de données qui peut être généré. Si les résultats de ce premier test ne sont pas bons il faut améliorer la méthode ou partir sur une autre piste, sinon il faut tester et ajuster la méthode avec un grand nombre de données(jeux de données libres : chapitre 4, jeux de données d'autres équipes : chapitres 4, 3 et développement d'outil d'extraction automatique des données : chapitre 2). Chaque chapitre est indépendant, ils traitent tous de l'adaptation et l'utilisation de méthodes d'apprentissage automatique dans l'industrie. Le premier chapitre introduit toutes les notions utiles à plusieurs des chapitres et contient aussi un état de l'art général. Chaque chapitre commence par une introduction des notions propres au chapitre et un état de l'art spécifique. Un plan simple du contenu du manuscrit est disponible dans la partie suivante 1.2.

1.2 Structure du document

La suite du document est structuré de la manière suivante :

— Chapitre 1 : Dans ce chapitre on trouve une présentation de l'industrie du semiconducteurs et de ses enjeux ainsi qu'une introduction au machine learning et à l'algorithme random forest.

— Chapitre 2 : Dans ce chapitre, est présentée la méthode qui a été développée pour détecter les excursions topographiques grâce aux données des équipements de lithographie. La méthode a été développée à partir de l'algorithme DBSCAN, un algorithme de classification non supervisé qui permet une classification par densité, proche du jugement humain.

— Chapitre 3 : Dans ce chapitre, est présentée la méthode qui a été développée pour retrouver la cause d'une excursion. RF-DA est basé sur random forest et utilise les métriques de précision et d'importances des variables pour trouver le facteur discriminant d'une excursion. La méthode peut être utilisée pour trouver n'importe quel type d'excursion et à différents niveaux (machine / opération / recette / chambre ...).

— Chapitre 4 : Dans ce chapitre, sont présentées les recherches pour construire des algorithmes contraints par des connaissances métiers sur les données. Les connaissances traitées sont la symétrie et la monotonie des données. Ces connaissances sont intégrées en modifiant le système de vote de random forest.

1.3 Description de l'industrie du semi-conducteur

Un semi-conducteur est un matériau qui a les caractéristiques électriques d'un isolant avec une conductivité électrique intermédiaire entre les métaux et les isolants. Le silicium est le matériau semi-conducteur le plus utilisé commercialement car il a de bonnes propriétés. Les semi-conducteurs sont indispensables pour la fabrication d'ordinateurs , de smartphones, de voitures intelligentes ou tout autre matériel électronique moderne. L'avantage du semi-conducteur est sa propriété hybride qui lui permet d'être à la fois isolant et conducteur.
L'industrie du semi-conducteurs regroupe les activités de conception, fabrication et commercialisation de composants électroniques et de circuits intégrés. Les composants électroniques exécutent des tâches uniques. Un circuit intégré est une combinaison de composants électroniques connectés entre eux sur une plaque de silicium. Le circuit intégré effectue des tâches plus ou moins complexes. Un circuit intégré est divisé en deux parties. La première partie est le die qui a une forme rectangulaire et est la partie élémentaire du circuit intégré. La deuxième partie est le boîtier qui possède des "pattes" pour établir la connexion électrique à l'extérieur du boîtier 1.1.

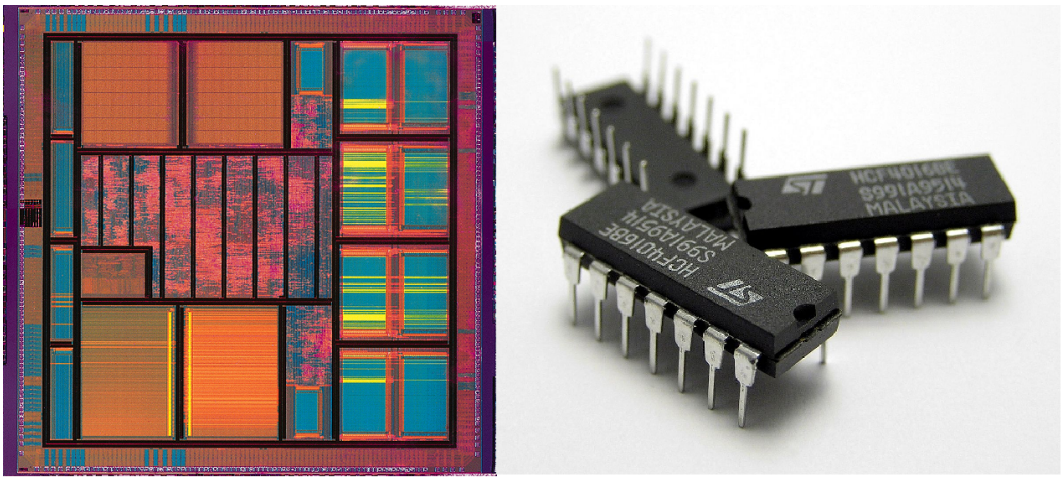


FIGURE 1.1 – Die à droite et boitiers à gauche

Les industries du semi-conducteur sont définies par trois activités : conceptualisation, fabrication et commercialisation. Il existe plusieurs profiles d'entreprise semi-conducteur. Les sociétés spécialisées dans la conception et la commercialisation appelées "fabless" (fab designe une usine de fabrication), les sous traitants qui fabriquent les produits des entreprises "fabless" appelés fonderie et les sociétés qui, comme STMicroelectronics, effectuent la conception, la fabrication et la commercialisation de leurs produits : les IDM. STMicroelectronics possède une quinzaine de site de fabrication, des ateliers de conception, des centres de R&D et des bureaux de ventes répartis à travers le monde. STMicroelectronics est quatorzième en terme de ventes au classement des industriels du semi-conducteur (figure 1.2).

1Q21 Top 15 Semiconductor Sales Leaders ($M, Including Foundries)

1Q21 Rank	1Q20 Rank	Company	Headquarters	1Q20 Total IC	1Q20 Total O-S-D	1Q20 Total Semi	1Q21 Total IC	1Q21 Total O-S-D	1Q21 Total Semi	1Q21/1Q20 % Change
1	1	Intel	U.S.	19,508	0	19,508	18,676	0	18,676	-4%
2	2	Samsung	South Korea	14,030	767	14,797	16,152	920	17,072	15%
3	3	TSMC (1)	Taiwan	10,319	0	10,319	12,911	0	12,911	25%
4	4	SK Hynix	South Korea	5,829	210	6,039	7,323	305	7,628	26%
5	5	Micron	U.S.	5,004	0	5,004	6,580	0	6,580	31%
6	7	Qualcomm (2)	U.S.	4,050	0	4,050	6,281	0	6,281	55%
7	6	Broadcom Inc. (2)	U.S.	3,673	409	4,082	4,355	485	4,840	19%
8	9	Nvidia (2)	U.S.	3,074	0	3,074	4,630	0	4,630	51%
9	8	TI	U.S.	2,974	190	3,164	3,793	235	4,028	27%
10	16	MediaTek (2)	Taiwan	2,022	0	2,022	3,849	0	3,849	90%
11	18	AMD (2)	U.S.	1,786	0	1,786	3,445	0	3,445	93%
12	11	Infineon	Europe	1,828	876	2,704	2,170	1,083	3,253	20%
13	10	Apple* (2)	U.S.	2,770	0	2,770	3,080	0	3,080	11%
14	14	ST	Europe	1,483	745	2,228	2,011	994	3,005	35%
15	13	Kioxia	Japan	2,567	0	2,567	2,585	0	2,585	1%
—	—	Top-15 Total		80,917	3,197	84,114	97,841	4,022	101,863	21%

(1) Foundry (2) Fabless

FIGURE 1.2 – Classement Q1 2021 du top 15 des vendeurs de semi-conducteur (source : company reports, IC insigths' Stategic Reviews database)

Le processus de fabrication des circuits intégrés est divisé en deux parties, le front end puis le back end. Dans la partie front end une plaque de silicium contenant les puces est créée puis cette plaque est découpée et les puces sont montées en boîtier. Il existe des usines front end (comme Crolles) et des usines back end.

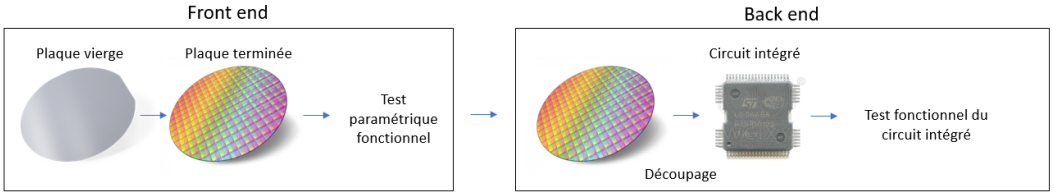


FIGURE 1.3 – Schéma processus de fabrication des circuits intégrés

Cette thèse se rapporte aux étapes front end qui sont la spécialité du site de Crolles. En effet, La dernière étape à ST Crolles est celle des tests paramétriques, ensuite les wafers sont envoyés dans une usine de découpe et d'assemblage.

Le processus front end regroupe de nombreuses étapes permettant de passer d'une plaque vierge à une plaque avec puces fonctionnelles. Les étapes principales de ce processus sont la lithographie et la gravure. La figure 1.4 est un schéma de toutes les étapes du processus.

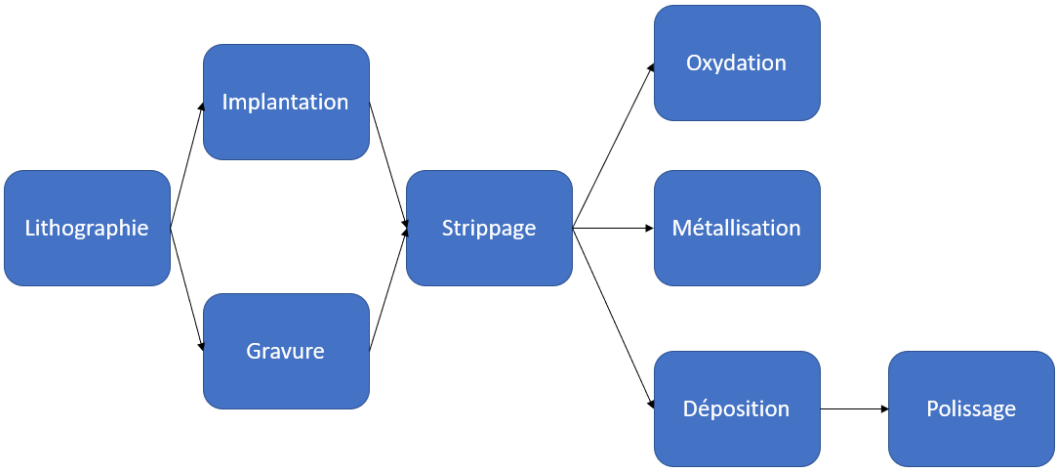


FIGURE 1.4 – Processus front end

Les étapes sont répétées de nombreuses fois pour transférer les motifs des circuits imprimés sur la plaque de silicium. C'est un processus très complexe comme le montre la figure 1.5.

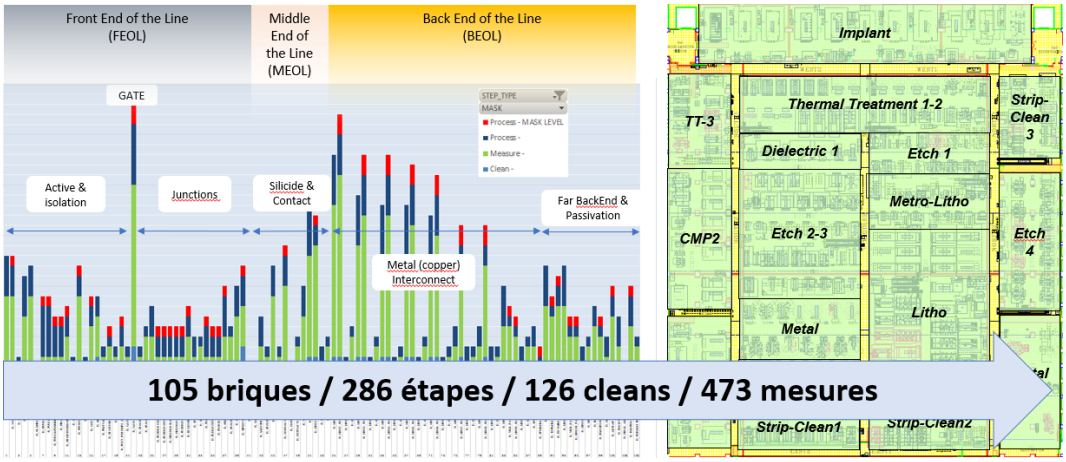


FIGURE 1.5 – Complexité du processus front end

De nombreuses étapes de fabrication s'enchaînent dans la salle blanche (pièce dont la concentration particulaire est très faible) (plan à droite sur la figure ci dessus) ainsi que des étapes de mesures qui permettent de surveiller les wafers en temps réel (section 1.5). Les plaquettes (wafers) sont fabriquées par lots de 25 wafer qui transitent dans la salle entre les différents ateliers (plan salle blanche). Après plusieurs centaines d'opérations les plaquettes contiennent des milliers de puces prêtes à être découpées.

Les différents ateliers sont :

— La **Lithographie** (LITHO) consiste à déposer un film photo-sensible pour être illuminé par une source lumineuse ultra-violet au travers d'un réticule qui contient les motifs du circuit intégré. Les étapes qui suivent sont celles de gravure et d'implantation ionique.

— La **Gravure** (ETCH) sert à transférer les motifs imprimés sur la couche de matériau cible. L'atelier gravure effectue deux étapes. Tout d'abord la gravure sèche par plasma puis l'élimination de résine (dit stripping). Le stripping sert à enlever la résine restante sur les zones non isolées après la gravure ou l'implantation.

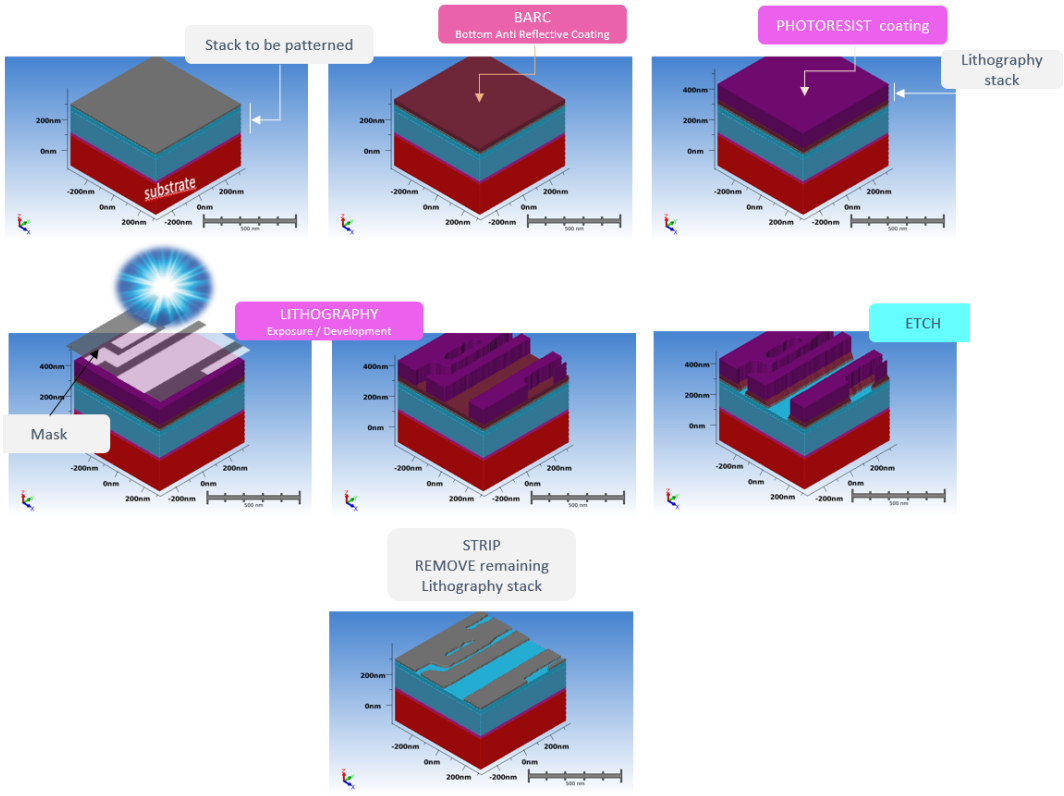


FIGURE 1.6 – Lithographie et gravure

— L'atelier **Implantation** (IMPLANT) effectue un dopage du silicium. Il existe deux types de dopage, le dopage de type N qui consiste à augmenter la densité d'électrons libres et le dopage de type P qui consiste à réduire la densité d'électrons et donc à avoir plus de trous. L'étape d'implant est toujours précédée par une étape de lithographie qui définit les zones à doper.

— L'atelier de **traitement thermique** (TT) réalise deux activités. La fabrication de couche d'isolants ou de semi-conducteurs et les recuits. La fabrication de couches est réalisée en faisant croître à très haute température une couche d'oxyde de silicium. Le recuit consiste à faire croître progressivement la température suivie d'un refroidissement pour activer les dopants et corriger les défauts introduits dans le silicium.

— L'atelier **WET** effectue le nettoyage de la surface de la plaque pour éviter la contamination de celle ci ou des machines des étapes suivantes. Cette atelier effectue aussi la gravure humide. La gravure humide grave de la même façon dans toutes les directions. La gravure humide est utilisée pour l'élimination d'une couche entière.

— L'atelier **DIEL** dépose des couches diélectriques (isolants électriques) sur la surface de la plaque. Ces isolants permettent, entre autre, de protéger le circuit intégré de l'environnement extérieur.

— L'atelier de **polissage** (CMP) a pour objectif de polir la surface de la plaque pour réduire l'épaisseur et la planariser. Pour cela des méthodes chimiques et mécaniques sont utilisées.

— L'atelier **Métal** a pour objectif le dépôt de couches conductrices comme les contacts, les vias et les pads. Les contacts permettent l'accès aux composants, les vias font le lien entre les composants et les pads servent à connecter le circuit intégré au boîtier.

1.4 Enjeux de l'apprentissage automatique dans l'industrie du semi-conducteur

Le marché du semi-conducteur était évalué à près de 419 milliards de dollars en 2019 et 464 milliards de dollars en 2020. Ceci représente une hausse d'environ 10% malgré la crise du covid-19 qui a impactée l'économie mondiale. Une nouvelle hausse de 17% est prévue pour 2021 (source : https ://www.idc.com/). Les confinements dans le monde et l'essor du télétravail ont fait augmenter la demande de semi-conducteurs pour les systèmes informatiques, comme les PC et les serveurs, à hauteur de 17%. Le marché des semi-conducteurs pour les téléphones mobiles et les voitures a aussi augmenté en 2020. Le marché du semi-conducteur semble destiné à une croissance constante malgré les récentes pénuries. L'importance critique des circuits intégrés dans de nombreux domaines est la raison de cette croissance. Dans ce marché porteur et hautement concurrentiel, une bonne compréhension des enjeux est cruciale pour une entreprise telle que STMicrolectronics.

Les enjeux du secteur du semi-conducteur sont multiples et seulement une partie sera présentée ici. Un domaine en plein croissance est le machine learning, le thème de ces recherches. Pour traiter des quantités toujours grandissantes de données les systèmes d'intelligence artificielle ont besoin de puces plus performantes. La technologie n'arrive pas à suivre la demande exponentielle de puissance de calcul. Investir dans ces technologies qui servent un domaine en pleine croissance est un bon choix stratégique. Les sociétés les mieux placées dans le secteur de l'IA sont listées sur la figure 1.7.

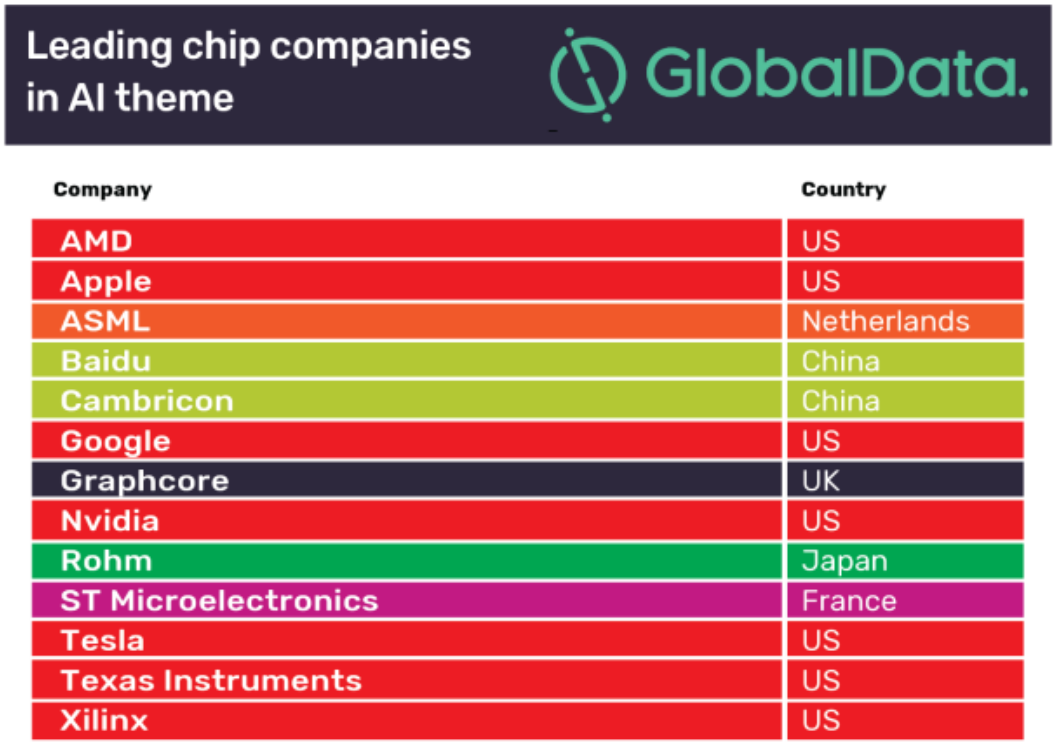


FIGURE 1.7 – Listes des entreprises de semi-conducteurs leaders dans le domaine de l'IA (source : actuia.com / GlobalData)

On retrouve AMD, entreprise qui fabrique des puces pour des cartes graphiques, en première position en terme de vente de puces destinées à l'IA. Des cartes graphiques puissantes sont nécessaires pour les calculs en machine learning. Apple est en deuxième position, l'entreprise connue pour ses smartphone a prévu la croissance de l'IA et a investi dans ce secteur. ST-

Microlectronics a aussi sa place parmi les leaders du domaine de l'intelligence artificielle. Le défi relevé par ces entreprises est de concevoir et fabriquer des produits toujours plus performants pour améliorer la puissance de calcul des systèmes d'intelligence artificielle. Cette amélioration des performances est cruciale pour le futur de l'IA car le volume de données à traiter est toujours plus grand. Pour tenir ses promesses l'intelligence artificielle a besoin de cette puissance de calcul comme elle a besoin de plus de données.

Le domaine de l'IA compte sur les entreprises du semi-conducteur autant que ces dernières comptent sur l'intégration des méthodes de machine learning à l'industrie. C'est un enjeu majeur pour l'industrie du semi-conducteur. D'après une étude d'Accenture, 77% des dirigeants de l'industrie du semiconducteurs ont déclaré avoir adopté l'IA au sein de leur entreprise et 63% s'attendent à ce que l'IA ait le plus grand impact sur leur activité au cours des 3 prochaines années (contre 41% dans les 20 secteurs d'activités concernés par l'étude). Les entreprises du semiconducteur évoluent dans un environnement hautement compétitif où le "winner-takes-all". C'est pour cela que la réduction du temps de fabrication et la poursuite de l'innovation sont critiques. Les coûts de la R&D et de la fabrication ont grandement augmentés comme le montre la figure 1.8

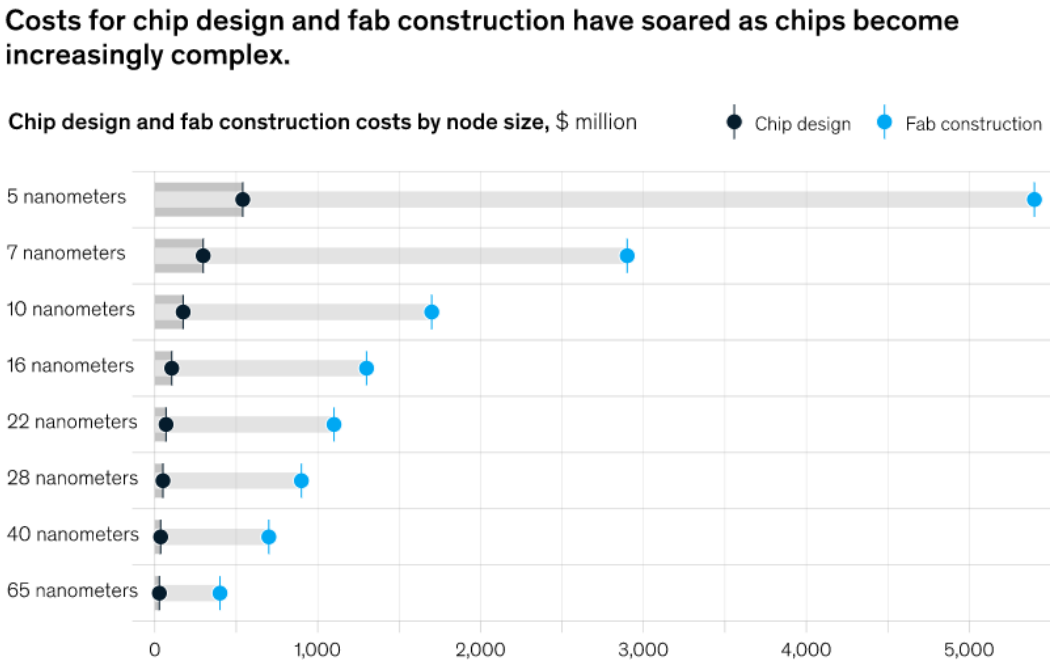


FIGURE 1.8 – Coût du design et de la fabrication de puce électronique en fonction de la taille des noeuds (source : McKinsey & Company, IBS : McKinsey analysis)

L'intelligence artificielle est un outil qui permettra d'aider à la production rapide de puces toujours plus innovantes et chères. Les recherches de McKinsey&Company (Scaling AI in the sector that enables it : Lessons for semiconductor-device makers) montrent que le machine learning contribue à environ 5 milliards de dollars de bénéfices pour les entreprises du semiconducteur. Ce chiffre représente uniquement 10% du potentiel de l'IA dans l'industrie. Dans 5 ans les bénéfices de l'IA pourraient s'élever à 90 milliards de dollars par an. L'avantage compétitif que cela représente est impossible à ignorer. Pour comprendre l'impact du machine learning sur l'industrie du semiconducteur il faut s'intéresser aux différentes use cases.

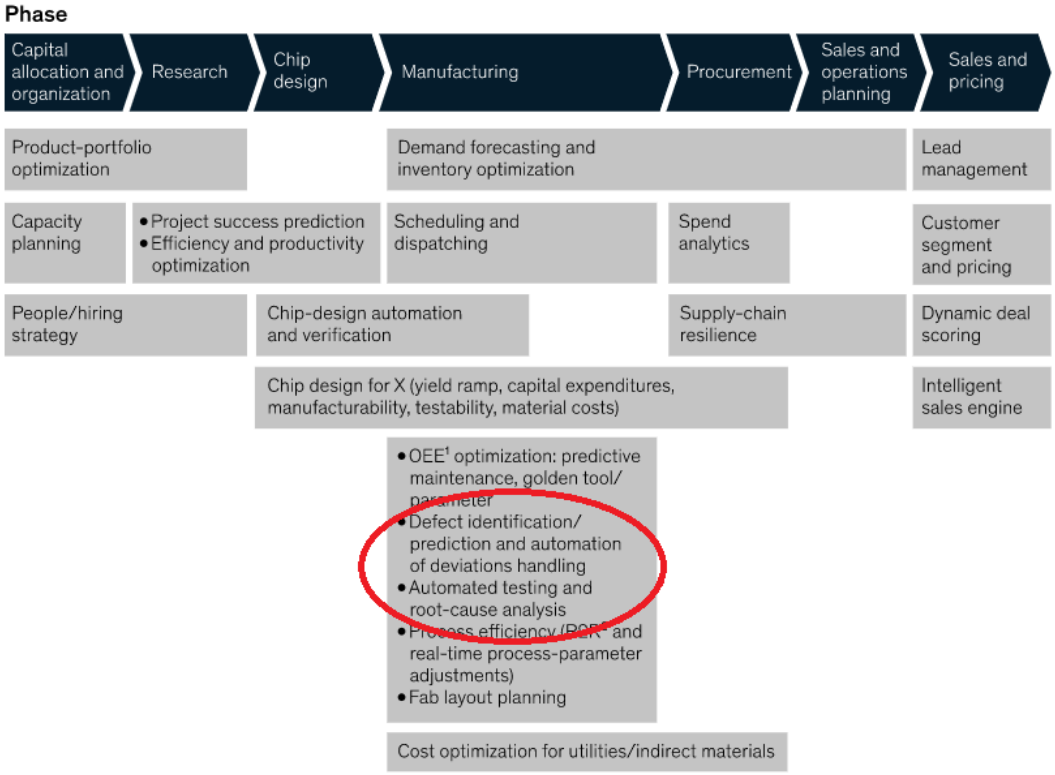


FIGURE 1.9 – Graphique des uses case ML dans la chaîne de valeur du semiconducteur (source : McKinsey & Company, IBS : McKinsey analysis)

Les sujets traités cette thèse sont entourés en rouge sur la figure 1.9 sauf les travaux décrits dans le chapitre 4 qui sont plus généraux et peuvent être utilisés pour différentes uses cases (les méthodes ont été testées sur des cas de scheduling). Comme on peut le voir les recherches présentées ici ne couvrent qu'une infime partie des uses cases possibles. Le machine learning peut être utilisé à toutes les phases, de la recherche à la vente et même pour l'organisation. Les uses cases IA de la phase R&D et manufacturing sont spécifiques à l'industrie du semi-conducteur, il est donc important d'effectuer des recherches sur ces sujets. Cette intégration du machine learning peut être difficile pour des entreprises qui ne sont pas spécialisées dans ce domaine. Pour ce faire il est nécessaire que les entreprises créent des services dédiés IA et recrutent des experts. STMicroelectronics a choisi une approche hybride qui repose sur un département spécialisé en IA appelé Manufacturing Data Analytics et sur des initiatives individuelles en IA ou en ML. C'est une approche qui a le mérite de faire rencontrer les innovations en IA avec les besoins du terrain. Les bénéfices liés à l'intelligence artificielle pour les entreprises du semiconducteur représenteront 10% du marché total.

ST Crolles 300 est un site de fabrication avec des problématiques et enjeux spécifiques. La problématique principale est la demande de produits plus performants, plus petits et moins chers comme illustré par la loi de Moore (figure 1.10). Gordon Moore a prédit en 1959 que la complexité des semiconducteurs doublerait chaque année. Ce postulat a été vérifié jusqu'à maintenant.

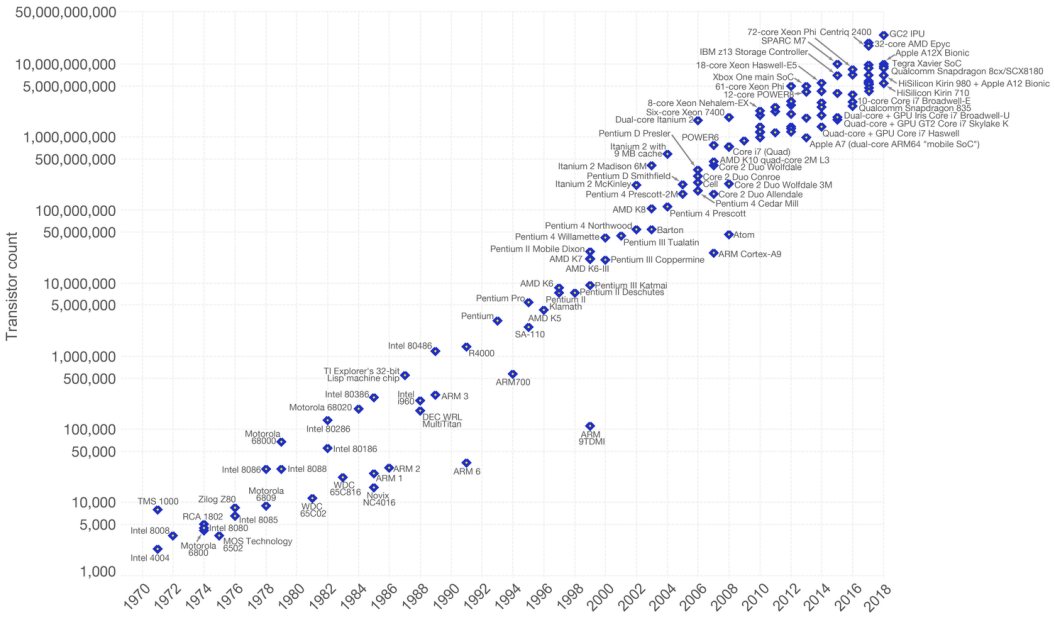


FIGURE 1.10 – Loi de Moore : nombre de transistors dans une puce électronique (1971-2018) (source : wikipedia / OurWorldInData.org)

L'évolution de la complexité est traduite par les noeuds technologiques. Le noeud technologique correspond à la largeur de la grille du transistor le plus fin présent sur la puce. STMicroelectronics a fait le choix d'arrêter la course à la miniaturisation à tout prix et se concentrer sur les fonctionnalités et les performances de la puce. Le développement de puces toujours plus innovantes impose un processus de fabrication toujours plus complexe :

- Temps de cycle important (environ 3 mois)
- Un nombre d'opérations importants (environ 300)
- Des équipements de production et de contrôle de plus en plus onéreux
- Une très grande variété de produits dans la même ligne de fabrication avec de nombreuses techniques de fabrication : cette problématique est appelée high mix low volume.
- Des produits de courtes durées (environ 2 ans). Par exemple, si ST fabrique les puces pour un nouveau smartphone : un très grand nombres de smartphones sera vendu dans la première année. Cela oblige le site de Crolles 300 à produire énormément de puces avant la sortie du smartphone. En 1 ou 2 ans le smartphone ne sera plus vendu et Crolles arrêtera la production de la puce.

Ces différentes contraintes obligent ST Crolles 300 à investir dans de nouvelles machines, dans des travaux de recherches et d'effectuer un contrôle de qualité tout au long de la production puis en sortie de ligne pour assurer un retour sur investissement. Les méthodes de contrôle de qualité doivent aussi être de plus en plus performantes. C'est l'objectif de cette thèse qui a pour but de rechercher des méthodes innovantes et performantes pour améliorer encore davantage la qualité.

1.5 Métrologie, process control, défectivité, PT et EWS

Le contrôle de la qualité à STMicroelectronics est assuré tout au long du processus de fabrication. Il y a plus d'étapes de contrôle que d'étapes de production pour les technologies les plus récentes. Les étapes de contrôle au cours du process sont effectuées à l'aide des mesures de métrologie et par le service process control. Le service process control a pour objectif de contrôler la variabilité des étapes de production pour détecter les dérives et assurer que le processus soit maîtrisé. La métrologie ne mesure pas toutes les plaques, un sampling est défini en fonction de la maturité de la technologie. C'est à dire que seulement quelques plaques par lots sont mesurées. Il y a un sampling car le coût de la métrologie est important.

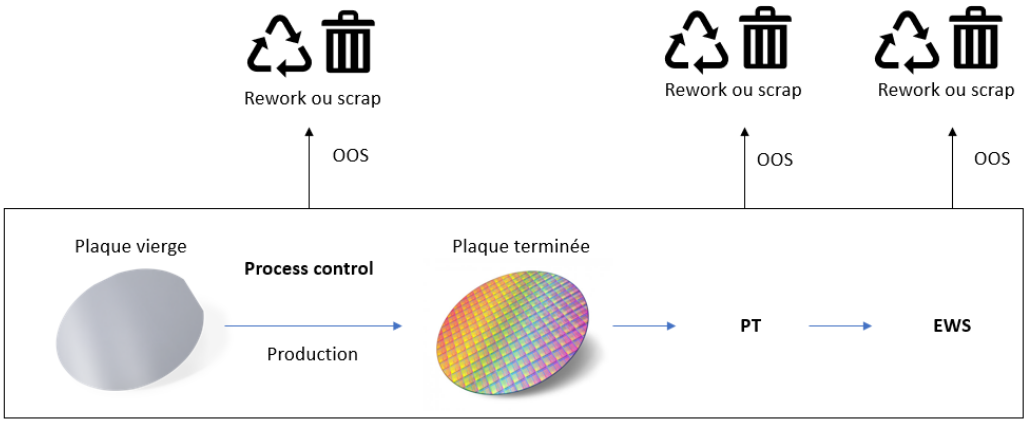


FIGURE 1.11 – Contrôle de qualité ST Crolles 300 (300 millimètre : diamètre du wafer

La figure 1.11 décrit l'enchaînement des étapes de contrôle de la qualité. Entre les étapes de production le **process control** vérifie que les plaques soient conformes en analysant les données de **métrologie**. Les plaques non conformes (Out Of Specifications : OOS) peuvent être "reworkées", on défait et refait les étapes de productions où le problème est apparu. Dans certains cas le rework est impossible. Dans ce cas on "scrap", c'est à dire qu'on jette la plaque. Une plaque scrap au milieu du processus de production plutôt qu'à la fin est un gain important compte tenu du cycle de production très long (3 mois environ), c'est pour cela qu'il est important de tester les plaques en cours de production et pas seulement à la fin. Une fois que la plaque est terminée la **défectivité** détecte les défauts présents sur la plaque, les **test paramétriques (PT)**sont effectués. Enfin les derniers tests effectués sont les **tests électriques (EWS)**, avec pour objectif de mesurer le rendement(pourcentage de puces fonctionnelles).

La métrologie regroupe l'ensemble des techniques, très diverses et utilisant de nombreux paramètres, de mesures des plaques. Pour effectuer ces mesures des équipements coûteux sont utilisés. Les travaux de recherches présentés dans ce manuscrit sont proches du process control. L'activité du process control est complexe. Quelques méthodes d'advanced process control (APC) sont présentées dans cette partie. L'APC regroupe les techniques statistiques et analytiques utilisées pour analyser les données de la métrologie, des outils de mesures et des outils de productions dans le but d'améliorer la qualité. La figure 1.12 regroupe les différents composants de l'APC.

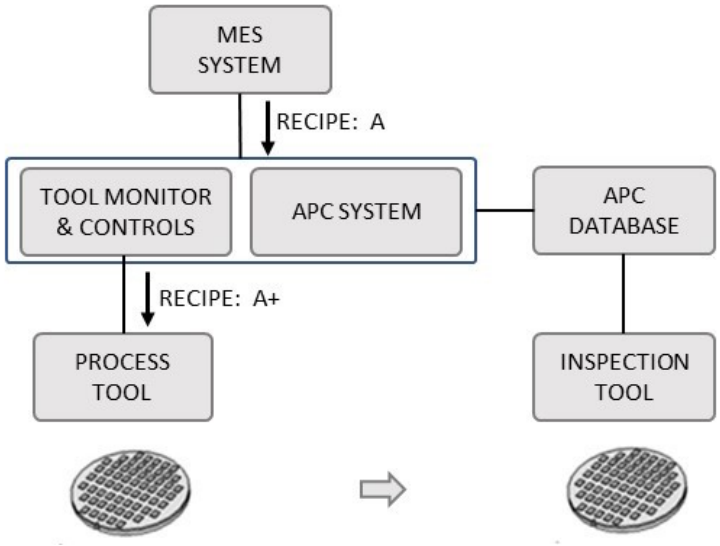


FIGURE 1.12 – Advanced process control

L'équipement d'inspection mesure des points prédéfinis sur la plaque puis les sauvegarde dans une base de données. Le système APC analyse les mesures et décide si la recette A doit être modifiée vers la recette A+ pour compenser des erreurs détectées par l'analyse. La recette correspond aux paramètres de production pour une étape (température, angle, focus...). Toutes les données de mesures sont conservées dans une base de données pour des analyses. Les méthodes utilisées par le process control sont principalement et historiquement statistiques. Les méthodes utilisées pour ces travaux de thèse sont des méthodes machine learning.

Il existe quelques malentendus sur la différence entre les modèles statistiques et les modèles d'apprentissage automatique (ils sont identiques, le machine learning est une évolution des statistiques, le machine learning est meilleur que les statistiques ...). Un exemple qui permet d'illustrer la différence entre les modèles statistiques et les modèles d'apprentissage automatique est la régression linéaire. Un modèle de régression linéaire similaire peut être obtenu statistiquement ou par apprentissage automatique. C'est dans la construction que réside la principale différence entre les deux types modèle. Pour construire le modèle statistique on trouve la ligne qui minimise l'erreur au carré avec **toutes les données** en **assumant que la distribution des données soit de nature gaussienne**. Pour construire le modèle d'apprentissage automatique on **divise le jeu de données** pour avoir un jeu d'entraînement et un jeu de test on apprend sur le jeu d'entraînement, **sans hypothèse sur la distribution sous jacente**, puis on teste les prédictions du modèle sur le jeu de test. La différence est donc une différence d'objectif, le modèle d'apprentissage est construit pour la prédiction alors que le modèle statistique est construit pour expliquer les relations entres les variables et la signification de ces relations. Il est aussi possible de faire des prédictions avec un modèle statistique ou de décrire les variables et leurs relations avec un modèle de machine learning (d'où la confusion) même si ce n'est pas leur objectif principal.

Outre leurs différences, comment faire coopérer les statistiques et le machine learning ? Les statistiques peuvent être utilisées indépendamment pour décrire les variables et leurs relations. Quand au machine learning il n'est pas supposé être utilisé sans une étude statistique préalable qui permettra de comprendre le problème (en décrivant les variables et leurs relations) pour choisir le bon type de modèle, les bonnes variables et les bons paramètres. Les meilleurs modèles de machine learning sont construit grâce à une très bonne compréhension des variables et de leurs relations obtenue suite à une étude statistique approfondie, pour vérifier ceci il est par exemple possible de regarder les solutions aux problèmes machine

learning sur Kaggle, les corrections les mieux notées avec les modèles les plus performants comporte toujours un analyse statistique complète (le code pour l'analyse statistique est souvent plus conséquent que la partie machine learning).

Comment est-ce que ces différences et cette relation entre modèle statistique et machine learning se traduisent elles à STMicrolectronics ? En industrie l'apprentissage automatique se nourrit du savoir accumulé par les modèles statistiques, l'innovation dans le domaine de l'apprentissage automatique est donc dépendant de la quantité et la qualité des méthodes/modèles statistiques disponibles (et donc de l'innovation dans ce domaine).

1.6 Introduction au machine learning

1.6.1 Introduction générale

L'apprentissage automatique (machine learning) est un des domaines de l'intelligence artificielle. L'apprentissage automatique est utilisé dans de nombreuses applications comme la conduite automatique en automobile, la publicité ciblée en marketing, la détection de tumeurs en médecine ect.

L'apprentissage automatique se fait en deux phases. Une première phase d'apprentissage où un algorithme va utiliser les données pour construire un modèle, puis une phase de production ou l'algorithme va recevoir des nouvelles données qu'il va traiter avec le modèle appris. Pour une tâche de reconnaissance d'images de chiens et de chats par exemple, la phase d'apprentissage consistera à construire un modèle de décision avec des images de chiens et de chats. L'algorithme déterminera comment exploiter les caractéristiques pour différencier un chat d'un chien. Lors de la phase de production, de nouvelles photos seront présentées à l'algorithme et il déterminera si ce sont des photos de chiens ou de chats avec le modèle de décision préalablement construit.

Il existe deux grandes familles de méthodes d'apprentissage, l'apprentissage supervisé et l'apprentissage non supervisé. Dans l'exemple d'apprentissage automatique pour une tâche de reconnaissance d'image, on parle d'apprentissage supervisé. Dans le cas d'un apprentissage non supervisé, les données ne sont pas étiquetées. Un apprentissage supervisé peut être une classification si les étiquettes sont discrètes ou une régression si les étiquettes sont continus. Il existe de nombreuses catégories et de nombreux algorithmes de machine learning dont voici une liste non exhaustive (figure 1.13)

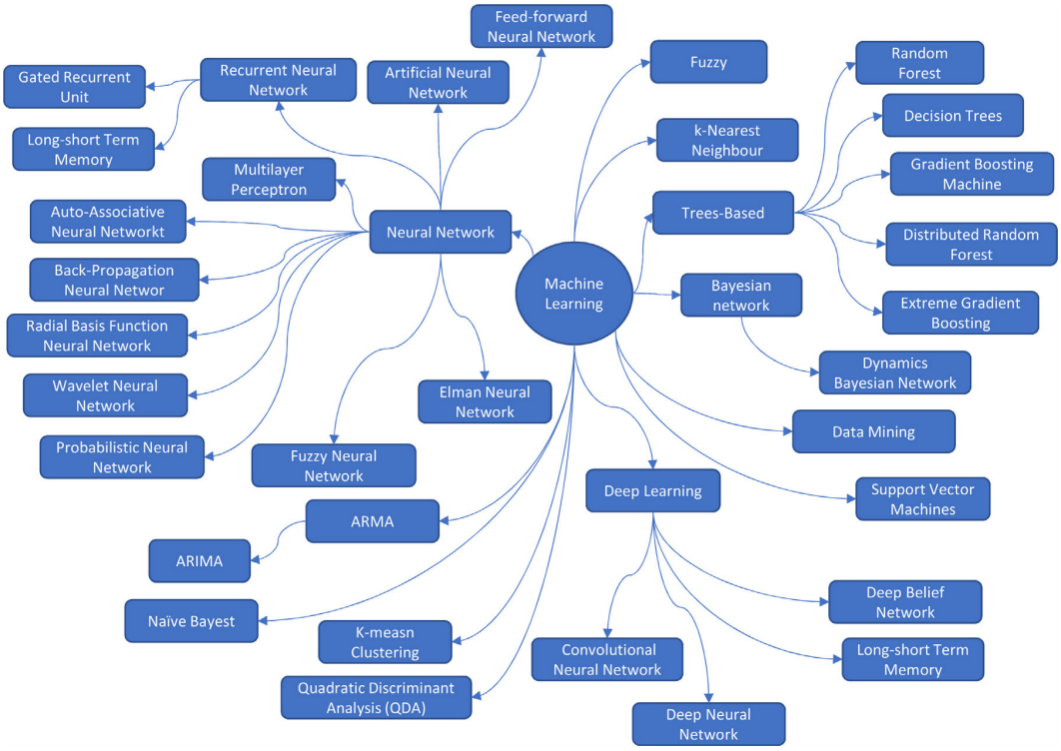


FIGURE 1.13 – Une Taxonomie machine learning d'un point de vue industriel (source : J. Dalzochio, E. Pignaton et al. / Computers in Industry 123 (2020) 103298)

1.6.2 Méthode simple : régression linéaire

Le modèle le plus simple en apprentissage automatique est la régression linéaire. L'objectif est de définir la relation entre une variable expliquée y et une ou plusieurs variables explicatives X, on cherche une droite qui est une fonction affine des variables X et permet de déterminer y. Pour trouver cette droite il existe différentes méthodes d'estimation. La plus connue est la méthode des moindres carrés que nous rappelons ci dessous,

voici la présentation formelle d'un modèle linéaire :

$$y_i = \beta_0 + \beta_1 x_{i,1} + ... + \beta_k x_{i,k} + \epsilon_i$$

y_i est la variables à expliquer, les x_i sont les différentes variables explicatives, ϵ_i représente l'erreur et les β_k sont les coefficients à estimer. On peut aussi écrire sous la forme vectorielle :

$$y_i = x_i'\beta + \epsilon_i$$

β est le vecteur des coefficients du modèle $(\beta_0, ..., \beta_K)$ et x_i' le vecteur des variables explicatives $(1, x_{i,1}, ..., x_{i,K})$. Et enfin la forme matricielle qui permet de décrire le problème pour les N individus :

$$Y = X\beta + \epsilon$$

$$\text{avec } Y = \begin{pmatrix} y_1 \\ y_2 \\ ... \\ y_n \end{pmatrix}, X = \begin{pmatrix} 1 & x_{1,1} & ... & x_{1,K} \\ 1 & x_{2,1} & ... & x_{2,K} \\ ... & ... & ... & ... \\ 1 & x_{n,1} & ... & x_{n,K} \end{pmatrix}, \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ ... \\ \beta_K \end{pmatrix} \text{ et } \epsilon = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ ... \\ \epsilon_n \end{pmatrix}$$

Pour estimer les coefficients on utilise la méthode des moindres carrés qui s'écrit formellement comme :

$$\hat{\beta} = (X^t X)^{-1} X^t y$$

En appliquant la méthode des moindres carrés avec les données d'entraînements on obtient les coefficients, c'est la phase d'apprentissage. On utilise les coefficients ainsi obtenus dans l'équation de y avec de nouvelles données pour effectuer les prédictions, c'est la phase de production. La régression linéaire est une méthode de base de l'apprentissage automatique. L'algorithme d'apprentissage automatique le plus utilisé au cours des recherches présentées dans ce manuscrit est Random Forest.

1.6.3 Méthodes avancées

Les algorithmes les plus populaires sont les algorithmes de deep learning. Le deep learning est une version plus complexe d'un réseau de neurones. Le réseau est composé de neurones interconnectés en différentes couches qui transportent l'information et créent des règles par essais et erreurs.

Un modèle deep learning est un réseau de neurones avec de multiples couches cachées de neurones. Il est utilisé pour tous les problèmes de reconnaissance d'images et traitement de signal (audio par exemple) [20]. Les performances du deep learning dans ces domaines sont meilleures que les autres algorithmes.

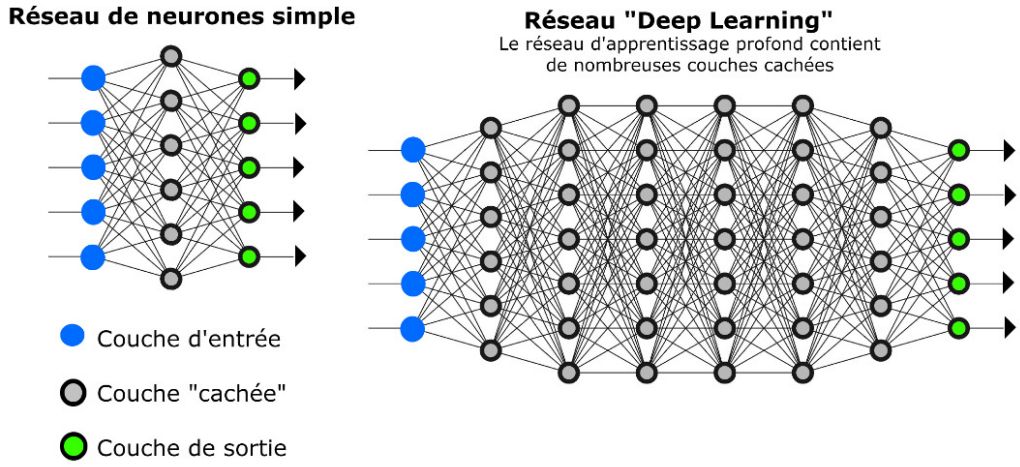


FIGURE 1.14 – Représentation de réseaux de neurones et deep learning (source : actuia.com)

L'apprentissage par renforcement est un autre domaine important de l'intelligence artificielle. L'apprentissage par renforcement est utilisé dans la robotique. Les robots formés par apprentissage de renforcement sont capable d'exploits proches, voir supérieurs à ceux des humains, comme AlphaStar un robot qui a battu les meilleurs joueurs du célèbre jeu de stratégie en temps réel Stacraft II [21].Dans l'apprentissage par renforcement, les agents sont formés selon un mécanisme de récompense et de punition. L'agent est récompensé pour les bons mouvements et puni pour les mauvais. Ce faisant, l'agent essaie de minimiser les mauvais mouvements et de maximiser les bons.

Bien que l'apprentissage profond et l'apprentissage par renforcement soient les plus utilisés dans le traitement de signaux et dans la robotique, le machine learning classique est toujours considéré comme très performant (voir plus performant) pour certaines problématiques industrielles. Un des avantages des approches plus simples est leur interprétabilité, certains chercheurs s'intéressent à rendre les réseau neuronaux plus facile à interpréter pour l'industrie [22]. Les réseaux de neuronaux et deep learning ont aussi besoin de beaucoup de données pour fonctionner, or nous ne disposons pas toujours de quantités de données assez grandes dans l'industrie. Pour nos recherches nous nous sommes donc concentré sur les algorithmes

de machine learning. Ce chapitre propose un aperçu des recherches en intelligence artificielle pour la fabrication (voir figure 1.15) de semiconducteur.

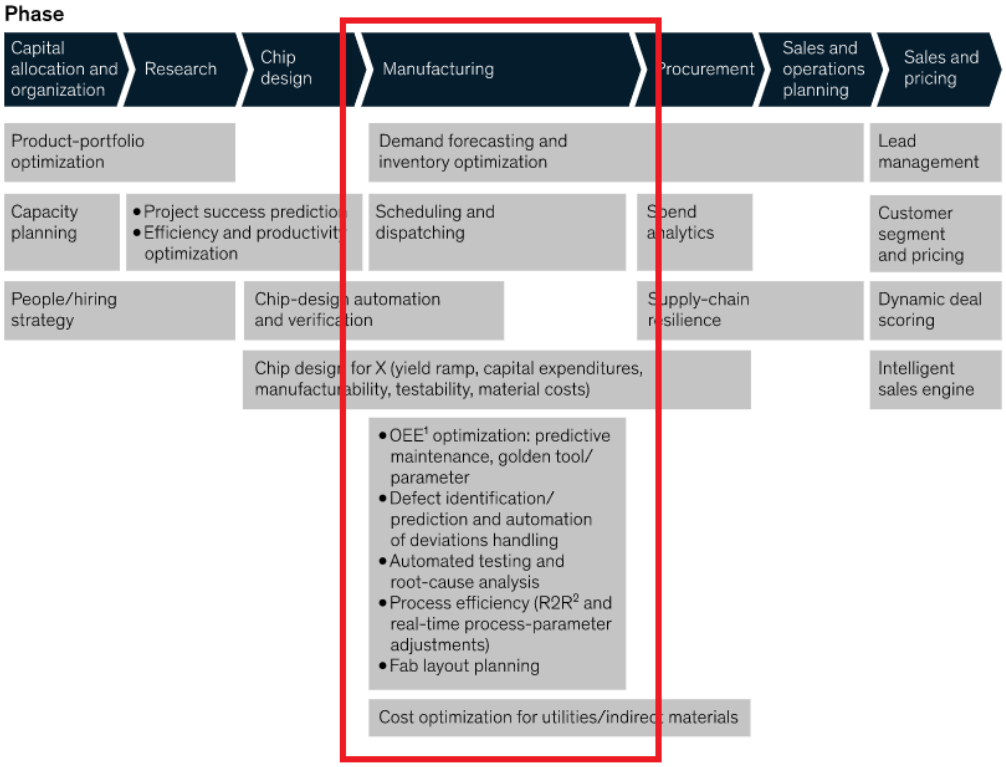


FIGURE 1.15 – Use cases spécifiques à la production dans l'industrie du semiconducteur

La maintenance prédictive permet de surveiller la santé des équipements afin d'éviter les pannes et donc de réduire les coûts. Des maintenances effectuées tardivement pourront causer des pannes et un arrêt de la machine ce qui perturbera la production. En revanche des maintenances trop fréquentes coûteront de l'argent inutilement. A STMicroelectronics la programation linéaire est une solution en cours de d'étude pour la maintenance prédictive [23]. De nombreuses méthodes d'intelligence artificielle sont aussi utilisées pour la maintenance prédictive. L'article [24] est une étude complète des différentes recherches existantes en maintenance prédictive. Dix des méthodes étudiées sont des réseaux de neurones, neuf sont des méthodes de machine learning classiques et quatre sont des méthodes de deep learning. Pour la maintenance prédictive le machine learning classique semble être une solution, au même niveau que les réseaux de neurones ou le deep learning.

L'identification et la détection de défauts, dont font parti les recherches du chapitre 2, sont un autre des sujets de recherche dans l'industrie du semi-conducteur. Le problème de détection de défauts peut être posé comme de l'analyse d'images avec des photos de qualités suffisantes (figure 1.16).

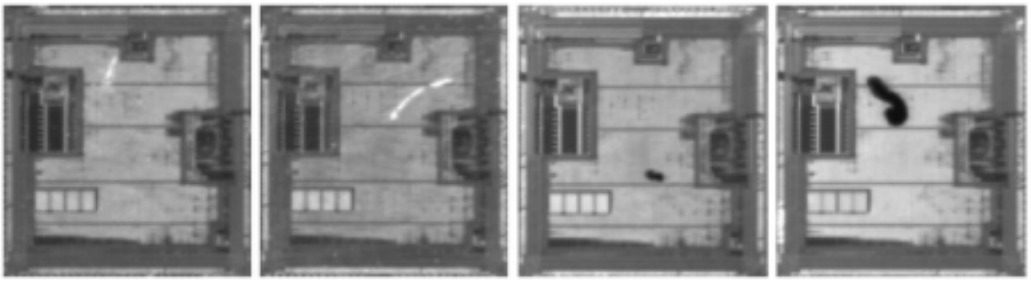


FIGURE 1.16 – Images de puces électroniques pour analyse (source : [1]

Ce n'est donc pas étonnant que de nombreuses recherches sur ce sujet portent sur des réseaux de neurones ou deep learning [25] [1] [26] [27] [28]. Pour pouvoir utiliser ces méthodes il faut une grande quantité de données étiquetées et une bonne compréhension du sujet. Des méthodes de classification non supervisées sont aussi utilisées, comme k-means [29] [30] [31] ou mixtures gaussiennes [32]. La plupart des données de défaut ne sont pas encore étiquetées ce qui rend impossible l'utilisation de classification supervisée. Des recherches sont en cours à STMicroelectronics pour utiliser un modèle Support Vector Machine (SVM) à une classe (One class SVM) [33]. L'avantage d'un SVM est qu'il n'a pas besoin de données sur les cas mauvais pour être entraîné. En effet il suffit de définir la zone des bons pour détecter les mauvais (One class : classe des bons dans le cas de la détection de défauts). C'est une méthode "à moitié supervisée" qui répond aux problématiques du milieu du semiconducteur. La difficulté est d'avoir assez de données pour apprendre correctement la "frontière" des cas bons/mauvais (problématique high mix low volume). Ceci n'est pas un problème pour les méthodes non supervisées qui fonctionnent sur peu de données.

La détection automatique de la root cause est un sujet critique dans l'industrie du semiconducteur. Les méthodes utilisées à STMicroelectronics ainsi que les recherches effectuées sur le sujet sont décrites dans le chapitre 3.

Le scheduling est au coeur de la production des semiconducteurs. En effet les machines complexes permettent d'effectuer de nombreuses tâches et il faut planifier l'ordre des wafers qui vont être processés par une machine. Il faut maximiser le nombre de wafers par jour tout en respectant de nombreuses contraintes. Historiquement le scheduling est effectué par des algorithmes de programation linéaire mais les méthodes machine learning commencent à être utilisées. Le deep learning [34], le reinforcement learning [35] et le machine learning classique [36] notamment.

1.6.4 Random Forest

Random forest est un algorithme de classification ou de régression très classique qui a été présenté la première fois par [37] et décrit en détail par [38]. Cet algorithme fonctionne en créant un certain nombre d'arbres de décisions [39] et en prenant comme résultat le mode des classes pour la classification ou la moyenne de celles ci pour la régression. Les données d'apprentissage de chaque arbre sont une sélection aléatoire d'un nombre défini d'attributs. L'algorithme de random forest seul (sans décrire l'arbre de décision) est assez court, le voici pour la classification :

Algorithm 1 Fit

```
1: procedure FIT(Data, NbSamples)
2:     RandomFeatures = random(list[Data]*NbSamples)
3:     Trees = for i in (0 : NbTree) append TrainTree(RandomFeatures[i])
4:     return Trees
```

Algorithm 2 TrainTree

```
1: procedure TRAINTREE(Data, MaxDepth)
2:     Tree = DesicionTreeClassifier(MaxDepth)
3:     Tree.Fit(Data)
4:     return Tree
```

Algorithm 3 Predict

```
1: procedure PREDICT(Feature)
2:     Predictions = []
3:     for Tree in Trees do
4:         Predictions.append(Tree.Predict(Feature))
5:     return Mode(Prediction)
```

Le paramètre $NbSamples$ correspond au nombre d'attributs aléatoires qui vont être sélectionnés. La fonction random(list[$data$]*$NbSamples$) est une pseudo fonction qui sélectionne $NbSamples$ attributs aléatoirement dans les données. On entraîne chaque arbre sur une sélection d'attributs différente, c'est ce qui fait la force de random forest en lui donnant une variance faible. Ce paramètre peut aussi être problématique quand le nombre d'exemples est limité. Dans ce cas, une petite valeur de $NbSamples$ pourra détériorer la qualité du modèle. Dans ce cas on choisit $NbSamples = N$ pour un petit échantillon.

On décrit random forest mathématiquement, suivant la définition de [38]. Nous ne considérons ici que le problème de classification binaire. Une random forest est un prédicteur constitué d'une collection de M arbres de régression aléatoires. Pour le j-ème arbre de la famille, la valeur prédite au point x est notée par $m_n(x; \Theta_j, D_n)$, où $\Theta_1, ..., \Theta_M$ sont des variables aléatoires indépendantes, distribuées de la même manière qu'une variable aléatoire générique Θ et indépendante de D_n. En pratique, la variable Θ est utilisée pour ré-échantillonner l'ensemble d'apprentissage avant la croissance des arbres individuels. Dans le cas binaire, un classificateur, ou règle de classification m_n est une fonction mesurable de Borel de X et D_n qui tente d'estimer l'étiquette Y à partir de X et D_n. Le classificateur m_n est cohérent si sa probabilité d'erreur :

$$L(m_n) = P[m_n(X) \neq Y] \xrightarrow{n \to \infty} L^* \tag{1.1}$$

Ou L^* est l'erreur du classificateur de Bayes optimal inconnu :

$$m^*(x) = \begin{cases} 1 & \text{if } P[Y = 1|X = x] > P[Y = 0|X = x] \\ 0 & \text{otherwise} \end{cases} \tag{1.2}$$

Dans le contexte de la classification, le classificateur de la forêt aléatoire est obtenu via un vote majoritaire parmi les arbres de classification, c'est-à-dire,

$$m_{M,n}\left(x;\Theta_1,...,\Theta_M,D_n\right) = \begin{cases} 1 & \text{if} \quad \frac{1}{M}\sum_{j=0}^{M} m_n(x;\Theta_j,D_n) > 1/2 \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

Les arbres de décisions générés par random forest sont des classificateurs supervisés. Un arbre de décision va apprendre des règles simples déduites des données pour prédire la valeur de la cible. Un arbre est un graphe non orienté, acyclique et connexe. Il possède trois catégories de noeuds, le noeud racine, les noeuds internes et les noeuds terminaux. Le noeud racine est le noeud d'accès à l'arbre. Les noeuds internes sont les noeuds qui possèdent des descendants et qui ne sont pas le noeud racine, les noeuds terminaux sont les noeuds qui n'ont pas de descendants. L'arbre est construit par partition recursive de chaque noeud en fonction de la valeur de l'attribut testé. Le critère optimisé est l'homogénéité des descendants par rapport à la variable cible. La variable testée dans un noeud est celle qui maximise cette homogénéité. Pour déterminer l'homogénéité l'arbre utilise le gain d'information dans les versions ID3 et C4.5 [40]. Pour calculer le gain d'information dans le noeud interne S sur l'attribut a l'algorithme partitionne S sur les valeurs de l'attribut a en k sous groupes S_1,...,S_k (k est le nombres de valeurs distinctes de l'attribut a). Le gain d'information est :

$$GI(S;a) = H(S) - \sum_{i=1}^{k} p_i H(S_i)$$

ou p_i est la probabilité qu'un élément de S appartienne à S_i. $H(S)$ mesure l'écart de la distribution de la variable cible par rapport à la distribution uniforme, c'est l'entropie. $H(S) = 0$ si S est homogène (tous les éléments sont dans la même classe : toutes les probabilités pi sont égales à 0, sauf une qui est égale à 1). $H(S) = max$ si toutes les probabilités pi sont égales (tous les groupes Ci ont la même taille : $1 = ... = pn = 1/m$). L'algorithme va calculer le gain d'information pour les attributs pas encore testés, choisir l'attribut a avec le gain d'information optimal puis il va créer un test de décision sur ce noeud S qui va générer des sous noeuds correspondant à la partition créée. Il y a ensuite récurrence sur les noeuds testés. La sortie de la récurrence se fait quand tous les attributs ont été testés, quand le nombre de niveaux de noeuds atteint correspond à l'argument max_depth définit par l'utilisateur ou quand tout les éléments du noeud S sont dans la même classe.

La prédiction pour un individu se fait en suivant les tests de décision des noeuds. Par exemple pour le jeu de données libre Titanic avec lequel on peut prédire les chances de survivre d'un passager en fonction de ses caractéristiques. La figure 1.17 est un arbre de décision possible.

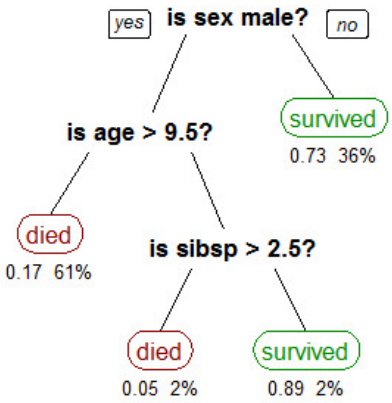


FIGURE 1.17 – Exemple arbre de décision titanic

La fonctionnalité de random forest qui nous intéresse pour l'analyse discriminante (Chapitre 3) est le calcul des features importance. En effet c'est en comparant les features importance du modèle qu'on peut déterminer quelle variable a le plus d'impact sur le modèle. Les variables les plus importantes sont de bons candidats pour être la cause de l'excursion. Pour l'algorithme random forest de Scikit-learn, la feature importance est calculée comme la diminution de l'impureté du nœud pondérée par la probabilité d'atteindre ce nœud. La probabilité du nœud peut être calculée par le nombre d'échantillons qui atteignent le nœud, divisé par le nombre total d'échantillons. Plus la valeur est élevée, plus la feature est importante.

Pour chaque arbre on calcule l'importance du noeud grâce à l'importance de Gini :

$$ni_j = W_j C_j - W_{left(j)} C_{left(j)} - W_{right(j)} C_{right(j)}$$

avec w_i nombre pondéré d'échantillons atteignant le noeud j, C_i la valeur d'impureté du noeud j, $left(j)$ noeud fils à gauche du noeud j et $right(j)$ noeud fils à droite du noeud j. L'importance de chaque feature pour un arbre de décision est calculé comme :

$$f_i = \frac{\sum_{j=0}^{N} n_j}{\sum_{k=0}^{M} n_k}$$

avec N : tout les noeuds splittés sur la feature i et M : les noeuds.

La feature importance est ensuite normalisée pour être comprise entre 0 et 1. Pour cela on divise par la somme de toutes les features importances :

$$norm(f_i) = \frac{f_i}{\sum_{j=0}^{N} f_j}$$

avec N = nombre de features.
Enfin la feature importance pour l'arbre est une moyenne de celle ci sur tout les arbres :

$$RFf_i = \frac{\sum_{j=0}^{T} norm(f_j)}{T}$$

avec T = nombre d'arbres.

Chapitre 2

Détection d'excursions topographiques avec un algorithme de classification non supervisé

L'industrie des semiconducteurs s'appuie sur la métrologie pour faire face à un environnement de production hautement compétitif et à la complexification de la technologie. En particulier, les mesures d'overlay, de focalisation et de dimensions critiques s'avèrent être un défi. Alors que la métrologie est nécessaire pour assurer la qualité en ligne, l'augmentation du nombre de mesures dégrade la productivité globale. Réduire d'échantillonage réduit les coûts de la métrologie, mais une politique trop agressive en matière de sampling est un vecteur d'événements de non-qualité. Ainsi, le compromis fondamental de la métrologie peut être formulé comme suit : comment est-il possible d'assurer la qualité tout en maintenant le nombre de mesures métrologiques au minimum ?

Les appareils modernes produisent une grande quantité de données qui ne sont pas utilisées pour le process control. Par exemple, les capteurs des scanners de lithographie mesurent la hauteur sur le wafer pour environ 35 000 points XY. Une idée suggérée est d'utiliser ces données pour le process control [3]. Ces données de leveling apportent des informations sur la topographie des wafers qui sont nécessaires aux scanners pour exposer les wafers. Dans ce document, nous proposons d'utiliser ces données, et non de les négliger, pour fournir des informations de qualité sur les lots ou wafers non-conformes.

En effet, le leveling est dans certains cas corrélé à la focalisation, aux dimensions critiques et aux mesures overlay [3, 41]. La détection d'anomalies de leveling sur les wafers dans les premières phases de production peut donc faire baisser les coûts de manière significative. De plus, en détectant les lots et les wafers suspects, nous pouvons cibler davantage de métrologie sur ces cas anormaux, ce qui permet une meilleure utilisation des capacités du système.

Pour y parvenir, une certaine préparation des données est nécessaire. Nous prenons en compte les variations non problématiques à l'échelle du wafer, comme les mesures inclinées d'un côté à l'autre ou près des bords. En raison de la forme des wafers, nous suggérons d'apprendre une tendance générale de la hauteur du wafer, avant de la soustraire, pour ne laisser qu'une erreur locale, avec des problèmes potentiels. En soustrayant cette forme moyenne, on se retrouve avec des défauts locaux (à haute fréquence) qui sont les plus susceptibles de causer des problèmes.

Ensuite, les données prétraitées sont utilisées pour alimenter un algorithme de machine learning non supervisé basé sur la densité (DBSCAN) qui peut détecter les valeurs aberrantes comme le ferait un expert humain.

Pour traiter les données nous proposons une méthode de nettoyage de données. Cette méthode est la régression polynomiale de Zernike. Les polynômes de Zernike sont continus et orthogonaux sur un cercle unitaire [42]. Ils sont une représentation appropriée des formes de signature de leveling que l'on trouve sur les wafers. Les polynômes de Zernike capturent les formes typiques des fréquences basses ; ainsi, l'apprentissage d'un tel régresseur polynomial produit une forme générale pour le wafer qui contient peu d'information sur les défauts. L'ajustement des données de leveling avec les polynômes de Zernike et la soustraction du modèle obtenu ont donné des résultats prometteurs pour le nettoyage de différentes formes optiques. Par exemple, l'utilisation du 4ème polynôme de Zernike supprime l'effet sphérique.

Ensuite, ces données nettoyées sont transmises à un algorithme non supervisé ; DBSCAN (Density-Based Spatial Clustering of Applications with Noise) qui suit les différentes variations de hauteur des wafers. La variation des données de leveling n'est pas la même sur toute la surface du wafer. Les bords des wafers ont une variabilité plus élevée que les centres des wafers. Pour contourner ce problème, une méthode de type ByPixel a été mise au point. Cette méthode consiste à comparer des points avec les autres wafers du même chuck (une machine de lithographie possède deux chucks vers lesquels sont dirigée chaque moitié des wafers du lot) pour chaque point mesuré sur le wafer afin de prendre en compte la variation de la variabilité. L'algorithme DBSCAN a été choisi comme noyau de cette méthode car il fonctionne sur de petits échantillons, et il est un algorithme non supervisé basé sur la densité, qui se comporte de manière anthropomorphique. Un tel algorithme peut éviter les décalages entre le jugement de l'expert humain et celui de la machine. Les résultats peuvent être tracés sur une wafer map pour montrer précisément où se situent les anomalies.

Cette méthode permet de détecter les chuck spot sur tous les wafers sans métrologie supplémentaire et sans aucun apprentissage préalable ni seuil. Des valeurs aberrantes de leveling provenant de cas d'excursion antérieurs ont également été détectées. Plus généralement, la méthode ByPixel DBSCAN est capable de détecter les changements de leveling de la topographie sur l'ensemble de la production.

2.1 Analyse des méthodes utilisées chez STMicroelectronics

Il existe peu de recherches sur le sujet de la détection d'excursions avec des données de leveling dans l'industrie du semiconducteur. Une méthode statistique appelée wafer profile est utilisée à STMicroelectronics pour détecter les problèmes de leveling.

La méthode wafer profile est une méthode statistique développée à STMicroelectronics [2]. Nous avons choisi de développer une nouvelle solution d'analyse du leveling car le process control standard ne suffisait pas à détecter toutes les excursions (figure 2.1).

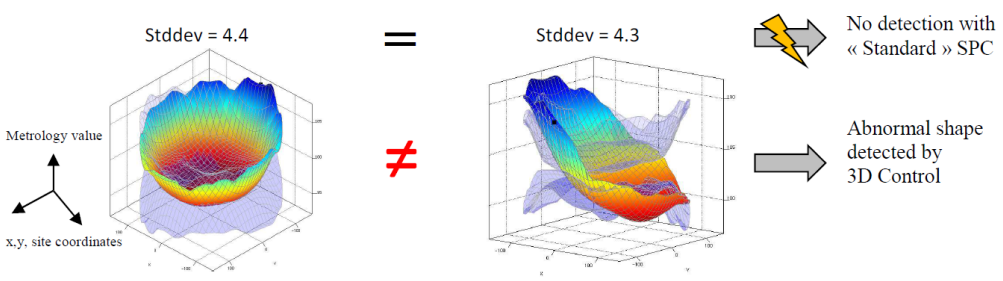


FIGURE 2.1 – Différence entre deux profil de wafer avec la même variance (source : [2])

Wafer Profile utilise les données de la métrologie. Les mesures ne recouvrent pas tout le wafer, il faut donc estimer le profil de leveling. Pour cela un modèle d'apprentissage est créé sur un échantillon de wafers références et la modélisation est utilisée pour estimer la distribution par site.

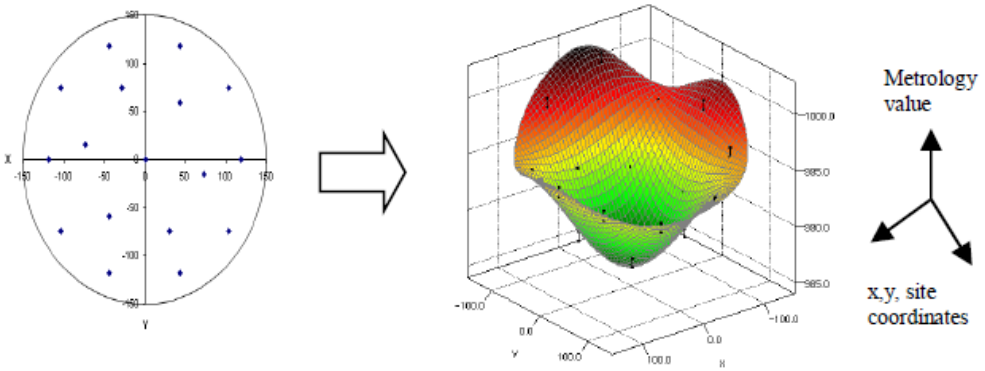


FIGURE 2.2 – Reconstruction de la forme du wafer, reconstruction complète à partir des données échantillonnées de métrologie (source : [2])

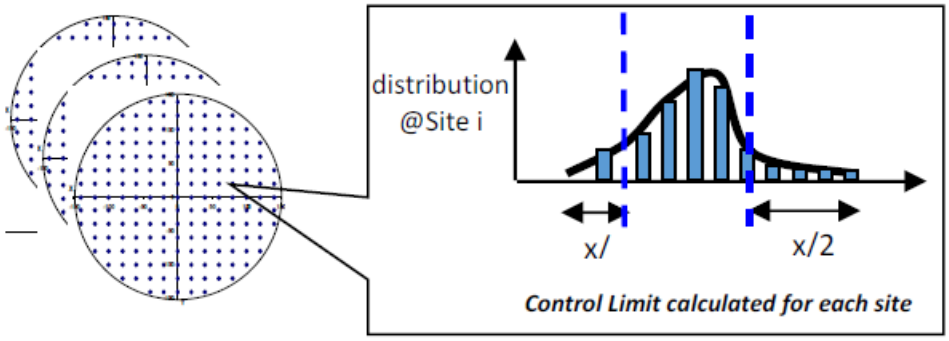


FIGURE 2.3 – Estimation de la distribution par site (source : [2])

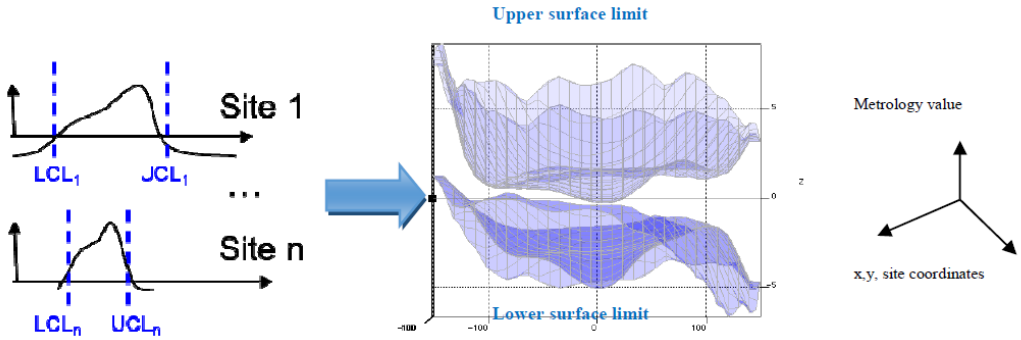


FIGURE 2.4 – Références utilisés comme limites de contrôles(source : [2])

A l'aide de wafers de références un profil contrôle est construit (figure 2.4). Une application

automatique en ligne estime le profil de chaque wafer mesuré et compare l'estimation au modèle de référence pour détecter les profils anormaux. Cette méthode est en production à STMicroelectronics et permet un contrôle du leveling. La limite de cette méthode est que les données utilisées sont mesurées sur un sampling de wafer (environ 2 par lot) et ne recouvrent pas tout le wafer (une centaine de points par wafer). La méthode permet de détecter les différences de profil global mais pas les excursions spécifiques comme les erreurs de focus. La solution présentée dans ce manuscrit utilise une autre source de données sans sampling avec 35 000 points par plaque.

Les recherches les plus proches sont celles de [3] qui utilisent aussi les données des équipements de lithographie pour détecter les excursions. La première étape pour l'utilisation des données de leveling de l'équipement de lithographie est de retirer les effets résiduels. Les auteurs de [3] identifient 3 effets résiduels. Le mooving average, le mooving standard deviation et l'effet chuck. Pour supprimer l'effet chuck et révéler la signature du processus la moyenne du chuck est retiré à chaque wafer.

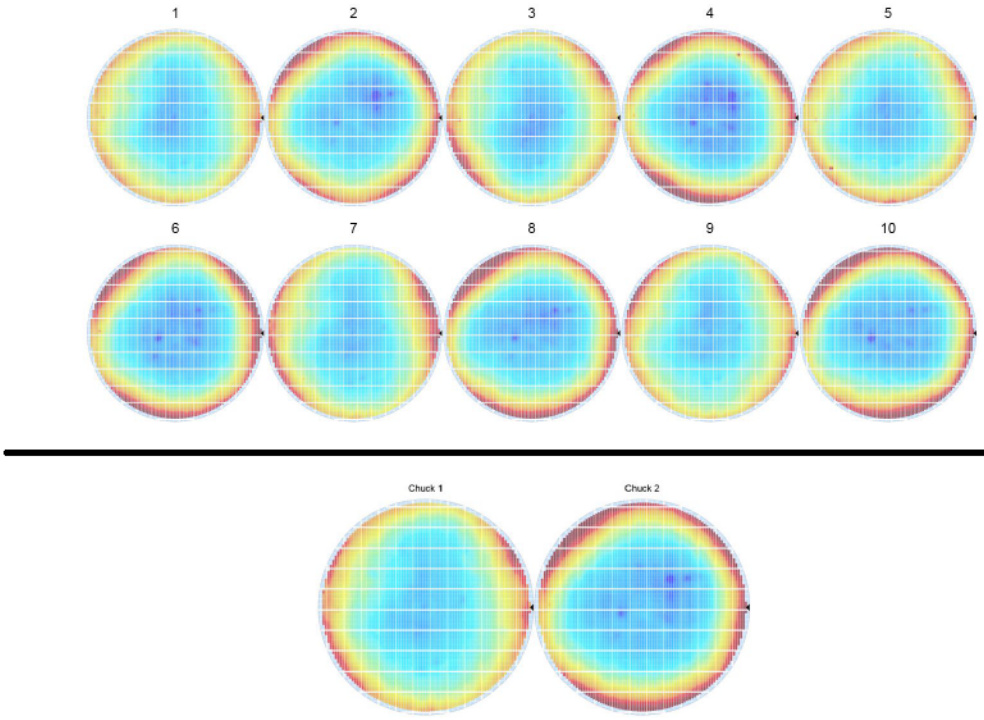


FIGURE 2.5 – Signature des wafers (haut) et moyenne du chuck (bas) (source [3])

Sur la figure 2.5, les données de leveling brutes de 10 wafers et la moyenne par chuck sont présentées. La moyenne par chuck est retirée à chaque wafer ce qui donne les signatures de la figure 2.6. Ces signatures sont les signatures process auxquelles il faut aussi retirer le mooving average et le mooving standard deviation de la même manière que l'effet chuck.
Pour analyser la signature du process le wafer est divisé en plusieurs zone (figure 2.7) ce qui permet de trouver plus précisément les excursions. Pour analyser les données de leveling des équipements de lithographie, [3] commence par nettoyer les données puis divise le wafer en différentes zones pour détecter précisément les excursions. Nous avons repris la même structure pour nos recherches.

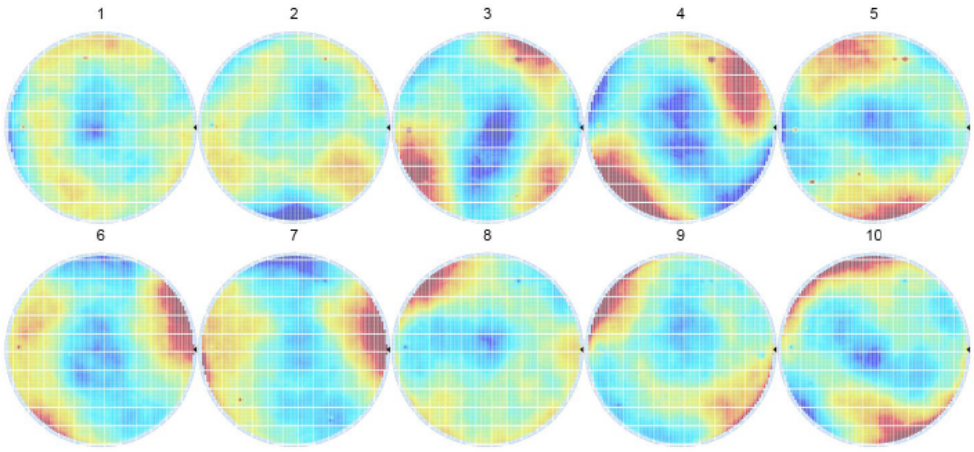


FIGURE 2.6 – Signature des wafers sans l'effet chuck(source [3])

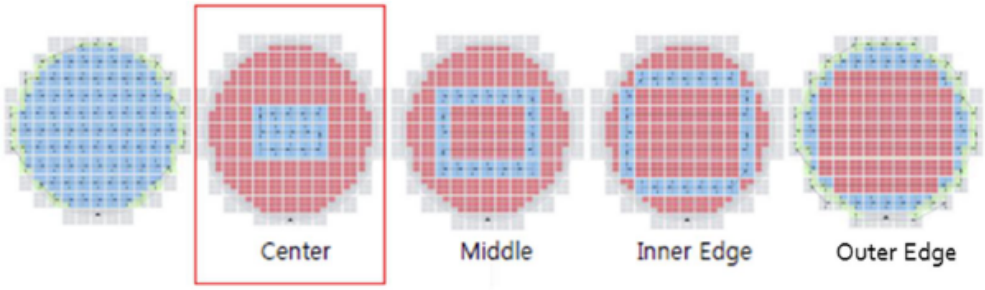


FIGURE 2.7 – Différentes zone d'analyse chuck(source [3])

2.2 Problématique

L'objectif est de détecter les anomalies topographiques sur les wafers pendant le process de fabrication en utilisant les données de leveling acquises par les équipements de lithographie.

Pour chaque plaquette passant par un équipement de lithographie, des points N ($\approx$ 35 000) sont mesurés, ce qui donne des triplets. $\{(x_i, y_i, z_i)\}_{i=1}^{N}$ où (x_i, y_i) représentent les coordonnées cartésiennes du i-ème point scanné et z_i sa hauteur.

L'objectif est de détecter les signatures anormales. Il n'est pas possible de former un algorithme supervisé pour détecter les excursions en raison de la fragmentation des données et du déséquilibre bon/mauvais. La fragmentation des données est une conséquence du high-mix low-volume qu'on retrouve dans l'industrie des semi-conducteurs. La quantité de données disponibles est insuffisante pour chaque contexte (un contexte est une agrégation possible de données avec les même caractéristiques. Exemple : 13 wafers de la même technologie, même niveau, même machine, même masque, même lot, même chuck). Ceci est accentué par le déséquilibre bon/mauvais ; il y a, heureusement, bien plus de données disponibles pour la situation normale que pour la situation anormale. Ci dessous un schéma qui illustre le problème de data fragmentation dans l'industrie du semi-conducteur.

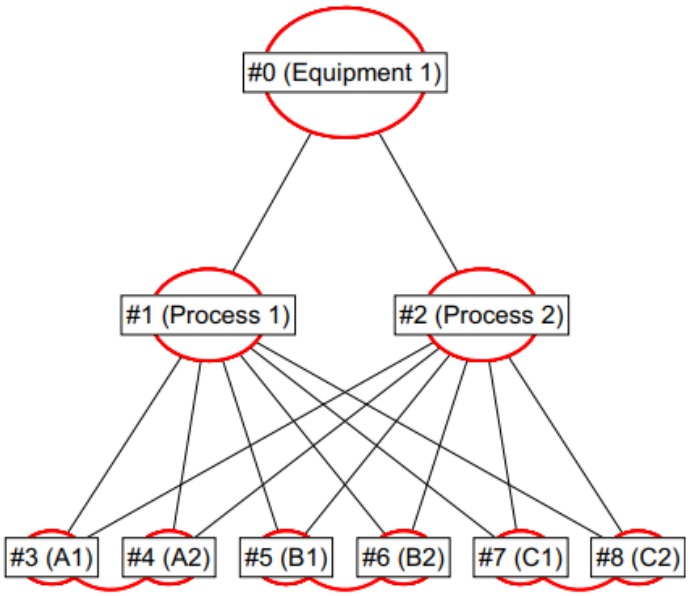


FIGURE 2.8 – Représentation en arborescence d'un outil CVD (Chemical Vapor Deposition) comprenant trois chambres (A, B, C) avec deux sous-chambres chacune (1 et 2), impliquées dans deux processus (Process 1 et Process 2).Ainsi, pour les wafers traités, douze configurations logistiques distinctes (c'est-à-dire des chemins) sont possibles. [4]

Il y a 12 contextes possible qui peuvent avoir différentes caractéristiques, ce qui empêche d'utiliser une agrégation des données pour effectuer un apprentissage.

Une autre problématique relative à l'apprentissage machine dans l'industrie est le bruit des données. Ce bruit est représenté par la variation de la variation de Z (hauteur) illustrée dans la figure 2.9.

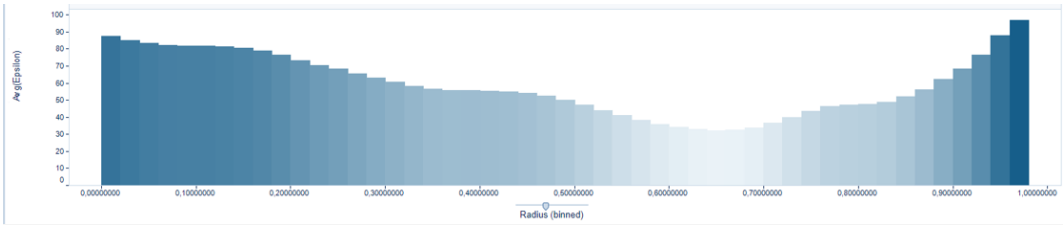


FIGURE 2.9 – Variation de Z sur le rayon du wafer

La variation Z n'est pas la même sur toute la surface du wafer. La solution de [3] est de diviser la plaquette en cinq zones avec une variation stable. C'est une façon possible de traiter le problème de la variation de la variance de Z.

2.3 Contributions

Les recherches sur le thème de la détection de signaux faibles sont les suivantes :

— Nettoyage des données de leveling avec Zernike : cette méthode fonctionne et peut être utilisée pour nettoyer des données sur une surface circulaire (comme les wafers).

— Détection d'outliers avec décomposition de Zernike et DBSCAN : moins performante et moins facile à interpréter que DBSCAN par pixel

— Détection des outliers en leveling avec DBSCAN par pixel : permet de détecter les outliers de leveling en temps réel mais avec quelques faux positif.

— Compte du nombre de clusters d'outliers avec DBSCAN : méthode simple qui peut être utilisée dans différents contextes pour analyser une outliers map. Le système d'alerte intègre cette méthode pour différencier les signatures.

— Classification des signatures avec random forest : Permet de différencier les signatures d'outliers de manière supervisée. Non utilisable avec peu de données. Il est difficile de récolter beaucoup de données d'excursions car une excursion est un évènement rare.

— Application d'analyse de données SpotFire : application d'analyse des outliers maps en détails. Non utilisée pour l'instant.

— Système d'alerte : système d'alerte sur les outliers en leveling utilisé à STMicroelectronics.

Excepté le système d'alerte et l'application d'analyse de données qui sont les solutions industrielles découlant des autres recherches, toutes les méthodes développées peuvent être utilisées seules.

2.4 Régression Polynomiale de Zernike

Les données brutes de leveling sont trop polluées pour être traitées directement. En effet, les variations de hauteur entre deux lots ou deux machines de lithographie ont des magnitudes du même ordre que les défauts de hauteur qui peuvent conduire à des faux positifs. Une condition préalable nécessaire est donc de se débarrasser de ces variations.

Pour ce faire, nous séparons les données en morceaux correspondant au même contexte de process (machine, produit, couche et chuck).

Nous nous débarrassons d'abord des informations qui ne dépendent pas du process et qui n'entraînent pas d'anomalies. Pour cela, nous devons faire mettre en place un modèle avec des valeurs variant lentement sur un disque représentant le wafer. Les polynômes de Zernike sont parfaitement adaptés à une telle représentation car ils forment une base de polynômes orthogonaux sur des disques correspondant à des formes *typiques* se produisant souvent en optique ou en électromagnétisme (inclinaison, diffraction, aberrations sphériques, etc.), voir [42] pour un aperçu récent.

Pour chaque degré (radial) $n \in \mathbb{N}$, il y a $n+1$ polynômes de Zernike indexés par Z_n^m, Z_n^{-m} pour tout $m > 0$ tel que m est pair si et seulement si n est pair. Par exemple, pour $n = 2$, il y a 3 polynômes de Zernike Z_2^2, Z_2^0, Z_2^{-2} et pour $n = 3$, il y a 4 polynômes de Zernike $Z_3^3, Z_3^1, Z_3^{-1}, Z_3^{-1}$.

La valeur d'un polynôme de Zernike en un point de coordonnées polaires (ρ, ϕ) est donnée pour un degré $n \in \mathbb{N}$ et $0 \leq m \leq n$ avec $n - m$ pair par la formule :

$$\begin{cases} & Z_n^m(\rho,\phi) = R_n^m(\rho)\cos(m\phi) \\ \text{or} & Z_n^{-m}(\rho,\phi) = R_n^m(\rho)\sin(m\phi) \end{cases} \quad \text{with } R_n^m(\rho) = \sum_{k=0}^{(n-m)/2} \frac{(-1)^k (n-k)!}{k!((n+m)/2-k)!((n-m)/2-k)!} \rho^{n-2k}$$

Comme nous voulons nous débarrasser uniquement du signal variant sur le disque, nous ne considérons que les premiers polynômes de Zernike $K = 3$ correspondant à $n \leq 2$.

Pour chaque point mesuré i, on forme ainsi le vecteur $\mathbf{p}_i \in \mathbb{R}^K$ des valeurs K des polynômes de Zernike choisis aux coordonnées polaires $(\rho_i = \sqrt{x_i^2 + y_i^2}, \phi_i = \text{atan2}(y_i, x_i))$ correspondant aux coordonnées cartésiennes (x_i, y_i) :

$$\mathbf{p}_i = \begin{bmatrix} Z_0^0(\rho_i,\phi_i) \\ Z_1^1(\rho_i,\phi_i) \\ Z_1^{-1}(\rho_i,\phi_i) \\ \vdots \end{bmatrix}.$$

Ensuite, nous ajustons le polynôme en minimisant la perte des moindres carrés

$$\boldsymbol{\lambda}^\star = \min_{\boldsymbol{\lambda} \in \mathbb{R}^K} \sum_{i=1}^{N} (z_i - \boldsymbol{\lambda}^\top \mathbf{p}_i)^2 \tag{2.1}$$

Après avoir choisi les polynômes de Zernike souhaités en fonction de l'effet à soustraire, la carte résiduelle est obtenue pour chaque point $i = 1, .., N$ comme suit :

$$r_i = \boldsymbol{\lambda}^\top \mathbf{p}_i - z_i \tag{2.2}$$

En utilisant les 3 premiers polynômes, il est possible de nettoyer la mise à zéro des équipements ainsi que l'inclinaison horizontale et verticale comme le montre la figure 2.10. On remarque que la tendance générale non informative est en effet supprimée alors que les défauts locaux ont tendance à se manifester.

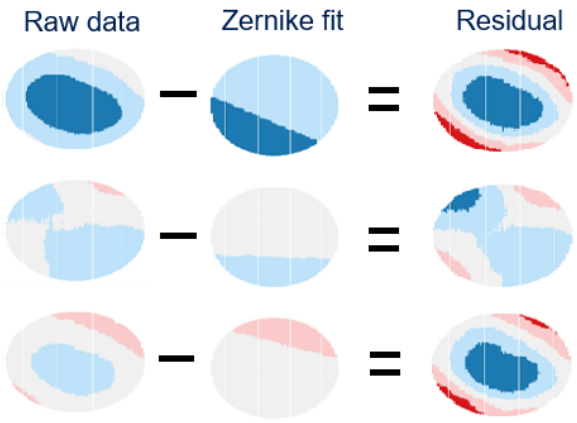


FIGURE 2.10 – nettoyage des données avec les trois premiers polynômes (exemples)

L'ajustement polynomial de Zernike est une représentation appropriée des formes optiques de leveling sur le wafer. Cette méthode est nécessaire car il n'est pas encore possible de nettoyer les données mathématiquement car certaines signatures non liées au process ne sont pas explicables.

Notez qu'il est possible de nettoyer d'autres signatures de formes optiques en fonction des besoins en utilisant d'autres polynômes de Zernike.

2.5 DBSCAN

DBSCAN (voir Algorithme 4) utilise deux paramètres ϵ et minPts, minPts est le nombre minimum de points dans une distance ϵ pour que ce regroupement de points soit considéré comme un cluster.

Algorithm 4 DBSCAN

```
1: clusters = [[]]
2: procedure DBSCAN(D, ϵ, minPts)
3:     c = 0
4:     nonVisited = D
5:     for Pt in nonVisited do
6:         nonVisited.remove(Pt)
7:         neighbors = EpsilonNeighborhood(D, Pt, ϵ)
8:         if size(neighbors) < minPts then
9:             Pt is classified as an outlier
10:        else
11:            c = c + 1
12:            ExpandCluster(D, Pt, neighbors, c, ϵ, minPts)
13: procedure ExpandCluster(D, Pt, neighbors, c, ϵ, minPts)
14:    clusters[c].add(Pt)
15:    for Pt2 in neighbors do
16:        if Pt2 in nonVisited then
17:            nonVisited.remove(Pt2)
18:            neighbors2 = EpsilonNeighborhood(D, Pt2, ϵ)
19:            if size(neighbors) ≥ minPts then
20:                neighbors.append(neighbors2)
21:        if Pt2 not in clusters then
22:            clusters[c].add(Pt2)
23: procedure EpsilonNeighborhood(D, Pt, ϵ)
24:    res = []
25:    for Pt2 in D do
26:        if distance(Pt, Pt2) < ϵ then res.add(Pt2)
27:    return res
```

Une telle solution non supervisée est une réponse à la problématique high-mix low-volume, typique dans la fabrication des semi-conducteurs, qui rend difficile la création d'un ensemble de données d'apprentissage suffisamment important. En effet, les excursions ne sont pas fréquentes et le contexte change rapidement, de sorte qu'il est presque impossible de disposer d'un ensemble de données significatif pour chaque excursion possible. Les algorithmes non supervisés ne sont pas affectés par ce problème car ils ne nécessitent pas d'apprentissage pour des excursions particulières et détectent des comportements anormaux de nouveaux types, contrairement aux algorithmes supervisés. Un autre avantage des méthodes basées sur la densité telles que DBSCAN est leur capacité à reproduire un comportement similaire à celui de l'homme pour la classification. Ceci est illustré par la figure 2.11. Les deux groupes sont détectés par DBSCAN même s'ils ne sont pas distincts en termes de distance. Dans une telle situation, un algorithme utilisant uniquement la distance comme kmeans ne serait pas en mesure de trouver ces deux groupes. D'autres avantages de DBSCAN sont que i) il ne nécessite pas de connaître le nombre de classes, ii) il peut détecter les valeurs aberrantes, et iii) il fonctionne sur des échantillons de petite taille.

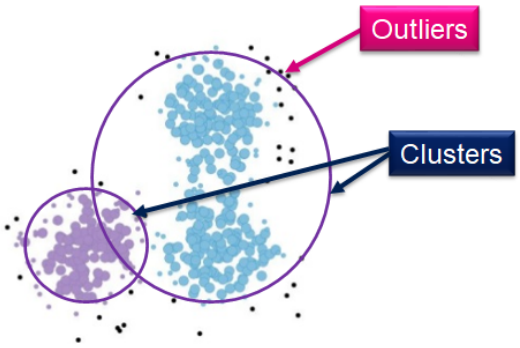


FIGURE 2.11 – DBSCAN density based schema

Pour choisir l'algorithme DBSCAN, une analyse des algorithmes de classification non supervisé a été effectuée. En commençant par ce graphique on peut déjà éliminer un grand nombre d'algorithmes :

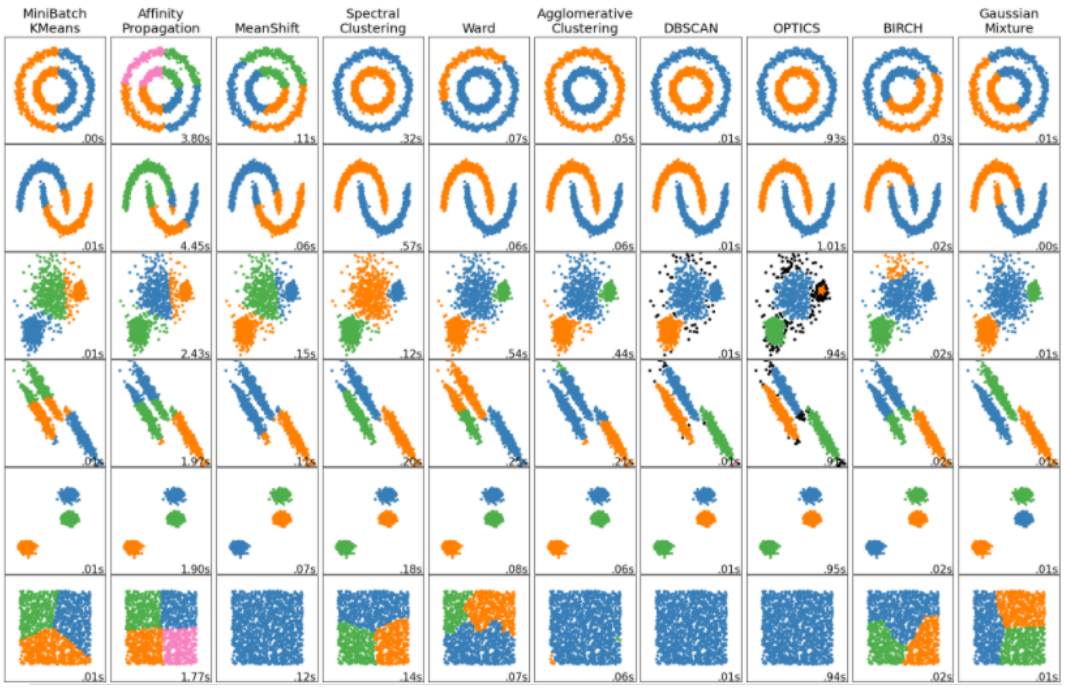


FIGURE 2.12 – Comparaison des algorithmes non supervisés

Les deux algorithmes qui semblent répondre à la problématique sont DBSCAN et OPTICS car ce sont les deux seuls qui permettent de détecter automatiquement les outliers (points noirs sur la figure 2.12). Ce sont aussi, avec gaussian mixture, les algorithmes qui déterminent le mieux les classes. DBSCAN a été choisi plutôt que OPTICS car son temps d'exécution était légèrement plus rapide [43], ce qui permet de traiter les lots en temps réel. OPTICS est une version améliorée de DBSCAN mais les ajouts ne sont pas intéressants pour la détection d'outlier de leveling sur les wafers. D'autres versions de DBSCAN existent comme [44] qui est une version de DBSCAN avec contraintes et [16] qui est une version pour les données temporelles. Ces deux versions ne sont pas les seules, ci-dessous un tableau comparant les différentes versions de DBSCAN. Ce tableau provient de l'article [5].

TABLE I. CRITICAL EVALUATION OF VARIOUS DBSCAN ENHANCEMENTS

DBSCAN Enhancements	Enhanced Features	Density Threshold *Eps & MinPts*	Computational Cost
Fahim et al.	can perform efficient clustering without requiring any user supplied input parameters	Not Required	*O(nk+ m2k+nm)* K= no of circles over the data space m=average points in each circle. m=n/k (n= number of points in each circle
Liu et al.	Clustering uneven dataset Efficiently varied in density	Computed automatically	*O (n * log n)*
EI-Sonbaty et al.	Scalability & Better performance than DBSCAN with speed up faster unto 5 times	User Dependent	Not discussed
Liu	Introduction of Kernel function to make clustering more accurate	User Dependent but less dependency on *Eps*	- Complexity Linear - Much less than that of *O (n * log n)*
Uncu et al.	Efficient clustering results of the datasets having different densities	Computed automatically	Expensive in terms of Computational
Borah et al.	Memory efficient & I/O cost minimized	Computed automatically (uses only *Eps*)	*O(n * log n)*
Ram	Limited the amount of allowed local density variation to achieve better results	User Dependent	Not discussed
Borah et al.	α = used to limit the amount of allowed local density variation.	User Dependent	*O (n * log n)* n = number of object
Xiaoyun et al.	Reduced Computational Cost & Efficient cluster results for large dataset	User Dependent	Not Discussed
Viswanath et al.	Excellent Clustering results as compare to its DBSCAN & early Variations of it	User Dependent	*O (n + k2)*
Birant et al.	1.Discovering Cluster on spatial-temporal data, 2.Identification of adjacent Clusters. 3.Identification of Noise objects from cluster with different densities	Calculated Automatically	Same as that of DBSCAN technique complexity
Mahran et al.	-Provide high performance with the advantage of high degree of parallelism	Calculated Automatically	N^2 / C C = Total number of cells in each grid
Yu et al.	Offer DBSCAN to determine density threshold in an unsupervised way	Calculated Automatically	*O(n * log n)*

FIGURE 2.13 – Comparaison des versions de DBSCAN [5] ([6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18])

Parmi les versions existantes, aucune ne semble répondre spécifiquement à la problématique de détection d'outlier en leveling sur un wafer. Certaines fonctionnalités supplémentaires pourraient être utiles mais pas assez pour justifier le coût en temps d'exécution. La problématique de détection de signatures anormales en leveling est très spécifique, il a donc été décidé de partir de la version de base de DBSCAN et de l'adapter.

2.6 By Pixel DBSCAN

Pour chaque chuck, nous effectuons un clustering non supervisé unidimensionnel avec DBSCAN, un pour chaque point de mesure sur les wafers. Cela signifie que le DBSCAN est exécuté sur 12 ou 13 points, ce qui correspond à la taille des deux chucks (il y a deux chucks par lot). Afin de contrôler le nombre et la qualité des valeurs aberrantes, le paramètre ϵ est défini comme suit :

$$\epsilon = k * \sigma(E) \tag{2.3}$$

$$E = \{z > median(Z) - k * \sigma(Z), z < median(Z) + k * \sigma(Z)\} \tag{2.4}$$

où $Z = (z_i^{(1)}, z_i^{(2)}, .., z_i^{(c)})$ est l'ensemble des valeurs de leveling pour un pixel et k est une constante fixée empiriquement pour réguler le nombre de valeurs aberrantes. De cette manière, epsilon suit la variation observée sur la figure 2.9. $E \neq Z$ de sorte qu'un premier ajustement est effectué pour calculer epsilon sans aberrations statistiques. Le second paramètre, $minPts$ est fixé à la moitié de la taille du chuck. BP-DBSCAN ne fonctionne pas comme une limite basée sur la variation car les points voisins d'un groupe de densité ne peuvent pas être des outliers, BP-DBSCAN est donc plus proche du jugement humain qu'une limite.

Cette méthode permet de traiter automatiquement tous les wafers et de générer des outliers maps. Chaque valeur aberrante sera marquée ce qui permet de détecter les lots et wafer dissidents à l'aide un seuil de valeur aberrante. BP-DBSCAN peut traiter un lot toutes les 2 minutes grâce à une programmation parallèle.

Pour comprendre l'algorithme voici la description d'une exécution de celui ci sur une zone avec outlier du wafer. La zone choisie est entourée en rouge sur la figure 2.14

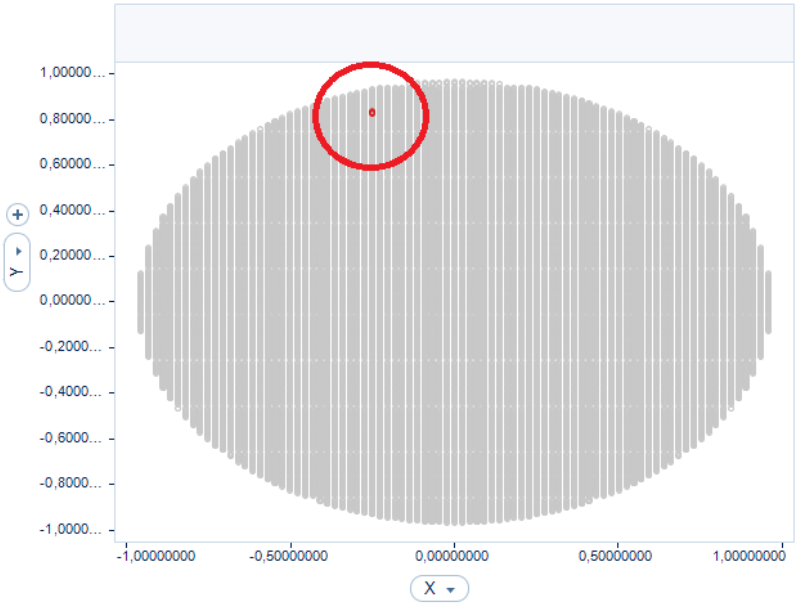


FIGURE 2.14 – Outlier spot sur le wafer

Après un zoom sur la zone entourée en rouge, le nuage de point de la figure 2.15 est obtenu.

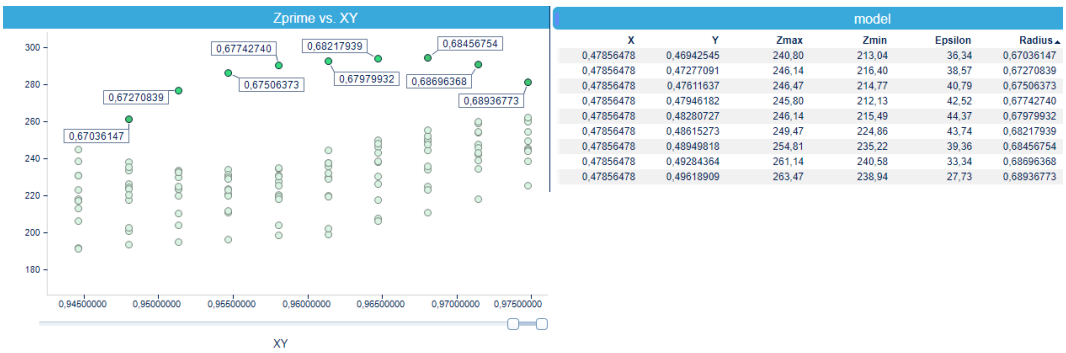
FIGURE 2.15 – Nuage de points avec numéro de wafer. En ordonnée la hauteur Z et en abscisse X+Y. A gauche la boite en rouge correspond au premier filtre

L'algorithme BP-DBSCAN est utilisé sur tous les wafers d'un chuck qui sont comparés entre eux. On peut voir ici en label le numéro des wafers qui correspondent aux points du nuage. Epsilon va être calculé pour chaque coordonnée après l'application d'un premier filtre. Le premier filtre est basé uniquement sur la variance par point et évite de calculer epsilon avec des points extrêmes. Les données filtrées E sont :

$$E = \{z > median(Z) - k * \sigma(Z), z < median(Z) + k * \sigma(Z)\}$$

Epsilon est ensuite calculé sur les points filtrés :

$$epsilon = k * \sigma(E)$$



X	Y	Zmax	Zmin	Epsilon	Radius
0,47856478	0,46942545	240,80	213,04	36,34	0,67036147
0,47856478	0,47277091	246,14	216,40	38,57	0,67270839
0,47856478	0,47611637	246,47	214,77	40,79	0,67506373
0,47856478	0,47946182	245,80	212,13	42,52	0,67742740
0,47856478	0,48280727	246,14	215,49	44,37	0,67979932
0,47856478	0,48615273	249,47	224,86	43,74	0,68217939
0,47856478	0,48949818	254,81	235,22	39,36	0,68456754
0,47856478	0,49284364	261,14	240,58	33,34	0,68696368
0,47856478	0,49618909	263,47	238,94	27,73	0,68936773

FIGURE 2.16 – Gauche : Nuage de points avec radius. Droite : tableau des epsilons par coordonnées / radius

On peut voir sur le tableau de la figure 2.16 les valeurs d'epsilon pour chaque coordonnée sur lesquelles DBSCAN va être utilisé avec comme paramètres tout les points qui des wafers du même chuck et la valeur d'epsilon correspondante.

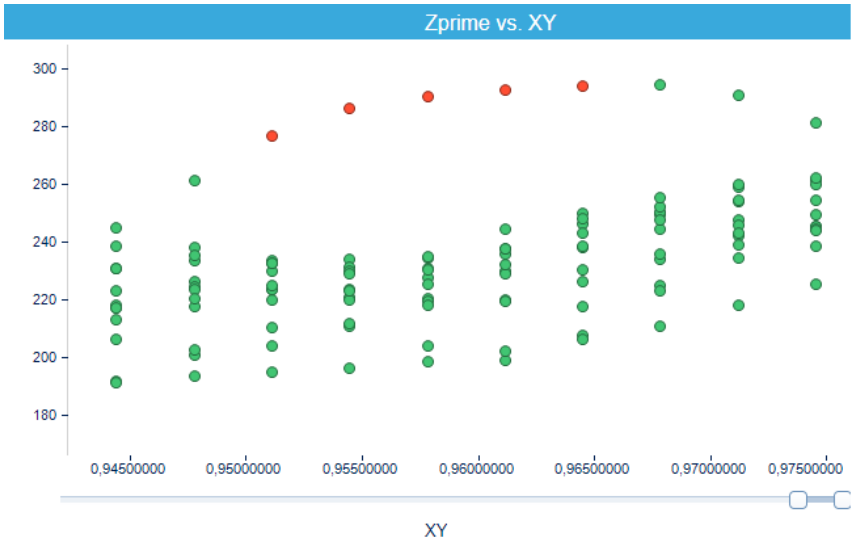


FIGURE 2.17 – Nuage de points avec outliers en rouge

Après la détection des outliers par DBSCAN voici le nuage de points obtenu 2.17. Les outliers sont reportés sur l'outlier map 2.14. Le même processus est répété sur toute la plaque.

2.7 Nombre de clusters d'outlier avec DBSCAN

Après avoir tracé l'outliers map on veut pouvoir compter le nombre de clusters d'outliers. Jusqu'ici DBSCAN est utilisé sur chaque point $[x, y]$ du wafer. Pour chacun de ces points DBSCAN trouve les outliers en terme de hauteur (z) parmi les wafers du chuck. Pour trouver le nombre de cluster on va utiliser DBSCAN de manière plus conventionnelle en lui donnant comme paramètre les coordonnées de tous les outliers pour un wafer et la distance physique sur la plaque qui sépare deux clusters (*epsilon*). Le résultat est une map des cluster d'outliers.

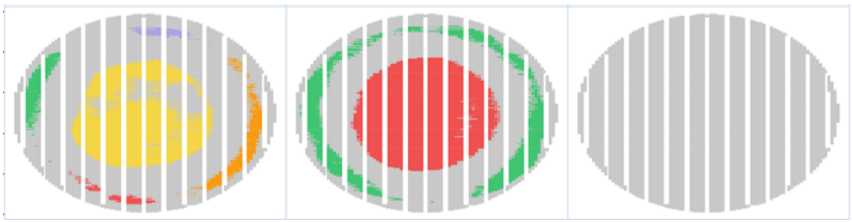


FIGURE 2.18 – Outliers clusters map

Chaque couleur correspond à un cluster. Le nombre de clusters et leurs tailles permettent, entre autre, de différencier un problèmes global 2.18 d'un focus spot 2.19

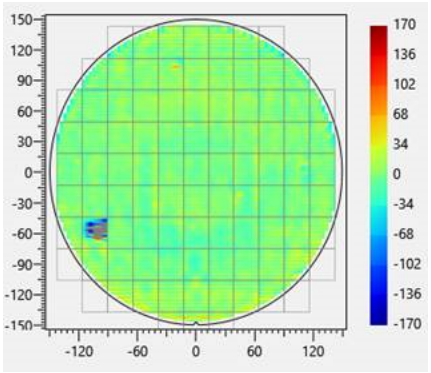


FIGURE 2.19 – Focus spot

Ces informations serviront aussi à classifier les différentes formes d'outliers mais ne sont pas suffisantes, c'est pourquoi des travaux de recherches sur la classification de signatures d'outliers ont été effectuées et sont présentées dans la section suivante.

2.8 Classification de signature d'outlier par random forest

Une des difficultés de la détection de défauts avec DBSCAN est un taux de faux positifs trop élevé. Ce taux de faux positifs est du au fait que la machine compense une partie des erreurs de leveling automatiquement. Pour pallier à ce problème il est possible de classifier les différentes outlier maps et de leur attribuer un risque qui est déterminé en fonction de la capacité de la machine à compenser cette signature anormale. C'est un exercice de classification supervisé des formes d'outliers map qui est un cas difficile à traiter car une des principales problématique de l'industrie est le manque de cas d'excursions. Il faudra attendre plusieurs années afin d'avoir assez d'excursions pour créer un jeu de données suffisamment conséquent pour construire un modèle performant. Pour pouvoir développer un première version du futur algorithme de classification on décide de prendre un epsilon très faible, de cette manière l'algorithme captera assez d'excursions pour effectuer un apprentissage et les difficultés futures pourront être anticipées. Voici des exemples d'outliers maps.

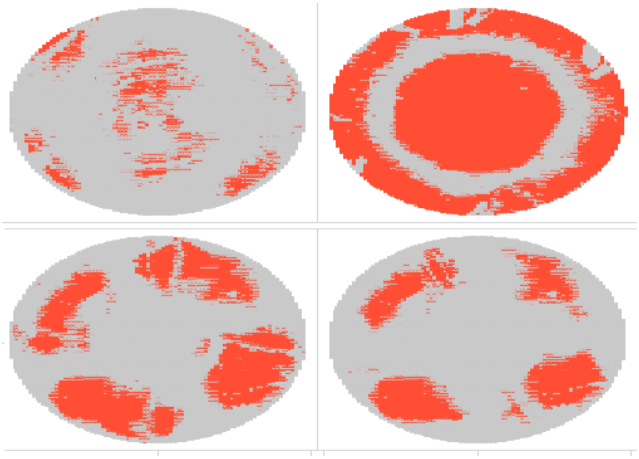


FIGURE 2.20 – Exemple d'outliers maps

Les formes des outliers maps sont distinctives ce qui devrait permettre l'apprentissage. Toutes les formes sont labellisées à la main.

LotId	WaferNr	Label
	4	SOMBRERO
	25	SOMBRERO
	13	INCONNU
	15	MANGEKYOU
	1	INCONNU
	17	INCONNU

FIGURE 2.21 – Exemple labels

Il y a 319 wafers anormaux dans l'échantillon. Les formes redondantes sont nommées et les formes exceptionnelles sont labellisées "INCONNU". De cette manière, il est attendu que le modèle créé puisse classifier toute nouvelle forme non répertoriée comme inconnue.

La technique utilisée pour classifier ces formes est de diviser le wafer en différentes zones, puis pour chaque zone on extrait le pourcentage d'outliers de la zone qui sont les attributs qui permette à l'algorithme de classifier les différentes signatures (figure 2.22.

LotId	WaferNr	Label	OuterCircleThin	OuterCircleLarge	InnerCircleThin	InnerCircleLarge	CenterDotSmall	CenterDotLarge	QuarterCirclesPerpendicular	QuarterCirclesDigonal
	4	SOMBRERO	91.86	80.56	99.72	98.10	100	99.67	40.59	45.60
	25	SOMBRERO	91.86	80.56	99.72	98.10	100	99.67	40.59	45.60
	13	INCONNU	0	10.53	92.17	88.05	6.78	60.32	30.57	68.4
	15	MANGEKYOU	9.60	25.71	3.07	8.78	0	23.99	79.89	36.90

FIGURE 2.22 – Data classification

Les zones sélectionnées doivent correspondre aux spécificités des signatures globales. Par exemple pour la signature mangekyou, voici la représentation graphique de QuarterCircles-Diagonal :

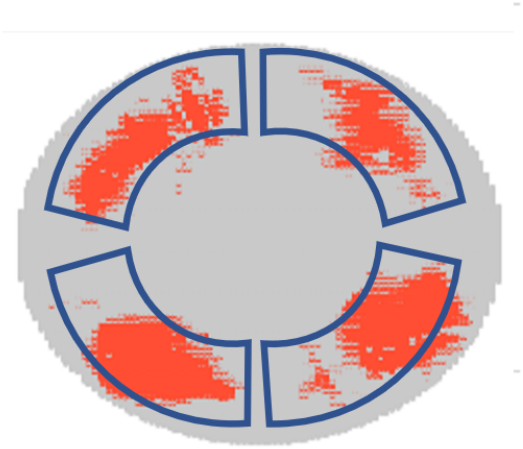


FIGURE 2.23 – Zone QuarterCirclesDiagonal superposée sur signature Mangekyou

La zone représente bien la spécificité de la signature. Pour chaque nouvelle signature ajoutée dans la liste des signatures connues, il peut être nécessaire d'ajouter des zones comme attributs pour que le modèle puisse distinguer la nouvelle signature. L'algorithme utilisé est random forest et sa précision pour la classification des formes sur 100 RUNS (la taille du jeu de test égale à 20% de la taille de l'échantillon) est de 91% avec la classe inconnu et de 98% sans la classe inconnu. La méthode utilisée pour classifier les nouvelles signatures comme inconnues n'est pas la bonne car une nouvelle forme peut être plus proche d'une autre classe qu'elle ne l'est de la classe inconnu. Cette difficulté peut être contournée avec une très grande quantité de données. Sinon, d'autres recherches seront nécessaires pour la classification de la classe inconnu quand une quantité suffisante de données de signatures anormales aura été récoltée.

Une autre méthode a été développée à STMicroelectronics pour éviter les faux positifs. Les différentes signatures corrigées par les équipements de lithographie ont été expliquées. En normalisant les données brutes par rapport à ces signatures on obtient des données propres sur lesquelles DBSCAN ne détectera pas de faux positifs. Ces signatures corrigibles n'existent pas pour tous les cas d'études, les recherches sur la classification des signatures anormales doivent donc tout de même être poursuivies.

2.9 Résultats

La méthode BP-DBSCAN s'est révélée capable de détecter les incidents lithographiques. La figure 2.24 et la figure 2.25 sont respectivement une outliers map et une carte de défectuosité d'une excursion détectée. Nous voyons clairement qu'il y a une forte corrélation entre l'outliers map et la carte de défectuosité. L'outlier map a été générée par BP-DBSCAN au cours du process, en amont de la carte de défectuosité. Cela illustre l'avantage d'utiliser des données de leveling pour détecter les problèmes liés au process qui n'ont pas été repérés par le contrôle standard du process.

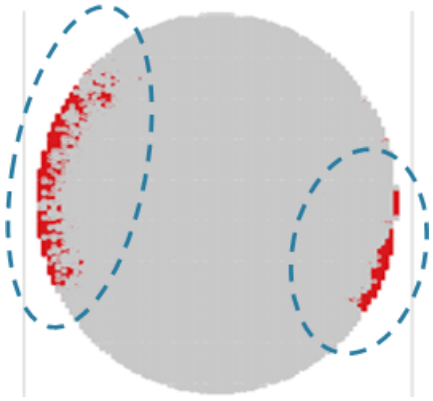


FIGURE 2.24 – BP-DBSCAN excursion outlier map

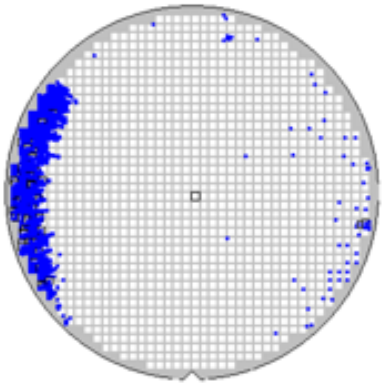


FIGURE 2.25 – defectivity excursion outlier map

Deux autres excursions liés à des incidents lithographiques ont été détectés par BP-DBSCAN. Ces incidents ont été signalés pendant le processus de fabrication par inspection visuelle, il n'y a donc pas de résultats de défectuosité disponibles.

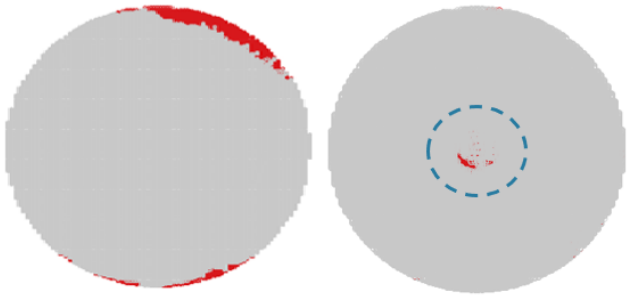


FIGURE 2.26 – BP-DBSCAN excursions outlier maps

D'autres wafers avec outliers qui n'ont pas créé de problèmes au cours du process ont été détectés (figure 2.27. Ces wafers doivent être classé "sans risque" par l'algorithme de classification automatique des signatures d'outliers.

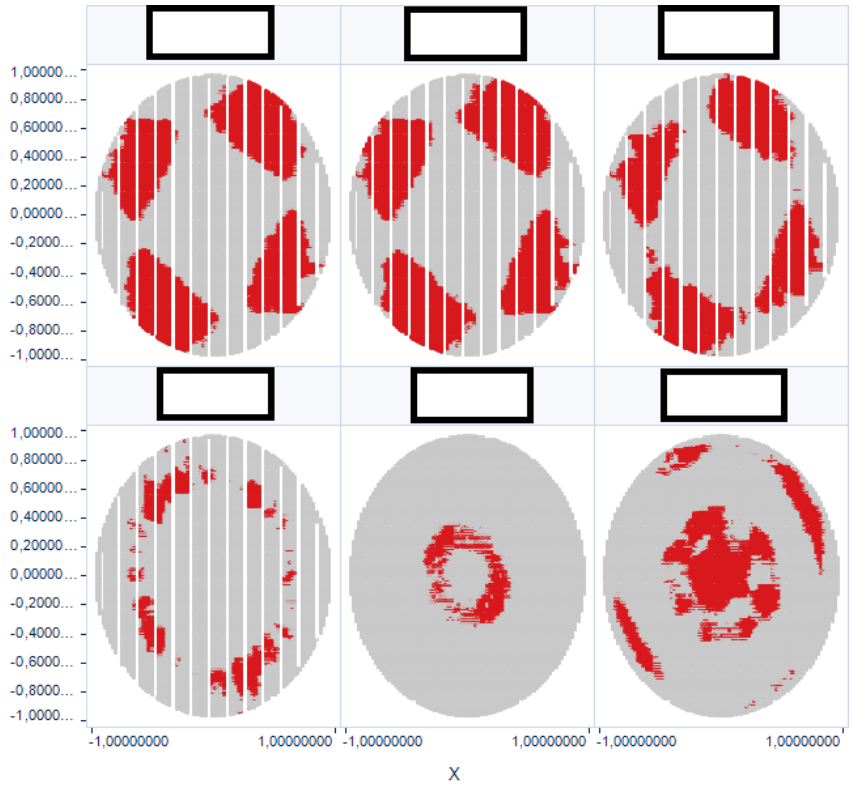


FIGURE 2.27 – Outlier maps à comparer avec les résultats EWS

L'intérêt d'utiliser la détection de signatures de leveling anormales avec BP-DBSCAN après le signalement d'un problème est de déterminer si ce problème est lié au leveling ce qui peut aider à la résolution de celui ci. L'utilisation optimale de la méthode BP-DBSCAN est la détection des excursions avant qu'elles ne soient signalées au cours du process (coût moyen) ou en defectivity (coût important). Détecter les excursions le plus tôt possible est obligatoire pour réduire les coûts.

Pour pouvoir utiliser BP-DBSCAN de manière optimale (en amont) il faut compléter la méthode par un tri des faux positifs 2.8 et une détection de root cause 3.

2.10 Implémentation des solutions

Pour obtenir les résultats présentés dans ce chapitre il a fallu traiter 6 mois de données sur une machine. Il est nécessaire de couvrir un volume important de données car il y a très peu d'excursions, l'algorithme doit donc traiter plusieurs mois de données déjà stockées mais aussi les nouvelles données en temps réel pour espérer capturer les excursions. Les résultats doivent être rendus disponibles pour être analysés par les ingénieurs process. Pour répondre à ces problématiques, une application automatique, un système d'alerte et deux applications d'analyse de données ont été développées. Cette section décrira les différentes méthodes et choix d'architecture utilisés.

Les figures 2.28 et 2.29 sont un schéma deux parties décrivant l'architecture et le fonctionnement de la solution.

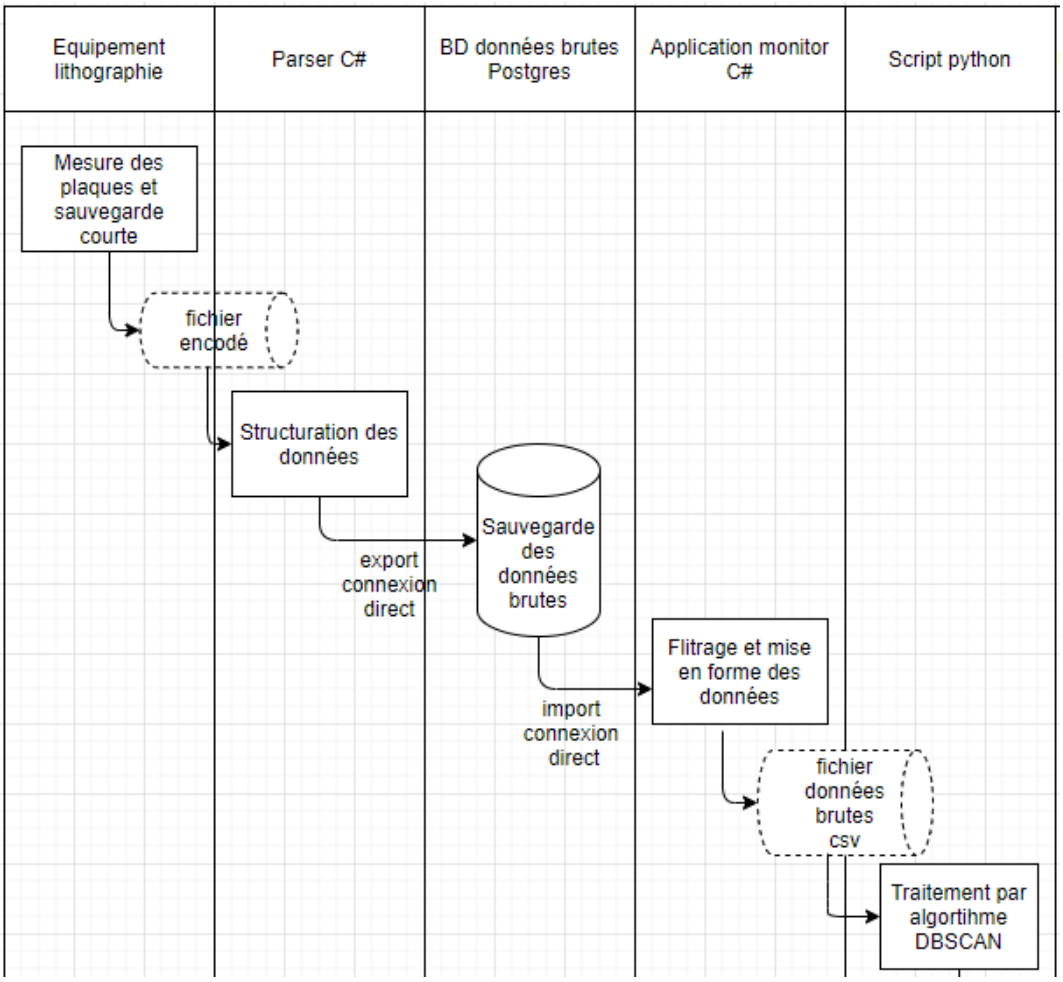


FIGURE 2.28 – Schéma architecture partie 1

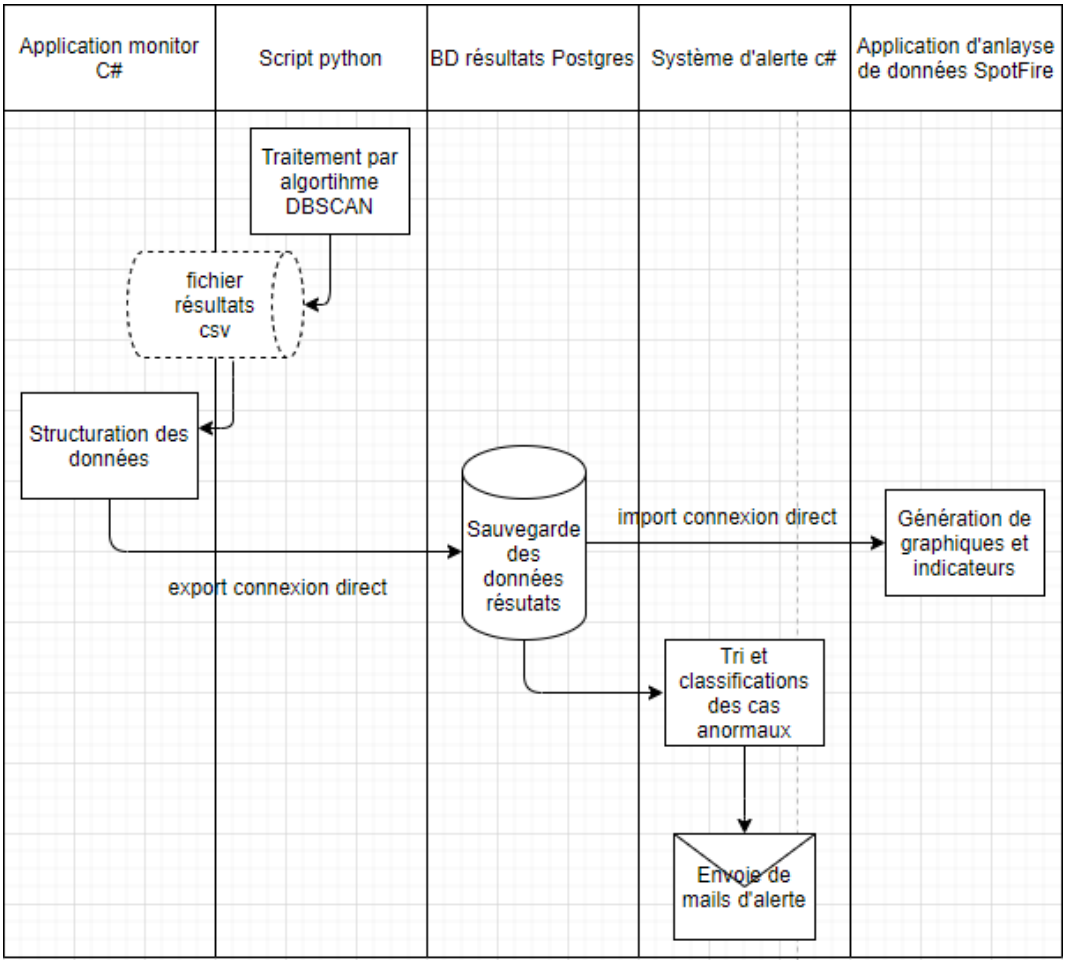


FIGURE 2.29 – Schéma architecture partie 2

Le leveling sensor de l'équipement de lithographie mesure le leveling sur la plaque. Il mesure les données de leveling d'environ 35 000 points par plaque qui sont encodées et sauvegardées pendant une courte période pour servir au focus de la plaque. Une première application C# décrypte les données puis les sauvegarde dans une base de données PostgreSQL (Postresql est un système de gestion de base de données open source géré par une communauté internationale de développeurs).

Une fois que les données sont stockées, une application C# va récupérer les données nécessaires pour la détection de signatures anormales et les mettre en forme. Il y a deux applications différentes, une qui permet de traiter toute la base de données qui doit être utilisée en premier et une autre qui fonctionne en temps réel. L'application de traitement de toutes les données déjà stockées va parcourir les différentes machines, technologie, niveaux et masques disponibles dans la base. Pour chacun de ces contextes (machine/technologie/niveau/masque) les données de chaque lot disponible sont exportées 30 par 30. Les fichiers sont divisés par chuck, les données d'environ 12 à 13 wafers sont disponibles sur chaque fichier.

Le script python traite chaque fichier indépendamment pour générer les résultats wafer à wafer et simultanément pour le lot à lot. Une spécificité de la méthode ByPixel est qu'il faut effectuer 35 000 clusterings DBSCAN sur environ 12 à 13 individus 2 fois par lot. Cette méthode est gourmande en temps d'exécution. Il faut cependant réussir à respecter un niveau de performance assez élevée pour traiter les lots en temps réel (1 lot toute les 2 minutes).

Pour atteindre le niveau de performance requis on exécute une partie du code en parallèle

ce qui permet d'exécuter 20 clusterings simultanément. La librairie JobLib disponible sur python permet de facilement paralléliser le code. La programmation parallèle (multithreading) est une solution intéressante pour les problèmes de performance des solutions machine learning.

L'application monitor C# va ensuite sauvegarder les fichiers de résultats csv générés par le script python sur la base de données de résultats PostgreSQL. Ces résultats sont accessibles par le système d'alerte et l'application d'analyse de données SpotFire. Une application web permet aux utilisateurs de s'abonner au système d'alerte en sélectionnant les contextes qui les intéressent. Le système d'alerte va ensuite trier les cas anormaux et envoyer les alertes mails aux utilisateurs abonnés.

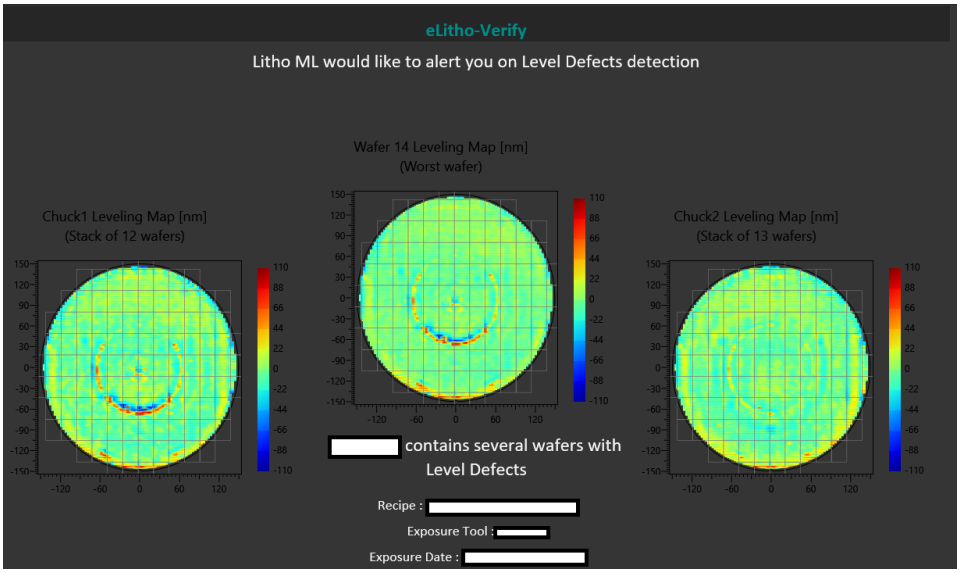


FIGURE 2.30 – Exemple système d'alerte mail : defect

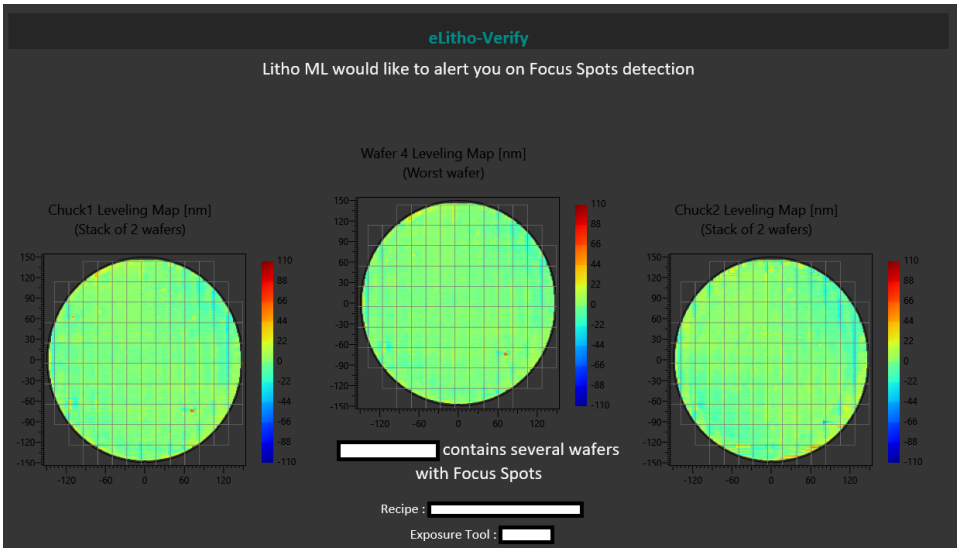


FIGURE 2.31 – Exemple système d'alerte mail : focus spot

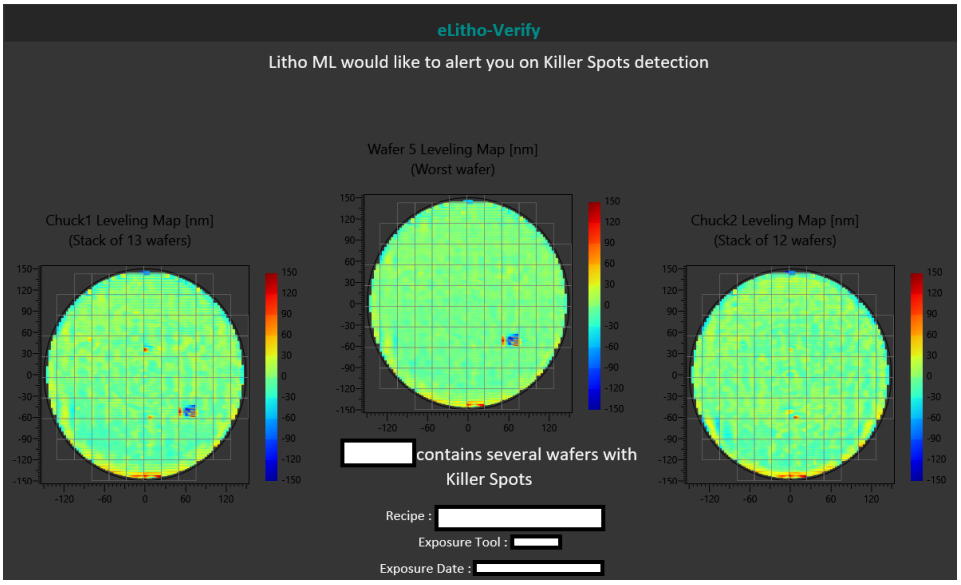


FIGURE 2.32 – Exemple système d'alerte mail : killer spot

Les 3 figures 2.30 2.31 2.32 sont les 3 mails d'alerte possible. Chaque type de mail correspond à une catégorie d'excursion différente. Il existe 3 catégories pour le moment, ce nombre devrait augmenter avec l'apparition de nouveaux types d'excursions récurrentes. La catégorie est déterminée à l'aide de la version DBSCAN qui compte le nombre de clusters d'outliers (figure 2.7), si il n'y a qu'un seul cluster important, la catégorie est focus spot (figure 2.31), si le nombre d'outliers dans ce cluster est grand, la catégorie est killer spot (Un killer spot est une excursion dont la taille va réduire le rendement, voir figure 2.32), enfin, tout les autres types d'excursions sont classés comme level defect (figure 2.30). En plus de la catégorie, l'utilisateur connaît le nom de la machine d'exposition, la recette et le numéro du lot. L'outliers map du pire wafer du lot est affichée ainsi que des outliers maps empilées par chuck. A terme il faudra inclure plus d'informations pour aider à la prise de décision. Des informations comme le nombre d'occurrences du problème, la root cause (chapitre 3) et plus de détail sur le type de signature 2.8 sont importantes.

Pour détecter les outliers, BP-DBSCAN n'a pas été utilisé dans l'application industrielle. Une autre méthode qui supprime la problématique de la variation de la variation du leveling sur le wafer a été développée (pas dans le cadre de mes travaux de thèse). La méthode consiste à supprimer tous les effets "polluants" sur la plaque. Pour y arriver il a fallu déterminer les origines des différentes signatures qui polluaient les données de leveling sur le wafer. La figure 2.33 décrit le processus qui permet d'obtenir des wafers maps de leveling propres.

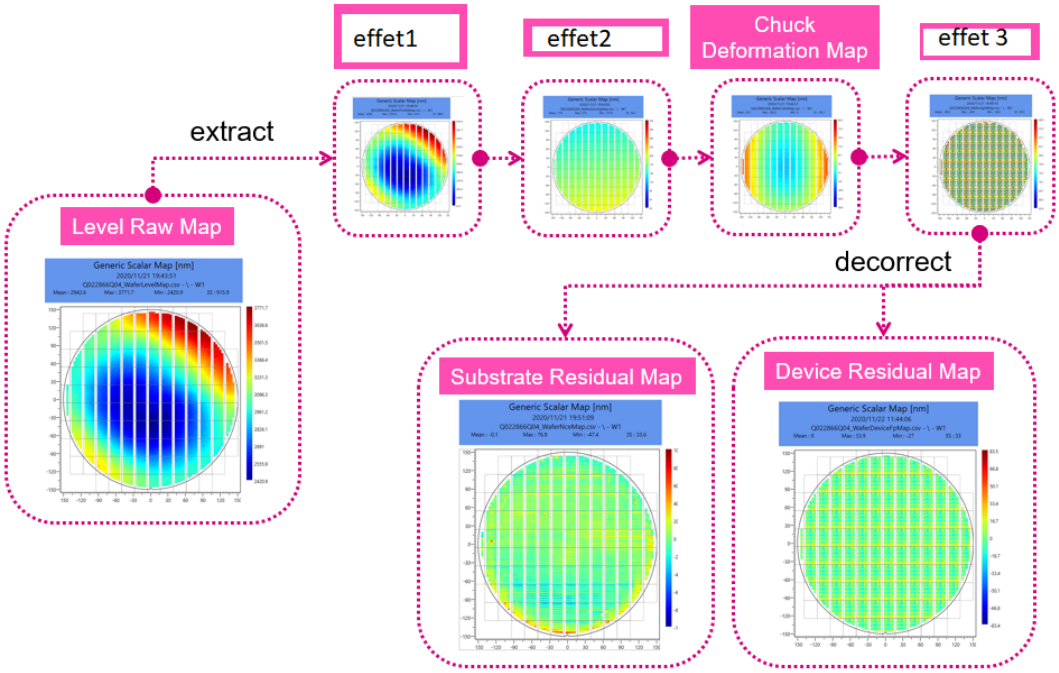


FIGURE 2.33 – Décomposition des signatures leveling

Cette méthode remplace la méthode Zernike qui permettait de supprimer les effets 1, 2 et la déformation du chuck mais pas l'effet 3. La méthode Zernike est une estimation des signatures polluantes alors que cette méthode est un calcul mathématique des effets physiques. Les données obtenues après avoir supprimé les effets sont assez propres pour être utilisées telles quelles (un simple seuil suffit pour détecter les outliers). La méthode BP-DBSCAN sera utile pour l'analyse de certains effets comme la déformation du chuck.

L'utilisateur pourra utiliser l'application SpotFire (logiciel d'analyse de données) pour analyser les données en détail. Le développement des différentes solutions industrielles est en phase beta et certains ingénieurs peuvent déjà les utiliser. Les recherches théoriques et méthodologiques effectuées ont permis la mise en place des solutions industrielles présentées et devraient permettre d'ajouter de nouvelles fonctionnalités (classification en détail avec 2.8).

2.11 Conclusion

Les leveling sensors des équipements de lithographie mesurent 35 000 points par plaque sur tout les wafers, l'utilisation de ces données permet de réduire les coûts en détectant des évènements de non qualités, qui sont normalement détectés par la défectuosité ou les tests EWS, en amont pendant le process. Ces données ont été nettoyées avec Zernike data clean puis les excursions ont été détectées avec BP-DBSCAN sur un échantillon très important qui a pu être obtenu en développant une application d'extraction des données en temps réel. Cette solution de détection à permit de créer un système d'alerte automatique par mail. Ce système peut être améliorer avec une meilleure classification des types d'excursion, la détection de la root cause et une application SpotFire dédiée.

Chapitre 3

Analyse discriminante Random Forest

3.1 Problématique

L'analyse de root cause est le processus qui a pour objectif de détecter la cause d'un problème en analysant les paramètres du système. Il faut trouver les paramètres discriminants, ceux qui causent le problème. L'analyse de root cause est critique pour la qualité à STMicroelectronics. Une excursion mesurée à une étape ne peut pas être expliquée par l'étape à laquelle elle est mesurée car plusieurs étapes ont pu précéder la mesure. Il est possible que l'excursion n'ait pas déclenchée d'alarme tout de suite et que le problème se soit dégradé. C'est pour cela qu'il est nécessaire de développer une méthode de détection de la root cause. Une méthode de détection de root cause peut être utilisée dans de nombreux cas à STMicroelectronics. En effet, à chaque fois qu'un problème de process est détecté, la première chose à faire est de trouver la cause pour résoudre le problème. Les excursions de process peuvent être lié au CD, overlay, thickness, focus spot... qu'importe le type de problème, la cause doit être détectée. Il existe une méthode de détection de root cause à STMicroelectronics. Cette méthode est Partial Least Square Discriminant Analysis (PLS-DA). Elle est notamment utilisée pour contrôler les causes du chamber mismatching [45]. Cette méthode permet de trouver la root cause de plusieurs types de problèmes. cependant, pour les cas triangle up et triangle down 3.12 , les résultats obtenus avec l'algorithme PLS-DA ne sont pas les bons. C'est pour cela que nous proposons de développer une nouvelle méthode de détection de root cause utilisant random forest (RF-DA). Cette méthode pourra être utilisée pour trouver la cause des cas anormaux en leveling détectés par BP-DBSCAN. Elle sera aussi intégrée dans le logiciel d'analyse de données SpotFire pour être utilisée pour détecter les causes d'autres types d'excursions. Nous pensons que croiser les résultats des deux méthodes (PLS-DA et RF-DA) permettra une plus grande confiance.

Les excursions à expliquer sont les outliers en leveling mais aussi toutes excursions provenant du process détectées à différentes étapes (IN-LINE, DEFECTIVITY, EWS). Le format des données d'entrée est le suivant :

GOOD/BAD (LOT)	OPERATION1	OPERATION2
GOOD	EQUIPEMENT1	EQUIPEMENT1
BAD	EQUIPEMENT2	EQUIPEMENT1
BAD	EQUIPEMENT2	EQUIPEMENT1
GOOD	EQUIPEMENT1	EQUIPEMENT3

GOOD/BAD (WAFER)	OPERATION1_EQUIPEMENT1	OPERATION1_EQUIPEMENT2
GOOD	CHAMBRE1	CHAMBRE1
BAD	CHAMBRE2	CHAMBRE1
BAD	CHAMBRE2	CHAMBRE1
GOOD	CHAMBRE1	CHAMBRE3

FIGURE 3.1 – format d'entrée cause équipement (gauche) et cause chambre (droite)

La figure 3.1 décrit le format des données dans le cas où l'on pense que la cause est un équipement. Pour chaque opération, on a l'équipement sur lequel est passé chaque lot. On veut savoir quel opération/équipement, ou quelle combinaison d'opérations et d'équipements

a causé l'excursion. Si le test sur les équipements ne donne aucun résultat on peut affiner la recherche avec les chambres.

Pourquoi utiliser un algorithme d'apprentissage automatique comme analyse discriminante ? Comme PLS-DA, random forest est un algorithme multi usage. Il peut être utilisé pour de la classification ou de la régression supervisé. Avec random forest on peut obtenir un score d'importance des variables. Ce score d'importance des variables est la métrique que nous proposons d'utiliser pour créer une méthode de détection de root cause.

Quelle sont les différences entre la méthode PLS-DA et RF-DA ? Une différence importante est que PLS-DA crée des composants. Les composants sont obtenus en sélectionnant une structure sous jacente de X qui maximise la covariance entre un composant et les réponses Y. Cette méthode permet de simplifier le problème quand le nombre de variables explicatives est élevé comme c'est le cas dans l'industrie du semi-conducteur (il y a de très nombreuses causes potentielles pour une excursion). Avec la méthode RF-DA on supprime les variables progressivement. Le modèle est plus facile à interpréter sans composants.
La deuxième différence majeure est que PLS-DA est une méthode linéaire. La méthode RF-DA est non linéaire. Sachant qu'une grande partie des systèmes physiques sont non linéaires, c'est un avantage pour la méthode RF-DA qui n'est pas limitée aux modèles linéaires. Les deux méthodes pourront être utilisées de manière complémentaire. De cette manière les résultats obtenus avec une des deux méthodes pourront être confirmés en utilisant l'autre méthode.

3.2 Contributions

Les travaux sur random forest discriminant analysis ont été faits avec pour objectif une industrialisation rapide car la faisabilité était connue. Une seule méthode a été développée et testée sur différents jeux de données :

- Triangle up, triangle down : données récupérées après détection d'outliers récurrents en leveling
- Test paramétrique : un problème de drift a été détecté en fin de processus
- Inline : données récupérées après détection de hors spécifications en leveling

Le code sera intégré dans SpotFire et pourra être utilisé par tout les employés de STMicrolectronics Crolles.

3.3 Présentation de PLS-DA

L'algorithme PLS a premièrement été introduit comme algorithme de régression puis il est devenu un algorithme de classification : PLS-DA. L'algorithme PLS-DA peut être utilisé pour construire des modèles prédictifs ou descriptifs mais aussi pour effectuer des analyses discriminantes de sélection de variables [46]. C'est cette utilisation qui est intéressante pour la détection de root cause. En déterminant les variables ou combinaisons de variables qui expliquent le mieux une réponse mauvaise (excursion) dans un jeux de données on doit trouver la cause ou combinaison de causes qui explique cette excursion.

PLS-DA transforme les prédicteurs et les réponses en composantes sous jacente individuelles. Les composants sont déterminés de manière a décrire le jeux de données explicatives X et a maximiser la corrélation entre les composants X (variables explicatives) et Y (réponse).

Soit X les données d'entrée $N \times J$ et y les données de sortie $N \times 1$. On commence par calculer les poids w, $(J \times 1)$:

$$w = X'y$$

Puis on détermine les x-scores t, $(N \times 1)$ comme :

$$t = \frac{Xw}{\sqrt{\sum w^2}}$$

Les matrices X-loading et Y-loading, p $(1 \times J)$ et q (1×1) sont ensuite calculées :

$$p = \frac{t'X}{\sqrt{\sum t^2}}$$

$$q = \frac{y't}{\sqrt{\sum t^2}}$$

Les résidus res_X, $(N \times J)$ et res_y, $(N \times 1)$ découlant de la part de variance non expliquée par les composants créés sont égaux à :

$$res_X = X - tp$$

$$res_y = y - tq$$

Les résidus res_X et res_y sont ensuite utilisés comme données d'entrées et données de sortie pour construire les prochains composants. On répète le procédé jusqu'à ce qu'il n'y ait plus de résidus. Ci dessous un schéma récapitulatif :

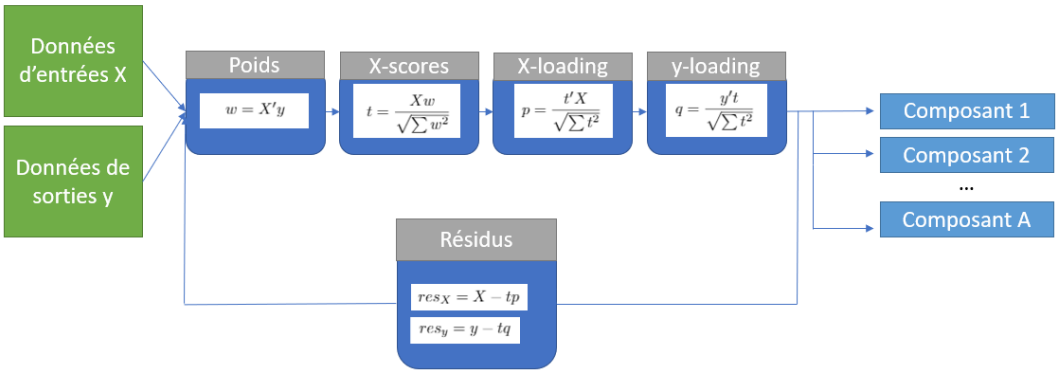


FIGURE 3.2 – PLS-DA : création des composants

Après la phase de création de composants, la construction du modèle de prédiction commence par l'estimation des coefficients de régression b :

$$b = w(pw)^{-1}q$$

qui nous donne pour tous les composants la matrice de coefficient B, $(J \times A)$.
Pour un jeu de test X_{test} les valeurs prédites $\hat{Y}$, $(N \times A)$ sont égales à :

$$\hat{Y} = X_{test}B$$

3.4 Travaux connexes

Les méthodes d'analyse de root cause sont d'abord des méthodes statistiques comme PLS-DA, une des méthodes utilisées actuellement à STMicroelectronics. Une autre technique statistique est le Chi2 [47] qui peut être utilisé sur des cas simples . L'utilisation d'un algorithme de classification pour la détection de root cause n'est pas courant mais il existe plusieurs exemples. Un exemple est l'utilisation d'un raisonnement bayésien [48] pour l'identification de la root cause.

Plus spécifiquement, random forest a aussi été utilisé comme algorithme de détection de root cause. Dans [49] l'objectif est de détecter les causes de problèmes de performance dans des datacenters. L'article souligne les capacités scaling de l'algorithme, c'est à dire qu'il reste performant sur un jeu de données de grande taille et avec beaucoup de paramètres. C'est un des avantages qui nous intéressent car les jeux de données à traiter pour la détection de root cause à STMicroelectronics peuvent contenir plusieurs milliers de paramètres.

Dans un article très récent [50], les différentes métriques de sélection de feature utilisées par les algorithmes de machine learning sont comparées pour la détection de root cause. Les différentes approches décrites sont les méthodes filtres, les méthodes intégrées (Le gain d'information intégré à Random forest), les méthodes wrappers et les réseaux bayésiens. Les tests effectués par [50] semblent indiquer que le gain d'information est la meilleure métrique pour la détection de root cause. Le gain d'information est justement la métrique utilisée par random forest 1.6. Nous avons choisit d'utiliser random forest et non pas simplement la métrique du gain d'information pour pouvoir évaluer la pertinence des résultats (voir les sections méthode et résultats).

3.5 Méthode

La méthode fonctionne avec les VIP (variable importance) de random forest. Ci-dessous un flowchart résumé de l'algorithme :

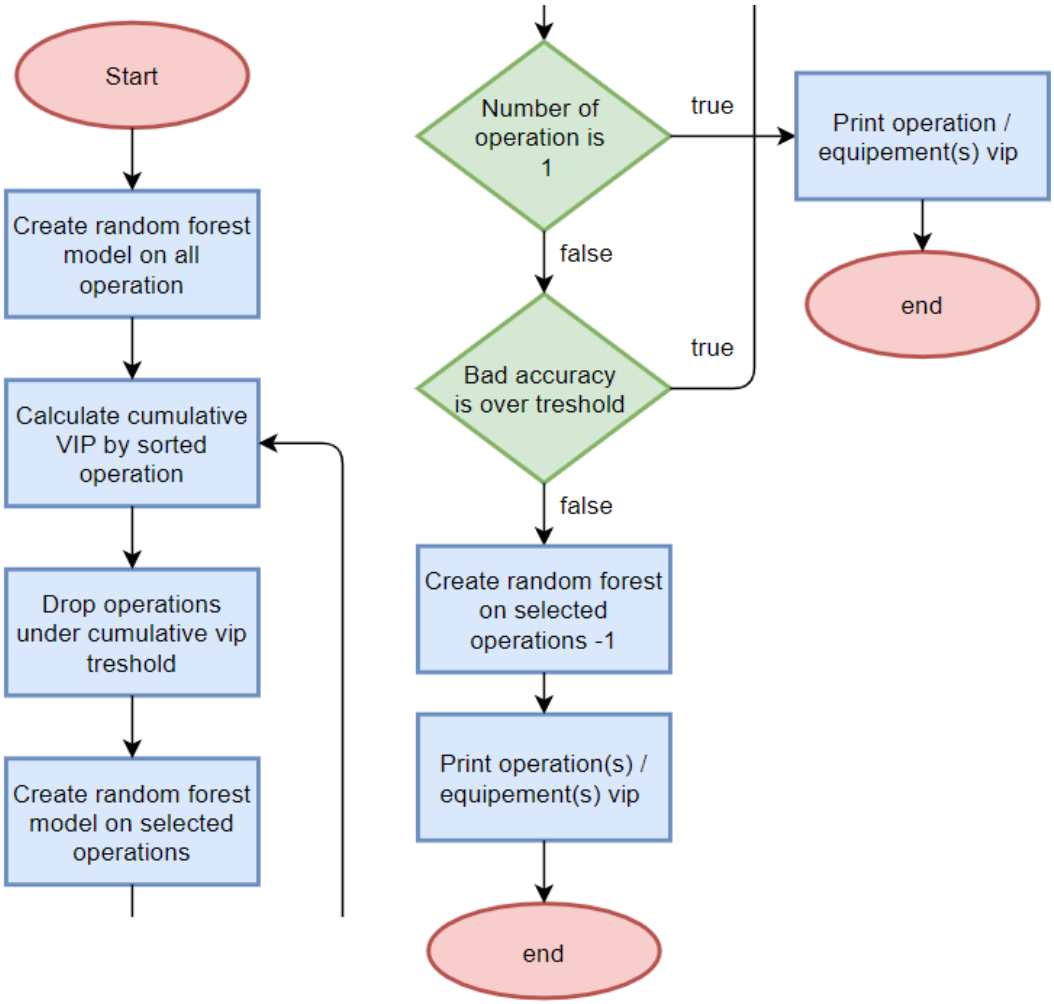


FIGURE 3.3 – Random Forest Discriminant Analysis flowchart

Le flowchart 3.3 décrit l'idée générale derrière la méthode Random Forest Discriminant Analysis. Le premier modèle random forest est créé avec toutes les opérations. Ensuite on fait un cumul des VIP opération_équipement par opération. La valeur cumulée du VIP permet de sélectionner les opérations qui expliquent le mieux la séparation entre les deux classes (good et bad). Ce sont les opérations qui expliquent le mieux l'excursion. Une fois qu'il ne reste plus qu'une seule opération, on affiche les VIP de chaque équipement de l'opération pour trouver la cause (VIP le plus élevé). Si le modèle a n opérations a une précision trop basse on crée le modèle à $n-1$ opérations. On affiche les VIP pour trouver la combinaison d'opérations et d'équipements qui a causé l'excursion.

Cette méthode est une méthode algorithmique. Pour mieux la comprendre voici une partie de l'algorithme en python (certaines méthodes sont spécifiques à ce langage et des librairies python sont utilisées). Les librairies utilisées sont pandas (pd), numpy (np) et scikitlearn. Toutes les procédures ne sont pas détaillées.

Algorithm 5 get_vip

```
1: procedure GET_VIP
2:     importances = forest.feature_importances_
3:     indicies = np.argsort(importance)[ : :-1]
4:     operations = []
5:     vip = []
6:     print("Feature ranking :")
7:     for f in range(x.shape[1]) do
8:         print(x.columns[indices[f]] + " : " + str(importances[indices[f]]))
9:         operations.append(x.columns[indices[f])
10:        vip.append(importances[indices[f]])
11:    vip_df = pd.DataFrame('OPERATION' : operations, 'VIP' : vip)
12:    return vip_df
   )
```

Algorithm 6 get_vip_operation_sum

```
1: procedure GET_VIP_OPERATION_SUM(vip_df)
2:     vip_gp = vip_df.groupby(['OPERATION'])
3:     vip_gp = pd.DataFrame('VIP_SUM' : vip_gp['VIP'].apply(np.sum)).reset_index()
```

Algorithm 7 find_root_cause

```
1: procedure FIND_ROOT_CAUSE
2:     X = pd.get_dummies(df, prefix_sep = " ;")
3:     rfda = RandomForestDA(nb_trees = 100, max_depth = 4)
4:     rf = rfda.fit(X,y)
5:     preds = rf.predict(X)
6:     vip = rf.get_vip()
7:     vip_ope = get_vip_by_operations_sum(vip)
8:     cols = vip_ope['OPERATION']
9:     treshold = cumulative_vip_treshold
10:    min_bad_acc = rf.bad_accuracy(y,preds) - bad_accuracy_lower_limit_reductor
11:    while rf.bad_accuracy(y,preds) >= min_bad_acc and len(cols) > 1 do
12:        vip_ope_sorted = vip_ope.sort(vip_ope.sort(by = ['VIP_SUM'],descending))
13:        vip_ope_sorted['CUMUL_VIP_SUM] = vip_ope_sorted['VIP_SUM].cumsum()
14:        for i in range(0,len(vip_ope_sorted)) do :
15:            if vip_ope_sorted.iloc[i]['CUMUL_VIP_SUM] > treshold then
16:                idx = i
17:                break
18:        vip_ope_sorted = vip_ope_sorted[ :(idx+1)]
19:        old_cols = copy.copy(cols)
20:        cols = vip_ope_sorted['OPERATION']
21:        if len(old_cols)==len(cols) then
22:            treshold = treshold - 0.05
23:        new_X = pd.get_dummies(df[cols], prefix_sep = " ;")
24:        rfda = RandomForestDA(100, 4)
25:        rf = rfda.fit(new_X, y)
26:        preds = rf.predict(new_x)
27:        vip = rf.get_vip()
28:        vip_ope = get_vip_by_operations_sum(vip)
29:    if rf.bad_accuracy(y,preds < min_bad_acc then
30:        new_X = pd.get_dummies(df[old_col, prefix_sep = " ;")
31:        rfda = RandomForestDA(100, 4)
32:        rf = rfda.fit(new_X, y)
33:        preds = rf.predict(new_x)
34:        vip = rf.get_vip()
35:        vip_ope = get_vip_by_operations_sum(vip)
36:
```

La première procédure est celle permettant de calculer le VIP par opération. C'est une procédure classique en machine learning.

La procédure *get_vip_by_operation_sum* est un simple group by. On choisit de regrouper les VIP des équipements par opération avec la somme. De cette manière la somme de toutes les VIP groupées est bien 1.

La procédure find_root_cause est le coeur de l'algorithme. Pour comprendre l'algorithme des captures d'écrans résumant une utilisation de cet algorithme sont présentées. L'exécution de l'algorithme présentée correspond à une excursion détectée au cours du process. Le problème est plus simple quand l'excursion est détectée au cours du process car il est possible d'éliminer d'office toutes les opérations ultérieures à celle où la détection a été effectuée.

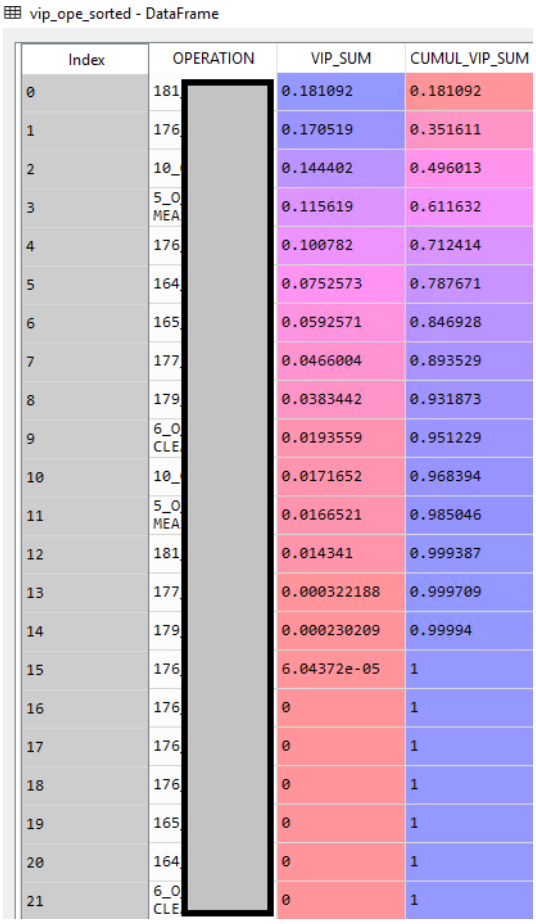
L'algorithme commence par fixer les valeurs des paramètres *trehsold* et *min_bad_acc*. Le paramètre *treshold* détermine le pourcentage de VIP cumulé à partir duquel on supprime des opérations 3.6 pour réduire la complexité du modèle. Plus le paramètre treshold est petit plus la réduction du nombre d'opérations sera rapide. Réduire le nombre d'opérations lentement permet de tester de plus nombreux modèles. Le paramètre *minc_bad_acc* est un paramètre de sortie de boucle. Pour chaque modèle généré, la précision (pour les individus bad uniquement) est calculée. Si un modèle a une précision pour les bad en dessous de *minc_bad_acc* alors la boucle est arrêtée et on revient au modèle précédent comme solution. C'est donc un paramètre très important. Pour changer la valeur de *min_bad_acc* on fixe la valeur de *bad_accuracy_lower_limit_reductor*. Ce paramètre va être soustrait à la précision pour les bad du modèle avec toutes les variables. La précision du modèle avec toutes les variables est considéré comme la précision maximale. On choisit l'écart du modèle final à cette précision avec *bad_accuracy_lower_limit_reductor*. Une valeur élevée de *bad_accuracy_lower_limit_reductor* réduira les chances de sortie de boucle à chaque nouveau modèle et assurera que le modèle final ne contienne qu'une seule variable. En contrepartie le modèle final pourra être beaucoup moins précis que le modèle avec toutes les variables. Il est important de comprendre que l'objectif n'est pas de construire un modèle prédictif mais bien de faire ressortir les causes potentielles d'une excursion. La précision du modèle est un indicateur utilisé pour mesurer la pertinence du résultat. Une variable importante d'un modèle précis a plus de chance d'être la cause qu'une variable importante d'un modèle moins précis. Si il est impossible de construire un modèle avec une précision relativement élevée cela peut vouloir dire que la cause de l'excursion ne se trouve pas dans les données. Après avoir fixé les paramètres importants, un modèle random forest avec toutes les variables est créé. Chaque variable correspond à un couple opération, équipement. L'importance de chaque variable (VIP) est calculée pour puis agrégée par opération (somme) 3.4.

	VIP
OPERATION1;EQUIPEMENT1	0.12049
OPERATION2;EQUIPEMENT1	0.0969216
OPERATION3;EQUIPEMENT2	0.0860618
OPERATION2;EQUIPEMENT3	0.0756708
OPERATION4;EQUIPEMENT4	0.0565229

OPERATION	VIP_SUM
OPERATION1	0.18642
OPERATION2	0.166626
OPERATION3	0.149783
OPERATION4	0.125382

FIGURE 3.4 – VIP par opérations ;équipements (gauche) et VIP par opérations(droite)

Les opérations sont triées par VIP décroissant et le cumul du VIP est calculé 3.5. Le cumul du VIP est égal à 1. On peut voir que pour cette exécution de la méthode, il y a 4 opérations représentants 50% du cumul du VIP. L'algorithme supprime toutes les autres opérations.

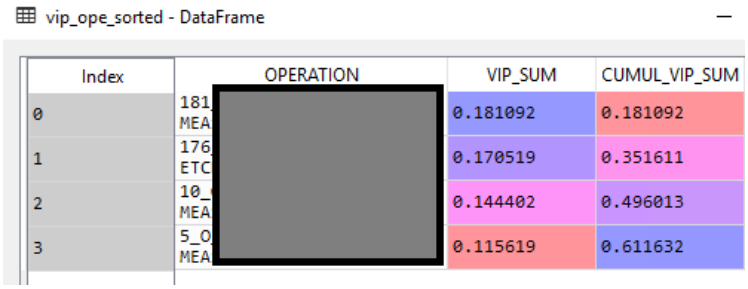


vip_ope_sorted - DataFrame

Index	OPERATION	VIP_SUM	CUMUL_VIP_SUM
0	181	0.181092	0.181092
1	176	0.170519	0.351611
2	10_	0.144402	0.496013
3	5_O MEA	0.115619	0.611632
4	176	0.100782	0.712414
5	164	0.0752573	0.787671
6	165	0.0592571	0.846928
7	177	0.0466004	0.893529
8	179	0.0383442	0.931873
9	6_O CLE	0.0193559	0.951229
10	10_	0.0171652	0.968394
11	5_O MEA	0.0166521	0.985046
12	181	0.014341	0.999387
13	177	0.000322188	0.999709
14	179	0.000230209	0.99994
15	176	6.04372e-05	1
16	176	0	1
17	176	0	1
18	176	0	1
19	165	0	1
20	164	0	1
21	6_O CLE	0	1

FIGURE 3.5 – VIP par opérations triés par opérations les plus importantes décroissantes avec cumul du VIP

Le tableau 3.6 est le tableau après suppression des opérations avec faible VIP. C'est avec ces opérations que l'algorithme va créer un deuxième modèle random forest.



vip_ope_sorted - DataFrame

Index	OPERATION	VIP_SUM	CUMUL_VIP_SUM
0	181 MEA	0.181092	0.181092
1	176 ETC	0.170519	0.351611
2	10_ MEA	0.144402	0.496013
3	5_O MEA	0.115619	0.611632

FIGURE 3.6 – VIP par opérations triés par opérations les plus importantes décroissantes avec cumul du VIP après première sélection des opérations les plus importantes

L'algorithme crée un deuxième modèle en prenant en compte uniquement les opérations sélectionnées puis les VIP sont calculés à nouveau pour ce modèle 3.7.

vip_ope_sorted - DataFrame

Index	OPERATION	VIP_SUM	CUMUL_VIP_SUM
0	176_	0.300737	0.300737
1	181_	0.282566	0.583303
2	10_	0.255081	0.838384
3	5_O MEA	0.161616	1

FIGURE 3.7 – VIP par opérations triées par opérations les plus importantes décroissantes avec cumul du VIP pour le deuxième modèle

De la même manière que pour la première itération, on écarte les opérations avec un cumul du VIP inférieur à 50%. Cette valeur correspond au seuil du VIP. C'est un paramètre de l'algorithme qui sera automatiquement réduit si l'algorithme n'arrive pas à diminuer le nombre d'opération.

vip_ope_sorted - DataFrame

Index	OPERATION	VIP_SUM	CUMUL_VIP_SUM
0	176 ETC	0.300737	0.300737
1	181 MEA	0.282566	0.583303

FIGURE 3.8 – VIP par opérations triées par opérations les plus importantes décroissantes avec cumul du VIP pour le deuxième modèle après sélection des opérations les plus importantes

La figure 3.9 correspond au tableau du cumul des VIP pour l'avant dernier modèle. Ici c'est l'opération 181 qui va être sélectionnée.

vip_ope_sorted - DataFrame

Index	OPERATION	VIP_SUM	CUMUL_VIP_SUM
0	181_ MEA	0.547522	0.547522
1	176_	0.452478	1

FIGURE 3.9 – VIP par opérations triés par opérations les plus importantes décroissantes avec cumul du VIP pour le troisième modèle

Après création du dernier modèle on affiche les VIPS de chaque couple opération / équipement pour déterminer quel est l'équipement qui a causé l'excursion 3.10. L'équipement xxxx05 a un vip très élevé de 0.83. C'est cet équipement qui est la cause du problème. Pour estimer la pertinence du résultat on va vérifier la précision du modèle. La précision pour les bad est mauvaise : 0.4 seulement alors que la précision globale est de 0.82. Ceci souligne la raison pour laquelle il faut utiliser la précision des bad comme condition de sortie et non pas la précision globale. L'objectif étant de déterminer la cause de l'excursion et donc des lots / wafers bad, il est logique d'utiliser la précision bad plutôt que la précision globale. Vu que la précision est mauvaise l'algorithme va automatiquement revenir au modèle précédent qui était un modèle à deux variables.

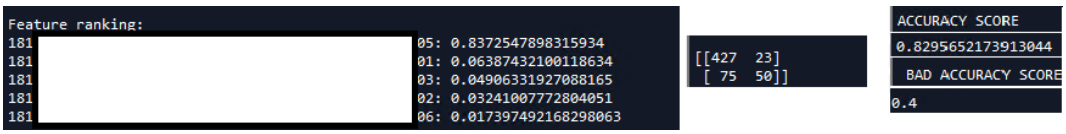


FIGURE 3.10 – Feature ranking du dernier modèle (une seule opération) avec précision

La figure 3.11 donne les VIP des couples opération / équipement pour le modèle à deux opérations. L'information que ce tableau nous donne est que l'équipement 005 cause aussi des excursions pour l'opération 176 et pas que pour la 181 comme nous laissait penser le modèle à une seule opération. Ce résultat est pertinent car la précision du modèle pour les bads est de 0.8. Le résultat est logique, c'est le même équipement sur deux opérations qui cause les excursions.

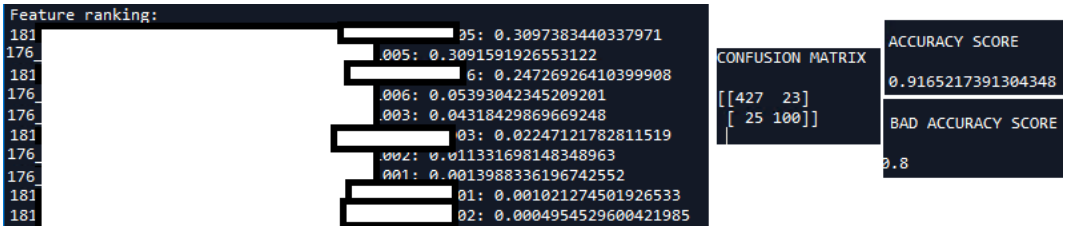


FIGURE 3.11 – Feature ranking de l'avant dernier modèle (deux opérations) avec précision

Cet exemple d'exécution d'algorithme permet de comprendre l'intérêt des paramètres *treshold* et *min_bad_accuracy* sur la vitesse de réduction du problème et la sortie de boucle. Un autre paramètre important de l'algorithme est la précision du modèle pour les bads. En effet l'importance des variables seules n'est pas pertinente car la somme sera toujours égale à 1 1.6 même si le modèle est faux. Le tableau d'importance des variables (VIP) fait sens uniquement quand la précision du modèle pour les bads est bonne. En réduisant progressivement le nombre de variables, on trouve le minimum de features pour avoir un modèle pertinent en terme de précision. Ce nombre d'attributs correspond au nombre d'opérations root cause.

3.6 Résultats

Le premier test effectué avec RF-DA porte sur des données de leveling. Des excursions récurrentes ont été détectées au cours du process. La figure 3.12 montre les excursions qui sont répétées sur plusieurs lots. Ces deux signatures semblent être les mêmes avec une orientation différente. Si une ligne est tracée entre les zones aberrantes, les signatures peuvent être vues comme un triangle pointant vers le haut et un triangle pointant vers le bas.

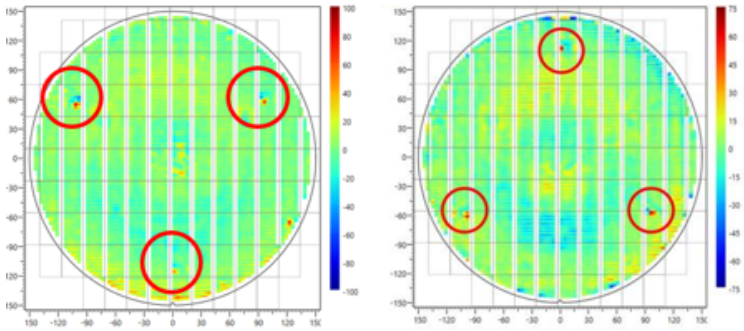


FIGURE 3.12 – TRIANGLE DOWN et TRIANGLE UP

L'algorithme Random Forest DA a permis d'obtenir un modèle d'une précision de 100%. La cause était une mono-variable (une seule opération cause). Une opération et trois machines étaient responsables de la signature. Deux machines sont responsables de la signature TRIANGLE DOWN et une autre est responsable de la signature TRIANGLE UP.

Ce cas était trivial et les mêmes résultats ont été obtenus avec PLS-DA et un test du khi2.

Pour le deuxième cas, il faut trouver la cause d'un défaut électrique détecté en EWS. L'EWS étant la dernière étape de vérification de la plaque, le cas où un problème y est détecté est le plus complexe car toutes les étapes du process sont potentiellement la cause. Cela fait plus de 500 opérations à tester. C'est pour ce type de cas que la diminution progressive du nombre d'opérations dans le modèle est utile. En effet les résultats du modèle à 500 opérations peuvent être dégradés par le nombre élevé de features. Le fait de réduire progressivement la taille du modèle en vérifiant la précision à chaque réduction du nombre de variables indique le nombre de causes du problème. C'est d'autant plus important quand on a un grand nombre de variables.

vip_ope_sorted - DataFrame

Index	OPERATION	VIP_SUM	CUMUL_VIP_SUM
0	1	0.266314	0.266314
1	2	0.173427	0.439741
2	3	0.135828	0.575568
3	4	0.086955	0.662523

```
[[502   8]
 [ 18 230]]
ACCURACY SCORE
0.9656992084432717
 BAD ACCURACY SCORE
0.9274193548387096
```

FIGURE 3.13 – VIP par opérations triées par opérations les plus importantes décroissantes avec cumul du VIP pour le premier modèle (4 premières opérations)

Dès la première boucle 3.13 on a 3 opérations qui se démarquent avec un VIP élevé.

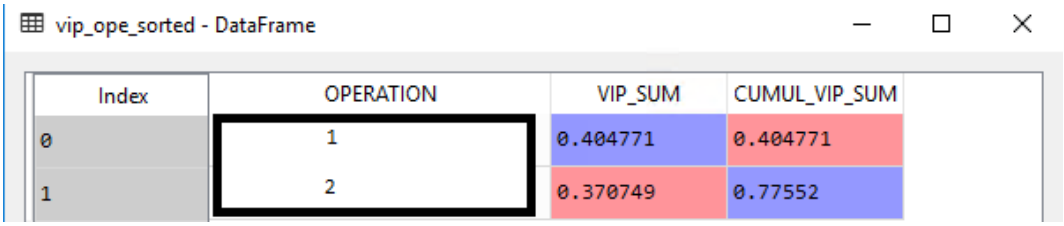


FIGURE 3.14 – VIP par opérations triés par opérations les plus importantes décroissantes avec cumul du VIP pour le dernier modèle

Dans le dernier modèle ce sont les deux premières opérations du premier modèle en terme de VIP qui ont été sélectionnées.

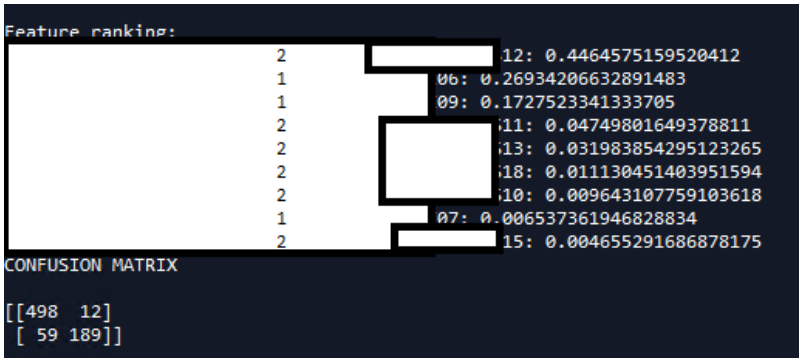


FIGURE 3.15 – Feature ranking dernier modèle

La figure 3.15 donne les features ranking en détail (par couple opération équipement) pour le dernier modèle. Le premier équipement en terme de VIP est un équipement de la deuxième opération. Les deuxième et troisième équipements sont des équipements de la première opération. Il est difficile de déterminer la root cause dans ce cas sans informations supplémentaires. La root cause pourrait être multiple ou être un seul équipement. Pour trancher, il faut utiliser des informations fonctionnelles comme l'ordre des opérations. Ceci n'est pas le rôle de l'algorithme mais de l'utilisateur qui maîtrise le processus de fabrication. Pour ce cas, la root cause validée est un problème sur la machine 06 à l'opération 1.

Le troisième essai de la méthode RF-DA est celui présenté dans la partie précédente comme exemple d'exécution. Les résultats obtenus sont les mêmes qu'avec la méthode PLS-DA. Pour obtenir des résultats plus surs il est possible de combiner les deux méthodes (RF-DA, PLS-DA) et de croiser les résultats.

3.7 Implémentation de la méthode RF-DA

Le développement de la méthode RF-DA a été motivée par plusieurs demandes. La première est la détection des root causes d'excursions en leveling. C'est donc la suite logique des recherches en détection d'outliers leveling. La deuxième demande provient de l'équipe en charge du développement d'applications SpotFire qui souhaitait proposer une nouvelle méthode plus simple d'utilisation, plus claire et plus performante pour la détection de root cause. La méthode RF-DA sera utilisée comme méthode de détection de root cause générale (pas spécifique au leveling ou autre mesure) dans SpotFire par de nombreux utilisateurs. C'est avec pour objectif que la méthode soit simple à utiliser que le développement a été fait. Pour que la méthode soit simple à utiliser il faut que les résultats soit compréhensibles. Or, tous les utilisateurs ne sont pas des statisticiens. Ils ne sont pas tous familiers avec random forest et les arbres de décision. Les détails des arbres peuvent être intéressants pour une analyse plus approfondie des résultats. Pour ceci un outil de traduction de forêt a été développé. Cette procédure va traduire tout les noeuds de tous les arbres de la forêt et les moyenner pour obtenir le tableau suivant :

ez_forest - DataFrame

Index	0	1	2	3
19	0.763722	des individus évalués par	3C06	sont mauvais
11	0.75743	des individus évalués par	iS12	sont mauvais
9	0.707307	des individus évalués par	iS11	sont mauvais
5	0.69934	des individus évalués par	17	sont mauvais
21	0.15	des individus évalués par	C07	sont mauvais
17	0.140567	des individus évalués par	i18	sont mauvais
1	0.134342	des individus évalués par	05	sont mauvais
13	0.106043	des individus évalués par	i13	sont mauvais
3	0.0968066	des individus évalués par	10	sont mauvais
15	0.0739837	des individus évalués par	iS15	sont mauvais
23	0.060586	des individus évalués par	3C09	sont mauvais
7	0.00177305	des individus évalués par	i10	sont mauvais

FIGURE 3.16 – Partie de forêt traduite

Ce tableau est compréhensible même sans connaissance en machine learning. Il donne un aperçu des facteurs déterminants des éléments mauvais du modèle. Le tableau est obtenu avec toutes les feuilles de tous les arbres de la forêt. Voici un exemple pour une feuille.

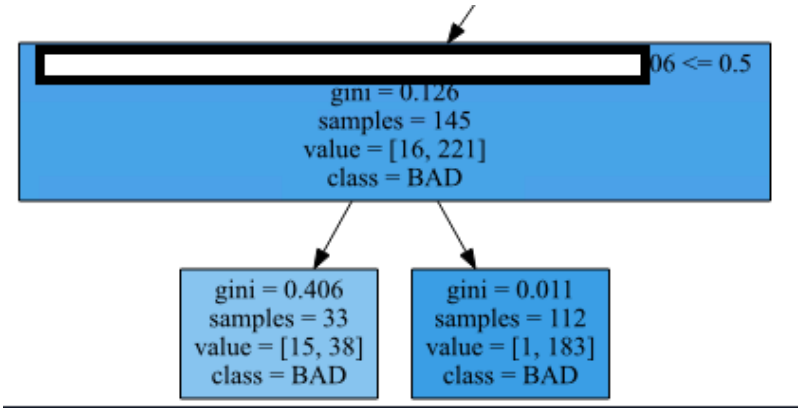


FIGURE 3.17 – Exemple de feuille : équipement 06

ftt - DataFrame

Index	Columns	Values	Good	Bad	good_percent	bad_percent
22	3C09	0	2977	12702	0.189872	0.810128
12	:13	0	2647	8968	0.227895	0.772105
19	:06	1	4662	15069	0.236278	0.763722
11	12	1	3485	10882	0.24257	0.75743
9	:11	1	701	1694	0.292693	0.707307
5	17	1	5374	12500	0.30066	0.69934
20	_ :07	0	943	1359	0.409644	0.590356
6	- _ _ _ _ . :10	0	2917	3424	0.460022	0.539978

FIGURE 3.18 – Exemple d'arbre pré-traduit

Pour ajouter cette feuille au tableau on ajoute 183 à la troisième ligne la colonne Bad et 1 à la troisième ligne de la colonne Good. Les individus qui vont au noeud à droite sont ceux qui sont passés sur l'équipement du noeud père. Ceux qui vont à gauche sont ceux qui ne sont pas passés par cet équipement. Cette information est donné par la colonne values. Pour finir la traduction de cette feuille il faudrait aussi compléter la ligne inverse de la ligne 3. La ligne inverse est la ligne avec Values = 0, les individus qui ne sont pas passés par cet équipement (équipement 06). En effectuant la même opération pour toutes les feuilles de tous les arbres on peut obtenir le tableau 3.16 qui permet à un utilisateur qui ne connaît pas les arbres de décision d'avoir un aperçu du détail de la forêt. La lisibilité des résultats est importante pour éviter la boite noire. Cette méthode sera utilisée comme outil d'aide à la décision, il faut donc que les résultats soit les plus clairs possible pour que l'interprétation de ceux ci soit bonne.

La méthode Random Forest Discriminant Analysis va être intégrée dans une application SpotFire qui sera accessible à tous les employés de STMicroelectronics.

3.8 Conclusion

Random Forest Discriminant Analysis (RF-DA) est une méthode de détection de root cause facile à utiliser et à interpréter. Tout au long du développement, la simplicité et compréhensibilité étaient les deux points les plus importants. L'algorithme peut être lancé avec les paramètres de base et détecter la root cause. Il est aussi possible de modifier les paramètres pour un résultat plus précis. Les performances semblent être très bonnes d'après les trois cas de test. En effet pour les trois cas RF-DA a détecté la root cause. Les performances sont meilleures que l'algorithme intégré dans l'application d'analyse de données SpotFire et au moins aussi bonnes que PLS-DA. L'intégration de la méthode PLS-DA à SpotFire est compliquée, c'est pour cela que RF-DA sera disponible via l'application d'analyse de données de STMicroelectronics, SpotFire. De cette manière tous les employés ST pourront utiliser la méthode RF-DA. Pour l'instant les analyses de root causes ne sont pas assez efficace. Il est attendu que RF-DA soit la méthode de détection de root cause principale à STMicroelectronics. RF-DA peut être utilisé qu'importe le contexte sans modification de code.

Ce travail est un exemple d'adaptation d'un algorithme de machine learning à un besoin industriel. L'algorithme choisi, random forest, est facile à utiliser et n'est pas une boite noire. Grâce à ces caractéristiques il est possible de développer une méthode industrialisable.

Chapitre 4

Intégration de connaissances

Les méthodes de classification automatique permettent de créer un modèle sans à priori sur la structure des données. C'est l'avantage des méthodes non paramétriques. Cependant dans de nombreux cas de classification dans l'industrie, un expert possède des connaissances sur la forme des données, telle que la monotonie, la convexité, concavité, super-modularité, uni-modalité ou la symétrie de celles ci. L'intégration de connaissances est un sujet qui propose de prendre en compte ce savoir expert et l'ajouter comme contrainte des algorithmes de classification supervisée pour créer un modèle semi paramétrique. Les figures suivantes sont un exemple de la résolution d'un problème jouet par un algorithme avec intégration de monotonie.

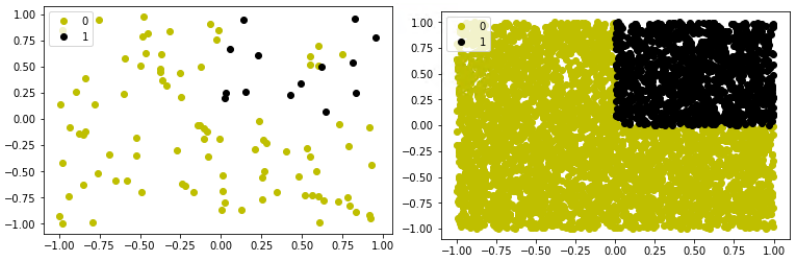


FIGURE 4.1 – Entraînement et test

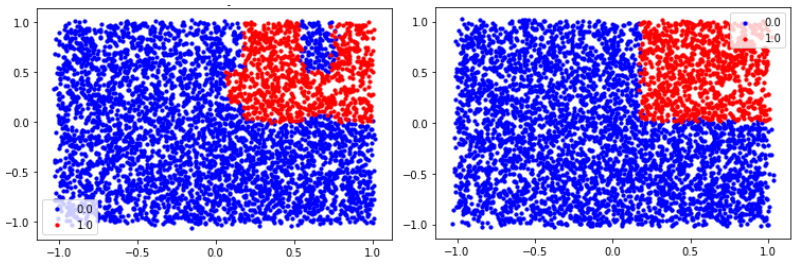


FIGURE 4.2 – Prédiction standard et prédiction monotone

Selon [51], un désavantage important des méthodes non-paramétriques est qu'elles sont très sensibles à des erreurs dans les observations. C'est une des faiblesses qui a motivée les recherches sur l'intégration de connaissances. En effet, les données industrielles dans le milieu du semi-conducteur peuvent contenir des erreurs de mesures, des outliers et sont de plus bruitées. Une des raisons à cela est que les mesures sont faites à l'échelle nanoscopique.

L'intégration de connaissances dans un algorithme de prédiction basé sur les données a

été étudiée pour différents types de connaissance et différents algorithmes. Par exemple, pour l'algorithme support vector machine (SVM), différents types de contraintes ont été imposées : dans [52], la connaissance à priori est que certaines zones polyhédrales de l'espace appartiennent à l'une ou l'autre des classes. Cette connaissance est ajoutée en reformulant un classificateur SVM linéaire, ce qui résulte en un programme linéaire solvable efficacement. Dans la méthode présentée par [53] la convexité est la connaissance intégrée à l'algorithme. Cette connaissance est intégrée par une série de matrice linéaire de contraintes qui est imposée à chaque point.

Une autre connaissance étudiée est la monotonie de l'objectif par rapport aux variables explicatives. La création d'un SVM monotone semi-paramétrique à l'aide de la dérivation primal dual est décrite dans [54].

Pour les méthodes de type Random Forest, un problème typique d'intégration de connaissances est de produire des classificateurs monotones en certaines caractéristiques [55, 56]. C'est à dire des classificateurs qui contraignent certains attributs à une relation monotone avec y. Dans sa version positive, la monotonie d'une variable x signifie que y ne peut qu'augmenter si x augmente (avec les autres variables fixes). Ces solutions utilisent l'algorithme Random Forest et le contraignent mathématiquement pour le rendre monotone. Une des méthodes décrites consiste a contraindre chaque arbre de décision au niveau des feuilles.

Toutes ces recherches proposent différentes méthodes mathématiques pour l'intégration de connaissances. Nous avons étudié deux méthodes mathématique et algorithmique adaptables à différentes connaissances. La première méthode modifie le mécanisme de vote de random forest et la deuxième consiste à ne sélectionner que les arbres qui respectent une contrainte dans la forêt.

4.1 Problématique

Nous avons choisi l'algorithme Random Forest car il permet une intégration de connaissances intuitive ; c'est un algorithme relativement simple qui est facile à modifier. C'est aussi un algorithme connu pour ses bonnes performances. Les formes de données que nous proposons d'étudier sont la symétrie et la monotonie. Le choix de la symétrie est motivée par les capteurs permettant la détection de défauts nanoscopiques par faisceau lumineux qui renvoie un résultat symétrique 4.3.

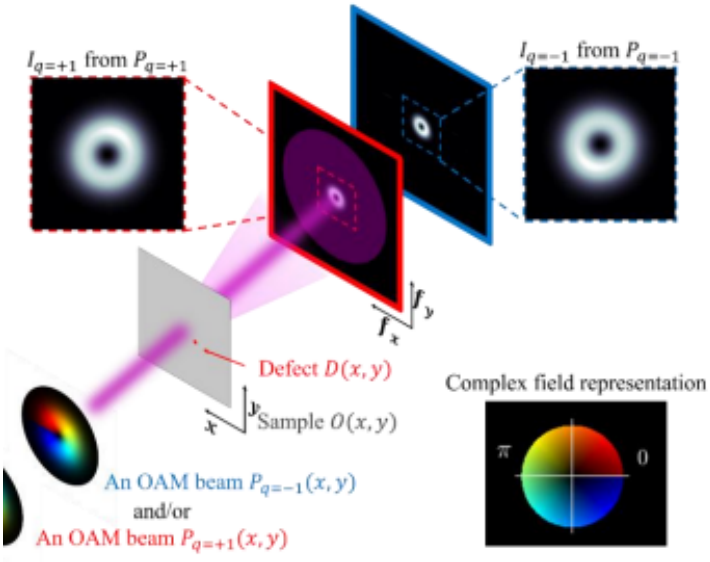


FIGURE 4.3 – Signal symétrique de [19]

L'objectif est de contraindre Random Forest pour forcer la symétrie du modèle. Nous proposons d'étudier théoriquement l'intégration de la symétrie axiale et symétrie radiale à l'aide de jeux de données générés. Ces travaux pourront ensuite être appliqués à des cas réels comme la détection de défauts nanoscopiques par faisceau lumineux.

La monotonie positive entre un objectif y et une variable explicative x signifie que pour $x_j > x_i$, $y_j \geq y_i$; y ne peut qu'augmenter pour des valeurs de x plus grandes. Comme pour la symétrie, l'objectif est de modifier Random Forest pour contraindre le modèle à respecter la monotonie. Les tests seront effectués sur les jeux de données libres utilisés par [56] pour permettre une comparaison des résultats. L'algorithme modifié doit permettre de traiter des cas multi-variés avec monotonie positive et négative. Pour le cas multi-varié, la monotonie positive de y par rapport à X_i signifie que pour toutes les autres variables fixes, y ne peut qu'augmenter pour des valeurs de X_i plus grandes. La version monotone de random forest devrait éviter d'apprendre de mauvaises règles qui ne respectent pas des contraintes de monotonie connues, par exemple un modèle qui apprendrait des règles ne respectant pas la monotonie entre la température et la pression d'un gaz. L'apprentissage de fausses règles peut être du au sur-apprentissage, aux données bruitées, aux outliers ou à un petit échantillon d'entraînement. Random forest monotone doit être plus robuste face à ces problématiques. Nous ne proposons pas de contraindre le modèle à 100% mais de manière "douce". Cela veut dire que le modèle doit tendre vers la monotonie. De cette manière le classificateur non paramétrique a encore une grande influence : nous avons pour objectif de créer un classificateur semi-paramétrique.

4.2 Contributions

Les recherches sur le thème de l'intégration de connaissances sont théoriques et ne sont pas motivées par un besoin concret à STMicroelectronics. Les méthodes développées pourront peut être être utilisées à l'avenir. Voici une liste des différents développements :

- Random forest symétrique modification de probabilité : testé sur jeux de données générées
- Random forest symétrique dropouts : testé sur jeux de données générées
- Random forest symétrie axiale modification de probabilité : testé sur jeux de données générées
- Random forest symétrie axiale dropouts : testé sur jeux de données générées
- Random forest monotone dropouts : testé sur jeux de données générées et jeux de données libres
- Random forest régression monotone dropouts : testé sur jeux de données générées, jeux de données libres et jeux de données STMicroelectronics

La méthode random forest régression monotone avec dropouts a obtenu des résultats intéressants sur les jeux de données STMicroelectronics.

4.3 Modification de probabilité

Étant donné un point x, chaque arbre de random forest fait une prédiction $\hat{y}$ pour y. Avec le vote de tous les arbres de la forêt on obtient une probabilité pour le point d'appartenir à une classe. En modifiant le mécanisme de vote et donc la probabilité, il est possible de contraindre la symétrie.

4.3.1 Symétrie axiale

Pour qu'il y ait symétrie axiale, les prédictions $r(x_1)$ et $r(x_2)$ doivent être les même pour $x_1 = |x_2|$.

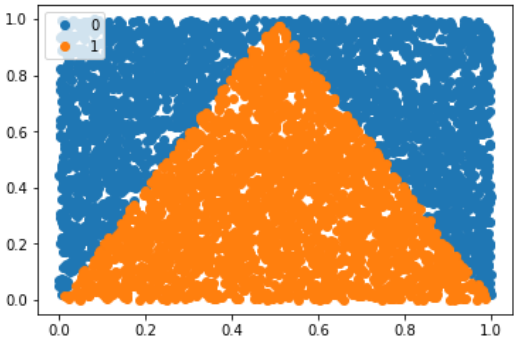


FIGURE 4.4 – Symétrie axiale exemple de jeu de données de test générées

La classe cible regroupe tous les points à l'intérieur du triangle. On veut construire une forêt d'arbres de décision où la symétrie par rapport à l'axe partant de la base du triangle à son sommet est exploitée. L'objectif étant d'améliorer la capacité de l'algorithme à prédire la classe cible même avec peu de données d'entraînement.

En partant de la définition du classificateur random forest standard on peut définir mathématiquement le classificateur avec modification de probabilité. Pour cela on ajoute la prédiction de la collection de M arbres pour le point x' qui est le point symétrique à x. On divise le tout par 2 pour que le résultat soit compris entre 0 et 1.

$$m_{M,n}\left(x;\Theta_1,...,\Theta_M,D_n\right) = \begin{cases} 1 & \text{if} \quad \frac{1}{M}\sum_{j=0}^{M}\frac{1}{2}\left(m_n(x;\Theta_j,D_n)+m_n(x';\Theta_j,D_n)\right) > 1/2 \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

La prédiction de la forêt symétrique est l'addition des prédictions des arbres pour le point x et des prédictions des arbres pour le point x' symétrique à x.

4.3.2 Symétrie radiale

Pour qu'il y ait symétrie axiale, les prédictions $r(x_1)$ et $r(x_2)$ doivent être les même pour $x_1 = \|x_2 - c\|^2$.

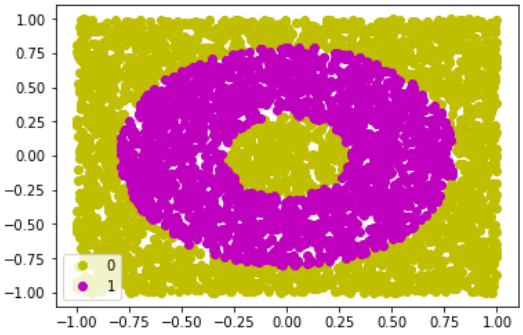


FIGURE 4.5 – Symétrie radiale exemple

La technique utilisée pour imposer la symétrie radiale au classificateur est de générer K points symétriques à x par rapport au centre du cercle supposé connu (il peut être estimé par médiane géométrique par exemple). Chaque point est généré comme suit :

$$\begin{aligned} p = h + r\cos(\theta), k + r\sin(\theta) \\ \theta = 2\pi z \end{aligned} \tag{4.2}$$

avec $c = (h, k)$ centre du cercle de rayon r. On ajoute la prédiction des arbres pour ces points à la prédiction (probabilité) pour le point x.

La définition mathématique du classificateur random forest pour la symétrie radiale ressemble à la définition pour la symétrie axiale. La différence étant que l'on ajoute la prédiction de la collection de M arbres pour tous les points K générés et pas uniquement pour le point x'.

$$m_{M,n}\left(x; \Theta_1, ..., \Theta_M, D_n\right) = \begin{cases} 1 & \text{if} \quad \frac{1}{M}\sum_{j=0}^{M}\frac{1}{K+1}\left(m_n(x; \Theta_j, D_n) \; + \; \sum_{i=0}^{K} m_n(P_i; \Theta_j, D_n)\right) > 1/2 \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

L'impact de la symétrie sur les résultats peut être modifié en augmentant ou en diminuant K.

Mais de meilleurs résultats ont été observés en utilisant un nouveau paramètre : le poids de prédiction de l'arbre pour le point x. On nomme ce paramètre symmetric strength (H), il permet d'augmenter le poids de la prédiction de base par rapport à la prédiction des points symétriques. Après avoir ajouté ce paramètre, le classificateur modifié est :

$$m_{M,n}\left(x; \Theta_1, ..., \Theta_M, D_n\right) = \begin{cases} 1 & \text{if} \quad \frac{1}{M}\sum_{j=0}^{M}\frac{1}{K+H}\left(H \times m_n(x; \Theta_j, D_n) \; + \; \sum_{i=0}^{K} m_n(P_i; \Theta_j, D_n)\right) > 1/2 \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

Ce paramètre est obligatoire dans certains cas où la forme de la classe cible n'est pas une symétrie parfaite, voir section 4.5.

4.4 Méthode dropouts

La méthode dropouts consiste à supprimer des arbres de la forêt pour améliorer le modèle. La méthode dropouts est décrite par [57] où les arbres sont supprimés en fonction de leur effet sur la performance du modèle global. Nous proposons de retirer les arbres qui ne respectent pas une contrainte de forme de données comme la symétrie ou la monotonie. La technique utilisée dépend de la contrainte à faire respecter.

4.4.1 Symétrie axiale

Le cas de la symétrie axiale est réalisable avec la méthode dropouts. Pour cela on va utiliser la métrique sp (Symetric Percentage) qui correspond au pourcentage de symétrie de l'arbre sur un plan. On génère N points répartis uniformément dans le demi-plan défini par $y > k$ et borné par les données. Le paramètre sp est calculé comme suit :

$$sp = \sum_{i=1}^{N} \frac{1}{N} \begin{cases} 1 \; if \; t(x_i) = t(x_i') \\ \\ 0 \; otherwise \end{cases}$$

Où t(x) est la fonction de classification de l'arbre formé T et x' le point symétrique à x par rapport à y. L'arbre respectant $ll < sp < ul$ avec $ll, ul \in [0, 1]$ et $ul > ll$ restera dans la forêt. La valeur ll correspond au pourcentage de symétrie minimum quand à ul, elle correspond au pourcentage de symétrie maximum. L'algorithme Random Forest standard crée M arbres, pour Random Forest dropouts on crée des arbres jusqu'à obtenir M arbres avec $ll < sp < ul$. Pour s'assurer que le nombre d'arbres voulu M soit atteint il faut générer une forêt standard de M arbres sans contrainte, calculer sp pour chaque arbre puis sélectionner un centile de la liste des sp pour enfin générer un forêt symétrique à environ sp%. Plus le centile choisi est élevé, plus la forêt finale sera symétrique et plus le temps d'exécution sera élevé.

4.4.2 Monotonie

Afin de répondre aux exigences des utilisateurs, plusieurs versions de random forest monotones ont été développées. Les techniques les plus populaires sont principalement basées sur i) l'application de divisions monotones dans les arbres de décision, comme dans le paquet R Arborist, XGBoost [58] [59] [60] ou ii) la repondération de l'agrégation ou de la décision dans les arbres [55] [56] [61].

Pour tester la monotonie des arbres on se base sur la définition d'un modèle multivarié monotone : la monotonie positive de y par rapport à X_i signifie que pour toutes les autres variables fixes, y ne peut qu'augmenter pour des valeurs de X_i plus grandes. L'idée est de générer N lignes de tests puis pour chaque X_i lié par une relation monotone avec y on fait augmenter la valeur correspondante à X_i k fois dans chaque ligne du minimum au maximum de X_i . Si la relation monotone est positive on vérifie que y est supérieur à $y - 1$. Le pourcentage de $y > y - 1$ correspond au pourcentage de monotonie pour la variable X_i. Un pourcentage cible est ensuite choisi pour chaque variable. On n'accepte que les arbres respectant cette contrainte. Dropouts fait tendre vers la monotonie mais ne l'impose pas. Il y a plusieurs paramètres modifiables pour contrôler l'impact de la monotonie sur le modèle. Pour durcir l'impact de la monotonie il faut augmenter N et K, cela a un coût sur le temps d'exécution de l'algorithme. Ci dessous l'algorithme en détail (aussi disponible sur GitHub en python : https ://github.com/Mathias38/Random-Forest-Monotone).

Algorithm 8 FitWithMonotonicityDropOut

```
1: procedure FitWithMonotonicityDropOut(data, values, nbRows, positiveVars,
   negativeVars, targetMonotonicity, nbTestMonoPos, nbTestMonoNeg, varTypes)
2:     trees = []
3:     rows = CreateValidationRows(values, nbRows, varTypes)
4:     passedTrees = 0
5:     totalTrees = 0
6:     while passedTrees ≤ nbTree do
7:         tree = (data)
8:         totalTrees = totalTrees + 1
9:         if TestTree(values, tree, positiveVars, negativeVars, TargetMonotonicity,
   nbTestMonoPos, nbTestMonoNeg, varTypes) then
10:            trees.append(tree)
11:            passedTrees = passedTrees + 1
       =0
```

Algorithm 9 CreateValidationRows

```
1: procedure CreateValidationRows(values, nbRows, varTypes)
2:     rows = []
3:     row = [0] * length(values)
4:     i = 0
5:     j = 0
6:     while j ≤ nbRows do
7:         while i ≤ length(row) do
8:             if var_types[i] == discrete then
9:                 row[i] = RadomInt(values[i][0], values[i][1])
10:            if var_types[i] == continuous then
11:                row[i] = RadomFloat(values[i][0], values[i][1])
12:            i = i + 1
13:        rows.append(row)
14:        j = j + 1
```

Algorithm 10 GetVariableMonotonicityPercent

```
1: procedure GetVariableMonotonicityPercent(sign, varValues idx, tree,
   nbTestMono, varType)
2:     totalCount = 0
3:     MonotonicityCount = 0
4:     for row in rows do
5:         test_row = copy(row)
6:         for i in 0 : nbTestMono do
7:             totalCount = totalCount + 1
8:             if var_type == discrete then
9:                 row[idx] = int(round((varValues[1] * (i / nbTestMono)), 0))
10:                testRow[idx] = int(round((varValues[1] * (i + 1) / nbTestMono), 0))
11:            if var_type == continuous then
12:                row[idx] = round((varValues[1] * (i / nbTestMono)), 3))
13:                testRow[idx] = round((varValues[1] * (i + 1) / nbTestMono), 3))
14:            testRowPred = tree.predict(testRow)
15:            rowPred = tree.predict(row)
16:            if sign == positivte then
17:                if ( thentestRowPred ≥ rowPred)
18:                    MonotonicityCount = MonotonicityCount + 1
19:            if sign == negative then
20:                if testRowPred ≤ rowPred then
21:                    MonotonicityCount = MonotonicityCount + 1
```

Algorithm 11 TestTree

```
1: procedure TestTree(values, tree, positiveVars, targetMonotonicity,
   NbTestMonoPos, NbTestMonoNeg, VarTypes)
2:     totalMonotonicityPercent = []
3:     res = True
4:     for k in 0 : length(postiveVars) do
5:         totalMonoticityPercent.append(GetVariableMonotonicityPercent("postivie",
   positiveVars[k], values[positiveVars[k], tree, nbTestMonoPos[k],
   varTypes[positiveVars[k]])
6:     for k i, 0 : length(negativeVars) do
7:         totalMonoticityPercent.append(GetVariableMonotonicityPercent("negative",
   negativeVars[k], values[negativeVars[k], tree, nbTestMonNeg[k],
   varTypes[negativeVars[k]])
8:     for i in 0 : length(totalMonotonicityPercent) do
9:         if totalMonotonicityPercent ≤ TargetMonotonicity then
10:            res == false
11:    return res
```

Les différents paramètres de l'algorithme sont utilisés pour contrôler son comportement :

- Le paramètre *values* est une liste des valeurs minimum et maximum de chaque variable
- *nbTestMonoPos* et *NbTestMonoNeg* fixent le nombre de fois qu'une valeur sera augmentée entre la valeur minimum et la valeur maximum sur chaque *testRow*.
- Les paramètres *nbTestMonoPos* et *NbTestMonoNeg* sont aussi des listes pour permettre de traiter chaque variable différemment. Pour une variable positive monotone x dans $[0, 10]$, le paramètre *values* est fixé à $[0, 10]$. Si *nbTestMonoPos* est égal à 2, les différentes valeurs de x testées seront $[0, 5, 10]$, si *nbTestMonoPos* est égal à 10 les différentes valeurs de x testées seront $[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$.
- Le paramètre *targetMonoticity* est une liste contenant le pourcentage de monotonie cible pour chaque variable. Si il est fixé à 95 pour la variable x tous les arbres avec un pourcentage de monotonie inférieur à 95% seront écartés de la forêt. Ce paramètre doit être fixé à l'aide de connaissances métiers et d'analyses statistiques préalables sur les différentes variables. Le choix de ce paramètre influe sur le temps d'exécution car l'algorithme continuera à tester des arbres jusqu'à en avoir 100. Pour déterminer ce paramètre par calcul il est conseillé de faire tourner l'algorithme avec $targetMonoticity = 0$ et d'extraire le pourcentage de monotonie pour chaque arbre sur toutes les variables. Pour chaque variable, on sélectionne un centile c_x^i de la liste des pourcentages de monotonie de chaque arbre comme *targetMonoticity*. De cette manière l'algorithme sélectionnera les $(100 - c_x^i)\%$ arbres les plus monotones. Le paramètre *nbRow* correspond au nombre de lignes de test. Plus *nbRow* est élevé plus la monotonie réelle de la forêt sera proche de *targetMonoticity*. En effet la méthode dropouts n'est pas une méthode "hard" qui force la monotonie mais une méthode "soft" qui fait tendre vers la monotonie.

Une version de random forest monotone en régression a aussi été développée dans laquelle les arbres de régressions sont utilisés et le vote de la classification est remplacé par un calcul de moyenne.

Pour comprendre l'algorithme voici le détail d'une exécution avec les valeurs des paramètres importants. Le code de création de la random forest monotone est :

```
rf_monotone.fit_with_monotony_drop_out(x_train_list,
                                       [[0, 17], [0, 199], [0, 122], [0, 99], [0,846], [0, 67.1], [0.078, 2.42], [21, 81]] ,
                                       100,
                                       [0, 1, 2, 3, 4, 5, 6, 7],
                                       [],
                                       [99.5, 99.5, 99.5, 99.5, 99.5, 99.5, 99.5, 99.5],
                                       [10, 10, 10, 10, 10, 10, 10, 10],
                                       [],
                                       ["d", "d", "d", "d", "d", "c", "c", "d"])
```

FIGURE 4.6 – Déclaration de la random forest monotone

Sur la première ligne il y a les données d'entraînement. Sur la deuxième, les valeurs minimum et maximum de chaque colonne monotone. Le nombre 100 correspond au nombre de lignes de test à générer. La liste en quatrième paramètre contient les indices des colonnes monotones positives. La liste vide est celle des colonnes monotones négatives. Le sixième paramètre est la liste des seuils de monotonie pour chaque variable. La liste suivante correspond au nombre de tests à effectuer sur chaque ligne (détaillé après) pour les variables positives. La dernière liste correspond au type des variables (d pour discrète et c pour continue).

L'algorithme commence par créer un arbre comme random forest standard mais contrairement à la version normale, la monotonie de l'arbre est testé. Pour les tests 100 lignes aléatoires sont générées. Ensuite la première variable est incrémentée 10 fois et l'égalité entre la prédiction pour la ligne n et n+1 est vérifiée :

```
Ligne n
[12, 124, 4, 2, 53, 56.544345944506006, 2.0847387129737096, 75]
Ligne n+1
[14, 124, 4, 2, 53, 56.544345944506006, 2.0847387129737096, 75]
Prédiction pour la ligne n
0.0
Prédiction pour la ligne n + 1
0.0
```

FIGURE 4.7 – Test d'égalité entre la prédiction pour les lignes n et n+1 : monotone

Entouré en rouge on peut voir l'incrémentation de la première variable monotone sur une des ligne générée. Les prédictions sont identiques, on ajoute donc 1 au compteur de monotonie.

```
[0, 124, 69, 52, 539, 17.651978093573177, 1.8744052218052565, 25]
Ligne n+1
[2, 124, 69, 52, 539, 17.651978093573177, 1.8744052218052565, 25]
Prédiction pour la ligne n
1.0
Prédiction pour la ligne n + 1
0.0
```

FIGURE 4.8 – Test d'égalité entre la prédiction pour les lignes n et n+1 : non monotone

La figure 4.8 montre le cas où les deux lignes ne sont pas monotones positives. La prédiction de la ligne $n+1$ est inférieure à la prédiction de la ligne n. On ajoute un au compteur de non monotonie. L'incrémentation est répétée pour chaque variable 10 fois (pour cette exécution 10 est la valeur du paramètre nombre de tests) par ligne de test.

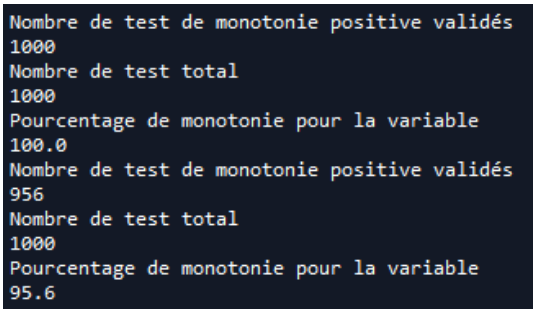


FIGURE 4.9 – Pourcentage de monotonie par variable

La figure 4.9 montre le pourcentage de monotonie de deux variables pour un arbre. La deuxième variables a un pourcentage de monotonie inférieure au seuil de $99,5\%$, cet arbre sera donc supprimé. L'algorithme va tester chaque variable monotone de chaque arbre de la même façon jusqu'à atteindre 100 arbres validés par les tests de monotonie. La prédiction est identique à la prédiction de random forest standard.

4.4.3 A propos de la cohérence du classificateur random forest avec test des arbres et dropout

L'analyse des classificateurs random forest est connue pour être difficile. En effet, à l'exception des forêts purement aléatoires de Breiman, peu de résultats sont connus en raison de la construction complexe des arbres de décision aléatoires ; voir par ex. [62] [63]. Cependant, la cohérence des classificateurs *avec vote* a été prouvée dès lors que les classificateurs agrégés sont eux-mêmes cohérents [64]. On peut donc se poser la question suivante : Le test de monotonie et le drop-out préservent-ils la cohérence du classificateur de vote ? Ou, en d'autres termes, sous quelles conditions le classificateur est-il cohérent ?

Puisque nous souhaitons révéler la monotonie de la distribution sous-jacente, il est naturel de caractériser ce comportement. Nous dirons que la distribution $\mathcal{D}$ est *strictement monotone* en ce qui concerne $p \in \mathcal{M}^+$ si pour presque tous les $\mathbf{x}$:

$$\mathbb{P}\left[Y = 1 | \mathbf{X} = [\mathbf{x}^1, ..., \mathbf{x}^{p-1}, t, \mathbf{x}^{p+1}, .., \mathbf{x}^d\right] \tag{4.5}$$

$$< \mathbb{P}\left[Y = 1 | \mathbf{X} = [\mathbf{x}^1, ..., \mathbf{x}^{p-1}, u, \mathbf{x}^{p+1}, .., \mathbf{x}^d\right] \tag{4.6}$$

pour tout $t < u$. Cette notion peut être définie de manière similaire pour une caractéristique négative.

Maintenant, nous pouvons montrer que si la distribution se comporte comme prévu, alors la probabilité que le test de monotonie soit vérifié converge vers 1 lorsque le nombre d'échantillons $nEchantillons$ va vers l'infini. Ceci nous permet de montrer que le classifieur que nous proposons est alors cohérent.

Theorem 1 *Supposons que la séquence de classificateurs binaires aléatoires (g_n) soit cohérente pour une certaine distribution $\mathcal{D}$ de $(\mathbf{X}, Y)$ qui est strictement monotone par rapport aux caractéristiques dans $\mathcal{M}^+, \mathcal{M}^-$. Alors, pour tout $\mu \in [0, 1)$, le classificateur de vote associé avec test aléatoire et abandon $\tilde{g}_{m;n}$ est également cohérent.*

La première partie de la preuve suit celle de [64, Prop. 1]. Pour une séquence de classificateurs binaires aléatoires (g_n), la cohérence signifie que

$$\mathbb{P}\left[g_n(\mathbf{X}; \Theta) \neq Y\right] \xrightarrow{n \to \infty} \mathbb{P}\left[g^\star(\mathbf{X}) \neq Y\right] \tag{4.7}$$

mais comme $\mathbb{P}\left[g_n(\mathbf{X}; \Theta) \neq Y | \mathbf{X} = \mathbf{x}\right]$ est supérieur ou égal à $\mathbb{P}\left[g^\star(\mathbf{X}) \neq Y | \mathbf{X} = \mathbf{x}\right]$ pour tous les $\mathbf{x}$

N, la cohérence signifie que pour presque tous les $\mathbf{x}$,

$$\mathbb{P}\left[g_n(\mathbf{X};\Theta) \neq Y|\mathbf{X}=\mathbf{x}\right] \xrightarrow{n\to\infty} \mathbb{P}\left[g^\star(\mathbf{X}) \neq Y|\mathbf{X}=\mathbf{x}\right]. \tag{4.8}$$

Maintenant, on peut remarquer que si $\mathbb{P}[Y=1|\mathbf{X}=\mathbf{x}]=1/2$, le problème de classification est dégénéré. Supposons que $\eta(\mathbf{x}) := \mathbb{P}[Y=1|\mathbf{X}=\mathbf{x}] > 1/2$ (le cas contraire peut être traité de manière similaire). Alors, nous avons :

$$\mathbb{P}\left[g^\star(\mathbf{X}) \neq Y|\mathbf{X}=\mathbf{x}\right] = 1-\eta(\mathbf{x})$$

et, puisque g_n est un classificateur binaire :

$$\begin{aligned}
&\mathbb{P}\left[g_n(\mathbf{X};\Theta) \neq Y|\mathbf{X}=\mathbf{x}\right] && (4.9)\\
&\quad = (2\eta(\mathbf{x})-1)\mathbb{P}\left[g_n(\mathbf{x};\Theta)=0\right] + 1 - \eta(\mathbf{x}) && (4.10)
\end{aligned}$$

ce qui signifie que $\mathbb{P}\left[g_n(\mathbf{x};\Theta)=0\right] \to 0$ par cohérence. Le raisonnement inverse montre que pour prouver la cohérence de $\tilde{g}_{m;n}$, il suffit de montrer que $\mathbb{P}\left[\tilde{g}_{m;n}(\mathbf{x};(\Theta_j),(\zeta_j))=0\right] \to 0$ pour presque tous les $\mathbf{x}$ tels que $\mathbb{P}[Y=1|\mathbf{X}=\mathbf{x}] > 1/2$.
Par conséquent, prenons tout $\mathbf{x} \in \mathbb{R}^d$ tel que $\mathbb{P}[Y=1|\mathbf{X}=\mathbf{x}] > 1/2$. Fixons $\tilde{m}$, alors m est une variable aléatoire, dépendant de $\tilde{m}$, (Θ_j,ζ_j). Nous devons donc reformuler notre expression pour remplacer le facteur $1/m$ par $1/\tilde{m}$. Nous avons :

$$\begin{aligned}
&\mathbb{P}\left[\tilde{g}_{m;n}(\mathbf{x};(\Theta_j),(\zeta_j))=0\right] && (4.11)\\
&= \mathbb{P}\left[\frac{1}{m}\sum_{j=1}^{\tilde{m}} g_n(\mathbf{x};\Theta_j)\mathbb{1}_{\hat{m}(\Theta_j;\zeta_j,\mu)} < \frac{1}{2}\right] && (4.12)\\
&= \mathbb{P}\left[\frac{1}{m}\sum_{j=1}^{\tilde{m}} \mathbb{1}_{g_n(\mathbf{x};\Theta_j)=0,\hat{m}(\Theta_j;\zeta_j,\mu)} > \frac{1}{2}\right] && (4.13)\\
&= \mathbb{P}\left[\frac{1}{\tilde{m}}\sum_{j=1}^{\tilde{m}} \Big(\mathbb{1}_{g_n(\mathbf{x};\Theta_j)=0,\hat{m}(\Theta_j;\zeta_j,\mu)} && (4.14)\\
&\qquad + \frac{1}{2}\mathbb{1}_{\overline{\hat{m}(\Theta_j;\zeta_j,\mu)}}\Big) > \frac{1}{2}\right] && (4.15)\\
& && (4.16)
\end{aligned}$$

où nous avons utilisé l'inégalité de Markov, le fait que (Θ_j,ζ_j) sont i.i.d., et noté par $\overline{A}$ le complément logique de A.
Puisque les classificateurs (g_n) sont cohérents, nous avons $\mathbb{P}\left[g_n(\mathbf{x};\Theta)=0\right] \to 0$. Ainsi, la principale différence avec le classificateur de vote habituel est la présence de la probabilité de rejeter un arbre : $\mathbb{P}\left[\overline{\hat{m}(\Theta;\zeta,\mu)}\right]$ (qui est indépendante de $\mathbf{x}$).
Prenez n'importe quelle caractéristique positivement monotone $p \in \mathcal{M}^+$ (le raisonnement est le même pour une caractéristique négative). Alors ,

$$\begin{aligned}
\mathbb{P}\left[\overline{\hat{m}^p(\Theta;\zeta,\mu)}\right] &= \mathbb{P}\left[\hat{m}^p_+(\Theta_j;\zeta) < \mu\right] && (4.17)\\
&= \mathbb{P}\left[1-\hat{m}^p_+(\Theta_j;\zeta) > 1-\mu\right] && (4.18)\\
&\leq \frac{1-\mathbb{E}\left[\hat{m}^p_+(\Theta_j;\zeta)\right]}{1-\mu} && (4.19)
\end{aligned}$$

où nous avons utilisé à nouveau l'inégalité de Markov et le fait que $\mu < 1$.
Nous nous tournons donc vers l'analyse

$$\mathbb{E}\left[\hat{m}_+^p(\Theta_j;\zeta)\right] \tag{4.20}$$

$$= \frac{1}{s(t-1)} \sum_{i=1}^{s} \sum_{k=1}^{t-1} \mathbb{P}\left[g_n(\zeta_k^p;\Theta_j) \leq g_n(\zeta_{k+1}^p;\Theta_j)\right] \tag{4.21}$$

$$= \frac{1}{t-1} \sum_{k=1}^{t-1} \mathbb{P}\left[g_n(\zeta_k^p;\Theta_j) \leq g_n(\zeta_{k+1}^p;\Theta_j)\right]. \tag{4.22}$$

Pour ce faire, prenons un $\mathbf{z}$ arbitraire dans le support de $\mathbf{X}$ (ou dans $\mathbb{R}^d$ pour fixer les idées) et laissons $t < u$ être deux valeurs dans l'intervalle de $\mathbf{X}^p$. On note $\mathbf{X}^{-p}$ les caractéristiques de $\mathbf{X}$ sauf la p-ième. Pour simplifier la notation, nous désignons par $\mathbf{z}_1$ (resp. $\mathbf{z}_2$), le vecteur de $\mathbb{R}^d$ tel que $\mathbf{z}_1^{-p} = \mathbf{z}$ et $\mathbf{z}_1^p = t$ (resp. $\mathbf{z}_2^p = u$). La monotonie positive stricte de la distribution $\mathcal{D}$ par rapport à la caractéristique p signifie que

$$\mathbb{P}\left[Y = 1|\mathbf{X} = \mathbf{z}_1\right] = \mathbb{P}\left[Y = 1|\mathbf{X}^p = t, \mathbf{X}^{-p} = \mathbf{z}^{-p}\right] \tag{4.23}$$

$$< \ \mathbb{P}\left[Y = 1|\mathbf{X}^p = u, \mathbf{X}^{-p} = \mathbf{z}^{-p}\right] = \mathbb{P}\left[Y = 1|\mathbf{X} = \mathbf{z}_2\right] \tag{4.24}$$

ce qui implique que

$$g^\star(\mathbf{z}_1) = \mathbb{1}_{\mathbb{P}[Y=1|\mathbf{X}=\mathbf{z}_1]\geq 1/2} \leq \ \mathbb{1}_{\mathbb{P}[Y=1|\mathbf{X}=\mathbf{z}_2]} = g^\star(\mathbf{z}_2) \tag{4.25}$$

ce qui signifie que le test de monotonicité est satisfait avec une probabilité de 1 par le classificateur de Bayes optimal. Maintenant, puisque g_n est un classifieur binaire, la seule possibilité pour qu'il n'y est pas

$$g_n(\mathbf{z}_1) \leq g_n(\mathbf{z}_2) \tag{4.26}$$

est le cas où $g_n(\mathbf{z}_1) = 1$ et $g_n(\mathbf{z}_{12}) = 0$.

1) Si $\mathbb{P}\left[Y = 1|\mathbf{X} = \mathbf{z}_1\right] \geq 1/2$. alors, $\mathbb{P}\left[Y = 1|\mathbf{X} = \mathbf{z}_2\right] > 1/2$ et puisque g_n est consistant :

$$\mathbb{P}\left[g_n(\mathbf{z}_2) = 0 \mid \mathbb{P}\left[Y = 1|\mathbf{X} = \mathbf{z}_2\right] > 1/2\right] \to 0. \tag{4.27}$$

2) If $\mathbb{P}\left[Y = 1|\mathbf{X} = \mathbf{z}_1\right] < 1/2$. alors, puisque g_n est consistant :

$$\mathbb{P}\left[g_n(\mathbf{z}_1) = 1 \mid \mathbb{P}\left[Y = 1|\mathbf{X} = \mathbf{z}_2\right] < 1/2\right] \to 0. \tag{4.28}$$

Ainsi, nous obtenons que

$$\mathbb{P}\left[g_n(\mathbf{z}_1) \leq g_n(\mathbf{z}_2)\right] = 1 - \mathbb{P}\left[g_n(\mathbf{z}_1) = 1, g_n(\mathbf{z}_2) = 0\right] \to 1 \tag{4.29}$$

ce qui montre que $\mathbb{E}\left[\hat{m}_+^p(\Theta_j;\zeta)\right] \to 1$ et enfin que

$$\mathbb{P}\left[\overline{\hat{m}^p(\Theta;\zeta,\mu)}\right] \leq \frac{1 - \mathbb{E}\left[\hat{m}_+^p(\Theta_j;\zeta)\right]}{1-\mu} \to 0. \tag{4.30}$$

Le raisonnement peut être effectué de manière similaire pour les autres caractéristiques monotones et donc $\mathbb{P}\left[\overline{\hat{m}(\Theta;\zeta,\mu)}\right] \to 0$. Enfin, nous avons par (4.16) que $\mathbb{P}\left[\tilde{g}_{m;n}(\mathbf{x};(\Theta_j),(\zeta_j)) = 0\right] \to 0$ ce qui implique à son tour que les classificateurs $(\tilde{g}_{m;n})_n$ sont cohérents.

4.5 Résultats Random Forest symétrique sur données générées

Les résultats graphiques représentent une seule exécution de l'algorithme. Aux résultats graphiques on ajoute un tableau qui contient les résultats moyens sur 100 exécutions pour N=100, N=500 et N=1000.

4.5.1 Symétrie axiale : triangle

Les tests pour la symétrie sont réalisés sur des données générées. L'utilité d'un classificateur random forest symétrique sur des vrais jeux de données reste à démontrer mais la symétrie est un très bon moyen de comprendre l'intégration de connaissances. On génère 100 points aléatoires pour les données d'entraînements. On génère 5000 points pour le jeu de test de façon à ce qu'il couvre l'ensemble de l'espace.

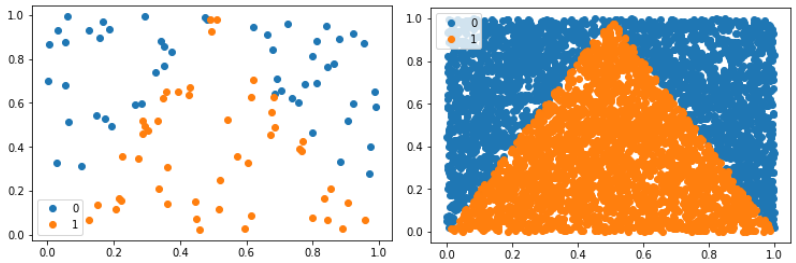


FIGURE 4.10 – Triangle données d'entraînement et de test

Générer un jeux de données de 100 points seulement permettra de tester la robustesse de l'algorithme avec peu de données. Voici les résultats pour le cas triangle.

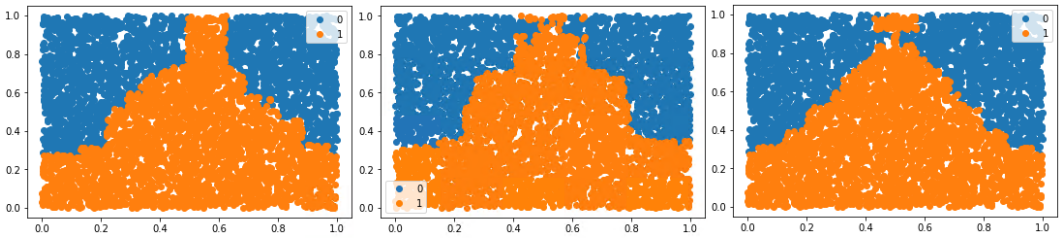


FIGURE 4.11 – Triangle résultat. A gauche : standard, au milieu : dropouts, à droite : probabilité modifiée

On remarque que le résultat de la version standard n'est pas symétrique et que les angles de la base du triangle ne sont pas bien définis. Pour la version dropouts le triangle résultat est symétrique mais les cotés ne sont pas lisses et les angles de la base du triangle ne sont pas bien définis non plus. A première vue dropouts semble donner de moins bon résultats que la version standard même si le triangle obtenu est symétrique. Le meilleur résultat est celui de la version probabilité modifiée ; le triangle est symétrique et les bords sont lisses. Aucune des versions n'est arrivée à prédire les angles de la base du triangle.

Pour confirmer ces résultats on lance 100 tests avec $N = 100$, $N = 500$ et $N = 1000$ où N est le nombre de points générés aléatoirement. On fixe le nombre d'arbres dans la forêt à 100. Dans le tableau de résultat on utilise le score F1. Le score F1 est définit comme :

$$F1 = 2 * (precision * recall)/(precision + recall)$$

La précision quantifie le nombre de prédictions de classe positive qui appartiennent réellement

à la classe positive. Le recall quantifie le nombre de prédictions de classe positives faites à partir de tous les exemples positifs de l'ensemble de données. La contribution de la précision et du recall est équivalente. Cette métrique permet de prendre en compte les vrais positifs et négatifs mais aussi les faux négatifs. Ci dessous un schéma explicatif du F score :

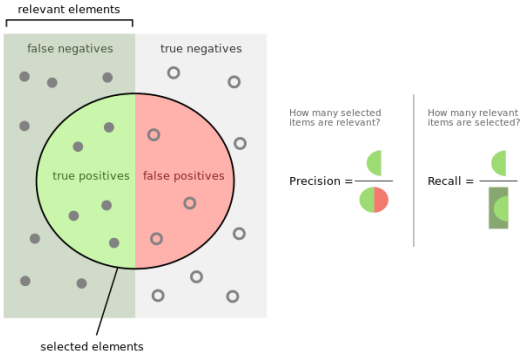


FIGURE 4.12 – F-score, source : wikipedia

SP est le nombre de points symétrisés (Symmetrised Points) , c'est à dire le nombre de prédictions qui ont été changées par la modification de la probabilité. $SI\%$ est le pourcentage de prédictions modifiées qui étaient mauvaises et sont devenues bonnes.

TABLE 4.1 – Triangle

		F1	SP	SI%
N=100	RF	0.885		
	RF DropOut	0.879		
	RF Modified Probability	**0.904**	729	74.36
	RF Modified Probability symetric strength 0.35%			
N=500	RF	0.921		
	RF DropOut	0.906		
	RF Modified Probability	**0.956**	807	84.86
	RF Modified Probability symetric strength 0.35%			
N=1000	RF	0.926		
	RF DropOut	0.908		
	RF Modified Probability	0.953	772	0.867
	RF Modified Probability symetric strength 0.35%			

Comme pressenti avec les nuages de points, le meilleur classificateur est la random forest avec probabilité modifiée et le pire, celui avec dropouts. On peut mesurer l'impact de la modification de probabilité avec SP et SI%, pour le cas triangle on peut voir que les points modifiés par symétrie deviennent bons plus de 74% des fois.

4.5.2 Symétrie radiale : donut

Une autre forme de symétrie est la symétrie radiale. Pour tester ce cas on choisit une forme de donut dont on ne connait pas le centre. Ci dessous les jeux de données d'entraînement et de test pour le cas donut (N=100) :

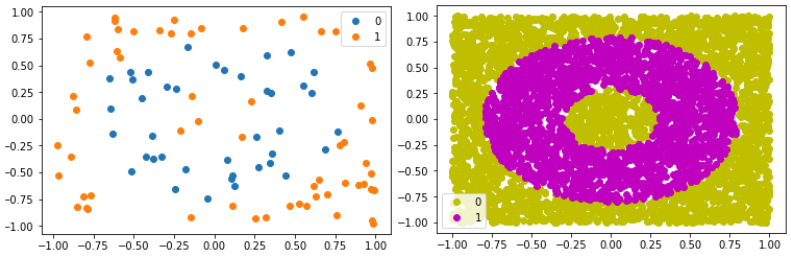


FIGURE 4.13 – Donut données d'entraînement (gauche) et de test (droite)

Voici un résultat graphique pour le cas donut.

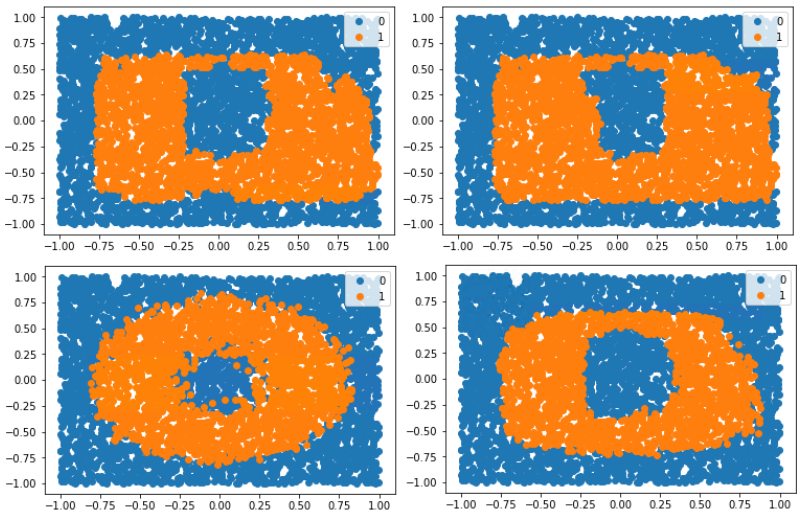


FIGURE 4.14 – Donut résultat. En haut à gauche : standard, en haut à droite : dropouts, en bas à gauche : probabilité modifiée, en bas à droite : probabilité modifiée avec symmetric strength à 35%

Le meilleur résultat est celui obtenu avec la random forest avec probabilité modifié et symmetric strength à 100%. Ce résultat est logique car la forme de donut peut être prédite uniquement par symétrie.

TABLE 4.2 – Donut

F1	SP	SI%
0.788		
0.790		
0.860	1242	80.01
0.856	1067	80.86
0.835		
0.840		
0.912	1242	86.81
0.903	1111	87.89
0.846		
0.847		
0.926	1240	88.87
0.914	1153	90.01

Les résultats sur 100 exécutions confirment les résultats graphiques. Ces résultats ne sont pas significatifs car le problème du donut est complètement symétrique. Le cas donut avec outlier et donut incomplet permettent d'obtenir des résultats plus représentatifs.

4.5.3 Symétrie radiale : donut avec outlier

Pour rendre le problème du donut plus compliqué, des outliers sont ajoutés. Il est attendu de l'algorithme modifié qu'il soit plus robuste face aux outliers que l'algorithme standard. Voici les données d'entraînement et de test pour le donut avec outliers :

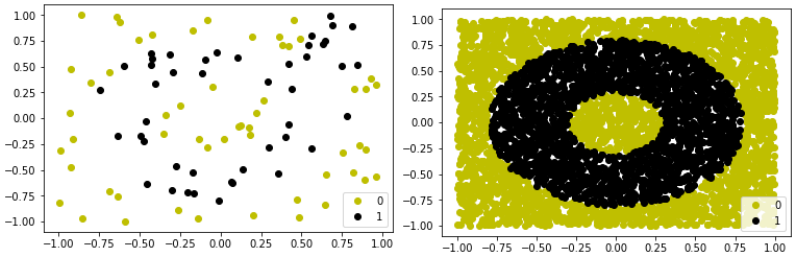


FIGURE 4.15 – Donut outlier données d'entraînement et de test

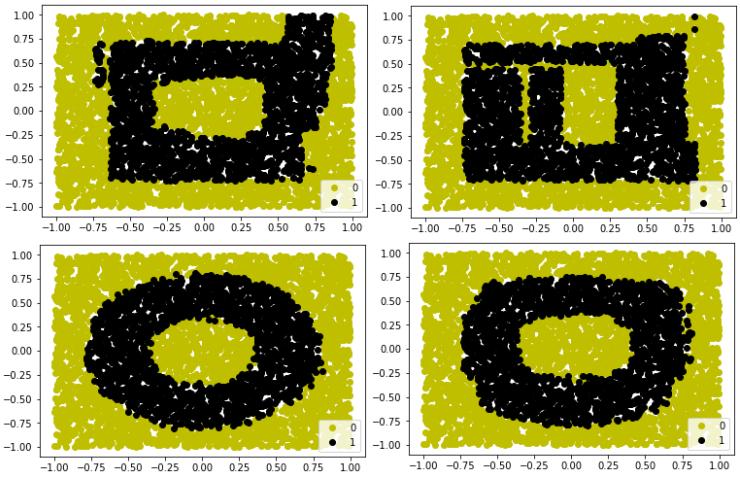


FIGURE 4.16 – Donut outlier résultat. En haut à gauche : standard, en haut à droite : dropouts, en bas à gauche : probabilité modifiée, en bas à droite : probabilité modifiée avec symmetric strength à 35%

Les résultats graphiques montrent que les forêts avec contrainte de symétrie sont plus robustes aux outliers. La forêt standard a été perturbée par les outliers d'après la partie haute droite de la prédiction. La méthode dropouts est moins sensible aux outliers mais n'arrive pas à prédire la forme circulaire. La modification de probabilité n'a pas été perturbée par les outliers et a prédit la forme circulaire.

TABLE 4.3 – Donut outlier

F1	SP	SI%
0.756		
0.673		
0.804	1527	76.62
0.827	1068	81.82
0.807		
0.814		
0.903	1389	84.93
0.884	1210	87.63
0.812		
0.820		
0.910	1459	86.58
0.888	1230	89.25

Ces résultats confirment confirment l'hypothèse que la random forest avec contrainte de symétrie est plus robuste aux outliers que la forêt standard. Ces résultats sont importants car ils confirment l'intérêt de l'intégration de connaissances évoquée dans l'introduction.

4.5.4 Symétrie radiale : donut incomplet

La forme du donut étant complètement prédictible par symétrie on décide de tester l'algorithme sur une forme de donut incomplet. Une partie du donut est retirée, la prédiction ne pourra pas reposer que sur la symétrie. Cette forme permettra de démontrer l'utilité du paramètre symmetric strength qui permet de contrôler l'impact de la symétrie sur les prédictions. Ici les donnés d'entraînement et de test pour ces données (N = 100).

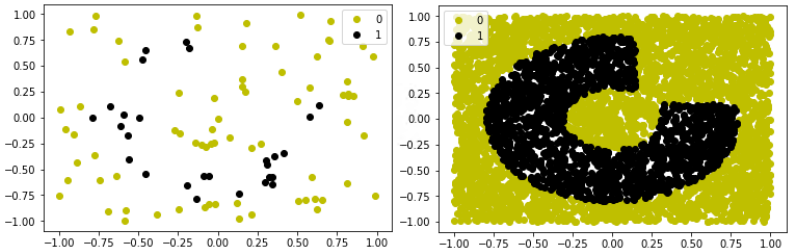


FIGURE 4.17 – Donut incomplet données d'entraînement et de test

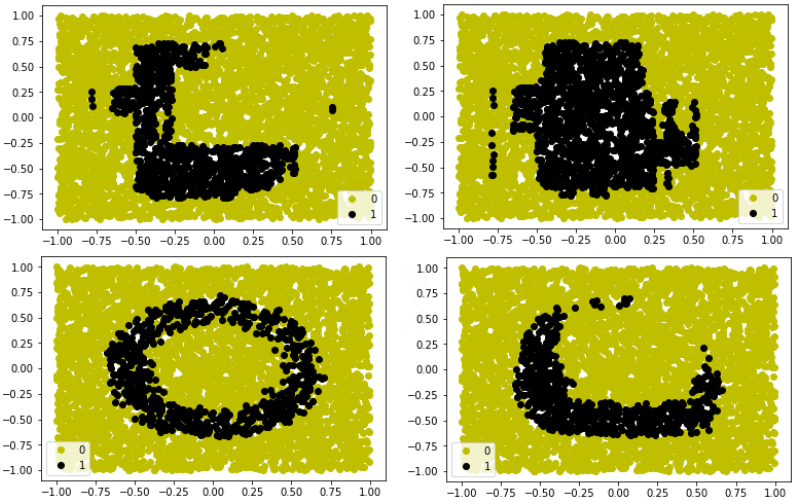


FIGURE 4.18 – Donut incomplet résultats. En haut à gauche : standard, en haut à droite : dropouts, en bas à gauche : probabilité modifiée, en bas à droite : probabilité modifiée avec symmetric strength à 35%

Les résultats graphiques semblent montrer que la random forest symétrique par modification de probabilité avec symmetric strength 35% est la plus performante sur ce cas car c'est la seule qui a su prédire la forme réelle des données (bien que la densité de points de la classe cible prédite soit trop faible).

TABLE 4.4 – Donut incomplet

F1	SP	SI%
0.708		
0.713		
0.553	1358	54.26
0.703	996	71.37
0.767		
0.78		
0.656	1390	60.60
0.757	1108	75.68
0.773		
0.782		
0.678	1334	61.37
0.779	1170	77.84

Les résultats pour la forme donut incomplet sont inattendus. En effet bien que la random forest symétrique avec modification de probabilité et symétrique strength à 35% donne une meilleure prédiction de la forme globale du donut, le score F1 est meilleur pour la random forest standard et la méthode dropouts. C'est un cas particulier intéressant où l'on peut voir un modèle plus performant (random forest dropouts) qui ne prédit pas correctement la forme de données. Dans cet exemple random forest avec probabilité modifiée et symmetric strength à 35% arrive à prédire la forme mais prédit moins de points que random forest dropouts. Ceci illustre une faiblesse des modèles non paramétriques, un bon score de classification ou une bonne précision ne signifie pas que le modèle est juste. On préférera le modèle probabilité modifié et symmetric strength à 35% car il est plus robuste (bonne forme). La méthode modification de probabilité sans réduire l'impact de la symétrie donne les plus mauvais résultats. Cette méthode donne de très bons résultats sur la forme du donut complet car c'est une forme parfaitement symétrique qui peut être prédite seulement par symétrie.

4.6 Résultats Random Forest monotone

Les résultats pour la méthode random forest avec symétrie concernaient des jeux de données générées. Pour la méthode random forest avec monotonie des jeux de données réels seront aussi utilisés. Ces jeux de données sont certains des jeux de données choisis par [55] [56] ainsi que deux autres jeux de données libres. Un premier essai de la méthode a été effectué avec des données générées. Il permet de visualiser ce que l'on attend de cet algorithme. Ci-dessous les données d'entraînements et de test pour les données générées :

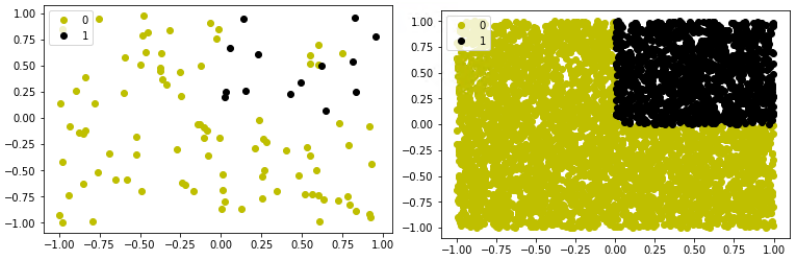


FIGURE 4.19 – Monotonie données d'entraînement et de test

On pollue volontairement le jeu d'entraînement pour simuler un bruit dans les données. Un des objectifs est d'éviter à l'algorithme d'apprendre des règles qui ne respectent pas des contraintes connues. Cela peut arriver à cause de données bruitées.

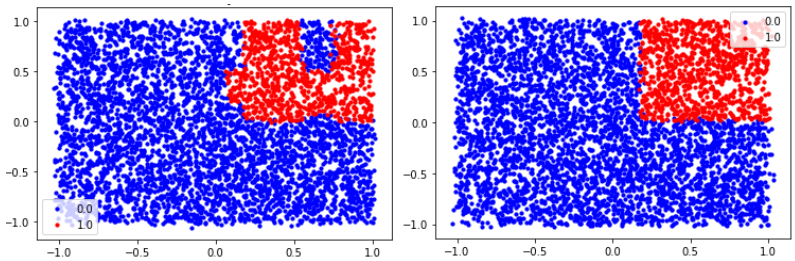


FIGURE 4.20 – Monotonie résultats

L'avantage de la random forest monotone est bien visible ici. Le modèle de la random forest standard a été pollué par le bruit ce qui a créé une mauvaise zone de prédiction contrairement au modèle de la random forest monotone qui est visiblement robuste à l'ajout de données bruitées pour ce cas simple.

La méthode est ensuite testée sur des jeux de données libres disponibles sur UCI [65] représentant des cas réels. Les jeux de données sont décrits dans le tableau 4.5.

Pour chaque jeu de données, voici une présentation détaillée de la réponse et des variables monotones.

German Credit Ratings

L'objectif est de classer les gens avec un bon risque de crédit ou un mauvais risque de crédit. Les personnes avec un mauvais risque de crédit ne pourront pas recevoir d'emprunt de la part de la banque contrairement aux personnes avec un bon risque de crédit. Les notes des clients (bon risque ou mauvais risque) du jeu de données sont des notes attribuées par des banquiers.
Il y a 6 variables qui ont une relation monotone positive avec y.La colonne *job* correspond à la catégorie du métier du client, *savings_quite_rich* et *savings_rich* indiquent que le client possède un épargne confortable, l'attribut *check_rich* veut dire que le client a beaucoup d'argent sur son compte courant et $owns_house$: possède une maison. Il y 5 variables qui ont

TABLE 4.5 – Présentations des jeux de données

Jeux de données	Source	Nb de lignes	Nb de colonnes	Réponse	Atrributs contraints par monotonie
German Credit Ratings	Kaggle	1000	24	Good (0) Bad (1)	positives : duration, other_debtor, renting négatives : chk_acct, saving_acct, present_emp, age, other_install, owns house
Pima Indians Diabetes	UCI	768	9	no diabete (0) diabete (1)	positives : Pregnancies, Glucose, BloodPressure, SkinThikness, Insulin, BMI, DiabetesPedigreeFunction
Cleveland Heart Disease	UCI	297	16	absence (0) presence (1)	positives : age, trestbp, chol, slope, male, typical_angina, atypical_angina, ecg1, ecg2
South African Heart Disease	KEEL	462	9	negative (0) positive (1)	positives : Sdp, Tobacco, Ldl, Adiposty, Famhist, Typea, Obesity, Alcohol, Age
Car Acceptability	UCI	1728	7	unacc (0) , acc (1), good (2), vgood (3)	postivies : persons, safety négatives : buying, maint
Abalone	UCI	2584	19	Age Category lower half (0) upper half (1)	positives : Sex, Length, Height, ViceraWeight, ShellWeight
Sesmic bump	UCI	4177	8	no mining hazard (0) mining hazard (1)	positives : seismoacoustic, shift, genergy, ghazard, nbumps, nbumps2, nbumps3, nbumps4, nbumps5

une relation monotone négative avec y. Dans ce cas, une valeur élevée d'une de ces variables détériorera la note obtenu par le client. La variable *duration* est la durée de remboursement demandée par le client, $credit_amount$ indique la somme du crédit que désire le client, *renting* si le client est locataire, $savings_moderate$ vaut 1 si les économies du client sont modérées et l'âge.

Pima Indian Diabetes

Avec ce jeux de données on veut prédire si un patient a un diabète ou non. Il y a 7 variables qui ont une relation monotone positive avec y. Des valeurs élevées pour ces attributs augmentent les chances de diabètes. La colonne $Pregnancies$ est le nombre de grossesses de la patiente, $Glucose$: quantité de glucose dans le sang, $BloodPressure$: la pression sanguin (mm hg), $SkinThickness$: épaisseur de la peau (mm), $Insulin$: insuline (mu U/ml), BMI : indice de masse corporelle (poids kg/ taille m^2) et $DiabetesPedigreeFunction$ est une synthèse des antécédents de diabète chez les parents et de la relation génétique de ces parents avec le sujet.

Cleveland Heart Disease

Le but de ce cas est de prédire la présence ou non d'une maladie cardiaque. Il y a 9 variables qui ont une relation monotone positive avec la réponse, des valeurs plus grandes de ces variables augmentent les chances d'une maladie cardiaque. Ces variables sont l'âge, $tresbp$: pression sanguine au repos, $chol$: cholestérol sérique (mg/dl), $slope$: la pente du pic de l'électrocardiogramme après un exercice, $male$: le patient est un homme, $typical\_angina$: angine de poitrine typique, $atypical\_angina$: angine de poitrine atypique, $ecg1$: anormalité dans l'électrocardiogramme au repos et $ecg2$: hypertrophie ventriculaire gauche probable ou certaine selon les critères d'Estes.

South African Heart Disease

Comme pour le cas précédent, on veut trouver la présence ou non d'une maladie cardiaque chez les patients. Il y a 9 variables positives, Sdp : pression sanguine systolique, $Tobacco$: cumul de tabac dans le corps (kg), Ldl : cholestérol lipoprotéine à faible densité, $Adiposity$: accumulation de graisse dans le tissu cellulaire sous-cutané (adiposité), $Famhist$: présence ou non de problème(s) cardiaque(s) dans la famille du patient, $Typea$: comportement hyperactif de type A, $Obesity$: le patient est obèse, $Alcohol$: consommation d'alcool et l'âge.

Car Acceptability

L'objectif est d'évaluer des voitures dans l'optique d'un achat suivant différents critères. La réponse a 4 niveaux : mauvais(unacc), acceptable (acc), bon (good) et très bon (vgood). Il y 2 variables liées positivement et 2 variables liées négativement à y. Une valeur élevée pour un attribut lié qui a une relation monotone positive avec y implique une meilleure évaluation. Les colonnes avec une relation monotone positive sont $persons$: capacité et $safety$: sécurité estimée de la voiture. Les colonnes avec une relation monotone négative sont $buying$: prix et $maintenance$: prix de la maintenance.

Abalone

L'abalone ou Ormeau en Français est un type de coquillage. On veut prédire l'âge des coquillages en fonction de plusieurs attributs. Pour la classification on transforme y (l'âge) en catégorie : 0 première moitié, 1 deuxième moitié. Il y a 5 colonnes qui ont une relation monotone positive avec y. Ces colonnes sont le sexe, la taille, la hauteur le poids des viscères et le poids de la coquille. Plus la valeur de ces colonnes est élevée, plus le coquillage est vieux.

Sesmic

Pour assurer la sécurité des mineurs il faut prédire les dangers miniers. Il y a 9 attributs liés positivement à y. Plus la valeur de ces variables est élevée, plus les chances de dangers miniers sont élevées. Les colonnes sont $seismoacoustic$: résultat de l'évaluation de l'aléa sismique de l'exploitation minière obtenu par la méthode sismoacoustique, $shift$: type d'équipe (préparation ou minage), $genergy$: l'énergie sismique enregistrée au cours du déplacement précédent par le géophone le plus actif (GMax), $nbumps_n$: nombres de secousses avec une intensité de $[10^n, 10^{n+1}[$ dans l'équipe précédente.

Pour obtenir les résultats ci-dessous on divise les jeux de données en données d'entraînement et données de test de manière aléatoire pour chaque exécution. La taille des jeux de test est fixée à 20% du nombres de lignes. Les résultats du tableau sont une moyenne sur 100 exécutions. Le pourcentage de monotonie est fixé empiriquement après plusieurs exécutions d'essais, il est le même pour chaque variable. Pour un problème réel ce pourcentage doit être fixé grâce à des connaissances à priori et après une étude plus détaillée des données.

Le tableau 4.6 contient les résultats sur 100 exécutions pour les différents jeux de données. Les colonnes sont le F-score pour la random forest standard, le F-score pour la random forest monotone, le pourcentage de monotonie souhaité pour chaque variable monotone et le pourcentage d'arbres acceptés.

Les tests (table 4.6) montrent une légère augmentation de score f1 et de précision avec random forest monotone pour tous les jeux de données comme pour [56]. On peut voir que le pourcentage d'arbres acceptés varie pour chaque jeu de données, cette valeur dépend des paramètres $target_monotony$, nb_test_points et nb_test_values ainsi que du jeu de données. Les deux jeux de données sur lesquels l'amélioration est la plus grande sont $lubjana$ et $cleveland$. Le pourcentage d'arbres accepté est différent pour ces deux cas (15% et 2%), à première vue le pourcentage d'arbres acceptés n'est pas le seul facteur d'amélioration des scores f1 et de la précision.

Des tests ont aussi été effectués pour la régression random forest monotone avec dropouts. Des jeux de données provenant de STMicroelectronics sont utilisés. Les variables ne seront pas décrites pour des soucis de confidentialité. Les données représentent des problèmes de planification. Les jeux de données sont idle time, cycle time et operation time. L'objectif est de prédire des temps d'inactivité machine, des temps de cycle et des retards sur certaines

Dataset	F1 Score		accuracy		parameters			
	Standard	Monotone	Standard	Monotone	target_monotony	nb_test_points	nb_test_values	Approved trees
abalone	0.780	**0.781**	0.780	**0.781**	[0.995, 0.95, 0.995, 0.995, 0.98, 0.98, 0.98]	100	10	8%
german_numeric	0.868	**0.869**	0.871	**0.872**	[0.98, 0.99, 0.98, 0.98, 0.98, 0.98, 0.99, 0.99, 1]	200	10	55%
indian_diabetes	0.756	**0.761**	0.758	**0.764**	[0.97, 0.95, 0.97, 0.97, 0.97, 0.95, 0.97, 0.97]	100	10	22%
cleveland	0.737	**0.749**	0.74	**0.75**	0.98	200	[20, 2, 2, 2, 20, 10, 2, 2, 4 , 20]	15%
sa_heart	0.681	**0.687**	0.690	**0.691**	[0.96, 0.96, 0.96, 0.98, 0.98, 0.98, 0.97, 0.97]	100	[10, 10, 10, 10, 2, 10, 10, 10]	12 %
seismic	0.905	**0.906**	0.928	0.928	[0.99, 0.98, 0.99, 0.99, 0.99, 0.98, 0.99, 0.99, 0.99]	100	[2, 10, 10, 10, 9, 8, 4, 10, 10]	5%
haberman	0.672	**0.677**	0.673	**0.677**	[0.98, 0.92, 0.95]	100	20	15%
ljubljana	0.685	**0.694**	0.683	**0.692**	[0.98, 0.98, 0.95, 0.98, 0.98]	100	[10 , 10 , 3 , 10 ,2]	2%
car	0.976	**0.979**	0.976	**0.979**	[0.99, 1, 0.99, 1]	100	4	3%

TABLE 4.6 – Comparaison entre random forest standard ("Standard") et la random forest monotone avec drop out ("Monotone").

opérations en fonction d'attributs de planification. En analysant les données au préalable et en utilisant les connaissances sur celles ci, les pourcentages de monotonie sont déterminés pour chaque variable indépendamment. Random forest standard et random forest monotone sont utilisés et on compare les résultats :

TABLE 4.7 – Résultats monotonie : cas STMicroelectronics

Jeu de données	r^2 standard	r^2 monotone	% trees
IdleTime	**0.981**	0.979	2.52
CycleTime	0.807	**0.818**	0.17
OperationTime	0.978	**0.989**	5.44

Pour le cas IdleTime, le régresseur random forest standard a un meilleur r^2 que la random forest avec monotonie. Pour les cas CycleTime et OperationTime c'est l'inverse, random forest monotone a un meilleur r^2. Le pourcentage d'arbres sélectionnés est très faible surtout pour CycleTime. C'est parce ce que le pourcentage de monotonie cible choisi est très élevé pour les différentes variables. C'est aussi parce ce que la monotonie des arbres est très variable dans ce cas. Ci-dessous les tableaux de variations pour les cas IdleTime et CycleTime :

variable	min	max	std
0	92.8	100	1.62103
1	90.7	100	2.10201
2	97.5	100	0.502867

FIGURE 4.21 – Tableau de variation pour le cas IdleTime

variable	min	max	std
0	84.3	100	3.0117
1	90.1	100	1.69058
2	90.1	100	2.17286
3	90	100	2.2627
4	84.7	100	3.78268

FIGURE 4.22 – Tableau de variation pour le cas CycleTime

La monotonie des arbres pour chaque variable a une plus forte variation dans le cas CycleTime que le cas IdleTime. Contraindre la monotonie dans un cas comme CycleTime ou la monotonie des arbres est variable permettra d'obtenir un modèle final plus robuste. La capacité de la random forest monotone à améliorer le modèle peut être liée à cette variation de monotonie des arbres pour une forêt non contrainte.

L'intégration de connaissances n'a pas comme prétention d'améliorer grandement la précision mais de rendre le modèle plus robuste (au bruit et aux outliers notamment). La robustesse d'un modèle est difficile à calculer avec des métriques. L'amélioration apportée par random forest monotone est visible sur quelques rare points :

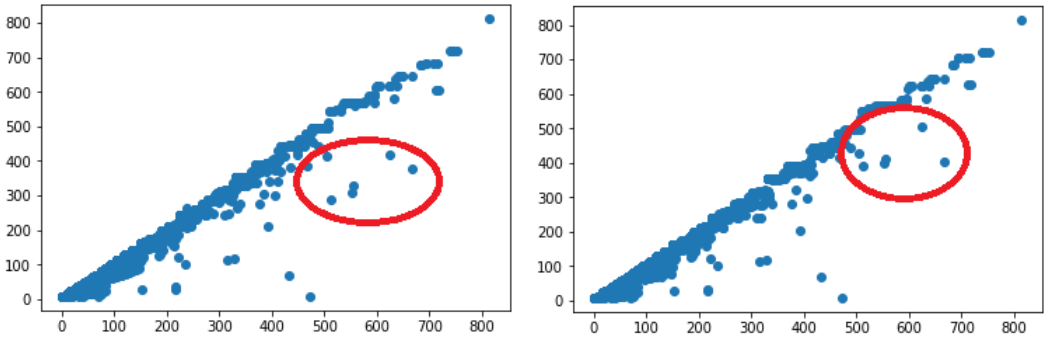


FIGURE 4.23 – Exemple cas OperationTime : y vs prediction (à gauche random forest standard et à droite random forest monotone

Certaines prédictions non monotones du modèle random forest standard ont été améliorées par la random forest monotone comme le montrent les graphiques. Ces points éloignés étaient sous estimés par le régresser standard et la monotonie permet de les ajuster.

4.6.1 Régions de décisions

Tout d'abord, considérons un exemple simple constitué de $n = 100$ points $(\mathbf{x}_i)$ uniformément tirés dans $[-1, 1] \times [-1, 1]$. Les exemples sont affectés à la classe 1 s'ils sont dans l'orthant positif et -1 sinon ; c'est-à-dire que $y_i = 1$ si et seulement si $\mathbf{x}_i^1 \geq 0$ et $\mathbf{x}_i^2 \geq 0$. Les deux caractéristiques 1 et 2 sont donc positivement monotones.

Nous entraînons des forêts aléatoires sur ces données avec `n_estimateurs` $= 20$ et `max_depth` $= 5$ arbres. La forêt aléatoire standard et notre approche atteignent toutes deux une précision d'apprentissage de 100%. Cependant, même dans ce cas très simple, des arbres non monotones sont générés, comme le montre la figure 4.24. Plus précisément, notre test aléatoire identifie 13% des arbres comme non monotones avec 50 de points de test et de valeurs de test. Ces arbres (et en particulier celui de la Figure 4.24) correspondent clairement à un surajustement des données d'apprentissage.

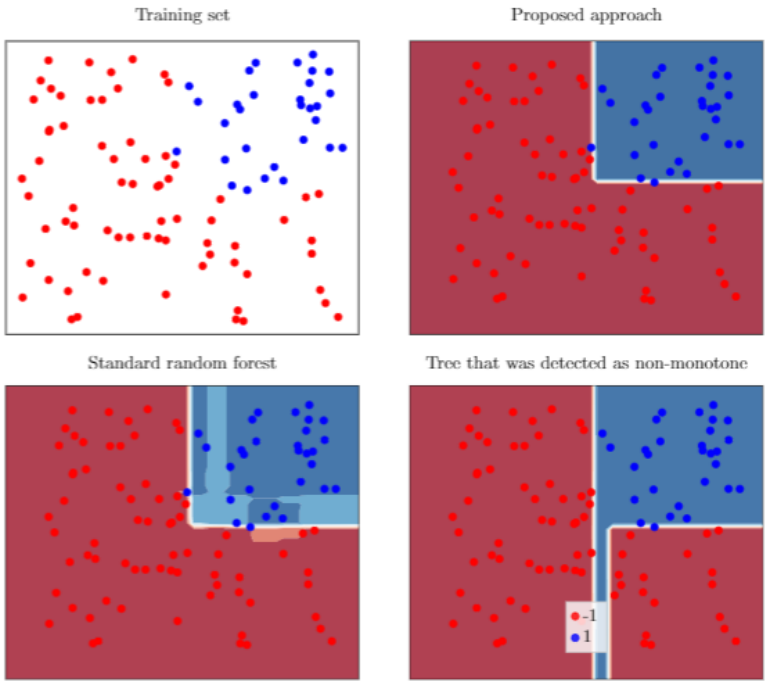


FIGURE 4.24 – Régions de décisions des classificateurs random forest pour un cas monotone.

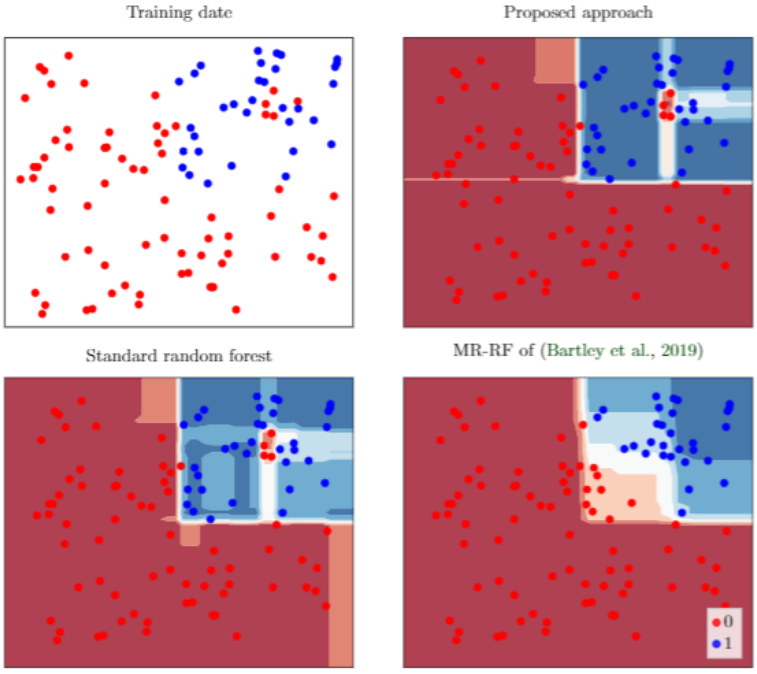


FIGURE 4.25 – Régions de décisions des classificateurs random forest pour un cas monotone avec 5 outliers.

Maintenant, nous modifions cet ensemble de données en ajoutant 5 points dans l'orthant positif mais avec la classe -1 afin de simuler des données bruitées ou incorrectement étiquetées. Dans cette situation, le nombre d'arbres identifiés comme non monotones passe à 40%. Dans cette situation, le score f1 d'apprentissage du classificateur random forest standard et random forest monotone chute à $0,985$. Notre approche monotone avec une monotonie cible une monotonie de 98% par défaut et ne conduit pas à un classifieur globalement monotone comme on le voit sur la figure 4.25 mais conserve une bonne performance de prédiction. Cela s'explique par notre choix de ne pas éliminer strictement la non-monotonie mais plutôt d'essayer d'éviter de grandes régions de l'espace avec des comportements non monotones (une alternative serait d'augmenter la monotonie cible et le nombre de points de test mais cela conduirait à un surcoût de calcul important).

Ce point de vue le rend différent et complémentaire du MR-RF de [56] dans son approche. En utilisant le code disponible sur `https://github.com/chriswbartley/monoensemble` avec les caractéristiques par défaut, nous voyons sur la Figure 4.25 que les régions de décision obtenues sont effectivement plus monotones mais au prix d'un score f1 d'apprentissage diminué de $0,774$ (vs $0,985$).

4.6.2 Conclusion sur l'intégration de connaissances

.

La random forest avec symétrie ne peut pas être utilisée sur le cas 4.3 qui a motivé ce travail. Aucun cas n'a été trouvé pour la version symétrique de random forest. En revanche c'est ce travail sur la symétrie qui a permis le développement de la méthode dropouts. Nous avons réorienté les recherches vers la monotonie car c'est une forme de données commune dans l'industrie. Ce choix a aussi été motivé par l'existence d'articles sur le sujet [55] [56] [53] [57]. Ces recherches ont abouti en une méthode utilisant random forest permettant de contraindre la monotonie de manière "douce" (la méthode dropouts). Cette méthode fonctionne pour n'importe quelle contrainte tant qu'il existe un test pour cette contrainte à effectuer sur les arbres. Les scores de précision et la fidélité à la distribution du jeu de données de test de la random forest monotone sont meilleurs que pour la random forest standard sur les jeux de tests générées 4.7. La random forest monotone a obtenu des scores de précision similaires à la random forest standard pour les jeux de données réels libres. Les résultats sont proches de ceux présentés dans [55] et peuvent être améliorés avec plus de feature engineering et d'analyse de données préalable. Le jeu de données CycleTime de STMicroelectronics semble un cas intéressant pour la random forest monotone. En effet il existe plusieurs relations monotones entre les variables explicatives et la variable de réponse et les arbres générés par la random forest standard ont une forte variation en terme de monotonie. Ce sujet de recherche doit être approfondi pour une utilisation industrielle. Il est nécessaire de tester la random forest monotone sur d'autres cas réels pour pouvoir ressortir des statistiques significatives sur l'amélioration du modèle en fonction de la variabilité de la monotonie des arbres non contraints. De manière générale l'intégration de connaissances à priori dans des algorithmes de machine learning pourrait permettre une plus grande utilisation du machine learning dans l'industrie. Elle permet d'utiliser le savoir acquis par l'entreprise pour améliorer la robustesse des algorithmes de machine learning. Elle permet aussi une plus grande confiance en ces algorithmes souvent perçus comme des boite noires [66] en assurant le respect de certaines contraintes connues là où un algorithme standard pourrait trouver des relations impossibles entre les variables. L'intégration de connaissances est donc une étape possible vers une "explainable AI" utilisable en industrie.

Chapitre 5

Conclusions et perspectives

5.1 Conclusions

Le fabrication de puces électroniques est un processus long et complexe. A partir d'une plaque de silicium des centaines d'étapes vont s'enchaîner pour arriver à une plaque terminée sur laquelle on trouve des milliers de puces. La taille des puces est de plus en plus petite et les méthodes de fabrication sont de plus en plus complexes. Les produits se diversifient et leur durée de vie se réduit. A toutes ces contraintes s'ajoute un objectif de 100% qualité. Assurer la qualité dans ce contexte avec la progression technologique des puces est coûteux et difficile. Différents services sont responsables de la maîtrise du processus, le service métrologie qui utilise différentes techniques pour mesurer les indicateurs de qualité tout au long de la fabrication, le service process control qui utilise des méthodes statistiques pour analyser les données mesurées par le service métrologie, la défectivité détecte les défauts sur la plaque terminée puis les tests électriques (EWS) sont effectués pour mesurer le rendement. Les coûts de ces différents services augmentent, notamment les coûts de métrologie. En effet la proportion d'étapes de mesures a augmenté pour dépasser la proportion d'étapes de fabrication.

Pour réduire ces coûts la recherche de nouvelles méthodes pour le contrôle de qualité est nécessaire. Tout problème détecté en amont représente un gain important car le temps de cycle est long. La recherche se tourne vers les méthodes de machine learning qui ont fait leurs preuves dans de nombreux secteurs. L'utilisation du machine learning pour l'industrie du semiconducteur représentera 10% du chiffre d'affaires en réduisant les coûts à toutes les étapes, de la recherche à la vente. Il permettra notamment de faire de la maintenance prédictive, prédire les évènements de non qualité et trouver les causes de ces évènements. Pour cette thèse nous nous sommes intéressés aux méthodes de machine learning pour assurer la qualité pendant les étapes de production et plus précisément aux méthodes de détection automatique des signaux faibles et à l'analyse de root cause.

Les méthodes de machine learning doivent être adaptées à l'industrie du semiconducteur pour être utilisées. Des contraintes spécifiques comme la fragmentation des données qui résulte du high mix low volume ou le déséquilibre entre les cas bons et mauvais doivent être prises en compte. Pour que des nouvelles méthodes soit utilisées il faut éviter les boites noires qui empêchent de comprendre le fonctionnement, d'analyser les résultats en détails et donc de faire confiance à une méthode. L'intégration de ces méthodes dans des applications faciles d'utilisation pour des utilisateurs qui n'ont pas de connaissances en machine learning est aussi un objectif. Les recherches effectuées pendant cette thèse répondent à ces contraintes spécifiques.

Une nouvelle source de données a motivé les recherches en détection automatique de signaux faibles. Cette nouvelle source de données sont les machines de fabrication, plus

spécifiquement les équipement de lithographie. Les équipements de lithographie sont équipés de sensors qui mesurent des données de leveling, un total de 35 000 points sont mesurés sur chaque plaque (pas de sampling). Ces données étaient utilisées uniquement pour le paramétrage de la machine puis supprimées. Un travail d'extraction et de stockage des données a été fait pour que ces données soit utilisables. Utiliser la machine de lithographie comme machine de mesure offre une métrologie gratuite. La détection d'évènements de non qualité avec ces données représentera un gain important car la détection a un coût nul.

Les données étant fragmentées et le nombre d'évènements de non qualité très rare (1/1000) il est impossible d'utiliser une méthode supervisée. C'est pour cela que le choix de DBSCAN, un algorithme de classification non supervisé a été fait. Pour répondre à la problématique spécifique au leveling de variation de la variation sur la plaque, une version "ByPixel" de DBSCAN a été développée. Différentes sous méthodes ont été développées comme "Zernike data cleaning" pour supprimer les effets optiques des données et la détection du nombre de clusters avec DBSCAN. Une partie de ces méthodes a été utilisée pour le développement d'un système d'alerte par mail qui prévient les utilisateurs quand un évènement de non qualité est détecté.

Après avoir détecté un problème, comme un signal faible en leveling, il faut trouver sa cause. La problématique de détection de root cause est omniprésente à STMicroelectronics et dans le secteur du semiconducteur en général. La complexité grandissante de la fabrication rend la tâche plus compliquée, le nombre d'étapes a vérifier est très grand. Les problèmes de détection de root cause contiennent beaucoup d'attributs (les colonnes) pour peu de lignes (les wafers passés sur ces étapes), surtout quand l'excursion est détectée en fin de fabrication. Les méthodes statistiques utilisées jusqu'à maintenant commencent à montrer leur limites sur ces jeux de données complexes. C'est pour cela que nous avons développé la méthode Random Forest Discriminant Analysis. Cette méthode basée sur l'algorithme de machine learning Random Forest permet de détecter la root cause d'une excursion même sur un jeu de données complexe.

Dans l'optique de développer des algorithmes de machine learning adaptés à l'industrie nous avons décidé de rechercher les méthodes d'intégration de connaissances dans les algorithmes. L'idée est d'utiliser les connaissances à priori sur la forme des données pour améliorer la performance de l'algorithme. Deux formes de données ont été identifiées, la symétrie et la monotonie. Nous avons montré que la symétrie des données n'était pas une forme répandue contrairement à la monotonie. Après de nombreuses modifications et tests l'algorithme random forest monotone a permis une augmentation de 1% de précision environ sur des cas STMicroelectronics de planification. Une augmentation de 1% peut paraître futile mais elle cette augmentation correspond à des cas "extrêmes" de mauvaises prédictions. Éviter les rares fois où l'algorithme prédit une valeur très mauvaise peut prévenir la prise de mauvaises décisions qui pourraient entraîner de plus gros problèmes. Il est plus simple de faire confiance à un algorithme intégrant des connaissance métiers sur les données.

Chacune des méthodes développées est une adaptation de machine learning à l'industrie. Nous avons développé en essayant de garder les techniques les plus simples possibles et en évitant les boites noires pour permettre la réutilisation de celles ci. Ces recherches sont une étape vers une entreprise data driven où le machine learning est présent à toutes les étapes de fabrication pour accompagner la production. L'industrie data driven est une étape est l'industrie 4.0, la quatrième révolution industrielle qui sera la convergence entre le monde virtuel et les produits et objets du monde réel. L'industrie 4.0 devra proposer aux clients des produits toujours plus spécifiques et maintenir les gains même avec de faibles volumes de fabrication.

5.2 Perspectives

Ce travail de thèse ouvre de nombreuses perspectives ; à court terme et à plus long terme. La méthode **BP-DBSCAN**, méthode développée pour répondre à une problématique de variation de la variance du leveling, n'a pas été utilisée directement sur les données de topographie des plaquettes car la variation de ces données a pu être supprimée par calcul. Mais l'idée d'utiliser un algorithme de détection non supervisé pour détecter des outliers pourra servir pour d'autres sujets comme l'analyse des signatures "déformation du chuck". L'avantage de cette méthode est qu'elle fonctionne par densité et a donc un jugement proche de l'humain ce qui rend les résultats plus interprétables. La méthode **Zernike data clean** peut être utilisée pour laver les signatures à forme optique qui sont fréquentes dans le milieu des semiconducteurs. **DBSCAN cluster count** est une simple utilisation de DBSCAN pour compter le nombre de clusters après une classification. La méthode est utilisée dans l'application d'alerte et fonctionne pour toute classification par distance mesurable. La classification de signatures de défauts, Random Forest defect signatures classifier, permet de classifier facilement différents types de signature défauts connus sur des plaquettes. L'avantage de cette méthode sur une classification par image est la création des variables qui est personalisable et facile d'utilisation.

La méthode présentée dans le chapitre 2 est **Random Forest Discriminant Analysis** elle sera intégrée dans un application SpotFire qui est le logiciel d'analyse de données à STMicroelectronics et pourra être utilisée par tous les employés de Crolles pour rechercher et analyser la cause d'un problème de tout type. L'avantage de la méthode est la construction de plusieurs modèles random forest successifs avec réduction du nombre de variables et en vérifiant de la précision de ces modèles ce qui permet de détecter la ou les causes d'un problème automatiquement (problème mono-cause et problèmes multi-causes). Cette fonctionnalité, couplée avec les performances de random forest, permet aussi de traiter des jeux de de données avec un grand nombre de variables (causes potentielles), ce qui fait de RF-DA un outil pérenne pour l'industrie du semiconducteur où le nombre d'opérations est élevé et est en augmentation. Cet outil peut aussi être utilisé dans d'autres secteurs avec un besoin de trouver le facteur déterminant parmi un grand nombre de candidats potentiels.

De manière générale l'intégration de connaissances dans des algorithmes de machine learning pourrait permettre une plus grande utilisation du machine learning dans l'industrie. Elle permet d'utiliser le savoir acquis par l'entreprise pour améliorer la robustesse des algorithmes de machine learning. L'algorithme random forest monotone a été utilisé à STMicroelectronics sur des problèmes de prédiction en planification, il peut être utile pour la classification et la prédiction avec tous les jeux de données avec une relation monotone entre certaines variables explicatives et la réponse. Pour contraindre la monotonie la méthode drop out est utilisée, cette méthode est évolutive et permet de contraindre n'importe quelle connaissance si un test peut être créé. Il serait possible d'ajouter plusieurs types de contraintes différentes à la monotonie et la symétrie pour construire des modèles prenant en compte différentes connaissances sur différentes variables. Un tel modèle serait très utile dans l'industrie de par sa robustesse aux petits échantillons et aux outliers.

Bibliographie

[1] Guojun Wen, Zhijun Gao, Qi Cai, Yudan Wang, and Shuang Mei. A novel method based on deep convolutional neural networks for wafer semiconductor surface defect inspection. *IEEE Transactions on Instrumentation and Measurement*, 69(12) :9668–9680, 2020.

[2] O Roule, F Pasqualini, and M Borde. Industrial implementation of spatial variability control by real-time spc. In *32nd European Mask and Lithography Conference*, volume 10032, page 100320P. International Society for Optics and Photonics, 2016.

[3] MinGyu Kim, Jaewuk Ju, Boris Habets, Georg Erley, Enrico Bellmann, and Seop Kim. Excursion detection using leveling data. In *Metrology, Inspection, and Process Control for Microlithography XXX*, volume 9778, page 97783O. International Society for Optics and Photonics, 2016.

[4] Gian Antonio Susto, Simone Pampuri, Andrea Schirru, Alessandro Beghi, and Giuseppe De Nicolao. Multi-step virtual metrology for semiconductor manufacturing : A multi-level and regularization methods-based approach. *Computers & Operations Research*, 53 :328–337, 2015.

[5] Kamran Khan, Saif Ur Rehman, Kamran Aziz, Simon Fong, and Sababady Sarasvady. Dbscan : Past, present and future. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, pages 232–238. IEEE, 2014.

[6] AM Fahim, AM Salem, FA Torkey, and MA Ramadan. Density clustering based on radius of data (dcbrd). *World Academy of Science, Engineering and Technology*, 2006.

[7] Peng Liu, Dong Zhou, and Naijun Wu. Vdbscan : varied density based spatial clustering of applications with noise. In *2007 International conference on service systems and service management*, pages 1–4. IEEE, 2007.

[8] Yasser El-Sonbaty, Mohamed A Ismail, and Mohamed Farouk. An efficient density based clustering algorithm for large databases. In *16th IEEE international conference on tools with artificial intelligence*, pages 673–677. IEEE, 2004.

[9] Bing Liu. A fast density-based clustering algorithm for large databases. In *2006 International Conference on Machine Learning and Cybernetics*, pages 996–1000. IEEE, 2006.

[10] Ozge Uncu, William A Gruver, Dilip B Kotak, Dorian Sabaz, Zafeer Alibhai, and Colin Ng. Gridbscan : Grid density-based spatial clustering of applications with noise. In *2006 IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 2976–2981. IEEE, 2006.

[11] Bhogeswar Borah and Dhruba K Bhattacharyya. An improved sampling-based dbscan for large spatial databases. In *International conference on intelligent sensing and information processing, 2004. proceedings of*, pages 92–96. IEEE, 2004.

[12] Anant Ram, Ashish Sharma, Anand S Jalal, Ankur Agrawal, and Raghuraj Singh. An enhanced density based spatial clustering of applications with noise. In *2009 IEEE International Advance Computing Conference*, pages 1475–1478. IEEE, 2009.

[13] B Borah and DK Bhattacharyya. A clustering technique using density difference. In *2007 International Conference on Signal Processing, Communications and Networking*, pages 585–588. IEEE, 2007.

[14] Chen Xiaoyun, Min Yufang, Zhao Yan, and Wang Ping. Gmdbscan : multi-density dbscan cluster based on grid. In *2008 IEEE International Conference on e-Business Engineering*, pages 780–783. IEEE, 2008.

[15] P Viswanath and Rajwala Pinkesh. l-dbscan : A fast hybrid density based clustering method. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 1, pages 912–915. IEEE, 2006.

[16] Derya Birant and Alp Kut. St-dbscan : An algorithm for clustering spatial–temporal data. *Data & knowledge engineering*, 60(1) :208–221, 2007.

[17] Shaaban Mahran and Khaled Mahar. Using grid for accelerating density-based clustering. In *2008 8th IEEE International Conference on Computer and Information Technology*, pages 35–40. IEEE, 2008.

[18] Xiaopeng Yu, Deyi Zhou, and Yan Zhou. A new clustering algorithm based on distance and density. In *Proceedings of ICSSSM'05. 2005 International Conference on Services Systems and Services Management, 2005.*, volume 2, pages 1016–1021. IEEE, 2005.

[19] Bin Wang, Michael Tanksalvala, Zhe Zhang, Yuka Esashi, Nicholas W. Jenkins, Margaret M. Murnane, Henry C. Kapteyn, and Chen-Ting Liao. A new metrology technique for defect inspection via coherent Fourier scatterometry using orbital angular momentum beams. In Ofer Adan and John C. Robinson, editors, *Metrology, Inspection, and Process Control for Semiconductor Manufacturing XXXV*, volume 11611, pages 66 – 80. International Society for Optics and Photonics, SPIE, 2021.

[20] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning : Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5) :1–36, 2018.

[21] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782) :350–354, 2019.

[22] Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J Anders, and Klaus-Robert Müller. Toward interpretable machine learning : Transparent deep neural networks and beyond. *arXiv e-prints*, pages arXiv–2003, 2020.

[23] Alexandre Moritz, Stéphane Dauzère-Pérès, Oussama Ben-Ammar, and Philippe Vialletelle. Maintenance with production planning constraints in semiconductor manufacturing. In *2020 Winter Simulation Conference (WSC)*, pages 1921–1930. IEEE, 2020.

[24] Jovani Dalzochio, Rafael Kunst, Edison Pignaton, Alecio Binotto, Srijnan Sanyal, Jose Favilla, and Jorge Barbosa. Machine learning and reasoning for predictive maintenance in industry 4.0 : Current status and challenges. *Computers in Industry*, 123 :103298, 2020.

[25] Jong-Chih Chien, Ming-Tao Wu, and Jiann-Der Lee. Inspection and classification of semiconductor wafer surface defects using cnn deep learning networks. *Applied Sciences*, 10(15) :5340, 2020.

[26] Chia-Yu Hsu and Wei-Chen Liu. Multiple time-series convolutional neural network for fault detection and diagnosis and empirical study in semiconductor manufacturing. *Journal of Intelligent Manufacturing*, 32 :823–836, 2021.

[27] Wang Yong, Chen Xu, and Wei Zhengying. Fault detection of sensor data in semiconductor processing with variational autoencoder neural network. In *2020 China Semiconductor Technology International Conference (CSTIC)*, pages 1–3. IEEE, 2020.

[28] Wang Yong, Lu Jingjing, Chen Xu, and Wei Zhengying. Fault detection of sensor data in semiconductor processing using neural network with dynamic time wrapping loss. In *2021 China Semiconductor Technology International Conference (CSTIC)*, pages 1–3. IEEE, 2021.

[29] Q Peter He and Jin Wang. Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes. *IEEE transactions on semiconductor manufacturing*, 20(4) :345–354, 2007.

[30] Ghislain Verdier and Ariane Ferreira. Adaptive mahalanobis distance and k-nearest neighbor rule for fault detection in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 24(1) :59–68, 2010.

[31] Zhe Zhou, Chenglin Wen, and Chunjie Yang. Fault detection using random projections and k-nearest neighbor rule for semiconductor manufacturing processes. *IEEE Transactions on Semiconductor Manufacturing*, 28(1) :70–79, 2014.

[32] Jianbo Yu. Fault detection using principal components-based gaussian mixture model for semiconductor manufacturing processes. *IEEE Transactions on Semiconductor Manufacturing*, 24(3) :432–444, 2011.

[33] Ilham Rabhi, Agnès Roussy, François Pasqualini, and Cyril Alegret. Out-of-control detection in semiconductor manufacturing using one-class support vector machines. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 1628–1633. IEEE, 2021.

[34] Bernd Waschneck, André Reichstaller, Lenz Belzner, Thomas Altenmüller, Thomas Bauernhansl, Alexander Knapp, and Andreas Kyek. Deep reinforcement learning for semiconductor production scheduling. In *2018 29th annual SEMI advanced semiconductor manufacturing conference (ASMC)*, pages 301–306. IEEE, 2018.

[35] In-Beom Park, Jaeseok Huh, Joongkyun Kim, and Jonghun Park. A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities. *IEEE Transactions on Automation Science and Engineering*, 17(3) :1420–1431, 2019.

[36] Yair Meidan, Boaz Lerner, Gad Rabinowitz, and Michael Hassoun. Cycle-time key factor identification and prediction in semiconductor manufacturing using machine learning and data mining. *IEEE transactions on semiconductor manufacturing*, 24(2) :237–248, 2011.

[37] Leo Breiman. Random forests. *Machine learning*, 45(1) :5–32, 2001.

[38] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25(2) :197–227, 2016.

[39] Lior Rokach and Oded Maimon. Decision trees. In *Data mining and knowledge discovery handbook*, pages 165–192. Springer, 2005.

[40] Badr Hssina, Abdelkarim Merbouha, Hanane Ezzikouri, and Mohammed Erritali. A comparative study of decision tree id3 and c4. 5. *International Journal of Advanced Computer Science and Applications*, 4(2) :13–19, 2014.

[41] John Francis Valley, Noel Poduje, Jaydeep Sinha, Neil Judell, Jie Wu, Marc Boonman, Sjef Tempelaars, Youri van Dommelen, Hans Kattouw, Jan Hauschild, et al. Approaching new metrics for wafer flatness : an investigation of the lithographic consequences of wafer non-flatness. In *Metrology, Inspection, and Process Control for Microlithography XVIII*, volume 5375, pages 1098–1108. International Society for Optics and Photonics, 2004.

[42] Vasudevan Lakshminarayanan and Andre Fleck. Zernike polynomials : a guide. *Journal of Modern Optics*, 58(7) :545–561, 2011.

[43] Hari Krishna Kanagala and VV Jaya Rama Krishnaiah. A comparative study of k-means, dbscan and optics. In *2016 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6. IEEE, 2016.

[44] Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas. C-dbscan : Density-based clustering with constraints. In *International workshop on rough sets, fuzzy sets, data mining, and granular-soft computing*, pages 216–223. Springer, 2007.

[45] Aabir Chouichi, Jakey Blue, Claude Yugma, and Francois Pasqualini. Heterogranular multivariate analytics for detecting and controlling the root causes of the mismatching machines in semiconductor manufacturing. In *2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, pages 334–339. IEEE, 2018.

[46] Loong Chuen Lee, Choong-Yeun Liong, and Abdul Aziz Jemain. Partial least squares-discriminant analysis (pls-da) for classification of high-dimensional (hd) data : a review of contemporary practice strategies and knowledge gaps. *Analyst*, 143(15) :3526–3539, 2018.

[47] Huan Liu and Rudy Setiono. Chi2 : Feature selection and discretization of numeric attributes. In *Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence*, pages 388–391. IEEE, 1995.

[48] Ira Cohen, Jeffrey S Chase, Moises Goldszmidt, Terence Kelly, and Julie Symons. Correlating instrumentation data to system states : A building block for automated diagnosis and control. In *OSDI*, volume 4, pages 16–16, 2004.

[49] Dhan V Sagar, P Bagavathi Sivakumar, and R Vijay Anand. Random forest and change point detection for root cause localization in large scale systems. In *2014 IEEE International Conference on Computational Intelligence and Computing Research*, pages 1–5. IEEE, 2014.

[50] Alexander Detzner and Martin Eigner. Feature selection methods for root-cause analysis among top-level product attributes. *Quality and Reliability Engineering International*, 37(1) :335–351, 2021.

[51] Rajiv D Banker and Ajay Maindiratta. Maximum likelihood estimation of monotone and concave production frontiers. *Journal of Productivity Analysis*, 3(4) :401–415, 1992.

[52] Glenn Fung, Olvi L Mangasarian, and Jude W Shavlik. Knowledge-based support vector machine classifiers. In *NIPS*, pages 521–528. Citeseer, 2002.

[53] Yongqiao Wang and He Ni. Multivariate convex support vector regression with semidefinite programming. *Knowledge-Based Systems*, 30 :87–94, 2012.

[54] Kristiaan Pelckmans, Marcelo Espinoza, Jos De Brabanter, Johan AK Suykens, and Bart De Moor. Primal-dual monotone kernel regression. *Neural Processing Letters*, 22(2) :171–182, 2005.

[55] Chris Bartley, Wei Liu, and Mark Reynolds. A novel technique for integrating monotone domain knowledge into the random forest classifier. In *Fourteenth Australasian Data Mining Conference (AusDM 2016)*, volume 170, 2016.

[56] Christopher Bartley, Wei Liu, and Mark Reynolds. Enhanced random forest algorithms for partially monotone ordinal classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3224–3231, 2019.

[57] Rashmi Korlakai Vinayak and Ran Gilad-Bachrach. Dart : Dropouts meet multiple additive regression trees. In *Artificial Intelligence and Statistics*, pages 489–497. PMLR, 2015.

[58] 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.

[59] Arie Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, 19(1) :29–43, 1995.

[60] Sergio González, Francisco Herrera, and Salvador García. Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity. *New Generation Computing*, 33(4) :367–388, 2015.

[61] Matt Bonakdarpour, Sabyasachi Chatterjee, Rina Foygel Barber, and John Lafferty. Prediction rule reshaping. In *International Conference on Machine Learning*, pages 630–638. PMLR, 2018.

[62] Erwan Scornet, Gérard Biau, and Jean-Philippe Vert. Consistency of random forests. *The Annals of Statistics*, 43(4) :1716–1741, 2015.

[63] Wei Gao and Zhi-Hua Zhou. Towards convergence rate analysis of random forests for classification. *Advances in Neural Information Processing Systems*, 33, 2020.

[64] Gérard Biau, Luc Devroye, and Gäbor Lugosi. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9(9), 2008.

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

[66] Andreas Holzinger, Peter Kieseberg, Edgar Weippl, and A Min Tjoa. Current advances, trends and challenges of machine learning and knowledge extraction : from machine learning to explainable ai. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 1–8. Springer, 2018.