UQAC
Université du Québec
à Chicoutimi

AUTOMATISATION DU PRE-TRAITEMENT DES DONNEES PAR L'OPTIMISATION METAHEURISTIQUE

PAR ISSOUF OUEDRAOGO

MÉMOIRE PRÉSENTÉ À L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI EN VUE DE L'OBTENTION DU GRADE DE MAITRE ES SCIENCES (M. SC.) EN INFORMATIQUE PROFIL RECHERCHE

Québec, Canada

RÉSUMÉ

Depuis quelques années, plusieurs entreprises ont commencé à exploiter les données qu'ils ont emmagasinées pour pouvoir extraire des connaissances et faire des prévisions. Une bonne utilisation de ces données permet aux entreprises de comprendre leurs clients et pouvoir améliorer leurs services. Les techniques d'intelligence artificielle se présentent comme des moyens pour révolutionner les prises de décisions. Plus d'entreprises embauchent des data scientists pour concevoir des modèles de prédiction. Le but des data scientists est de concevoir des modèles et de les performer. L'un des défis majeurs rencontré par ces data scientists est le prétraitement des données. Le prétraitement des données impact la qualité du modèle mis en place. Ce problème est dû au fait qu'il n'existe pas une seule bonne manière de prétraité les données. Les techniques utilisées dépendent du problème et du type de donnée. Ce mémoire a pour objectif d'évaluer la possibilité de développer une solution qui permet d'automatiser les opérations de prétraitement des données pour les acteurs de données (data scientistes, chercheurs, etc.). Nous utiliserons le terme pipeline afin de nommer la chaîne de pré-traitement et de traitement des données (de la donnée brute jusqu'à la classification de la donnée). En effet, nous proposons un pipeline qui permet de choisir les meilleures techniques de prétraitements en fonction du problème. Plusieurs techniques sont répertoriées et nous utilisons des techniques d'optimisation métaheuristique pour minimiser le temps de recherche du meilleur modèle. Ces techniques sont par la suite comparées avec des techniques d'optimisation standard comme la recherche par grille. Le pipeline est par la suite testé sur des données d'une entreprise d'assurance automobile du Canada. Les résultats nous montrent que les méthodes d'optimisation métaheuristique sont plus efficaces sur des *big data*. Elles ont permis de diminuer d'environ 30% le temps nécessaire pour retrouver un bon modèle. En plus, le pipeline se présente comme un outil qui permet au personnel expert ou non de passer moins de temps pour le prétraitement. Le pipeline applique de façon automatisé différentes techniques de prétraitement et propose celui qui convient.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS

AUC	:	AREA UNDER THE CURVE
CART	:	CLASSIFICATION AND REGRESSION TREES
ML	:	MACHINE LEARNING
MEA	:	MEAN ABSOLUTE ERROR
MDE	:	MOTOR DRIVE END
MSE	:	MEAN SQUARED ERROR
MNDE	:	MOTOR NON-DRIVE END
KNN	:	K NEAREST NEIGHBOR
MAR	:	MISSING AT RANDOM
MCAR	:	MISSING COMPLETELY AT RANDOM
MNAR	:	MISSING NOT AT RANDOM
GA	:	GENETIC ALGORITHM
GLMS	:	GENERALIZED LINEAR MODELS
GWO	:	GREEN WOLF OPTIMISER
RF	:	RANDOM FORESTS
RMSE	:	ROOT MEAN SQUARED ERROR
UCI	:	UNIVERSITY OF CALIFORNIA
UQAC	:	UNIVERSITE DU QUEBEC A CHICOUTIMI
SVM	:	SUPPORT VECTOR MACHINES
DIM	:	DEPARTEMENT D'INFORMATIQUE ET DE MATHEMATIQUE

DÉDICACE

Je dédie ce travail à ma famille et à mes amis.
Merci pour le soutien et l'affection qui m'ont été offerts tout au long de ce parcours.
Sachez que j'en suis très reconnaissant.

REMERCIEMENTS

Je tiens à remercier tous ceux qui m'ont soutenu pendant la réalisation de ma maîtrise et la rédaction de ce mémoire. Je tiens à remercier également toute l'équipe du Département d'informatique et de mathématique (DIM) de l'UQAC pour tout l'attention dans les moments de compliqués tout au long de ce parcours. Plus particulièrement, j'aimerais remercier mon directeur de recherche Julien Maître pour son accompagnement et pour son expertise précieuse offerte tout au long de ma recherche. Je remercie aussi Bob-Antoine Jerry Ménélas et Bruno Bouchard pour m'avoir permis d'obtenir ce stage.

Je suis également reconnaissant à l'entreprise CO-OPERATORS pour son soutien financier, matériel et technique. CO-OPERATORS à mis à notre disposition un espace physique au sein de son bureau de Chicoutimi ainsi que plusieurs formations.

Je remercie également mes collègues de CO-OPERATORS particulièrement l'équipe de recherche et d'innovation pour le partage de leurs connaissances, pour leur aide et leur amitié durant ces deux dernières années. Finalement, j'aimerais remercier ma famille et mes amis pour leurs encouragements tout au long de mes étude

CHAPITRE 1

INTRODUCTION

1.1 CONTEXTE ET PROBLEMATIQUE

De nos jours, le prétraitement des données est l'une des étapes les plus chronophages et pertinentes du processus d'une analyse de données [1]. Une opération de prétraitement [2] de données peut avoir soit des impacts positifs ou soit des impacts négatifs sur l'analyse. Les utilisateurs experts des données ont des connaissances pour trouver les bonnes opérations de prétraitement basées sur leurs expériences. Cependant, lorsqu'il s'agit de non-experts dans ces mêmes données, il est difficile pour eux de trouver les bonnes opérations qui auraient un impact positif sur leur analyse, comme augmenter la valeur prédictive d'un modèle de classification.

Cette problématique de prétraitement des données est omniprésente chez les grands acteurs du domaine des assurances, comme que CO-OPERATORS, DESJARDINS, TD ASSURANCE qui exploitent de grands ensembles de données pouvant et devant être valorisées. Avec la révolution numérique accélérée[3] que nous observons et vivons, il serait regrettable de ne pas exploiter celle-ci afin de s'attaquer à cette problématique et de soutenir la croissance et la compétitivité chez CO-OPERATORS.

1.2 OBJECTIFS

L'objectif du présent projet de recherche est d'évaluer la possibilité de développer une solution reposant sur une intelligence artificielle afin d'automatiser les opérations de prétraitement des données de CO-OPERATORS. Cette solution, en plus d'automatiser le processus de pré-traitement, doit permettre de minimiser le temps passé sur cette étape dans l'analyse des données, d'améliorer ce processus[4]. Enfin, la solution se présentera comme un

---

[1] Par exemple : une tâche de classification, de régression, de clustering, etc.
[2] Par exemple : une tâche de normalisation ou d'encodage des données
[3] Une intégration totale de l'intelligence artificielle et la croissance exponentielle des quantités de données
[4] Par exemple obtenir les meilleures performances dans une tâche de classification

outil logiciel d'aide au prétraitement des données pour les non-experts et experts en science des données.

Plusieurs travaux en cours tentent d'apporter de nouvelles solutions afin d'assister les utilisateurs non-experts et experts dans les différentes étapes de la science des données. Ces travaux peuvent se diviser en 3 catégories distinctes :

- support pour le prétraitement des données ;
- support pour le forage de données ;
- support pour la découverte de connaissances[5] [1].

Dans ce projet de recherche, nous nous concentrerons sur le prétraitement des données afin d'optimiser les performances lors de l'analyse des données[6]. En effet, le prétraitement des données représente l'étape la plus importante selon la Data Science Methodology (DSM) [2] afin d'obtenir des modèles de classification ou de régression pouvant être déployés après un long processus de travail itératif. Le temps passé à prétraiter les données constitue environ 50% à 80% du temps total d'un projet d'apprentissage machine. C'est pourquoi il est essentiel de s'attarder un peu plus sur cette étape qui est la clé pour réussir un projet en science des données.

Pour la bonne conduite de ce projet nous nous posons des questions du genre Quoi ? Comment ? et Pourquoi ? Cette méthodologie nous emmène à nous intéresser d'abord aux tâches que le système doit supporter[7] et comment le système doit fournir l'assistance à savoir, soit automatiquement ou soit interactivement. Ensuite nous cherchons un moyen pour que le système puisse fournir cette assistance. En d'autres termes, nous cherchons à répondre à la question : comment les données transitent entre les différentes couches du systèmes de

[5] Encore appelés analyse des données
[6] p.ex. : découverte de règles, classification, prédiction, etc.
[7] p.ex. : conversion des formats des données en un format unique, nettoyage des données aberrantes, etc.

l'entrée à la sortie ? Enfin, quelle est l'intention réelle du système. Par exemple : est-ce pour impacter l'analyse des données[8] ou non ?

1.3 RÉSULTATS ET CONSTRIBUTIONS

Nous avons mis en place un système qui permet d'optimiser le processus de prétraitement et de choisir le meilleur modèle en fonction du problème. Chaque variable cible qui peut être classifié, peut-être supposé comme un problème à résoudre. Par conséquent, résoudre un problème revient à classifier une variable cible. Notre solution à la possibilité de créer plusieurs modèles et de choisir le meilleur. Le choix du meilleur modèle devient compliqué voire impossible quand le nombre de modèle à tester augmente. L'une des contributions est de rendre la tâche réalisable en utilisant une approche basée sur les métaheuristiques. Cette approche ne prend pas en compte toutes les combinaisons mais fournit des résultats intéressants en un temps record. Notre hypothèse est confirmée puisqu'en comparant les deux méthodes d'optimisation[9] sur une base de données réclamation de sinistres, nous retrouvons en 4 minutes un modèle qui permet de classifier les types de couvertures avec 74,0% de précision. Quant aux validations croisées, la méthode brute force prend en moyenne 14 minutes pour évaluer toutes les possibilités et le meilleur modèle a une précision de 74,1%. L'utilisation a vraiment diminué le temps de recherche et permet d'avoir une solution malgré la complexité du problème. Force est de reconnaitre que la deuxième méthode donne le meilleur modèle, mais le deuxième aussi peut atteindre ce meilleur résultat en peu de temps.

1.4 METHODOLOGIE

Le travail est réalisé en plusieurs étapes. Chaque étape est constituée de plusieurs sous tâches et vise un objectif bien précis. La durée moyenne d'une étape est de 12 semaines. Chaque étape donne lieu à un ou plusieurs livrables. Chaque livrable est validé avant de

[8] Les résultats d'apprentissage machine
[9] Basée sur la validation croisée et les métaheuristiques

passer à l'étape suivant. Un diagramme de Gantt présente les tâches à effectués ainsi que les délais prévisionnels. Les principales étapes de notre travail sont :

- **Étape 1 (Revue de la littérature et des outils rivaux)** : cette étape du projet a pour objectif de mieux comprendre le problème. Pour cela nous nous sommes documentés sur le problème. La méthodologie utiliser pour repêcher nos sujets est celle utilisée par la plupart des chercheurs [3]. Elle consiste à utiliser les bases de données et moteur de recherche comme « Google Scholar », « Sofia » ou « ResearchGate » avec une combinaison de mots-clés. Les mots-clés les plus utilisés sont :
  - automatisation du pré-traitement,
  - Optimisation,
  - Optimisation métaheuristique,
  - algorithmes d'apprentissage automatique,
  - pré-traitement et traitement des données,
  - traitement des données d'assurance avec l'intelligence artificielle.

  Ensuite, interroger les revues actuarielles comme RISKS *ou European Actuarial Journal (EAJ)* pour comprendre le domaine des assurances. Pour terminer, rechercher dans les références des contributions similaires de chaque article pertinent. Les articles repêcher seront classés en des sous-paragraphes. Chaque sous-paragraphes présente l'application d'une ou plusieurs techniques pour résoudre un problème.

- **Étape 2 (Contribution théorique) :** mettre en place une approche pour automatiser le pré-traitement des données par l'optimisation métaheuristique. Ensuite faire une étude comparative des méthodes d'optimisation standard et les méthodes d'optimisation métaheuristique.
- **Étape 3 (Contribution algorithmique) :** proposer un schéma et un pipeline qui représentent le processus proposé à l'étape précédente.

- **Étape 4 (Développement et mise en œuvre) :** implémenter le pipeline et les différents algorithmes identifiés avec le langage Python. Ce langage est utilisé parce qu'il est le plus utilisé pour la mise en place des modèles d'apprentissage automatique.
- **Étape 5 (Expérimentation et validation) :** Exploiter les données d'une entreprise d'assurance pour tester notre modèle. Les performances du modèle seront par la suite évaluées et comparer.

1.5 ORGANISATION

Nous verrons dans un premier temps qu'il est nécessaire de présenter des études semblables autour du sujet afin de mieux appréhender le sujet et avoir une idée des travaux antérieurs. Nous présenterons par la suite dans le chapitre suivant les algorithmes de « Machine Learning » pour le prétraitement ainsi que l'apprentissage et les techniques méta-heuristiques utilisées pour l'optimisation des résultats. Aussi sera présenté une description des données utilisés comme données de test. Cette description nous permettra de mieux connaître les données qui nous permettront de les préparer pour leurs utilisations par les algorithmes d'apprentissage automatique. La dernière partie concerne l'évaluation des résultats de ses techniques sur les données.

CHAPITRE 2

REVUE DE LA LITTERATURE

Les travaux présentés dans ce mémoire tentent d'automatiser le processus de pré-traitement des données et la classification de celles-ci à l'aide d'algorithmes d'optimisation métaheuristique. Ainsi, afin de bien choisir nos outils pour la réalisation de ce projet de recherche et positionner nos travaux par rapport à la littérature, nous commençons, dans ce chapitre, par présenter une revue de littérature sur les travaux qui optimisent les résultats de classification ou de prédiction. Dans un deuxième temps, nous résumons les travaux qui optimisent les résultats de classification ou de prédiction à l'aide de pré-traitement et traitement des données. Enfin, pour clôturer ce chapitre, nous présentons des travaux traitant de données d'assurance en exploitant les concepts de l'intelligence artificielle (IA) au sens large (science des données, apprentissage automatique).

2.1 OPTIMISATION PAR ALGORITHMES D'APPRENTISSAGE AUTOMATIQUE

Plusieurs algorithmes cherchent à optimiser les résultats de classification ou de prédiction à l'aide d'algorithmes d'apprentissage automatique, le premier qui est cité est l'article de [4], l'article présente l'algorithme *competitive swarm optimizer* (CSO) afin d'effectuer une réduction de dimensionnalité par une sélection de caractéristiques pour la classification. Cet algorithme est un variant du *particle swarm optimization* (PSO) et il vise à compenser les lacunes du PSO. En effet, PSO a été développé pour des problèmes d'optimisation continu [5]. L'optimisation continue consiste à traiter des variables réelles (non catégorielle). Néanmoins, la sélection de caractéristique est considérée comme un problème d'optimisation combinatoires [6]. Selon un article [4], PSO ne permet pas d'obtenir des meilleurs performances sur ce genre de problème d'optimisation. Les auteurs de l'article mentionnent tout de même que PSO montre des performances prometteuses sur des problèmes à faible dimension, mais peine à résoudre des problèmes de sélection de caractéristiques pour des ensembles de données à très grande dimension. Pour ces raisons, CSO a été proposé afin de résoudre les problèmes d'optimisation combinatoire à très grande dimension. Tel qu'illustré par les résultats des travaux, PSO n'obtient pas les meilleures performances dans le cas d'une

sélection de caractéristiques. Par exemple, avec l'ensemble de donnée *movement* (90 caractéristiques, 360 instances et 15 classes), CSO a réduit le nombre de caractéristiques pour optimiser la classification avec l'algorithme des k-plus proches voisins (k-NN). Les auteurs ont paramétré k-NN avec k=5 et calculé le taux d'erreur moyen (AER) en ayant définit 10 plis pour la validation croisée. Le AER obtenu est de 0.2394 et le nombre de caractéristiques sélectionnées est de 49.40 en moyenne avec CSO tandis que PSO a permis d'obtenir un AER de 0.2850 et 41.70 caractéristiques sélectionnées en moyenne. Cinq autres ensembles de données ont été exploités afin de démontrer la supériorité de CSO sur PSO. Les cinq ensembles de données sont *musk* (167 caractéristiques, 6 598 instances et 2 classes), *arrhythmia* (279 caractéristiques, 452 instances et 16 classes), *madelon* (500 caractéristiques, 2 600 instances et 2 classes), *isolet5* (617 caractéristiques, 1 559 instances et 26 classes) et *InterAd* (1 588 caractéristiques, 3 279 instances et 2 classes). Pour ces cinq ensembles de données, CSO sélectionne non seulement un nombre beaucoup plus petit de caractéristiques, mais aboutit à de meilleures performances de classification également. Cette amélioration est rendue possible en supprimant de PSO, le *global* et *personal best positions*. Ainsi, les particules de CSO apprennent à partir de compétiteurs choisis au hasard. CSO a montré qu'il pouvait gérer des problèmes avec une dimensionnalité avoisinant 5 000 dimensions. Pour plus d'informations concernant le fonctionnement de l'algorithme CSO, les lecteurs sont invités à lire l'annexe A du présent mémoire.

Dans un autre article [7] est présenté l'algorithme d'optimisation *Binary SailFish* (BSF). Il permet de résoudre, en partie, les problèmes de sélection de caractéristiques et représente un variant de *SailFish Optimizer* (SFO). SFO imite le comportement de la chasse en groupe de voiliers[10] qui suit une stratégie d'alternance d'attaques pour chasser un banc de sardines. La principale différence entre SFO et BSF est que ce dernier utilise la fonction de transfert sigmoïde pour faire correspondre l'espace de recherche continue de SFO sur un espace binaire. La performance de l'optimiseur BSF peut être améliorée en l'associant notamment

[10] Les voiliers (genre Istiophorus) forment un genre de poissons pélagiques, de la famille des Istiophoridae.

avec l'algorithme *Adaptive β-Hill Climbing* (Aβ-HC). Les contributions de cet article est d'associé l'optimiseur BSF et l'algorithme méta-heuristique Aβ-HC. L'approche est testée sur des problèmes de classification en utilisant KNN comme algorithme de classification. Les données sont constitués de 18 jeux de données provenant du site *UCI Machine Learning Repository* [8]. La nouvelle approche est comparée à 10 autres méthodes de sélection de caractéristiques reposant également sur des méta-heuristiques. Pour chaque ensemble de données, 80 % des instances sont utilisées pour entraîner le modèle et les 20 % restants sont utilisés pour tester le modèle entrainé. Ils ont effectué plusieurs tests et à chaque test, ils changent la taille des données et le nombre d'itérations maximum pour voir l'impact de ces deux paramètres sur la performance du modèle. En conclusion, ils précisent que BSF a atteint une précision supérieure à 90% dans le cas de 11 (61,1%) jeux de données. Aβ SF a atteint une précision de 100% pour 10 (55,5%) jeux de données tels que Breastcancer, BreastEW, CongressEW, Lymphography, etc., ce qui est assez impressionnant.

Dans [9], une nouvelle approche de sélection des caractéristiques est présentée. La méthode est appelée *Generalized wrapper-based Feature Selection* (GeFeS). Plus simplement, GeFeS repose sur un algorithme génétique (GA) et la stratégie de populations multiples est exploitée afin de paralléliser les calculs. Plus concrètement, l'algorithme crée une population initiale de chromosomes. Chaque chromosome représente un sous-ensemble de caractéristiques de l'ensemble de données et est évalué (évaluation de la qualité du chromosome) avec l'algorithme k-NN. Les paramètres de k-NN sont réglés en utilisant une validation croisée imbriquée dans le processus. Bien évidemment, k-NN peut être remplacer par n'importe quels autres algorithmes d'apprentissage automatique. Une fois que tous les chromosomes de la population ont un score dénotant leur qualité, un opérateur de sélection choisit des chromosomes qui vont aller dans la population de la génération suivante. Les améliorations de GeFeS par rapport aux autres GA résident dans l'opérateur de croisement, l'opérateur de mutation et un nouvel opérateur de pondération appelé *inverse*. Les auteurs attestent que les opérations de croisement et de mutation proposées offrent un bon compromis entre l'exploration et l'exploitation de l'espace de recherche. En ce qui concerne l'opérateur

inverse, il met efficacement à jour les poids des caractéristiques en observant le comportement du problème pendant l'optimisation. Enfin, après l'opérateur inverse, l'algorithme applique un opérateur de remplacement. Les auteurs estiment que GeFeS permet d'augmenter la précision, la fiabilité et la généralisation d'un classeur d'apprentissage automatique. De surcroît, bien que GeFeS ne semble pas intégrer de concepts liés à l'intelligence artificielle, les auteurs l'appellent aussi un algorithme génétique intelligent. L'algorithme d'optimisation proposé a permis d'obtenir des moyennes d'*accuracy* de 95.83%, 97.62%, 99.02%, 98.51% et 94.28% en réduisant le nombre de caractéristiques de 56 à 28, 34 à 18, 279 à 135, 30 à 16, et de 19 à 9 pour les ensembles de données *lung cancer*, *dermatology*, *arythmia*, *WDBC*, et *hepatitis* (tous disponible sur UCI), respectivement. Ils ont comparé leurs résultats avec les algorithmes *ReliefF*, *Correlation feature selection* (Cfs), *Principal Components Analysis* (PCA) et plusieurs algorithmes dans la littérature [9]. GeFeS obtient quasiment pour chaque ensemble de données les meilleurs résultats de classification. Malheureusement, le nombre de classes maximum était de 6 ce qui explique, en partie, les bonnes performances obtenues.

L'article [10], les auteurs présentent un nouvel algorithme intitulé *Grey Wolf Optimizer* (GWO). L'objectif est de réduire la dimensionnalité des données par la sélection des caractéristiques les plus pertinentes afin d'augmenter les performances de classification et réduire, par la même occasion, les temps de calculs nécessaires. Le principe de l'algorithme GWO s'inspire de la manière dont se comporte les meutes de loups gris. Il existe une hiérarchie reposant sur le pouvoir dans un groupe de loups gris (les alpha, beta, oméga et gamma). Le modèle mathématique de GWO consiste à encercler, chasser et attaquer la proie. Aussi, GWO est approprié que pour les problèmes de recherche continue. L'article présente l'utilisation de GWO dans plusieurs domaines tels que : la reconnaissance des émotions faciales, la classification des signaux EMG, le diagnostic des maladies, la sélection des gènes et systèmes de détection des intrusions. Aussi un tableau illustre son utilisation sur des bases de données, ainsi que les méthodes et classifiers utilisés.Il est aussi précisé que les méthodes de sélection sont classées en deux groupes, voire trois groupes par certains chercheurs. Ce sont les le groupes des wrappers, des filtres et du méthodes embarquées. Ces méthodes de sélection de

caractéristiques sont combinées avec des classifiers pour apprendre et prédire des modèles. Ce sont les classifiers comme le voisin le plus proche K (KNN), la machine à vecteurs de support (SVM), l'arbre de décision, le réseau de neurones artificiels (ANN) et Bayes naïf.

L'article [11], propose une métaheuristique appelée *Multi-Verse Optimizer* (MVO). Son objectif est de sélectionner les caractéristiques optimales et d'optimiser les paramètres de l'algorithme *Support Vector Machine* (SVM). SVM est un algorithme d'apprentissage automatique largement appliqué aux problèmes de classification et de régression. L'efficacité de SVM et sa précision de classification dépendent fortement de son paramétrage ainsi que des caractéristiques sélectionnées pour former le sous-ensemble de données. Le principe de MVO est qu'il imite les règles d'une des théories du multivers. Il s'inspire de la théorie de l'existence d'univers multiples et de leurs interactions via les trous noirs, blancs et de vers. C'est un algorithme stochastique basé sur la population et se rapproche de l'optimum global pour les problèmes d'optimisation avec un ensemble de solutions. Une solution est codée comme un vecteur de nombres réels. Plus exactement, le nombre d'éléments du vecteur représentent le nombre de caractéristiques de l'ensemble de données et deux paramètres supplémentaires pour SVM. Les deux paramètres pour SVM sont le coût (pénalité pour une erreur de classification) et gamma (paramètre du noyau *Radial Basic Function*). Encore une fois, les auteurs ont implémenté la stratégie de la validation croisée pour l'entraînement du modèle de classification. Un total de 10 ensembles de données a été utilisé pour vérifier l'efficacité de MVO. Ces ensembles de données sont *heart* (13 caractéristiques, 270 instances et 2 classes), *ionosphere* (34 caractéristiques, 351 instances et 2 classes), *sonar* (60 caractéristiques, 208 instances et 2 classes), *german* (24 caractéristiques, 1 000 instances et 2 classes), *vowel* (10 caractéristiques, 528 instances et 11 classes), *wine* (13 caractéristiques, 178 instances et 3 classes), *vehicle* (18 caractéristiques, 846 instances et 4 classes), *breast cancer* (10 caractéristiques, 683 instances et 2 classes), *parkinsons* (22 caractéristiques, 195 instances et 2 classes), *spectf* (44 caractéristiques, 276 instances et 2 classes). Tous ces ensembles de données sont disponibles sur le site web UCI *repository*. De surcroît, les auteurs ont comparé MVO avec 4 autres algorithmes d'optimisation, à savoir, un GA, PSO, *Bat*

algorithm (BAT) et *Firefly algorithm* (FF). MVO ne surpasse pas vraiment les autres algorithmes d'optimisation de manière significative.

Les auteurs dans [12], propose d'appliquer la méthode *Fast Correlation-Based Feature Selection* (FCBF) afin d'effectuer une sélection des caractéristiques les plus discriminantes (filtrage des caractéristiques redondantes) en vue d'une classification. Une fois les caractéristiques sélectionnées par FCBF, les auteurs combinent PSO et *Ant Colony Optimization* (ACO) pour optimiser la sélection des caractéristiques restantes. Malheureusement, il n'est pas décrit comment cette combinaison est réalisée. Seul l'ensemble de données *heart disease* a été exploité pour valider leur approche. Cet ensemble de données possède 13 caractéristiques et 2 classes. Aussi, plusieurs algorithmes d'apprentissage automatique ont été choisis pour la classification. Ces algorithmes sont k-NN, SVM, *Random Forest* (RF), *Naïve Bayes* (NB) et le *MultiLayer Perceptron* (MLP). Chaque modèle de classification a été entrainé en suivant la stratégie de la validation croisée en 10 plis et l'optimisation a été exécutée 10 fois. Les métriques de performances utilisées sont, entre autres, la précision, le rappel et le F1-score. Les résultats démontrent clairement l'efficacité de l'approche combinant FCBF, PSO et ACO. En effet, pour chaque algorithme de classification, l'approche associant FCBF, PSO et ACO surpasse les F1-scores de classification sans optimisation et de la classification en appliquant FCBF seulement. Par exemple, pour k-NN, FCBF, PSO et ACO obtiennent un F1-score de 0.996 alors que la classification sans optimisation et avec seulement FCBF obtiennent 0.746 et 0.781, respectivement. Les auteurs ont poussé la comparaison plus loin avec des travaux issus de la littérature (p.ex., *Bayes Net*, *Bagging*, *GA+SVM*). Encore une fois, les auteurs obtiennent les meilleures performances.

En résumé, nous pouvons confirmer la popularité et l'efficacité des algorithmes métaheuristiques pour la sélection des caractéristiques afin d'optimiser les performances de classification. Néanmoins, le point faible de ces travaux est un manque de comparaison avec des méthodes classiques de réduction de dimensionnalité telles que *SelectKBest* [11]ou encore

[11] Sélectionnez les caractéristiques en fonction des k scores les plus élevés.

VarianceThreshold[12], méthodes déjà implémentées dans la librairie *scikit-learn*[13]. En effet, le principe même d'exploiter un algorithme métaheuristique est de limiter le temps passé dans la recherche de sous-ensemble optimal de caractéristiques. Malheureusement, cette composante de temps n'est pas assez exploitée dans la présentation des résultats. En d'autres termes, la littérature ne nous permet pas de comparer adéquatement le gain de temps des algorithmes métaheuristiques entre eux et par rapport aux méthodes classiques.

Enfin, notre projet de recherche ne se limite pas seulement à l'optimisation de la classification par la sélection des caractéristiques optimales basé sur des algorithmes d'apprentissage automatique, mais inclut également l'optimisation des opérations de pré-traitement et de traitement des données ce qui complexifie grandement le problème. Ainsi, selon les performances obtenues dans la littérature et des propriétés de chaque algorithmes métaheuristiques, nous avons sélectionné GA et GWO pour nos travaux.

2.2 OPTIMISATION PAR PRE-TRAITEMENT ET TRAITEMENT DES DONNEES

Nous retrouvons dans la littérature une autre manière d'optimiser les performances de classification. Cette manière tente d'optimiser les opérations de pré-traitement (p.ex., la gestion des valeurs manquantes et des valeurs aberrantes) et traitement (p.ex., la collecte et la manipulation) des données [13]. De plus, les auteurs dans [1] estiment que le prétraitement des données est l'une des étapes les plus chronophages et les plus pertinentes d'un processus d'analyse de données (p.ex., tâche de classification). Dans le monde réel, les données contiennent du bruit, des valeurs manquantes, elles sont incomplètes et parfois dans un format inutilisable qui ne peut pas être directement utilisé pour les modèles d'apprentissage automatique. L'utilisation de ces données douteuses et sales peuvent entraîner des conséquences sur le résultat et souvent conduit à des conclusions erronées. Le prétraitement des données est un moyen d'obtenir des résultats plus significatives et fiables. Pour obtenir

[12] Sélecteur de caractéristiques qui supprime toutes les fonctionnalités à faible variance
[13] Scikit-learn est une bibliothèque libre Python destinée à l'apprentissage automatique, accéssible sur le web via : https://scikit-learn.org/stable/

ces résultats, le prétraitement des données utilisent plusieurs techniques. Ces techniques permettent de transformer les données brutes en un ensemble de données propres, utilisables et significatives en redimensionnant, normalisant, binarisant, etc. Plusieurs articles abordent l'optimisation par le biais du pré-traitement ou traitement des données. Dans cette section nous listerons quelques techniques utilisées en fonction du problème. Les problèmes les plus fréquents selon la littérature sont : les valeurs manquantes, le temps de calculs, modélisation des interactions entre les variables, la présence des colonnes non informatives ou redondantes, les valeurs non-cohérentes, etc.

2.2.1 LES VALEURS MANQUANTES

Les valeurs manquantes sont dues souvent selon [14, 15] à : l'erreur humaine lors du traitement des données, l'erreur de la machine due au dysfonctionnement de l'équipement, le refus des répondants de répondre à certaines questions, l'abandon dans les études et la fusion de données sans rapport. Néanmoins, il est primordial de traiter ces valeurs manquantes avant d'analyser les données car ignorer ou omettre les valeurs manquantes peut entraîner une dégradation des performances, une analyse biaisée ou mal informée [16]. Plusieurs techniques permettent de traiter les valeurs manquantes. Chaque technique d'imputation des valeurs manquantes à des caractéristiques, une performance et des limites. En plus, une technique peut être plus ou moins adaptée en fonction du type de données. Appliquer la bonne technique permet d'augmenter considérablement la précision du modèle.

Les techniques comme la suppression d'instances et le remplacement par des valeurs potentielles ou estimées permettent de traiter les valeurs manquantes dans les recherches [17-19]. Un autre technique appelé imputation est utilisé pour traiter les données manquantes[20] . Dans la littérature, d'autres techniques traditionnelles d'imputation statistique et d'apprentissage automatiques telles que la moyenne, la régression, K plus proche voisin (*nearest neighbor*), basée sur l'ensemble (*ensemble based*), etc., sont proposés par les chercheurs [21],[22]. D'autres chercheurs ont proposés des approches hybrides pour pallier au faiblesses des techniques d'imputation traditionnelles [23],[24],[25]. Le *floue c-means*

clustering est un exemple d'approche hybride réalisé par [23]. Cette approche combine la régression des vecteurs de support (*support vector regression*) et un algorithme génétique (*GA*) pour imputer les valeurs manquantes. Une meilleure précision d'imputation est obtenue par rapport aux méthodes d'imputation *FcmGa, SvrGa, Zero.* Il est important de noter que la bonne méthode pour traiter les valeurs manquantes n'existe pas, il suffit de bien analysé le problème étudier. Cependant, avec l'augmentation des quantités de données (big data), les données deviennent plus complexes de sorte qu'il est difficile de traiter les données manquantes à l'aide de méthodes d'apprentissage traditionnelles. Les ingénieurs de données sont obligés d'utiliser des techniques d'apprentissage automatique car les techniques traditionnelles ne sont conçus pour les mégadonnées [26]. Les méthodes d'imputation basés sur l'apprentissage automatique sont des techniques sophistiquées qui impliquent principalement le développement d'une approche prédictive pour gérer les valeurs manquantes à l'aide d'un apprentissage non supervisé ou supervisé [14]. Ces techniques devinent les données manquantes en fonction des informations disponibles à partir des valeurs non manquantes dans les données à l'aide de données étiquetées ou non étiquetées. Nous discutons ci-dessous de certaines des techniques d'imputation d'apprentissage automatique les plus fréquentes telles que : *K nearest neighbour classification, Support vector machine (SVM), Decision tree, Clustering imputation, etc*.

2.2.1.1 SUPPRESSION

C'est l'une des méthodes traditionnelles les plus utilisés. Pour cette approche, toutes les entrées avec des valeurs manquantes sont supprimées/éliminées lors de l'analyse. La suppression est l'approche la plus simple car la machine ne cherche pas à deviner la valeur la plus probable. Cependant, cette méthode à des faiblesses selon les chercheurs XXX. La suppression introduit un biais dans l'analyse lorsque les données manquantes ne sont pas distribuées au hasard. La suppression des valeurs manquantes s'effectue de deux manières : la suppression par paires et la supression par liste (*pairwise or list-wise deletion*). La suppression par liste consiste à supprimer toutes les entrées qui ont une ou plusieurs valeurs manquantes. La suppression par liste entraine un biais si les données ne sont pas

volumineuses et peut également entraîner la perte de certaines informations importantes. La suppression par liste peut être une approche raisonnable si les données sont suffisamment grandes. La suppression par paire est utilisée pour atténuer la perte d'informations lors de la suppression par liste. C'est une méthode permet de réduire les pertes pouvant survenir lors de la suppression par liste.

Une façon simple de penser au fonctionnement de la suppression par paires est de penser à une matrice de corrélation. Une corrélation mesure la force de la relation entre deux variables. Pour chaque paire de variables pour lesquelles des données sont disponibles, le coefficient de corrélation tiendra compte de ces données. Ainsi, la suppression par paires maximise toutes les données disponibles sur la base d'une analyse par analyse. L'un des points forts de cette technique est qu'elle augmente la puissance de vos analyses. Il y a aussi des inconvénients. L'un des inconvénients de la suppression par paires est que la norme d'erreurs calculée par la plupart des progiciels utilise la taille moyenne de l'échantillon dans les analyses. Cela tend à produire des erreurs standard qui sont sous-estimées ou surestimées.

2.2.1.2 IMPUTATION

Cette approche consiste à remplacer les valeurs manquantes par certaines valeurs prédites. Les lignes qui n'ont pas de données manquantes serviront à prédire les valeurs utilisées pour remplacer les valeurs manquantes [20]. Il existe plusieurs techniques d'imputation traditionnelles selon la littérature mais nous retenons : *simple imputation, regression imputation, hot-deck imputation, expectation–maximization et Multiple imputation, cold-desk, Mean substutition, Non-negative matrix factorization.*

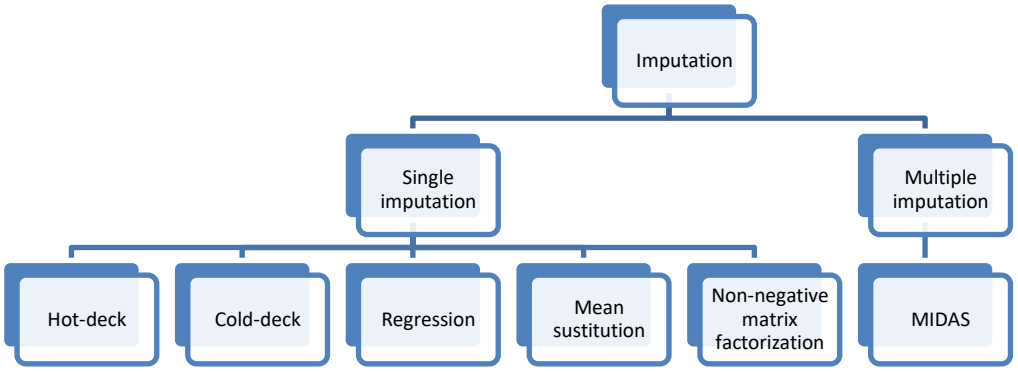

Figure 1: classification des méthodes d'imputation

L'approche d'imputation simple (*simple imputation)* consiste remplacer les valeurs manquantes à l'aide d'une statistique descriptive (par exemple moyenne, médiane ou la plus fréquente) le long de chaque colonne, ou à l'aide d'une valeur constante. Par exemple, cette technique est utilisés pour imputés les données manquantes dans un problème réel de cancer du sein [27]. Cependant, les méthodes d'imputation simples peuvent produire des biais ou des résultats irréalistes sur des ensembles de données de grande dimension. La méthode est donc déconseillée sur de tels ensembles de données.

L'imputation par régression (*regression imputation)* est une technique statistique. La technique est aussi appelée imputation moyenne conditionnelle. La technique est de remplacer les valeurs manquantes par une valeur prédite créée à partir d'un modèle de régression. Le processus de régression global est une méthode en deux phases : la première étape utilise toutes les observations complètes pour construire un modèle de régression et la deuxième étape impute les données manquantes en fonction du modèle de régression construit [28]. L'avantage de cette méthode est qu'elle maintient la taille de l'échantillon en préservant toutes les observations avec des valeurs manquantes. Néanmoins, elle peut nécessiter un grand échantillon de données pour produire des résultats stables. Aussi, une seule courbe de

régression est suivie pour toutes les valeurs imputées et aucune variation inhérente n'est présentée dans les données. La technique de régression mise en œuvre dépend de la nature des données. S'il y a deux caractéristiques manquantes ou plus, un modèle de régression multivariée doit être utilisé pour l'imputation.

La technique d'imputation par régression est utilisée par Sherwood et al. [29] pour estimer les valeurs manquantes sur des données de la santé. Les auteurs ont utilisé cette approche sur les données de santé car elle est généralement attribuée à un niveau élevé d'asymétrie, de variances hétéroscédastiques et l'estimateur de régression est cohérent, contrairement à l'estimateur naïf, et asymptotiquement normal, ce qui le rend approprié pour analyser ce type de les données selon [14]. En conclusion, la technique de régression quantile s'est montré efficace pour analyser les données numériques sur les coûts des soins de santé. Néanmoins, l'approche n'était pas robuste en raison de la spécification de la forme fonctionnelle et aurait pu introduire des résultats biaisés.

Des chercheurs ont proposés une méthode complète de traitement des valeurs manquantes par régression de cas en utilisant la composante principale fonctionnelle [30]. Les auteurs ont comparé la performance de la nouvelle approche : l'approche ou les valeurs manquantes sont imputés par la régression et l'approche ou les valeurs manquantes ne sont pas traités. En conclusion de leur étude, La régression linéaire fonctionnelle est conseiller et adapter à remplacer les valeurs manquantes. D'autres chercheurs ont appliqué une technique d'imputation multivariée pour imputer les valeurs manquantes dans les données multivariées normales [31]. Une séquence de régression a permis de d'obtenir les valeurs d'imputation. En effet, toutes les variables contenant des valeurs manquantes ont été régressées par rapport aux variables qui ne contenaient pas de valeurs manquantes comme variables prédictives en utilisant l'approche d'itération. L'approche marche avec plus d'une variable contenant des valeurs manquantes et des modèles non monotones.

La méthode d'imputation *hot-deck* est utilisés pour reconstruire les données manquantes d'une base de données sur des demandes de logement en Pays-Bas [32]. Lors de l'expérience, les chercheurs ont omis volontairement des données et ont par la suite appliqués la méthode hot-deck pour reconstruire la base. Les imputations sont ensuite comparées aux vraies valeurs. L'Imputation *hot-deck* est une méthode où une valeur manquante est imputée à partir d'un enregistrement similaire sélectionné au hasard. Dans cette méthode les informations donneurs proviennent du même jeu de données que les destinataires. Les variantes de hot-desk sont : *random hot-deck* et *weighted sequential hot deck*. *random hot-deck* ne limite pas le nombre de fois qu'un donneur est désigné ; cependant, les donneurs sont choisis au hasard dans le groupe de donneurs. En revanche, *weighted sequential hot deck* limite le temps pendant lequel un donneur peut être choisi pour éviter que le même donneur ne soit jumelé à un grand nombre de receveurs. L'imputation *cold-deck,* en revanche, sélectionne les donneurs à partir d'un autre ensemble de données. Il s'agit d'une méthode de remplacement par des valeurs de réponse d'éléments similaires dans des enquêtes [14]antérieures. Il est disponible dans les enquêtes qui mesurent des intervalles de temps.

Expectation–maximization est une méthode itérative et est souvent utilisés dans les ensembles de données numériques. L'approche qu'elle utilise est « *impute, estimate and iterate until convergence* ». En effet, elle utilise deux étapes (l'attente et la maximisation) pour chaque itération. L'espérance estime les valeurs manquantes compte tenu des données observées, tandis que dans la maximisation, les valeurs estimées actuelles sont utilisées pour maximiser la probabilité de toutes les données.

La méthode *Expectation–maximization* est appliqués sur une base de données analysant les impacts des comportements alimentaires chez les animaux traités et non traités, pour traités les données manquantes [33]. La méthode est par la suite comparée avec les

[14] Une enquête est l'équivalent d'un enregistrement. Un enregistrement est un élément d'un tableau à deux dimensions ou d'une base de données.

méthodes comme : la suppression par liste, l'approche bayésienne et la régression par substitution moyenne. La suppression par liste était la moins efficace tandis qu'*Expectation–maximization* était la meilleure dans leur étude.

L'imputation multiple est mise en place spécialement pour résoudre les limites de l'imputation unique[34]. Le principe d'imputation est que la distribution des données observées est utilisée pour approximer de nombreuses valeurs qui reflètent l'incertitude autour de la valeur réelle. L'analyse est effectuée sur un ensemble de données à l'aide des diverses techniques de données manquantes, et la moyenne des estimations des paramètres sur M échantillons est calculée en une seule estimation ponctuelle. La technique d'imputation multiple comprend trois phases : Les données manquantes sont traitées dans *M*, ce qui donne *M* ensembles de données complets ; Les *M* jeux de données complets sont ensuite analysés ; Les résultats de tous les *M* ensembles de données imputés sont combinés pour le résultat d'imputation final. Selon [14], c'est la méthodologie standard pour traiter les valeurs manquantes. Pour ces derniers, les chercheurs doivent utilisés des techniques d'imputation appropriés, afin de garantir l'obtention de résultats fiables. Des chercheurs [35, 36]ont démontrés que les techniques d'imputation multiples traditionnelles ne marche pas bien sur des données de grande dimension. L'étude de [35] conclu que la *Bayesian lasso regression* et ses extensions conviennent mieux à l'imputation multiple en présence de données de grande dimension que les autres méthodes de régression.

D'autres chercheurs [37] ont utilisés l'imputation multiple à l'aide de la *Least Squares Support Vector Machine (LSSVM*) sur un ensemble de données de patients. Leur technique imputait avec précision les valeurs manquantes dans cinq bases de données différentes. Ils concluent que leur méthode surpassait les méthodes d'imputation standard et qu'en plus la technique était plus robuste puisqu'elle générait des valeurs plus proches de celle qui manquait. Une deuxième méthode intitulée *Clustered Z-score Least Square Support Vector Machine (CZLSSVM)* est proposé dans l'article. *CZLSSVM* est testé efficace sur deux problèmes de classification. Ils ont comparé CZLSSVM avec d'autres techniques d'imputation

telles que *SVM, arbre de décision, KNN, rough sets* et *artificial neural networks* et ont conclu que la précision de la classification était augmentée avec CZLSSVM.

Les données manquantes sont un problème aussi courant dans le domaine médical. Des recherches [20, 38-40] prouvent que leurs traitements permettent d'optimiser le processus de prise de décision. Dans une étude effectué [20], les auteurs démontrent que les techniques simples de traitement des données manquantes comme l'analyse complète des cas (*case analysis*), l'imputation de la moyenne globale (*overall mean imputation*) et la méthode des indicateurs manquants (*the missing-indicator method)* produisent des résultats biaisés tandis que les techniques d'imputation donnent des résultats valables sans compliquer l'analyse. Ils démontrent que l'imputation multiple donne des erreurs types et des intervalles de confiance correctement estimés. Une simulation est faite sur des données de diagnostic médicale. L'expérience consiste à simuler 1000 échantillons de 500 patients en utilisant le logiciel R. Les échantillons sont tirés d'une population composée d'un nombre égal de patients malades et non malades. Parmi les patients non malades (bien portants), 80 % ont reçu au moins une valeur manquante au test. Pour les patients malades n'ont pas de données manquantes. Les résultats montrent que la procédure d'imputation unique semble plus précise en raison de l'erreur standard plus petits et un intervalle de confiance de 90%. L'imputation multiple conduit à une erreur-type plus grande et à des intervalles de confiance plus larges, mais les erreurs-types estimées sont plus correctes et l'intervalle de confiance est de 90,3 %. Cet article conclu que, contrairement à l'imputation simple, l'imputation multiple donne des résultats valables tant en ce qui concerne le biais que la précision.

2.2.1.3 K NEAREST NEIGHBOUR CLASSIFICATION

Le principe de la méthode *KNN* est qu'elle classe les voisins les plus proches des valeurs manquantes et utilise ces voisins pour l'imputation en utilisant une mesure de distance entre les instances [41]. Plusieurs mesures de distance sont utilisées avec l'imputation KNN : *Minkowski distance, Manhattan Distance, Cosinus Distance, Jaccard Distance, Hamming Distance et Euclidean Distance.* Des études [42, 43] menées sur ces mesures de distance

montrent que la distance euclidienne est efficace, productive et est la plus utilisée. L'imputation KNN est reconnue comme une méthode flexible pour gérer les données discrètes et continues. Aussi, elle peut être utilisés comme techniques d'imputation multiples selon des études[41]. Les inconvénients de l'imputation KNN est qu'elle est naïve, c'est-à-dire qu'elle parcourt tous l'ensemble de données. Ce parcours naïf augmente le temps de calcul selon des études [44]. Plusieurs chercheurs ont travaillé pour améliorer la manière dont parcours l'algorithme d'imputation KNN. Des approches satisfaisantes ont été développés par ces études [45], [46] et [47] pour améliorer la méthode classique.

Dans une recherche, les chercheurs [48] ont développé une méthode d'imputation itérative KNN qui utilise comme mesure de distance *grey relational grade* au lieu de la distance euclidienne pour rechercher les k instances voisines les plus proches. Cette imputation itérative utilise toutes les valeurs de l'itération précédente pour estimer les valeurs manquantes. Le résultat montre que la nouvelle méthode estime les données manquent de façon fiable. Elle marche aussi quel que soit le taux de données manquantes dans l'ensemble de données. Une étude comparative conclu que la méthode proposée donne de meilleures performances par rapport aux autres méthodes en ce qui concerne la précision de l'imputation et la vitesse de convergence.

Une autre étude à développer l'approche *Cross-validation based K nearest neighbor imputation* (CVBkNN) [49]. L'approche utilise la validation croisée pour améliorer les paramètres pour chaque valeur manquante. L'approche est testée sur huit (08) ensembles de données différents. Comme résultat, les expériences montrent que leur approche était supérieure aux autres approches et améliore la précision de la classification. Le petit problème est que la validation croisée fait toutes les combinaisons possibles qui peut prendre plus de temps. La solution est d'obtenir les meilleurs paramètres en avance (Ceci est un autre problème d'Optimization).

Emmanuel et al. [50] proposent et évaluent deux méthodes, le *k* plus proche voisin et une méthode d'imputation itérative (*missForest*) basée sur l'algorithme de forêt aléatoire. Les deux méthodes sont évaluées sur deux bases de données (la base de données *iris* et de ventilateur de centrale électrique) un taux de des valeurs manquantes de 5 % à 20 %.

Plusieurs études ont permis d'améliorer l'imputation basé sur KNN, nous listerons quelques-uns sans rentrer dans les détails. Une étude l'a expérimenté pour évaluer son efficacité en tant que méthode d'imputation pour traiter les données manquantes et a comparé ses performances à d'autres algorithmes tels que par *le C4.5* et *CN2* et l'imputation moyenne ou modale [51]. Une autre étude a incorporé une matrice de corrélation pour la conception de l'algorithme KNN [52]. Une autre étude consistait à combiner l'algorithme génétique pour améliorer le choix du plus proche voisin[53]. Cet algorithme est intitulé *EvlKNNImpute* et s'est montré efficace pour gérer les données manquantes sur une base de données de levure. En somme, la méthode d'imputation KNN est l'une des plus utilisés et plus performant selon la littérature. Cependant, le problème est le choix des paramètres KNN qui donneront le meilleur résultat. Un autre problème est son utilisation dans le Big Data.

2.2.1.4 AUTRES ALGORITHMES ET TECHNIQUES

D'autres algorithmes d'apprentissage automatique et stratégies comme *Support vector machine* (SVM), *Decision tree, Clustering imputation* et *Ensemble methods* sont largement utilisés pour le traitement des données manquantes[50]. Chacun de algorithmes cités a des avantages et a des particularités.

2.2.2 LES VALEURS ABERRANTES

L'un des sujets importants et à ne pas négliger est la gestion des valeurs aberrantes lors de la mise en place d'un système décisionnel. Les valeurs aberrantes sont encore appelées valeurs extrêmes ou valeurs non-coherentes. Les valeurs aberrantes sont très importantes quand le système décisionnel utilise de la régression. En effet, ces systèmes sont plus touchés puisqu'il suffit qu'une grande partie des données soit regroupé vers le minimum ou le maximum

et les autres données reculées seront considées comme des valeurs aberrantes ou extrêmes [54]. Les valeurs aberrantes sont des observations qui se situent sur une distance anormale par rapport aux autres observations dans les données. L'une des approches pour résoudre les valeurs aberrantes est de supprimer tout ce qui est 3 * écarts-types loin de la moyenne. La règle est définie comme suit :

- 68% des données sont réparties dans l'intervalle [moyenne - écart type, moyenne + écart type],
- 95% des données sont distribuées dans l'intervalle [moyenne - 2 * écart type, moyenne + 2 * écart type],
- 99,7 % des données sont distribuées dans l'intervalle [moyenne - 3 * écart-type, moyenne + 3 * écart-type].

En partant de cette règle, nous pouvons supposer que les valeurs qui sont hors de l'intervalle **[moyenne - 3*std, moyenne + 3*std]** sont des valeurs aberrantes et ces valeurs peuvent être supprimées.

2.2.3 LES DONNEES CATEGORIELLES ET LA MISE A L'ECHELLE DES CARACTERISTIQUES

VARIABLES CATEGORIELLES

Les donnée catégorielles ou qualitatives sont des variables qui ont des catégories spécifiques dans l'ensemble des données. On peut dire que les données constituées de valeurs possibles finies peuvent être considérées comme des données catégorielles selon [55]. Les données catégorielles sont des informations recueillies et qui sont divisées en groupes. Par exemple, une liste de plusieurs personnes avec leur groupe sanguin : A+, A-, B+, B-, AB+, AB-, O+, O- etc. Le *Dummy coding* (codage fictive) est l'une des méthodes les plus utilisés pour convertir une variable d'entrée catégorielle en une variable continue.

Une variable fictive est une variable numérique qui représente des données catégorielles, telles que le sexe, la race, l'affiliation politique, etc. Une étude présente le *Dummy coding*

comme un moyen d'incorporer des variables nominales dans l'analyse de régression. Cependant, il explique pourquoi ce type de codage est utilisé plus dans des analyses de régression [56]. Dans leur article, ils utilisent le *Dummy coding* pour prédire si la matière préférée d'un élève était équivalent à une moyenne pondérée cumulative ou GPA accru. Le principe de *Dummy coding* est d'utiliser un certain nombre de variable fictives pour représenter une variable catégorielle. Le nombre de variables fictives requises pour représenter une variable catégorielle particulière dépend du nombre de valeurs que la variable catégorielle peut prendre[57]. Pour représenter une variable catégorielle qui peut prendre k valeurs différentes, un chercheur devrait définir k - 1 variables fictives.

MISE EN ECHELLE

Les caractéristiques d'une base de données n'ont pas souvent les mêmes échelles. Par exemple, une base de données peut être constitue d'une caractéristique mesurée en kilogrammes tandis que l'autre est en grammes, une autre en litres, etc. Ces variations entre caractéristiques affectent beaucoup certains modèle d'apprentissage automatique si elles ne sont pas gérer selon [58]. L'auteur affirme que certains algorithmes d'apprentissage automatique sont sensibles à la mise à l'échelle des caractéristiques, tandis que d'autres y sont pratiquement invariants. Les algorithmes basés sur la descente de gradient (la régression linéaire, la régression logistique, le réseau de neurones, etc.) et sur la distance (KNN, K-means et SVM) fonctionnent bien quand les caractéristiques ont une même échelle tandique les algorithmes basés (*DecisionTree*, *Randomforest*) sur les arbres sont assez insensibles à l'échelle des caractéristiques.

Les deux techniques de mise en échelle les plus utilisés sont la normalisation et la standardisation. La normalisation est une technique de mise à l'échelle dans laquelle les valeurs sont décalées et remises à l'échelle de sorte qu'elles soient comprise entre 0 et 1. Elle est également connue sous le nom de mise à l'échelle Min-Max. La formule de la normalisation est définie comme suit :

$$X' = \frac{xi - \min(x)}{\max(x) - \min(x)}$$

Xmax et Xmin sont respectivement les valeurs maximale et minimale que prend la caractéristique.

- Lorsque la valeur de *X* est la valeur minimale dans la colonne, le numérateur sera 0, et donc *X'* est 0
- En revanche, lorsque la valeur de *X* est la valeur maximale dans la colonne, le numérateur est égal au dénominateur et donc la valeur de *X'* est 1
- Si la valeur de *X* est comprise entre la valeur minimale et la valeur maximale, alors la valeur de *X'* est comprise entre 0 et 1.

La standardisation est une technique de mise à l'échelle où les valeurs sont centrées autour de la moyenne avec un écart type unitaire. Dans ce cas, la moyenne de l'attribut devient nulle et la distribution résultante a un écart type unitaire. La formule de la standardisation est :

$$X' = \frac{X - \mu}{\sigma}$$

Avec μ: la moyenne des valeurs des caractéristiques et σ : Ecart-Type

Les deux techniques sont les plus utilisés et sont chacun adapté à des cas bien précis. Cependant, le choix est expliqué dans l'article de Bhandari [58]. Il explique que la normalisation est utile lorsque que la distribution de vos données ne suit pas une distribution gaussienne. D'autre part, la standardisation est utile dans les cas où les données suivent une distribution gaussienne. Cependant, cela ne doit pas nécessairement être vrai. La normalisation est conseiller dans les algorithmes comme *K-Nearest Neighbors et Neural Networks* qui ne supposent aucune distribution des données. L'une des particularités de la normalisation est qu'elle n'affecte pas les valeurs aberrantes et n'a pas de plage de délimitation. En conclusion, l'auteur précise qu'il est conseiller d'ajuster le modèle à des données brutes, normalisées et standardisées et comparer les performances pour obtenir les meilleurs résultats.

Toujours dans le même article de Bhandari [58], il a étudier l'effet de la mise à l'échelle sur trois algorithmes : *K-Nearest Neighbours, Support Vector Regressor et Decision Tree.* Il retient qu'avec KNN la mise à l'échelle des fonctionnalités a fait baisser le score RMSE et les données normalisées fonctionnent un peu mieux que les données standardisées. Avec SVM, D'une part la mise à l'échelle des fonctionnalités fait baisser le score *RMSE*. D'autre part, les données standardisées ont donné de meilleurs résultats que les données normalisées. Avec DecisionTree, le score RMSE n'a pas changé lors de la mise à l'échelle des fonctionnalités. Il confirme alors la théorie disant que la mise en échelle influence certaines techniques d'apprentissage plus que d'autres.

2.2.4 ECHANTILLONNAGE DES DONNEES

Le rééchantillonnage consiste à prélever à plusieurs reprises des échantillons à partir de données d'apprentissage et à les réajuster à un modèle d'intérêt sur chaque échantillon afin d'obtenir des informations supplémentaires sur le modèle ajusté. Les principales techniques d'échantillonnage sont : validation croisée et rééchantillonnage bootstrap [59]. Il existe 5 types de techniques de validation croisée dont les plus utilisés sont *HoldOut Method, Leave-One-Out Cross-Validation, k-Fold Cross-Validation, etc.* Une étude comparative menés par L'équipe *DataMites* [59] donne l'avantage de ses techniques dans le tableau 1.

Tableau 1: Comparaison de toutes les techniques de rééchantillonnage

Hold out Validation	Leave-One-Out Cross-Validation	k-Fold Cross-Validation	Bootstrap
Utilisez-le lorsque les données ont moins de fonctionnalités	Ensemble de données de taille moyenne	Vaste ensemble de données avec de grandes fonctionnalités	Utiliser avec n'importe quelle forme de données
Risque élevé de biais	Moins de biais	Bon compromis entre biais et variance	Bon compromis entre biais et variance

Risque élevé de surajustement	Modèle généralisé	Modèle généralisé	Modèle généralisé

Il n'existe pas de bonne technique, chaque technique est adaptée à un type de données et il est conseillé de tester le maximum de techniques avant de faire le choix. Dans une étude, les auteurs ont exploré dans quelle mesure le biais peut être réduit en utilisant la validation croisée et le rééchantillonnage bootstrap [60]. Ils comparent par la suite ces méthodes à d'autres techniques telles que l'approche *ad hoc* et à une méthode heuristique. Ils ont testé les techniques sur une base de données de cancer de sein. Les deux techniques ont conduit à une nette réduction des biais.

2.2.5 MULTICOLINEARITE

Ce problème existe chaque fois qu'une variable indépendante est fortement corrélée avec une ou plusieurs autres variables indépendantes dans une équation de régression multiple. La multicolinéarité peut entraîner de sérieux problèmes de validation, d'interprétation et d'analyse du modèle, tels que des estimations instables, des signes déraisonnables, des erreurs de type élevé, etc. [61]. L'une des manières pour détecter si la multicolinéarité existe est d'utiliser les *statsmodels [54]*. Des méthodes de réduction variable telles que *B2, B4, VIF* (*variance inflation factor*)*, KIF* et l'analyse factorielle (FA) peuvent être utilisées pour surmonter ce problème de multicolinéarité. Ces méthodes peuvent être utilisés conjointement avec des algorithmes d'apprentissage automatique telles que les réseaux de neurones artificiels (RNA) et la logique floue à condition que ces algorithmes n'automatisent pas à leur tour le processus de sélection de caractéristique.

Par la méthode VIF, sa valeur numérique indique le pourcentage d'augmentation de la variance pour chaque coefficient. Selon la littérature [54], un VIF de 1,9 nous indique que la variance d'un coefficient particulier est 90% plus grande que ce que nous attendrions s'il n'y avait pas de multicolinéarité. Des auteurs ont testé la multicolinéarité entre trois colonnes [54]. Dans leurs hypothèses de base il suppose que : si le kilométrage est plus petit, le prix sera

plus élevé. De plus, si la voiture est plus ancienne, le prix sera inférieur. Ils ont eu respectivement les valeurs 1.27, 1.14 et 1.50 pour *price, milleage et year*. Il conclut que lorsque la valeur VIF est égale à 1, il n'y a aucune multicolinéarité. Les valeurs comprises entre 1 et 5 sont considérées comme correctes malgré qu'il n'y ait pas de norme fixée. Puisque toutes les valeurs sont inférieures à 5 alors elles sont correctes.

Dans une étude [61], les auteurs ont proposé une approche intitulé procédure d'estimation imbriquée. Le concept de cette procédure est très simple et facile à exécuter. Elle est basée sur la méthode *Ordinary Least Square (OLS)*, en estimant les différents paramètres de la variable indépendante individuellement et séquentiellement à chaque itération. L'expérimentation est menée avec succès sur une base de données regroupant la concentration moyenne de dioxyde de soufre dans l'air. L'étude concerne la pollution atmosphérique dans les villes américaines. La base comprend sept variables explicatives et est recueillies sur une période de 3 ans.

Une étude présente comment la multicolinéarité peut influencer l'estimation de la teneur en graisse à l'intérieur du corps. Les auteurs présentent quelques techniques statistiques utilisés pour résoudre le problème de multicolinéarité, ce sont : *stepwise regression, radial basis function partial least squares, partial robust M-regression, ridge regression et principal component regression*. Ils présentent aussi des techniques d'apprentissage automatique pour résoudre le problème, ce sont : *Factor analysis-artificial neural network* (FA-ANN) and *genetic programming.*

2.2.6 LA SELECTION DE CARACTERISTIQUES

La sélection des variables est aussi appelé *Feature selection*, consiste principalement à supprimer les fonctionnalités (colonnes) non informatives ou redondantes du modèle [62]. C'est une opération automatique qui sélectionne les variables pertinentes avant de les transmettre au modèle d'apprentissage automatique. L'objectif est de réduire les données non pertinentes (bruit) pour l'apprentissage du modèle. Les bruits peuvent être : fonctionnalités

indésirables qui n'ont aucune pertinence pour obtenir des résultats précis, données corrompues, valeurs aberrantes, valeurs nulles, données non structurées, erreurs de saisie de données, valeurs manquantes, etc.

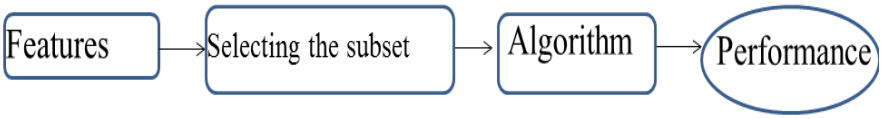

Figure 2: Etape de sélection de caractéristique

La suppression des fonctionnalités jugés supplémentaires ou indésirables n'ont aucun effet et augmente la précision du modèle selon Mohita Madaan [62]. Pour argumenter ses propos, Mohita Madaan propose un exemple ou il suppose un ensemble de données d'employés. Il propose une image de sa base de données illustré par la figure 3.

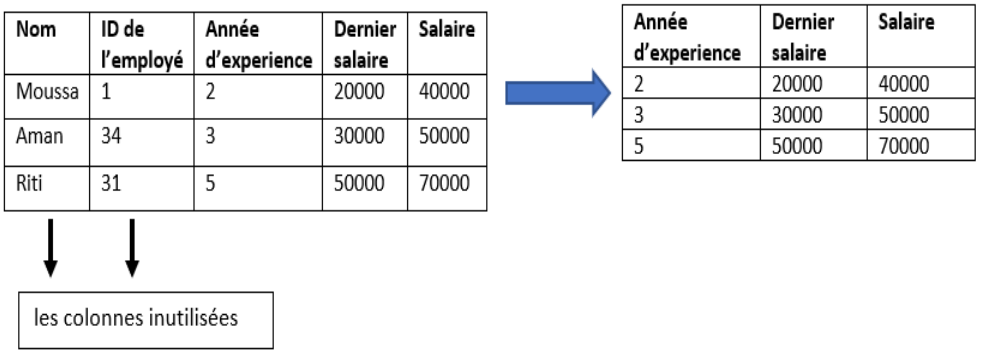

Figure 3: Exemple illustratif d'un ensemble de données d'employés

L'objectif du modèle est de prédire le salaire d'un employé. Ainsi la variable cible est le salaire et les autres colonnes tels que Nom, ID de l'employé, Nombre d'années d'expérience et salaire précédent sont les variables indépendantes. Pour l'auteur le Nom et l'ID de l'employé n'ont aucun rapport avec son salaire. En conclusion, Mohita Madaan suggère de supprimer les deux colonnes avant de former le modèle.

Plusieurs études ont présenté les avantages de réduire les caractéristiques d'une base de données. Nous retenons que cela permet de réduire le surajustement d'une part, c'est-à-dire que la sélection de caractéristiques élimine les doublons, ce qui réduit le phénomène de surajustement. Aussi, elle permet d'améliorer la précision, c'est-à-dire que si le phénomène de surajustement est faible alors le modèle fonctionnera bien sur les données de test et donnera une bonne précision. De plus, la sélection de caractéristiques permet de réduire le temps d'apprentissage du modèle, c'est-à-dire que si les données ne sont pas complexes alors le temps de calcul pour entrainer le modèle est faible. Enfin, la sélection de caractéristiques permet d'avoir un modèle généralisé et plus facile à expliquer et à interpréter. En fonction du type de données plusieurs techniques permettent de faire la sélection de caractéristiques. Les techniques peuvent être catégorisées comme suit : filter method, wrapper method et embedded methods.

La méthode de filtrage ou filter method consiste à supprimer des fonctionnalités en fonction du score obtenu par un test statistique (confère figure 4).

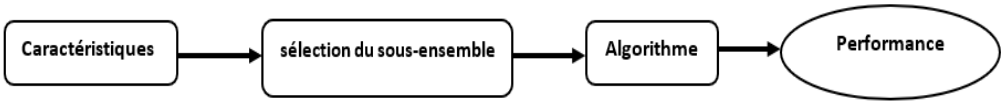

Figure 4: Méthode de filtrage de sélection de caractéristique

Ce score détermine la relation entre une caractéristique et la sortie. Les tests statistiques comme *Information gain, Fisher's score, Chi-square ou Pearson correlation* permettent d'évaluer une caractéristique [63]. Cette méthode classe les caractéristiques en fonction du score et supprime celles qui ont des scores inférieurs. C'est une méthode moins coûteuse en calcul et plus rapide que les *wrapper method.* La méthode de filtrage est recommandée pour traiter des données de grande dimension. Le Coefficient de corrélation ou *Pearson correlation* est une mesure de la relation entre deux ou plusieurs caractéristiques. Elle permet de sélectionner les caractéristiques qui ont une relation avec la variable cible(dépendant). Le

principe est que chaque caractéristique doit être corrélés avec la variable cible et non avec les autres caractéristiques. Selon [62], le seuil de sélection est de 0.5. Cependant, une caractéristique qui a un coefficient de corrélation inférieur à celui-ci avec la variable cible peut être supprimé. Le test du chi carré permet d'évaluer l'indépendance de deux caractéristiques catégorielles. Une valeur élevée du Chi-carré vérifié une dépendance élevé des deux caractéristiques. Alors, cette valeur permet de sélectionner les caractéristiques dépendantes du caractéristique cible.

Dans une étude de détection d'intrusion, Dogukan Aksu et al. [64] utilise la base de données CICIDS2017 qui est constitué d'attaques courantes bénignes et les plus avancées. Les chercheurs ont utilisé l'algorithme Fisher Score pour sélectionner les meilleures caractéristiques. Ils classifient les attaques grâces aux algorithmes comme *Support Vector Machine (*SVM*), K Nearest Neighbor* (KNN) et *Decision Tree* (DT). Les résultats sont intéressants et les taux de réussite atteints respectivement sont 0,9997%, 0,5776% et 0,99%.

La méthode d'emballage (*wrapper method)* permet de prendre un groupe de caractéristiques et de vérifier ce qui fonctionne le mieux avec un modèle.

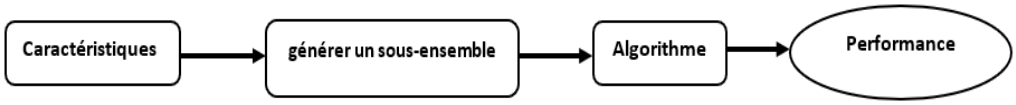

Figure 5: wrapper method pour la sélection de caractéristique

La méthode forme d'abord plusieurs groupes de caractéristiques, ensuite chaque groupe est utilisé pour former et évaluer le modèle d'apprentissage. Le groupe donnant la meilleure performance est sélectionné. Il existe plusieurs techniques pour former des groupes de caractéristiques (*forward selection, backward elimination* et *recursive feature elimination*). La première (*forward selection)* est une méthode itérative. Elle sélectionne d'abord la colonne la plus corrélé à la variable cible. Par la suite, elle ajoute une caractéristique et évalue la précision

du modèle. L'évaluation est effectuée à chaque itération. L'itération continue jusqu'à ce que l'ajout d'une nouvelle caractéristique n'améliore pas les performances du modèle. Le *Backward Elimination* est l'inverse *de Forward Selection*. Le *Backward* Elimination prend dès le début toutes les caractéristiques et supprime à chaque itération une caractéristique moins important. *Recursive Feature elimination* est une méthode qui cherche le sous-ensemble la caractéristique le plus performant. Il crée plusieurs modèles à chaque itération et supprime la caractéristique la moins corrélé.

Dans une étude Sutter et al. compare[65] *Forward Selection, Backward Elimination,* et *Generalized Simulated Annealing (GSA)* pour la sélection de caractéristique. L'étude est réalisée sur des trois ensembles de données de chimie. D'après leur étude comparative (GSA) s'avère plus performant. Néanmoins, les deux techniques améliorent considérablement la précision du modèle dans d'autres cas. Une troisième méthode appelé *Embedded methods* est aussi utilisé dans la littérature.

La méthode embarquées ou *Embedded methods* utilisent des modèles d'apprentissage automatique pour sélectionner les caractéristiques. L1 Regularization et Ridge regression sont les techniques utilisées pour pénaliser les modèles. Le *Regularization* est le plus utilisé selon [66].

Dans cet article [67] , il est question de traiter aussi des problèmes liés au distribution déséquilibrée des données et de la stratégie commune pour traiter les ensembles de données déséquilibrés. Il présente une méthode de sélection d'attributs qui représente un vecteur de poids basé sur une machine à vecteurs de support. Le modèle est capable de bien gérer les ensembles de données déséquilibrées. Son objectif fondamental était de supprimer les variables indésirables, améliorant ainsi le taux de précision de la classification. Leur méthodologie consiste à sélectionner les variables avec SVM et par la suite utiliser la cartographie auto-organisé ou *Emergent Self-Organizing Mapping (*ESOM) pour la classification non supervisée. Quelques bases de données populaires souffrent de ce

problème. Ce sont : la base de données de détection de fraude à la carte de crédit [68], détection de marées noires [69] et les données médicales [70]. Le problème est cité dans plusieurs revus comme [71], [72] et [73]. L'article annonce une nouvelle méthode hybride en fusionnant à la fois SVM et ESOM. Cette méthode est appelée *support-vector-based emergent self-organising map* (SVESOM). D'autres algorithmes se sont aussi inspirés de la nature comme OBBA, ce sont : l'algorithme Optimized Crow Search et l'algorithme Optimized Cuttlefish. Ils n'ont pas donné de meilleurs résultats comparés à OBBA.

Dans son cas, L'article [74] cherche à résoudre le problème de la distorsion des données, c'est-à-dire la répartition inégale des données entre les classes. En effet, trois algorithmes d'échantillonnage sont testés, ce sont *: Resampling, SpreadSubSampling* et *SMOTE*. SMOTE est une méthode de suréchantillonnage et SpreadSubSampling est méthode de sous-échantillonnage. Les trois techniques seront classées à l'aide de l'algorithme du K-plus proche voisin. L'expérience est menée plusieurs bases de données de santé comme celui du diabète, le cancer du sein et l'hépatite. Pour évaluer le modèle, les mesures de performance suivantes sont utilisées : *l'accuracy, positive predictive value, sensitivity,* et *F-score*.

2.3 INTELLIGENCE ARTIFICIELLE ET STATISTIQUES DANS LE DOMAINE DES ASSURANCES

Le secteur de l'assurance est un secteur actif et en plein expansion au sein du paysage économique mondial. Selon [75] « L'assurance est également un secteur assez particulier du point de vue du mode de fonctionnement de ses entreprises, puisque les entreprises et compagnies d'assurance ont des méthodes de gestion spécifiques adaptées à la nature de leur activité, qui est, rappelons-le, de savoir anticiper le risque pour pouvoir assurer ». Plusieurs domaines scientifiques cherchent à résoudre les problèmes rencontrés dans le domaine des assurances, notamment certains cherchent à prédire la qualité des chiffres comptables [[76], [77]] ou à améliorer la détection des irrégularités en cours [[78], [79]] , d'autres cherchent à expliquer pourquoi un client s'abonne ou abandonne une couverture d'assurance non-vie [80], d'autres présentent les facteurs dans le domaine de l'assurance qui

obligent les entreprises à utilisés les systèmes d'apprentissage automatique, de statistiques [81]. Il est évident que les données possédées par les assureurs peuvent être utilisées pour améliorer les prises de décision. Nous retenons une panoplie d'articles abordant plusieurs sujets traitant les données d'assurance. Ci-après quelques articles qui utilisent les techniques d'intelligence artificielle, de science des données, d'apprentissage automatique et de statistiques sur des données d'assurance.

Les articles repêchés tels que [[82],[83],[81],[84]] font une comparaison entre les algorithmes d'apprentissage machine et les GENERALIZED LINEAR MODELS (GLM). En effet, GENERALIZED LINEAR MODELS (GLMs) sont les modèles qui ont toujours été utilisés en pratique dans le domaine de l'assurance. Les GLMs sont des modèles d'apprentissage statistiques utilisés depuis 1980 par les actuaires. Selon [3], Ils sont utilisés pour la tarification et la réservation. Les GLMs ajoutent une fonction de lien pour exprimer la moyenne d'une variable aléatoire en fonction d'une relation linéaire entre les caractéristiques. Cet ajout permet la modélisation d'effets non linéaires simples. D'autres modèles comme les réseaux de neurones, les arbres de décision et les machines à vecteurs de support promettent des résultats satisfaisants pour la tarification d'après [82]. Selon [85], les raisons ayant favorisé une poussée de popularité des réseaux de neurones ces temps-ci sont : l'introduction de meilleures fonctions d'activation, les bases de données sont composés de beaucoup d'images et les GPU sont beaucoup plus puissants. L'article [3] présente les inconvénients des GLMs et montrent deux avantages du machine Learning. Le premier est que les modèles de ML apprennent les transformations non linéaires et les interactions entre les variables à partir des données sans les spécifier manuellement. Ceci est réalisé implicitement avec des modèles arborescents et explicitement avec des réseaux de neurones. Deuxièmement, il existe de nombreux modèles pour différents types de formats d'entités. En fonction du problème, il existera forcément un bon modèle. Aussi, il y'a des pertes d'information. Cela parvient au moment où les actuaires structurent les données. En somme, les GLM fonctionnent uniquement avec des données structures comparé aux modèles d'apprentissage machine.

Dans leur article [3], les chercheurs ont recensés les compétions sur KAGGLE et se sont rendus compte que les arbres de décision fonctionnent mieux pour les problèmes structurés, tandis que les réseaux de neurones fonctionnent mieux pour les problèmes non structurés. Ils ont aussi remarqué que la tarification utilise des données structurées et le boosting (XGBoost, GBT) est le cadre de tarification le plus populaire, tandis que la réservation utilise des données non structurées et les réseaux de neurones sont les plus populaires pour les modèles de réservation. Les modèles additifs généralisés (GAM) sont populaires pour la tarification puisqu'ils sont des généralisations des GLM. Alors les chercheurs ont utilisé un réseau de neurones de trois couches entièrement connectées à titre d'exemples. En effet, ils construisent une fonction qui retourne une ou plusieurs variables et prend en entrée des caractéristiques du modèle. Le modèle construit prend aussi des paramètres. La dernière couche est appelée « couche de sortie » puisqu'il s'agit de la sortie du modèle de classification ou de régression. Chaque fonction est non linéaire. Pour la régression, il existe une seule valeur de sortie représentant la valeur prédite. Un GLM correspond à un réseau de neurones sans couche cachée et à une seule couche de sortie. Par conséquent, un réseau de neurones avec de nombreuses couches cachées peut être considéré comme des GLM empilés. Les paramètres des variables aléatoires sont estimés à partir de la sortie du réseau de neurones en utilisant une approche pour la régression de Poisson non linéaire [86]. Plusieurs articles sont publiés dans le but de faciliter la tarification. Plusieurs techniques d'apprentissage automatique sont utilisées pour concevoir des modèles.

Selon [3], l'objectif de la tarification est de prévoir les coûts futurs associés au contrat d'assurance d'un nouveau client sans informations sur l'historique des réclamations pour ce client. Pour la tarification conventionnelle, les GAM et les réseaux de neurones permettent de rendre la fonction de score plus flexible. Cela leur permet de bien modéliser adéquatement la distribution des réponses. Une des méthodes utilisées se base sur des arbres, et elle offre des bons résultats. Aussi, [87] présente l'algorithme multiclass Adaboost pour classer le nombre de réclamations déposées au cours d'une période, le gradient boosting est appliqué à la prédiction des coûts des pertes par accident responsable [88], Les arbres de décision

multivariés sont appliqués pour modéliser la distribution conjointe des variables de réponse dans plusieurs couvertures [89], l'algorithme TDboost présenté par [89] utilise l'amplification de gradient pour estimer les paramètres d'une distribution de Tweedie. Pour ce qui est de la Tarification neuronale, plusieurs méthodes sont utilisées. Le premier cité est publié dans [90], l'article utilise des réseaux de neurones pour comparer des modèles d'apprentissage statistique pour estimer la prime pure. Ces derniers ont aussi fait la comparaison avec l'algorithme support vector regression. Ils ont conclu que leurs performances prédictives ne sont pas bonnes puisque les données sont asymétriques et que le modèle doit être équilibré pour apprendre quelque chose d'utile. Un critère d'équité est ensuite défini dans le but de s'assurer que la prime pure ne discrimine pas systématiquement une classe. Un autre article [91], utilise aussi un réseau de neurones pour prédire la fréquence a posteriori. Leur modèle prend en entrée la fréquence historique des réclamations pour un contrat, le facteur de crédibilité et la fréquence annuelle estimée des réclamations. Le modèle donne en sortie la fréquence annuelle estimée des sinistres pour l'année suivante. Le troisième travail [92], utilise un réseau de neurones pour modéliser des relations non linéaires qui n'ont pas été capturées par un modèle plus simple. L'approche est nommée Combined Actuarial Neural Network (CANN). Pour ce qui est de la tarification télématique, quelques travaux sont faits. Rappelons que les données télématiques sont des sources de données non structurées. Dans la section suivante est détaillés les articles cités.

L'algorithme Gradient Boosting (GB) est utilisé pour prédire des couts de pertes par accident automobiles dans [88] . Le but est de démontrer que GB donne plus de précision prédictive par rapport à l'approche conventionnelle du modèle linéaire généralisé (GLM). GB a l'avantages d'être un algorithme itératif qui combine des fonctions paramétrées simples avec des performances médiocres pour produire une règle de prédiction très précise. Aussi, il donne des résultats interprétables, tout en nécessitant peu de prétraitement des données et de réglage des paramètres. La prédiction dans ce cas d'étude consiste à prendre comme variables d'entrée, une collection d'attributs quantitatifs et qualitatifs du véhicule et de l'assuré, et la sortie est le coût réel des pertes. Deux modifications supplémentaires à l'algorithme de

gradient boosting ont été proposées, à savoir la régularisation par le rétrécissement des apprenants faibles contribués et l'injection d'aléatoire dans le processus d'ajustement. Les données utilisées par l'algorithmes se composent d'informations sur les politiques et les réclamations au niveau du véhicule individuel. L'ensemble de données comprend 426 838 expositions mesurées en années-véhicules de janvier 2006 à juin 2009, et 14 984 sinistres survenus au cours de la même période, avec des pertes basées sur les meilleures estimations des réserves en décembre 2009. Quelques caractéristiques du conducteur sont : L'Age, l'année de licence, le sexe ou État civil. Quelques Historiques des accidents/condamnations sont : Nombre d'accidents imputables (1 à 3 dernières années), Nombre d'accidents imputables (4 à 6 dernières années). Quelques Caractéristiques de la politique sont : années depuis la création de la politique, présence de multi-véhicules, franchise collision. Quelques caractéristiques de véhicules sont : marque du véhicule, véhicule acheté neuf ou d'occasion et rapport puissance/poids. L'expérience a consisté à d'abord partitionner les données en ensembles de données d'entraînement (70 %) et de test (30 %). L'analyse des résultats prouve que le niveau de précision de la prédiction s'est avéré plus élevé pour GB par rapport à l'approche conventionnelle du modèle linéaire généralisé. En plus Il ressoude d'autres défis qu'a le GLM. Ce sont par exemple : Le grand nombre de prédicteurs catégoriques et numériques, la présence de non-linéarités dans les données et les interactions complexes entre les entrées, les données non propres et/ou contenir des valeurs manquantes.

Le boosting est aussi utilisé pour prédire les coûts futurs associés à un contrat d'assurance dans [93]. Les auteurs ont comparé le résultat de prédiction de trois techniques comme le modèle GLM, le GAM, les arbres de décision. Leur but est de développer des plans tarifaires complets basés à la fois sur la fréquence et la gravité des réclamations. Le plan de tarification doit être transparents et interprétables et facilement explicables à toutes les parties prenantes. Trois types d'arbres sont utilisés à savoir : les arbres de régression simples ou Regression tree, random forests ou Forêt aléatoire et boosted trees ou Boosting machine. Les arbres utilisent la déviance gamma ou de poisson comme fonctions de perte. Ils sont parvenus à la conclusion que gradient boosting est la meilleure approche de modélisation. Le test est

effectué sur un portefeuille de responsabilité civile automobile (MTPL) d'un assureur belge en 1997. Les informations sont sur 163 212 assurés uniques. L'ensemble de données liste cinq facteurs de risque catégoriels, quatre continus et deux spatiaux, chacun d'eux informant sur des caractéristiques spécifiques de la police ou de l'assuré. Les paramètres optimaux ce chaque modèle est utilisé pour effectuer la comparaison. Les arbres boostés surpassent les GLM classiques, permettant à l'assureur de constituer des portefeuilles rentables et de se prémunir contre une éventuelle sélection défavorable des risques.

Les algorithmes *Regression tree* peut être améliorer pour capturer les effets covariables non linéaires et/ou d'interaction. Cette modification est prise en charge dans [94]. L'algorithme est utilisé pour une tarification a posteriori. Les auteurs ont présenté le modèle de crédibilité de l'arbre de régression, une méthode basée sur l'arbre pour la tarification avec la théorie de la crédibilité. La classification a priori permet d'exprimer correctement des informations a priori sur un nouvel assuré ou un assuré n'ayant jamais subi de sinistres. La prime de crédibilité classique de Bühlmann-Straub est appliquée dans chaque nœud terminal de l'arbre [94]. L'approche permet d'intégrer des informations sur les covariables dans la prédiction de la prime de crédibilité. L'algorithme binaire récursif partitionne un collectif de risques individuels en sous-collectifs mutuellement exclusifs et applique la formule de crédibilité classique de Bühlmann-Straub pour la prédiction des primes nettes individuelles. Il sélectionne à priori les variables influentes pour la prédiction de la prime et ne nécessite aucune procédure de sélection de variables supplémentaire.

Un autre algorithme de boosting appelé Delta Boosting (DB) est présenté dans [95]. Il est un semblable au populaire Gradient Boosting (GB). Le gradient boosting original repose sur trois actions : une base, une régression et un ajustement. La machine delta boosting est proposée, combinant les étapes de régression et d'ajustement. Par conséquent, l'algorithme est dit efficace en termes de calcul. DB tente de réduire la perte à chaque itération en ajustant séquentiellement un apprenant de base simple pour compléter les prédictions en cours. Au lieu de s'appuyer sur le gradient négatif, comme c'est le cas pour GB, DB adopte une nouvelle

mesure appelée delta comme base. Delta est défini comme le minimiseur de perte à un niveau d'observation. Les tests sont faits sur les données de réclamation d'assurance d'un assureur canadien. La base de données est formée de renseignements sur les polices et les réclamations au niveau du véhicule pour la couverture en cas de collision dans une province donnée. La *dataset* comprend les données d'accidents de 2001 à 2005 et 290147 années-véhicules. Les *target* étudiés sont : la fréquence des réclamations, le nombre de réclamations par année de véhicule, la gravité (paiement moyen d'un sinistre) et le coût des pertes (le paiement moyen par année de véhicule). Pour cela 85% des données sont sélectionnées au hasard pour l'apprentissage et le test (La suppression des observations avec des valeurs manquantes est utilisé sur la même *dataset* dans [84]). En pratique, l'algorithme DB donne plus d'importance aux nombres d'années passé auprès de l'assureur, c'est-à-dire la fidélisation de la clientèle (2,9 % pour GB et 6,5 % pour DB). Sachant que le meilleur modèle doit avoir la *Loglikelihood* négative la plus faible, DB est le meilleur avec −139.50 (GB avec −135.80 et GLM avec 0.0). Ils concluent alors que plus d'informations peuvent être extrait des données en utilisant DB que GB.

D'autres travaux portant sur l'extension des arbres sont proposés dans [89] où des arbres de décision multivariés sont appliqués pour modéliser la distribution conjointe des variables de réponse dans plusieurs couvertures. Des extensions aux forêts aléatoires et à l'amplification de gradient sont également présentées. Il est utilisé pour la tarification conventionnelle. Aussi dans [96], l'expérience est de modifier l'algorithme CART (Classification And Regression Tree) pour prendre en compte les spécificités de l'assurance non-vie. Alors, au lieu de diviser le total des sinistres par l'exposition pour revenir à la base d'exposition unitaire, la compensation est incorporée dans la fonction de déviance, qui a servi de critère de répartition.

Les travaux [97] présentent une approche pour catégoriser les styles de conduite à travers l'utilisation de cartes thermiques vitesse-accélération. Ces cartes thermiques sont générées pour différentes tranches de vitesse afin de représenter les tendances de vitesse et

d'accélération des conducteurs. Ensuite, l'algorithme k -means est appliqué pour créer des clusters similaires de conducteur. Pour ce faire, les compagnies d'assurance automobiles collectent les données de localisation GPS haute fréquence de leurs conducteurs. Ces informations seront analysées à l'aide de techniques de reconnaissance de formes et d'apprentissage automatique. L'expérience à consister à prendre des données de 1753 automobilistes. Pour chacun de ces automobilistes, ils ont enregistré 200 trajets individuels. Les données telles que la vitesse, la valeur absolue de l'accélération et du freinage et l'intensité des virages sont les caractéristiques analysées. L'objectif est de classer les conducteurs qui ont les *heatmaps* similaires. Ces conducteurs seront considérés comme ayant des styles de conduite similaires et sont donc placés dans les mêmes catégories de catégories pour la tarification de l'assurance automobile. Après dans leurs article suivant [98], ils utilisent l'analyse en composantes principales (PCA) et des auto-encodeurs pour projeter les cartes thermiques vitesse-accélération dans un vecteur bidimensionnel pouvant être utilisé dans le cadre de variables de notation. Le but est de montrer qu'une représentation bidimensionnelle est suffisante pour reconstruire les cartes thermiques avec une grande précision. Pour cela leur idée est de réduire la dimension à l'aide de l'analyse en composantes principales (PCA). La méthodologie est de remplacer les classes catégorielles fournies par l'algorithme K -means par des traits continus de faible dimension. Deux possibilités sont présentées : Utiliser SVD pour une PCA des *heatmaps* ou utiliser des réseaux de neurones qui conduisent à une PCA non linéaire. L'expérience a montré que les réseaux de neurones et SVD parviennent à fournir des représentations continues de faible dimension d'objets complexes tels que heatmaps. Les assureurs peuvent utilisés ces représentations continues dans des modèles de régression en tant que covariables continues.

L'apprentissage automatique améliore aussi les estimations comptables. Rappelons que la comptabilité utilise beaucoup d'estimations. Les éléments comme le bilan, le compte de résultat, les dépenses liées aux régimes de retraite, les options d'achat d'actions des clients découlent d'estimations. Cependant quelques erreurs d'estimation objectives ou des manipulations managériales peuvent affectées négativement l'estimation. Pour cela l'article

[99] utilise l’apprentissage automatique pour améliorer les estimations managériales. Le test est effectué sur des données de compagnies d’assurance. Les résultats prouvent que les estimations faites par l’apprentissage machine sont supérieures à celles effectues par l’estimations managériales. Pendant que plusieurs études capturent la discrétion managériale excessive ou les anomalies de risque de crédit grâce à l'analyse des chiffres des états financiers, cet article présente comment l'apprentissage automatique peut améliorer directement l'estimation du solde d'un compte, révélant ainsi les mécanismes par lesquels l'apprentissage automatique peut atténuer les erreurs intentionnelles et non intentionnelles. Les auteurs utilisent quatre algorithmes de ML pour la prédiction des pertes d’assurance. Les quatre algorithmes utilisés sont la régression linéaire*, random forest, gradient boosting machine*, et *artificial neural network.* Pour la régression linéaire, la méthode M5 est utilisé pour sélectionner les attributs de données. *Cartesian grid search* est utilisé pour La détermination de la combinaison optimale d'hyper-paramètres. Il a permis de configurer les hyper-paramètres optimaux. L’approche formation, validation et test sont utilisés sur les échantillons. La méthode de validation croisée est utilisée pour développer le modèle d’apprentissage automatique de chaque algorithme. L’échantillons est constitués à partir des rapports annuels des compagnies d'assurance IARD basées aux États-Unis déposés auprès de la *National Association of Insurance Commissioners* (NAIC). Cinq secteurs d'activités sont concernés : la responsabilité civile automobile des particuliers, la responsabilité civile automobile commerciale, l'indemnisation des accidents du travail, les risques multiples commerciaux et le propriétaire/propriétaire agricole. Les métriques d'évaluation utilisés pour comparer les estimations aux chiffres réels sont : l'erreur absolue moyenne (MAE) et l'erreur quadratique moyenne (RMSE). Ces deux mesures de performance ont permis d’évaluer les performances de la solution (IA) en comparer les prévisions de perte de machine Learning avec les estimations des managers. Un exemple concret de résultat est que le modèle développé de 1996 à 2006 pour prédire les pertes encourues en 2007, les modèles forestiers aléatoires prédisaient respectivement 38 953 000 $ et 40 996 000 $. L'estimation des gestionnaires était

de 43 650 000$ pour la même année, tandis que les pertes réelles se sont avérées être de 39 791 000$.

Un modèle d'intelligence artificielle qui cherche à expliquer pourquoi un client s'abonne ou abandonne une couverture d'assurance non-vie est présenté dans [80]. Elle permet de comprendre immédiatement les facteurs qui contribuent aux décisions des clients. La méthode proposée consiste à prédire d'abord des valeurs à partir de l'algorithme de classification *l'Extrême Gradient Boosting* (XGBoost), ensuite ces valeurs obtenues seront clustérisées. Les valeurs obtenues par XGBoost sont des scores qui correspondent à l'importance des variables. Ces simples valeurs ne sont pas interprétables, c'est la raison pour laquelle ces données seront par la suite utilisées par le modèle d'IA pour surmonter le problème (rendre interprétable). L'IA utilise l'approche par la valeur de Shapley. Pour cette approche, la variabilité des prédictions est répartie entre les covariables disponibles. De cette manière, la contribution de chaque variable explicative à chaque prédiction ponctuelle peut être évaluée, quel que soit le modèle sous-jacent présenté dans [100], de manière indépendante du modèle. L'algorithme de *clustering* k-means est utilisé pour déterminer si les consommateurs peuvent être fusionnés en groupes. Les tests sont effectués sur des données de la société INSURTECH NEOSURANCE, basée en Italie. Les données concernées sont les achats d'instantanés et de micro-assurances dans le domaine du sport et du voyage. Les comportements étudies sont les tendances à acheter et le taux de désabonnement des clients. Ces données sont reparties en 3778 utilisateurs pour estimer la propension à acheter, et 1689 utilisateurs afin d'estimer le taux de désabonnement des clients. Comme variables explicatives, l'expérience utilise des informations démographiques (principalement le sexe, l'âge, l'emplacement approximatif et l'appareil utilisé) et des informations concernant l'historique et le comportement d'achat, l'utilisation de l'application et l'expérience utilisateur. La prédiction visée « acheter » dans le cas de la propension à acheter et l'événement « quitter » dans le cas du désabonnement. L'expérience donne en résultat 27,5% pour « acheter », tandis que pour l'étude de désabonnement, elle est de 53,3%.

Des sujets portant sur l'importance du machine Learning pour les industries d'assurance et les défis qui peuvent entraver leurs mise en œuvre sont discuter dans [81]. Une des raisons est le fait que la quantité de données augmente de manière exponentielle, ce qui rend les données inexploitables en totalité par les humains. Les mégadonnées sont le principal point d'information dans le secteur de l'assurance. Aussi la concurrence entre les entreprises dans le domaine est rude. Chaque entreprise veut proposer les meilleurs services pour se procurer plus de clients. Ainsi il présente l'apprentissage machine comme un moyen d'ajouter de la valeur ajouter à l'industrie de l'assurance en permettant par exemple : la souscription, l'évitement des confiscations, l'évitement des déchéances, la gestion des droits, la détection des fraudes, l'évaluation des produits, les transactions et la capacité des clients à adhérer ou à refuser un produit. Il fournit avec précision une chaîne de valeur en identifiant les risques, l'action du client et les droits en utilisant une précision de prévision innovante. L'article présente le fait que tous les algorithmes d'apprentissage (supervisé, non supervisé et par renforcement) sont utilisés en fonction de l'objectif visé. Selon l'article, tout type de données peut être exposé à des méthodes d'analyse de données pour recevoir une compréhension qui pourrait être appliquée pour améliorer les connaissances et les procédures. Pour cela, l'analyse de données est classée en 4 classes : Analyse descriptive, Analyse prédictive, Analyse prescriptive et l'analyse extrapolative. Dans l'article, il est précisé que parmi les entreprises du secteur de l'assurance, la majorité utilisent essentiellement des modèles linéaires généralisés (GLM) conventionnels pour examiner le risque de prix [101]. C'est une approche statistique appliquée pour prédire la possibilité d'avoir des liens entre diverses variables de risque. Plus loin l'article présente quatre motivations à utiliser le ML en assurance : les données ne cessent de croître, les modèles open source comme l'apprentissage automatique aident à la gestion des mégadonnées, La taille et la vitesse des données ouvrent la voie à l'émergence de l'apprentissage automatique dans la surveillance et l'élaboration avant que la structure de programmation devienne plus complexe, le ML ne se base pas sur l'humeur du client ou du gestionnaire pour prendre une décision, cela permet de créer plus de valeur. Aussi, des études de cas sont présentées. L'article présente le fonctionnement des compagnies d'assurance.

L'industrie de l'assurance effectue une inspection pour vérifier leurs décisions de souscription en fonction des informations recueillies sur le danger avant la couverture et après l'initiation avec le cycle de réactivation. Cela les aide à reconnaître tout danger réel ou potentiel et à aider leurs clients à gérer les catastrophes, réduisant ainsi leur exposition. Le premier étude cas vise à éliminer la période d'inspection et à améliorer la productivité des enquêteurs grâce à l'utilisation de l'apprentissage automatique. Cette approche sera non seulement rentable pour l'expert lors de la collecte du compte d'inspection, mais pourrait aider le souscripteur à prendre des choix de souscription bien informés et éventuellement aider les clients dans leur gestion des catastrophes. Le prochain cas, concerne l'internet des objets : il permet aux assureurs de proposer des services aux clients. Les services sont par exemple : un capteur d'humidité ou de fuite aux clients. Ces services sont offerts à bas prix et évite des dégâts. Un autre cas est l'accélération et l'automatisation du paiement des droits en quelques heures ou jours au lieu de semaines ou mois comme défis rencontrés lors de la mise en œuvre de l'apprentissage automatique, nous retenons : *Training Necessities* (L'entrainement), *The right source of data* (La bonne source de données*), Exertion in forecasting returns* (l'effort de prévision des rendements) *et Data security*. Pour l'entrainement, les données doivent être entraînés avec d'énormes volumes de documents ou de transactions pour couvrir toutes les situations potentielles [102]. Ensuite pour la qualité, les données doivent être symboliques et stables pour pouvoir prendre une image de qualité et échapper à la partialité. Il est aussi compliqué de budgétiser un produit d'IA et de préciser le rendement du projet. Enfin l'utilisation d'algorithmes d'IA ou d'infrastructure externe créé un risque de sécurité supplémentaire pour les données.

[103] et [104] proposent des réseaux de neurones pour modéliser la fréquence des réclamations, soit directement, soit via un GLM imbriqué. L'imbrication permet d'explorer la structure du modèle non capturée par le GLM simple. Il permet d'optimiser les résultats en se rapprochant plus de l'optimal. Il permet aussi de réduire le temps de calcul car l'algorithme d'ajustement utilisé débute avec un paramètre raisonnable. Comme résultats le Model GLM2 donne un *in-sample-loss* de 31.25674 et un *out-of-sample -loss* 32.14902 (soit un *average frequency* de 10.01%) comparé aux réseaux de neurones à deux dimensions qui donne un un

in-sample-loss de 30.16513 et un *out-of-sample -loss* 31.45327 (soit un *average frequency* de 9.70%). Cela montre l'importance de l'imbrication. Il est important de retenir que dans certains pays (exemple de loi du Conseil de l'Union européenne [105]) les modèles de tarification des assurances sont réglementés. Selon [106], la loi permet aux individus de demander une explication de la logique derrière la décision, cela oblige à utiliser dans la pratique des modèles transparents et faciles à expliquer aux intéressés. Ces conditions constituent un frein à certains algorithmes d'apprentissage automatique. Il est important de retenir que le plus important pour un assureur est la bonne mesure de risque. Bien que les GLM sont des fonctions explicables et compréhensible par les parties prenantes, il est important de préciser que plusieurs techniques d'intelligence artificielle comme les arbres de décision, les réseaux de neurones et bien d'autres offrent la possibilité d'avoir plus précision.

L'objectif de ce chapitre est de comprendre la question de recherche. Pour cela, il était donc question de présenter des travaux antérieurs sur le sujet. D'abord, les sujets intéressants qui ont portés sur l'optimisation des résultats de classification ou de prédiction à l'aide d'algorithmes d'apprentissage automatique sont présentés. Nous retenons que plusieurs techniques *comme competitive swarm optimizer (CSO), particle swarm optimization (PSO), l'algorithme optimiseur Binary Sailfish, du Sailfish Optimizer (SFO), Fast Correlation-Based Feature Selection* (FCBF) sont développés par les chercheurs dans le but d'optimiser les résultats.

Ensuite, est présenté les articles qui ont abordés l'optimisation des résultats de classification ou de prédiction par le pré-traitement et traitement des données. Quelques algorithmes recensés par la littérature sont : Resampling, SpreadSubSampling, SMOTE ou Optimized Binary Bat algorithm (OBBA).

Nous présentons une liste de quelques articles (confère tableau 2) qui traitent les données d'assurance à l'aide d'algorithme d'apprentissage. Ces articles présentent plusieurs techniques tels que XGBoost, RF, DT, SVR, SVM, etc.

Tableau 2: Présentation des articles sur l'apprentissage automatique en actuariat

Description	Méthodologies/Approches	Référence
Étude comparative	GLM, DT, GBT, NN	Noll et al. (2018) [107]
Étude comparative	GLM, RF, GBT, NN	Diana et al. (2019) [83]
Étude comparative	XGBoost, RF, LR, NN,	Maynard et al. (2019) [108]
Lecture notes	GLM, GAM, NN, RF, GBM, SVM	Wuthrich et Buser (2019) [109]
Étude comparative	SVR, Kernel LR	Kašćelan et al. (2016) [110]
Review	NN, RF, GBM, SVM	Corlosquet-Habart and Janssen (2018) [111]
Review	CART, NN, XGBoost	Grize et al. (2020) [112]
Report	RF, NN, GBM	Jamal et al. (2018) [113]
White paper	GLM, GBT, NN	Panlilio et al. (2018) [114]
Review	NN	Richman(2020a, 2020b) [115], [116]
Report	NN	Harej et al. (2017) [117]
White paper	NN	Richman et al. (2019)[118]

Enfin, les articles qui ont utilisés l'IA, la science des données, l'apprentissage automatique ou la statistique sur des données d'assurance sont présentés. Nous retenons que beaucoup d'articles sont présentés, nombreux sont ceux qui porte sur le processus de

prédiction et non de pré-traitement. Nous retenons que les GLM sont largement utilisés par les compagnies depuis 1980 car Ils ont permis d'améliorer la qualité des modèles de prédiction du risque [119]. Bien qu'il ait eu des améliorations par rapport à ses prédécesseurs comme le modèle de régression linéaire simple, cependant, plusieurs limites lui contraintes selon [119]. Cela conduit aux développements d'autres approches d'apprentissage automatique. Dans la section suivante nous aborderons la méthodologie utilisée pour mettre en place notre solution.

CHAPITRE 3

DESCRIPTION DE LA SOLUTION

L'objectif de ce chapitre est de présenter le pipeline de traitement des données et de classification issue de ce projet de recherche afin d'automatiser le processus de pré-traitement des données. Pour cela, nous présentons d'abord la solution initiale réalisée par deux étudiants de baccalauréat en informatique qui ont travaillé sur les mêmes données. Cette solution initiale servira d'élément de comparaison avec la solution que je propose dans ce mémoire. De plus et afin de bien comprendre la solution proposée, nous présentons chaque élément constituant la modélisation des deux pipelines. Nous décrivons également les méthodes classiques d'optimisation utilisées dans la solution initiale qui ont les meilleures performances de classification, mais aussi leurs limites. Nous terminons avec une présentation des algorithmes métaheuristiques de la solution proposée, qui permettent de contourner les problèmes rencontrés par les méthodes classiques d'optimisation.

3.1 APERÇU DE LA SOLUTION INITIALE

La solution initiale est une mise en place d'une chaîne d'opérations permettant de nettoyer et de transformer automatiquement les données en vue de réaliser une tâche de classification, incluant les étapes d'entraînement et de test des algorithmes d'apprentissage automatique. Plus précisément, le pipeline est constitué d'un prétraitement des données, d'une sélection de caractéristiques (variables), d'une prédiction (classification) et d'un post-traitement. De plus, nous paramétrons (*hyperparameter fine tunning*) les algorithmes d'apprentissage automatique afin d'obtenir les meilleures performances de classification. La solution initiale a été testée avec les données de réclamations d'assurances automobile d'une compagnie d'assurance. Les données sont présentées au chapitre suivant. La figure 6 illustre la solution initiale.

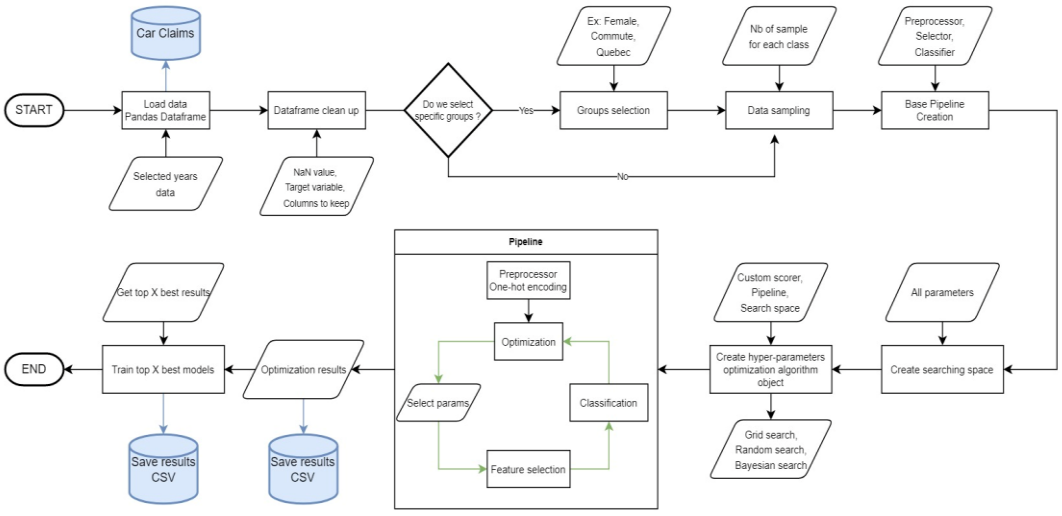

Figure 6: Pipeline de base avec optimisation brute force

3.2 CHARGEMENT ET PREPARATION DU JEU DE DONNEES

La première étape consiste à charger les données à partir d'une base de données de réclamations clientes concernant les autos assurées chez CO-OPERATORS. Le seul paramètre utilisateur disponible et qui permet de contrôler, dans une certaine mesure, la nature des données est une liste des années des réclamations que nous voulons traiter. La Figure 7 illustre cette étape.

Cette étape permet à l'utilisateur de définir les valeurs par défauts qui seront utilisées par les algorithmes. Il permet par exemple de :

- sélectionner une liste d'années qui seront utiliser pour constituer la base de données
- sélectionner une variable cible en fonction de l'objectif de prédiction visé.
- sélectionner un groupe particulier d'individus comme les femmes ou hommes
- Choisir l'emplacement ou sera sauvegardé les résultats en format CSV.
- Sélectionner les algorithmes ainsi que leurs paramètres utilisés pour l'optimisation

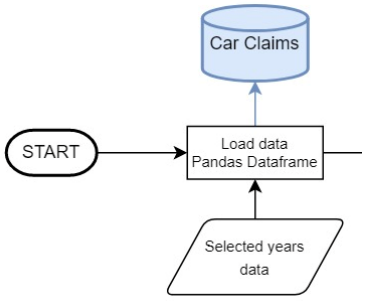

Figure 7: Etape 1 du pipeline

- Chargement des données de réclamations

selon des années

La seconde étape illustrée à la figure 8 est le nettoyage des données issues de l'étape 1. Plus précisément, cette étape se charge de supprimer toutes les observations (une observation = une réclamation) qui ont des données manquantes. Ces données manquantes sont représentées par les valeurs -1, 0, NaN (*Not a Number*), *Not Available,* None, etc. De plus, seulement les colonnes (variables) ayant peu de données sont supprimées. Les autres sont conservées. Enfin, nous uniformisons également certaines valeurs de caractéristiques. Par exemple, pour la colonne « TYPE DE COUVERTURE », il y a deux valeurs qui sont *accident benefits* et *accident benefits 2015*. Afin d'éviter la complexification de l'ensemble données pour quasiment aucun gain de généralisation des performances de classification, nous fusionnons les deux valeurs en remplaçant *accident benefits 2015* par *accident benefits.*

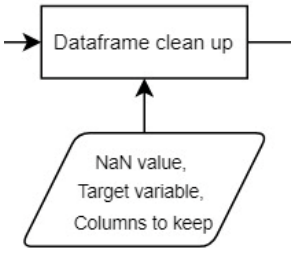

Figure 8: Etape 2 du pipeline – Nettoyage de l'ensemble de données

La troisième et dernière étape dans le chargement et la préparation de l'ensemble de données consiste à échantillonner l'ensemble de données afin de cibler plus précisément des clients (p. ex., les personnes de genre féminin où la réclamation a été effectuée en Ontario ou en Alberta), mais aussi d'alléger la charge mémoire et les temps de calcul liés à l'entraînement de certains algorithmes d'apprentissage automatique. En prenant tous les clients, il est possible qu'aucun modèle d'apprentissage automatique ne puisse modéliser adéquatement les relations entre les variables explicatives et la variable cible. Ainsi, la sélection d'un sous-groupe de l'ensemble de données permettrait de cibler plus précisément une des relations entre les variables explicatives et la variable cible. Par ailleurs, à un plus haut niveau, cette sélection d'un sous-groupe peut contribuer à réduire les risques de biais que le modèle peut avoir entre différents clients. Pour l'échantillonnage des instances, nous utilisons la classe *RandomUnderSampler* de la librairie *imbalanced-learn*. Elle possède l'avantage d'équilibrer le nombre d'instances pour chaque classe afin d'éviter d'obtenir un modèle biaisé par une classe majoritaire. L'inconvénient de l'échantillonnage des instances est la nécessité d'exécuter plusieurs fois l'échantillonnage pour valider les résultats obtenus. La figure 9 illustre l'étape 3. Le paramètre utilisateur nécessaire est la liste des valeurs de caractéristiques pour des variables explicatives particulière afin de sélectionner un groupe d'individus.

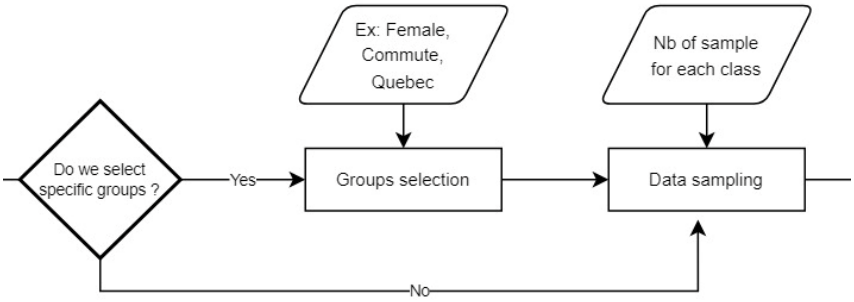

Figure 9: Etape 3 du pipeline - Echantillonnage de l'ensemble de données nettoyés

3.3 CONFIGURATION DE LA PARTIE APPRENTISSAGE AUTOMATIQUE

La configuration du pipeline pour la partie dédiée à l'apprentissage automatique consiste à définir les séquences d'actions qui vont automatiser le processus de prédiction. Dans notre projet de recherche, le processus de prédiction consiste à classifier une donnée. Les séquences d'actions définies dans le pipeline sont les transformations des caractéristiques, la sélection des caractéristiques et la classification des données. Ces séquences d'actions vont être implémentées dans un pipeline de base grâce à la classe Pipeline de la librairie scikit-learn. La figure 10 illustre cette étape 4. Pour la classification, le pipeline utilise trois algorithmes de classification qui sont : decision tree, random fores et catboost.

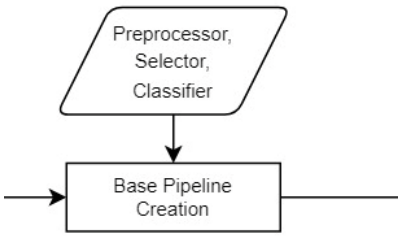

Figure 10: Etape 4 - Création du pipeline de base

3.3.1 TRANSFORMATION DES CARACTERISTIQUES

La transformation des caractéristiques est une étape essentielle dans le processus d'apprentissage automatique. En effet, elle permet d'éviter les biais dans les modèles générés liés à des valeurs de caractéristiques beaucoup plus grandes que les valeurs des autres caractéristiques. De plus, il existe des transformations qui consistent simplement à convertir des valeurs dans le format « string » en valeurs entières. Ceci est simplement lié au fait que les algorithmes acceptent seulement des « floats » ou des « integers ». Dans la solution initiale et proposée, plusieurs méthodes de transformation sont accessibles. Les méthodes utilisées sont : SimpleImputer (le strategie est median) et OneHotEncoder.

3.3.2 SELECTION DES CARACTERISTIQUES

La sélection des caractéristiques consiste à réduire le nombre de caractéristiques de l'ensemble de données. Cela permet de réduire la complexité d'un modèle d'apprentissage automatique et le rend plus facile à interpréter. L'étape de sélection des caractéristiques peut être effectuée manuellement ou automatiquement. Plusieurs algorithmes de sélection sont implémentés dans la solution initiale. Ces algorithmes sont supervisés ou non supervisés, les algorithmes non supervisés n'utilisent pas la variable cible pour assister la sélection tandis que les algorithmes supervisés l'utilisent.

Deux méthodes de filtrage intéressent qui sont utilisés dans le pipeline est la méthode qui permet de sélectionner les k meilleurs caractéristiques et la méthode qui permet de sélectionner les variables du centile supérieur. Il est difficile de désigner la meilleure méthode de sélection des caractéristiques ou le meilleur ensemble de variables pour le modèle d'apprentissage. Le choix dépend du problème à résoudre et c'est pour cela que l'optimisation rentre en jeux. Le rôle de l'optimisation est de choisir le meilleur sous ensemble ou la meilleure méthode en fonction du problème et cela vérifié notre choix à utilisés plusieurs les techniques.

3.3.3 CLASSIFICATION

La prédiction fait référence à la sortie d'un modèle (de classification) après que ce dernier a été entrainé sur un ensemble de données et appliqué à de nouvelles données. Il existe plusieurs algorithmes de prédiction comme les réseaux de neurones, *k-Nearest Neighbors, Decision Trees, Naive Bayes, Random Forest* ou encore *Gradient Boosting.* L'algorithme est choisi en fonction de l'objectif, du type de données ou du problème à résoudre. Dans notre projet de recherche, nous utilisons la classification. Les modèles de classification sont généralement simples à comprendre et à expliquer. Il est à noter que le domaine des assurances se concentre principalement autour de la compréhension du processus de prise de décision et de tarification.

Pour effectuer une classification, nous pouvons utiliser des d'algorithmes d'apprentissage automatique qui apprennent à attribuer une étiquette aux instances. Par exemple, un algorithme peut apprendre à classer les courriels comme « spam » ou « non spam ». Dans notre projet de recherche, nous avons sélectionné les algorithmes suivants : les arbres de décision (*Decision Tree*), les forets aléatoires (*Random Forest*) et *Catboost*. Cette sélection a été réalisée en fonction du temps d'entraînement et de la facilité de compréhension des modèles générés.

ARBRES DE DÉCISION

Les arbres de décision sont produits par des algorithmes (p. ex., ID3, C4.5, CART) qui partitionnent les données en sous-ensembles en fonction de la théorie de l'information. La théorie de l'information permet d'identifier les variables les plus discriminantes et les meilleures divisions. De plus, le modèle résultant est un arbre de classification qui permet de suivre le processus de décision. Ainsi l'assureur pourra aisément expliquer pourquoi une décision fut prise à l'égard d'un client. Les arbres de décision ont inspiré d'autres algorithmes plus complexes tels que XGBoost ou *Random Forest*.

Il est aussi important de bien définir la profondeur de l'arbre. En effet, sa profondeur est directement liée à sa complexité. Un modèle avec une grande complexité peut conduire à un sur-apprentissage (*overfitting*) et un modèle trop simple peut conduire à un sous-apprentissage (*underfitting*). L'utilisation des arbres de décision est conseillée dans des cas spécifiques selon [120] : « *Les arbres de décision sont utilisés lorsque l'interprétabilité des données a plus de valeur que la performance et lorsque l'algorithme ne peut pas être industrialisé*. ».

FORET ALÉATOIRE

L'algorithme de l'arbre de décision est très facile à comprendre et à interpréter. Dans beaucoup de cas, un seul arbre ne suffit pas pour produire des résultats efficaces. Dans [121], nous pouvons observer que l'arbre de décision fonctionne bien sur l'évaluation dans l'échantillon, mais ses performances diminuent considérablement sur l'évaluation hors

échantillon. C'est le phénomène de sur-apprentissage. Pour pallier le risque de sur-apprentissage des arbres de décision, les chercheurs [122] ont proposés l'algorithme *Random Forest*. *Random Forest* exploite la puissance de plusieurs arbres de décision. Comme son nom l'indique, c'est une « forêt » d'arbres. Le termes « forêt aléatoire » provient du fait que les arbres de décision qui constituent la forêt sont créés au hasard. En d'autres termes, les arbres sont construits à partir d'un ensemble (plus petit que le nombre de variables de l'ensemble de données) de variables sélectionnées aléatoirement. La forêt combine ensuite la sortie des arbres de décision pour générer la sortie finale selon la stratégie du vote majoritaire. Nous avons sélectionné cet algorithme car selon [121], l'utilisation de *Random Forest* convient aux situations où nous disposons d'un grand ensemble de données.

Il est à noter que déterminer le nombre d'arbre pour construire sa forêt est une tâche compliquée. Quelques travaux [123-125] ont conclu qu'au-delà d'un certain nombre d'arbres, l'ajout d'autres arbres ne permettrait pas d'améliorer les performances de classification.

CATBOOST

C'est un algorithme reposant sur la théorie du *Gradient Boosting*. Le *Gradient boosting* est une technique d'apprentissage automatique qui produit un modèle prédictif composé d'un ensemble de modèles prédictifs faibles, généralement des arbres de décision [126]. Dans la famille des méthodes de *boosting*, nous retrouvons aussi les algorithmes LightGBM et XGBoost. Comparativement aux autres algorithmes, CatBoost génère automatiquement les données de type *string* et numérique.

Catboost doit sa popularité par rapport à d'autres algorithmes en raison des caractéristiques suivantes [127]:

- gestion native des fonctionnalités catégorielles,
- entraînement sur GPU,
- visualisations et outils pour l'analyse des modèles et des fonctionnalités,

- utilisation d'arbres inconscients ou d'arbres symétriques pour une exécution plus rapide,
- *boosting* ordonné pour limiter le sur-apprentissage.

3.3.3 EVALUATION DU MODELE

Un modèle d'apprentissage automatique a besoin d'être évalué. En d'autres termes, nous devons évaluer les performances de la tâche de classification ou de régression du modèle. Il existe plusieurs mesures de performance populaires telle la précision, le rappel, le F1-score, la justesse (*accuracy*), le coefficient de Kappa. Ces mesures de performance sont toutes calculées à partir de la matrice de confusion. Dans notre projet de recherche, nous avons mis en place une règle permettant d'améliorer la recherche des meilleures performances. Cette règle est présentée ci-dessous.

Règle : Les résultats de prédictions sont utilisés pour générer la matrice de confusion. Par la suite un ratio est utilisé comme score de passage pour le test. Cette note de passage peut être modifiée selon les besoins. Enfin si la matrice ne passe pas le test, le score retourné est de zéro, dans le contraire *l'accuracy* est retourné.

3.4 OPTIMISATION PAR VALIDATION CROISEE

Les paramètres utilisateurs sont passés comme paramètres au pipeline. Chaque paramètre (ou hyperparamètres pour les algorithmes d'apprentissage automatique) prend une ou plusieurs valeurs. La combinaison de toutes les valeurs possibles pour chaque paramètre constitue l'espace de recherche, permettant de déterminer les meilleurs choix pour la classification.

3.4.1 ESPACE ET STRATEGIE DE RECHERCHE

L'espace de recherche définit l'ensemble de toutes les solutions possibles. Les stratégies de recherche des meilleures solutions permettent d'effectuer une recherche complète ou partielle de l'espace de recherche. Plusieurs algorithmes existent. Nous avons

grid search, bayesian search et *random search*. Ces algorithmes sont voraces et nécessitent beaucoup de temps d'exécution lorsque les hyperparamètres peuvent prendre beaucoup de valeurs (l'espace de recherche augmente). Dans notre projet de recherche, *Grid* et *Random Search* sont les deux stratégies utilisées.

3.4.2 GRID SEARCH

L'optimisation des hyperparamètres des algorithmes d'apprentissage automatique avec la méthode *Grid Search* utilise la stratégie de la validation croisée en k plis (*k-fold cross validation*). La validation croisée en k plis permet d'assurer dans une certaine mesure quelles sont les performances réelles du modèle d'apprentissage automatique généré. *Grid Search* est un brute force puisque cette méthode teste toutes les combinaisons possibles des valeurs définies des hyperparamètres des algorithmes d'apprentissage automatique. La figure 11 illustre les différentes solutions (combinaisons des valeurs des hyperparamètres) à tester dans le cas où nous avons 2 hyperparamètres avec 6 valeurs possibles pour le premier et 7 valeurs possibles pour le second. La nombre de combinaisons est donc de 42 (6 x 7). De plus, selon la littérature [128], l'espace de solutions des hyperparamètres de l'algorithme d'apprentissage automatique peut inclure des domaines avec des valeurs réelles ou illimitées, ce qui néssécite de spécifier des limites pour appliquer une optimisation à partir d'une grille. Par ailleurs, afin d'évaluer une solution d'hyperparamètres, il faut définir un score à utiliser. Les fonctions d'évaluation généralement utilisées peuvent être l'*accuracy*, l'aire sous la courbe ROC, le rappel, etc.

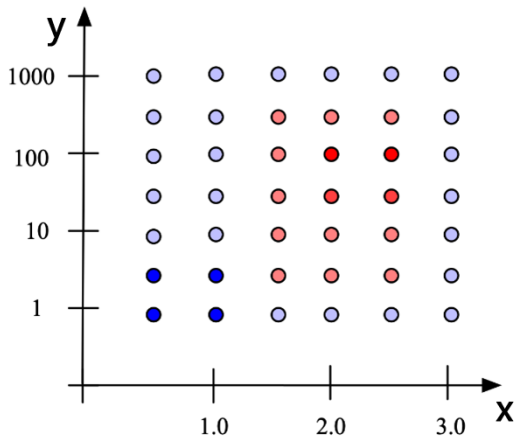

Figure 11: Une illustration d'un espace de recherche de grille.

Nous définissons manuellement une gamme de paramètres possibles et l'algorithme effectue une recherche complète sur eux. En d'autres termes, l'algorithme *grid search* est une force brute

3.4.3 RANDOM SEARCH

Random Search peut aussi être utilisé avec la validation croisé en k plis. Cet algorithme sélectionne au hasard les hyperparamètres afin d'évaluer moins de solutions de l'espace de recherche. *Random Search* est particulièrement pratique lorsque le nombre de solutions est significatif et lorsque des paramètres sont continus ou continus et discrets [128].

Bien que cet algorithme soient reconnus comme efficace pour régler les hyperparamètres pour un modèle d'apprentissage automatique, plusieurs travaux affirment qu'il devient significativement moins efficace lorsque l'espace de recherche devient grand [128, 129]. La figure 12 illustre le principe de *Random Search*. Pour pallier le problème des grands espaces de recherche et l'efficacité en terme de temps, plusieurs travaux exploitent les métaheuristiques [128].

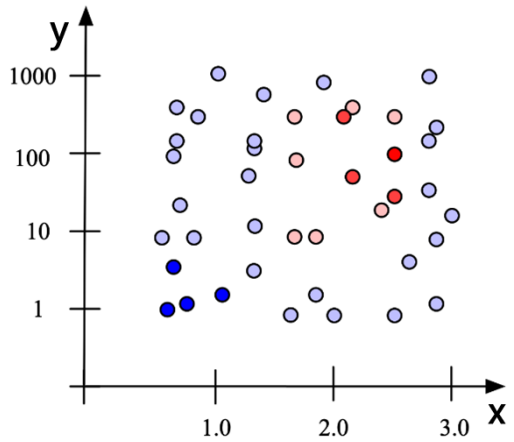

Figure 12: Une illustration d'un espace de recherche aléatoire.

Nous définissons manuellement une plage de limites des paramètres possibles et l'algorithme effectue une recherche sur eux pour le nombre d'itérations que nous avons défini.

Après avoir décrit les deux algorithmes d'optimisation, nous retenons que pour avoir de meilleures performances, il faut explorer le maximum de l'espace de recherche. Ceci implique d'avoir un nombre très important de combinaisons possibles (la bonne méthode de sélection de caractéristiques, le bon ensemble caractéristique, le bon algorithme de classification, etc.). La manière naïve est de crée toutes les combinaisons possibles (brut force). Elle est réaliste et efficace pour un petit jeu de donnée (avec une faible dimensionnalité) mais non efficace pour de gros jeu de données. Par exemple si nous prenons le temps mis pour explorer une espace de recherche de 243 combinaisons (5 paramètres avec chacun 3 valeurs, ce qui donne $3^5 = 243$) sur un jeu de donnée de 3 millions de lignes alors nous supposons que cela prendre *X heures*. Ce temps mis s'augmente *de X+Y heures* si on ajoute de l'information (lignes, des colonnes, des paramètres, des valeurs des paramètres) aux problèmes. Pour résoudre ce problème plusieurs recherches ont proposées les algorithmes d'optimisations métaheuristique. Ce sont des algorithmes qui permettent de réduire le temps de traitement tout en explorant un très grand espace de recherche. Le problème est qu'ils ne

garantissent pas l'obtention de l'optimum global. Néanmoins ils sont jusqu'à date la solution proposée. La prochaine section décrit ces algorithmes d'optimisation.

3.5 OPTIMISATION PAR DES METAHEURISTIQUES

En science des données, le temps d'entraînement des modèles d'apprentissage automatique est un enjeu primordial. De plus, lorsque nous cherchons à déterminer l'ensemble des meilleurs hyperparamètres, le nombre d'entraînement peut rapidement croître. Plusieurs travaux expliquent comment les métaheuristiques peuvent diminuer le nombre d'entraînement en laissant de côté les modèles inutiles. En effet, un métaheuristique est une méthode générique pour la résolution de problèmes combinatoires NP-difficiles [130, 131]. C'est une approche intelligente capable de contourner l'explosion combinatoire et résout ces problèmes difficiles en un temps raisonnable [132]. Pour contourner le problème combinatoire, les métaheuristiques explorent volontairement qu'une partie de l'espace des solutions. L'inconvénient est qu'elles ne trouvent pas forcément la solution optimale.

Il existe deux grandes familles de métaheuristiques : les approches perturbatrices et constructive [130]. Les approches perturbatrices construisent des combinaisons en modifiants des combinaisons existantes, tandis que les approches constructives génèrent des combinaisons de façon incrémentale en utilisant un modèle stochastique. Une des particularités des deux grandes familles de métaheuristique est qu'elles peuvent être hybrides. Les approches perturbatrices les plus connus sont les algorithmes génétiques et la recherche locale, tandis que les approches constructives les plus connues sont les stratégies gloutonnes aléatoires, les algorithmes par estimation de distribution et l'optimisation par colonies de fourmis.

Dans le cadre de notre projet de recherche et pour l'optimisation de notre modèle de classification, nous utilisons les algorithmes suivants : l'algorithme génétique, *Grey Wolf Optimizer* et *Differential Evolution*. La figure 13 regroupe les métaheuristiques en quatre

grandes familles : *Evolutionary, Physics-based, Swarm Intelligence based* et *humans-social-behaviour and idiology based.*

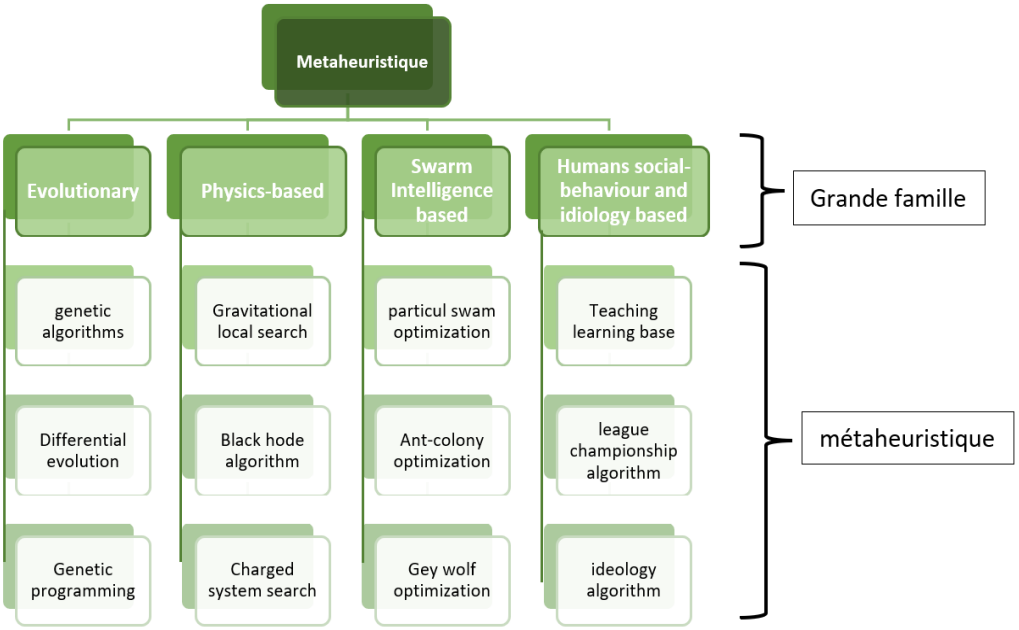

Figure 13: Classements des métaheuristiques en famille

3.5.1 PRINCIPES DE BASE

Les algorithmes d'optimisation métaheuristique repose sur l'exploitation d'une fonction objectif (*fitness*). En général, un algorithme d'optimisation peut s'exprimer comme suit [133]:

$$\text{Optimize, } f_1(x), \ldots, f_i(x), \ldots, f_N(x),\ x = (x\_1, \ldots, x\_d)$$
$$\text{subject to, } hj(x) = 0,\ (j = 1, 2, \ldots, J)$$
$$g_k(x) <= 0,\ (k=1, \ldots, K)$$

où $f_1, \ldots, f_N$ sont les objectifs, tandis que h_j et g_k sont respectivement les contraintes d'égalité et d'inégalité. Dans le cas où $N = 1$, on parle d'optimisation mono-objectif. Notre projet se concentre sur un problème d'optimisation mono-objectif (chercher le meilleur modèle de classification). Pour résoudre le problème d'optimisation mono-objectif, il sera important de définir une fonction objective, le problème et une liste des contraintes à respecter avant de passer à la recherche de solution.

PREPARATION A LA FONCTION OBJECTIVE

La fonction objective prend une solution (un vecteur de valeurs) et retourne un score (une valeur). Chaque solution sera évaluée et aura un score. Dans notre étude le score est *l'accuracy* du modèle évalué. Dans notre problème nous définissons des fonctions qui pénalisent certaines solutions. Le but des fonctions de pénalisation est de donner des mauvais scores à des solutions pour que ces derniers ne soient pas sélectionnés pour passer à la prochaine génération. Dans notre étude les fonctions de contraintes et de pénalisation vérifient la solution et les résultats du modèle évaluer.

Dans notre étude si aucune règle n'est violée alors le score de la solution est l'accuracy du modèle. Si au moins une règle est violée alors le score de 0 est donner à la solution. La valeur de punition dans notre cas est 0 car nous sommes dans un problème de maximisation. Le but est de trouvé *l'accuracy* le plut élever par conséquent la valeur la plus petite est 0. Ce score de pénalisation peut prendre n'importe quelle valeur et est défini en fonction du problème (par exemple le score peut avoir la valeur 1 si le problème est de minimiser la fonction de perte ou *Loss Function*). Les fonctions de contraintes et de pénalisation que nous utilisons sont :

- Une solution à le nombre d'élément nécessaire pour paramétrer la fonction objective
- Vérifier que chaque élément de la solution équivaut à un choix valable (Par exemple le premier élément de la solution codifier le *classifier* à utiliser et le deuxième le *selector* à utiliser).
- Vérifier que la matrice de confusion respect la règle mise en place au point 2.3.4
- Une solution peut contenir qu'un algorithme de sélection et un algorithme de classification
- Seulement les indices représentant les paramètres de l'algorithme de sélection ou de classification sont prisent en compte lors de l'évaluation du modèle.

PREPARATION DU PROBLEME

La préparation du problème permet de définir la liste des variables, les limites de chaque variable et le type de problème à résoudre (minimisation ou maximisation). Par exemple, dans un algorithme génétique, nous pouvons avoir un chromosome (une solution) constitué de cinq allèles, pouvant chacun seulement prendre les valeurs 0 ou 1. Les allèles n'ont pas nécessairement besoin d'être binaires, mais peuvent prendre des valeurs continues.

Il est aussi important de définir une ou plusieurs conditions d'arrêt. La condition d'arrêt par défaut est le nombre d'époques (générations, itérations, …). Néanmoins, il existe d'autres conditions d'arrêt en fonction du problème à optimiser. Nous avons par exemple le nombre de solution à évaluer, le temps de recherche, etc. Dans ce projet de recherche, nous avons opté pour le nombre d'itérations comme condition d'arrêt du processus d'optimisation.

SOLUTION ET FONCTION DE GENERATION

A chaque itération de l'algorithme métaheuristique, une ou plusieurs solutions sont utilisées par la fonction objectif afin d'assigner un score à chaque solution. Le ou les scores sont conservés et permettent d'ajuster les solutions de la prochaine itération de l'algorithme d'optimisation. En ce qui concerne la solution, un exemple est présenté à la figure 14.

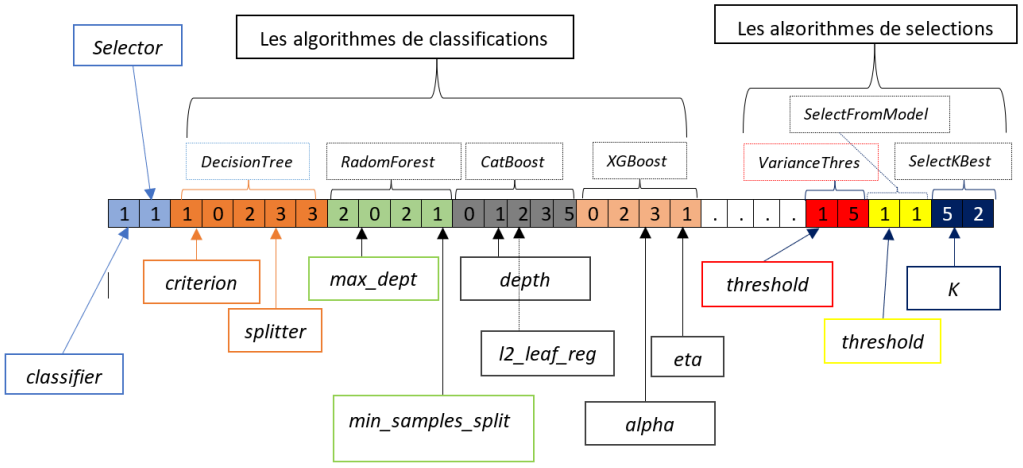

Figure 14: Un exemple de solution utilisée par une fonction objectif

La solution est représentée sous forme de matrice unidimensionnelle. Cette matrice peut avoir plusieurs noms en fonction de l'algorithme métaheuristique implémenté (La matrice est appelée chromosome dans le contexte de l'algorithme génétique et vecteur pour l'algorithme évolutionnaire). Il peut être nécessaire de définir dans certain cas la fonction qui génère les solutions. Cela évite à avoir plusieurs fonctions de contraintes et minimiser le temps de calcul. L'inconvénient est que cela réduit les chances de tombé sur quelques choses qui n'a jamais été prévu.

EXPLICATION DE LA MATRICE

Codage du choix du *classifier*, C0 = Colonne 1 du chromosome

Le premier élément de la matrice C0 est chargé de coder le classifier utiliser. Par exemple s'il existe 04 classifier comme *DecisionTreeClassifier, RandomForestClassifier, CatBoostClassifier* et *XGBClassifier*. Le pipeline se basera sur l'ordre d'énumération des *classifiers* pour décoder le chromosome. Le premier élément de la matrice peut prendre les valeurs entre 0,1,2,3 et 4. Chaque numéro correspond à un classifier. Ainsi l'équivalence est illustrée par la figure 15.

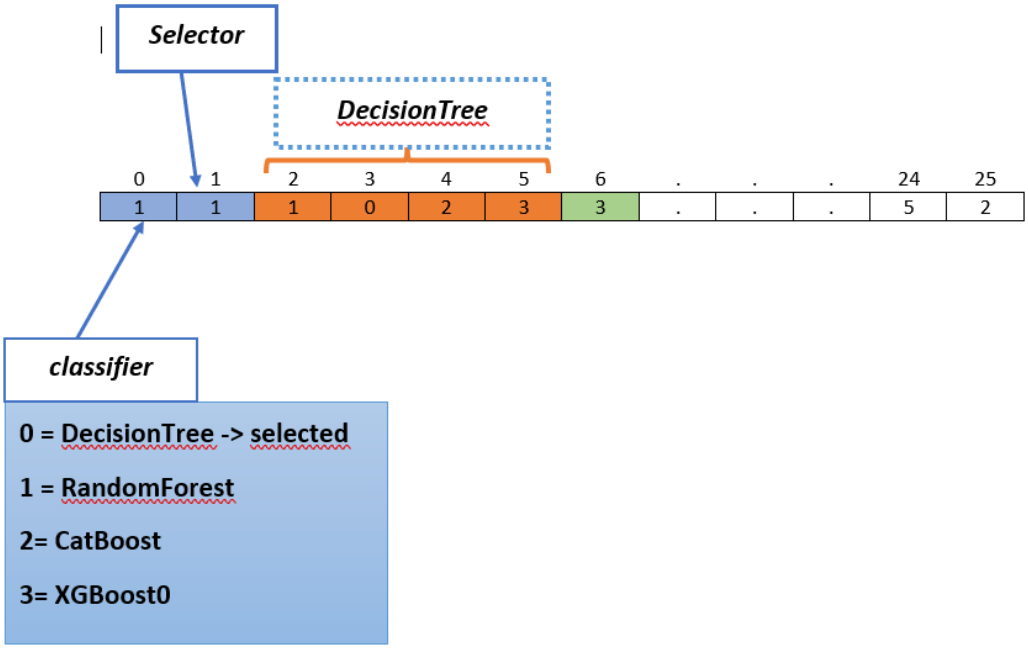

Figure 15: Matrice solution (Coder le classifier)

- 0 = DecisionTreeClassifier
- 1 = RandomForestClassifier
- 2 = CatBoostClassifier
- 3 = XGBoostClassifier

Une fonction de contrainte permet de vérifier que la valeur prise par le premier élément de la matrice ne dépasse pas le nombre de *classifier* déclarer.

Codage du choix du Selector, C1 = Colonne 2 du chromosome,

Le deuxième élément de la matrice code le *Selector* (Algorithme de sélection de caractéristique). Le pipeline décode le chromosome selon l'ordre d'énumération des *selectors*. Le deuxième élément de la matrice peut prendre les valeurs entre 0,1,2 et 3 si trois *selectors* sont déclarer (VarianceThreshold, SelectFromModel et SelectKBest). Dans le cas des selector, le choix None est réservé pour prendre en compte les cas où les *selectors* ne peuvent pas être utilisés. Aussi, ça permettra d'évaluer aussi le cas où toutes les caractéristiques sont

conservées. Chaque numéro correspond à un *selector*. Ainsi l'équivalence est illustrée par la figure 16.

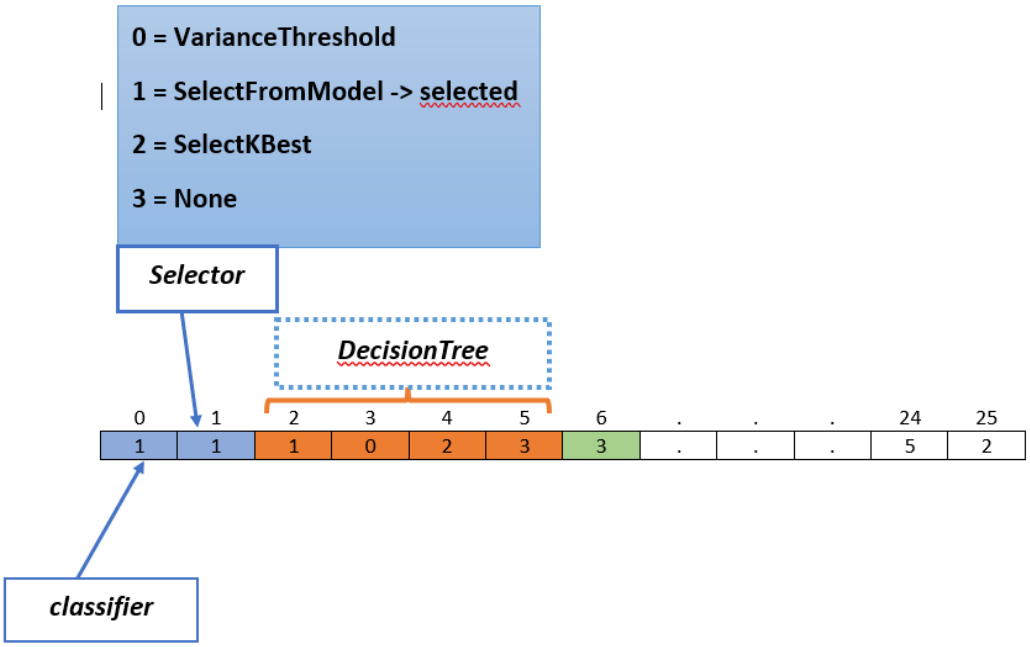

Figure 16: Matrice solution (Coder le Selector)

- 0 = VarianceThreshold
- 1 = SelectFromModel
- 2 = SelectKBest
- 3 = None (Toutes les caractéristiques sont utilisés pour l'apprentissage)

Une fonction de contrainte permet de vérifier que la valeur prise par le deuxième élément de la matrice ne dépasse pas le nombre de méthode de sélection déclarés. Dans l'exemple, le gène prend la valeur 1. Cela se traduit que la méthode de sélection utiliser est *SelectFromModel*.

Codage d'un *classifier*, Exemple de DecisionTree (C2 – C5)

Dans la matrice (chromosome) illustré dans la figure 17, les quatre 4 colonnes (gènes) C2, C3, C4 et C5 servent à coder les paramètres du *classifier DecisionTree*. Ces 04 colonnes

codent respectivement les paramètres splitter, criterion, max_depth et max_features. La colonne C2 de la matrice permet de coder le paramètre splitter de l'algorithme DecisionTree. La colonne peut prendre les valeurs 0 et 1. La valeur 0 code la valeur *best* et 1 la valeur *random*.

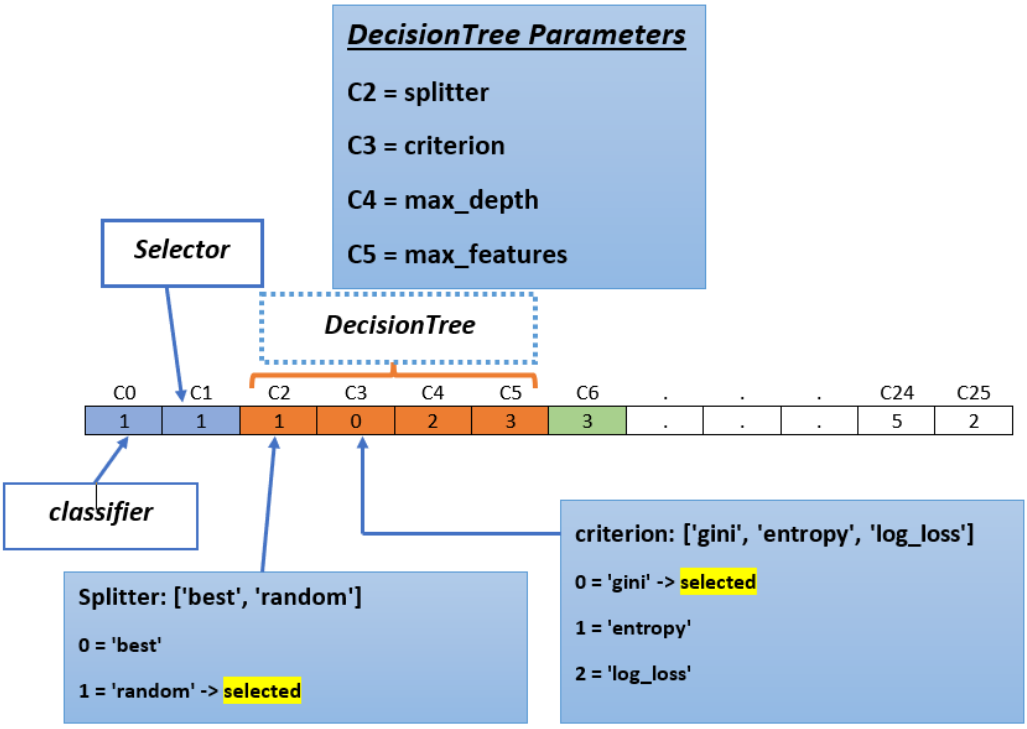

Figure 17: Matrice solution (Coder les paramètres d'un classifier, exemple de DecisionTree)

- C2 = splitter
  - C2 = 0 égale à C2 = best
  - C2 = 1 égale à C2= random
- C3 = criterion
  - C3 = 0 égale à C2 = gini
  - C3 = 1 égale à C2= entropy
  - C3 = 2 égales à C2= log_loss
- C4 = max_depth
- C5= max_features

Les colonnes ne peuvent prendre qu'une seule valeur. Par exemple la colonne splitter ne peut prendre que best ou random (C2= 0 ou C2=1). L'utilisateur peut modifier les paramètres utiliser pour chaque algorithme de classification. L'exemple illustre pour le classifier DecisionTree, mais les colonnes à partir du C6 servent à coder les paramètres d'autres algorithmes comme RandomForest, CatBoost ou XGBoost. Chacun de ses algorithmes utilise une colonne pour coder une de ses paramètres. Par exemple, C6, C7, C8, C9 sont utilisés par RandomForest pour coder ses paramètres criterion, n_estimators, max_depth et max_features. CatBoost utilise les colonnes C10, C11, C12 et C13 pour coder ses paramètres criterion, n_estimators, max_depth et max_features. Quant à XGBoost, il utilise les colonnes C14, C15, C16, C17 et C18 pour coder les paramètres iterations, learning_rate, depth et l2_leaf_reg. Les autres colonnes serviront à coder les autres algorithmes comme XGBoost et autres si l'utilisateur souhaite utilisés plus de 04 classifiers.

Codage d'un Selector, Exemple de SelectKBest (C24 – C25)

Dans la matrice (chromosome) illustrée à titre d'exemple dans la figure 18, les colonnes C24 et C25 servent à coder les paramètres du Selector 'SelectKBest'. Ces 02 colonnes code respectivement les paramètres k et score_func. La colonne C24 de la matrice permet de coder le paramètre k de l'algorithme de sélection de caractéristique SelectKBest. La colonne C24 peut prendre les valeurs 0, 1, 2, 3, 4 et 5. Ces valeurs représente pour le Selector les valeur 8, 9, 20, 15 et 5 respectivement. Ainsi dans l'exemple la valeur de k= 5 pour le Selector SelectKBest. La colonne 25 ne sera pas présenter car respecte la même règle, c'est-à-dire que le paramètre score_func à la valeur 2 pour cet exemple. La valeur 2 est l'indice d'un élément lister dans les valeurs possibles de score_func.

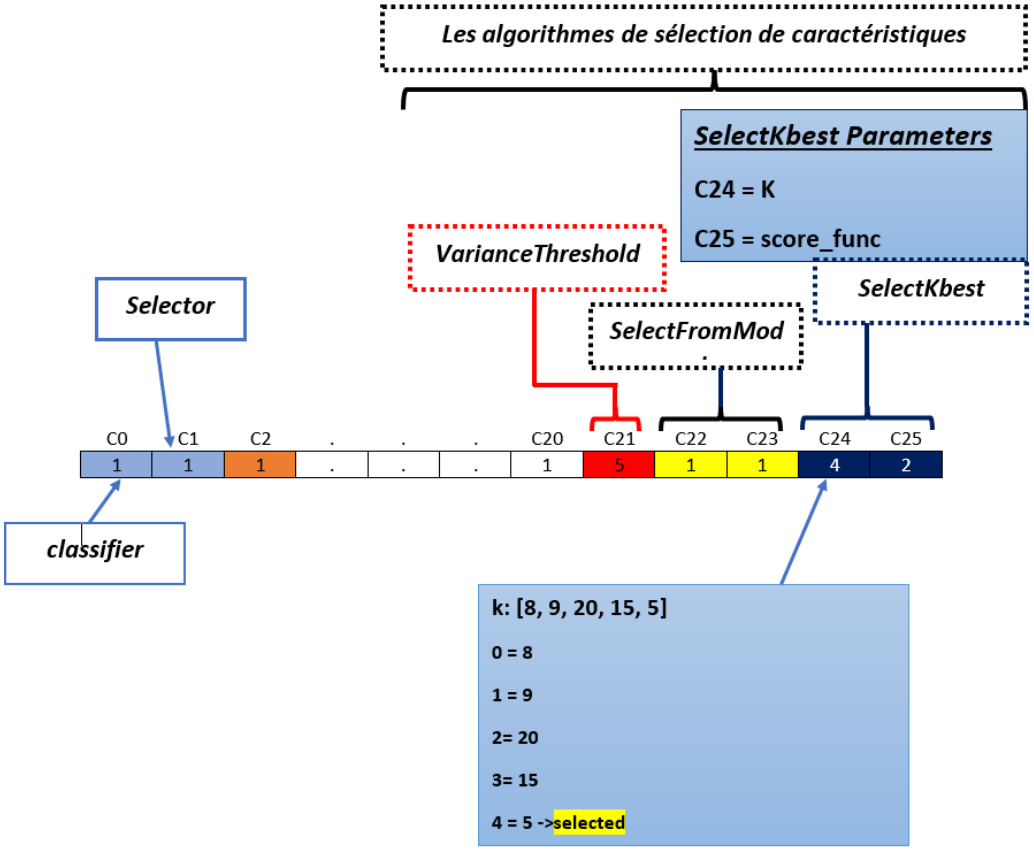

Figure 18: Matrice solution (Coder les paramètres d'un Selector, exemple de SelectKBest)

Seule la colonne est présentée, mais ici les colonnes C21 à C25 code les paramètres des algorithmes de sélection de caractéristiques. Chaque colonne appartient à un algorithme de sélection et dans l'exemple 3 algorithmes sont codés : *VarianceThreshold, SelectFromModel et SelectKBest*. Le premier à un seul paramètre tandis que les deux autres ont chacun deux paramètres. Néanmoins l'algorithme de sélection de caractéristique utilisera pour la solution le *SelectFromModel (confère figure 8)*. En effet l'indice de la colonne C1 est 1 tandis que 1 est l'indice de *SelectFromModel* dans l'exemple.

3.5.2 ALGORITHMES GENETIQUES

L'algorithme génétique est un algorithme de recherche évolutive utilisé pour résoudre des problèmes d'optimisation et de modélisation en sélectionnant, combinant et faisant varier séquentiellement des paramètres à l'aide de mécanismes qui ressemblent à l'évolution biologique [128] : croisements, mutations, sélection, etc. (confère figure 19).

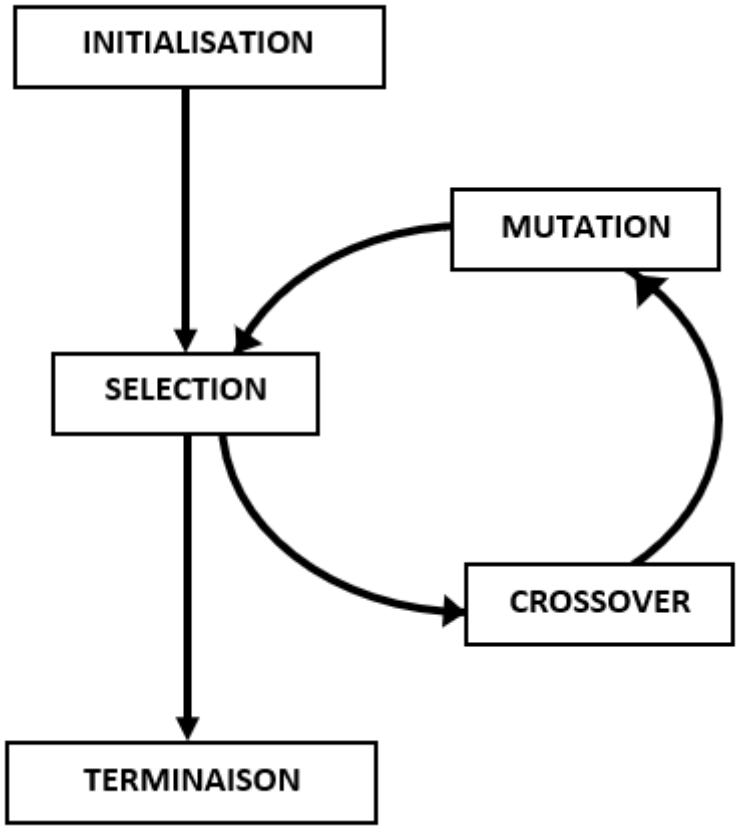

Figure 19: Processus de l'évolution biologique imité par l'algorithme génétique

PRINCIPES GENERAUX

Les algorithmes génétiques imitent le processus de sélection naturelle, c'est-à-dire que les espèces capables de s'adapter aux changements de leur environnement peuvent survivre, se reproduire et passer à la génération suivante. Chaque génération est constituée d'une population d'individus et chaque individu représente un point dans l'espace de recherche et

une solution possible. Chaque individu est représenté par une chaîne de gènes (caractère, entier, flottant ou bits). Cette chaîne est comparable au chromosome. L'algorithme génétique commence par une population de chromosomes générée aléatoirement. Ensuite, il procède à un processus de sélection et de recombinaison, basé sur le *fitness* de chaque chromosome (score). Le matériel génétique parent est recombiné pour générer des chromosomes enfants produisant la génération suivante. Ce processus est itéré jusqu'à ce qu'un certain critère d'arrêt soit atteint [129].

Pour utiliser l'algorithme génétique, cinq éléments sont nécessaires : un principe de codage de l'élément de population, un mécanisme de génération de la population initiale, une fonction à optimiser, des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état et enfin des paramètres de dimensionnement(taille de la population, nombre total de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation). Le principe général du fonctionnement d'un algorithme évolutionnaire est représenté est illustré par [134] sur la figure 20 .

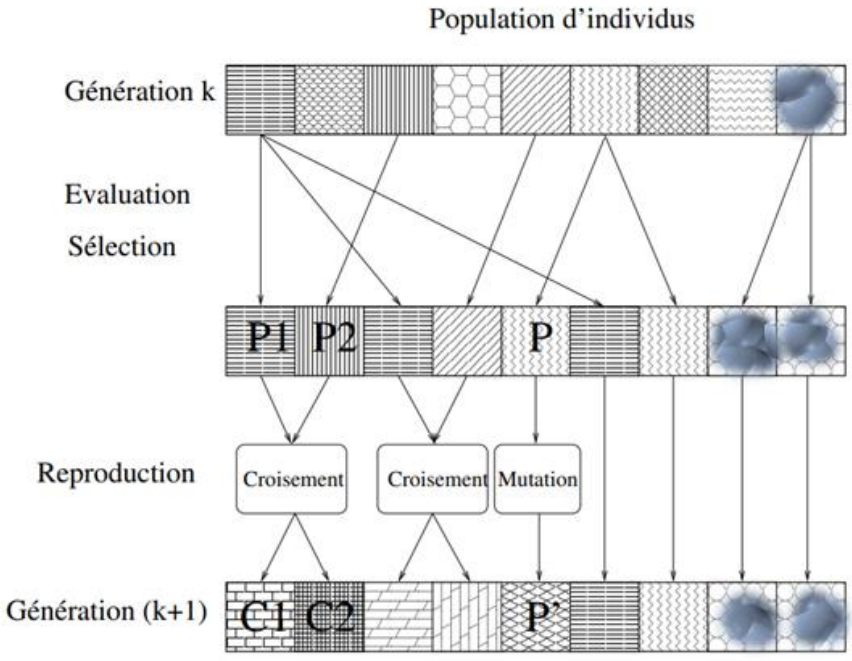

Figure 20: Principe général des algorithmes génétiques

Selon le principe, la première étape consiste à engendrer une population d'individus de façon aléatoire. Tous les individus d'une population *k* subissent trois opérations (Evaluation, Sélection et reproduction) pour passer d'une génération *k* à la génération suivante *k+1*. Les deux parents P_1 et P_2 sont sélectionnées et l'opération de croisement leur est appliqué avec une probabilité P_c. Ce croisement donne lieu à deux enfants C_1 et C_2. En fonction de leur adaptation d'autres éléments *P* sont sélectionnés. L'opération de mutation est appliquée aux P éléments avec la probabilité P_m (P_m inferieur à P_C généralement) pour engendrer des individus mutés *P'*. Les enfants C_1 et C_2 et les individus mutés *P'* sont ensuite évalués avant d'être insérer dans la nouvelle population. Pour terminer, un critère d'arrêt met fin à l'algorithme. Les critères les plus utilisés pour arrêter sont : définir un nombre de générations que l'on souhaite exécuter, arrêter l'algorithme lorsque la population n'évolue plus rapidement.

DESCRIPTION DETAILLEE

Codage des données est la manière dont un élément de l'espace de recherche est représenté. Les algorithmes génétiques utilisaient dans le passé une chaine de bits pour contenir toute l'information nécessaire à la description d'une solution au problème. Cette manière de coder l'information a pour avantage de créer des opérateurs de croisements et de mutation simples [135]. Ce type de codage était utilisé pour avoir les premiers résultats de convergence théorique. Cependant, ce type de codage n'est pas forcément le meilleur. Le premier problème est que deux éléments voisins en termes de distance de *Hamming* ne codent pas nécessairement deux éléments proches dans l'espace de recherche. Les chercheurs pour éviter ce problème utilise plutôt un codage de *Gray*. Le deuxième problème peut survenir quand l'espace de recherche a une grande dimension. Dans ce cas, chaque variable est représentée par une partie de la chaîne de bits et la structure du problème n'est pas bien reflétée, l'ordre des variables ayant une importance dans la structure du chromosome, alors qu'il n'en a pas forcément dans la structure du problème. La techniques proposé selon [136] est d'utiliser des vecteurs réels. Les vecteurs réels conservent les variables du problème dans le codage de l'élément de population. Ces techniques s'appellent RCGA (*Real Coded Genetic Algorithms*) et son but est de conservée la structure du problème dans le codage.

La génération aléatoire de la population initiale est une étape très importante, car conditionne fortement la rapidité de l'algorithme. Généralement la position de l'optimum dans l'espace de recherche est inconnue, alors il est important de s'assurer que les individus générer aléatoirement respectent les contraintes[137]. Aussi, il faut s'assurer que les tirages soient uniformes dans chacun des domaines associés aux composantes de l'espace d'état lors de la génération des individus. Dans le cas où les informations sont disponibles sur problème, alors cela peut servir à engendrer des individus dans le but d'accélérer la convergence. Dans certains cas la gestion des contraintes ne peut se faire lors de la génération de la population, alors elles sont inclues dans le critère à optimiser sous forme de pénalités.

La gestion des contraintes permet d'éliminer des individus. Dans le cas où un individu de la population ne respecte pas une contrainte, il se verra attribuer un mauvais fitness. Un individu a une forte probabilité d'être éliminé s'il a un mauvais fitness. Il est souvent conseiller de garder les éléments avec un mauvais fitness tout en les pénalisant car ils peuvent permettre de générer des individus de bonne qualité. Quand au moins une contrainte de séparation est saturée, cela peut se traduire pour certains problèmes l'atteinte de l'optimum. Quand les contraintes ne sont pas bien gérées, le processus se retrouve à rechercher des solutions admissibles au détriment de la recherche de l'optimum ou inversement. Le rôle des opérations de croisement et de mutation est de d'assurer que les individus ne soient pas homogènes entre les générations. Il doit permettre d'explorer le maximum de possibilité.

L'opérateur de croisement a pour but de diversifié la population en modifiant la structure des chromosomes. Généralement deux parents sont croisés pour donner naissance à deux enfants. Auparavant le croisement à découpage de chromosomes (*slicing crossover*) était utilisé par le codage par chaînes de bits. Pour effectuer ce type de croisement sur des chromosomes constitués de M gènes, on choisit aléatoirement une position dans chacun des parents. On remplace ensuite les deux sous-chaînes terminales de chacun des deux chromosomes, ce qui produit deux enfants C_1 et C_2. Le découpage peut se faire aussi en plusieurs points selon [138]. Ce type de croisement est très utilisé pour les problèmes discrets.

Il y'a aussi le croisement barycentrique qui marche plus avec des problèmes continus. Le croisement barycentrique peut être formalisé par la sélection de deux gènes P1(i) et P2(i) dans chacun des parents à la même position *i*. Ils définissent deux nouveaux gènes $C_1(i)$ et $C_2(i)$ par une combinaison linéaire :

$$\begin{cases} C_1(i) = \alpha P_1(i) + (1-\alpha)P_2(i) \\ C_2(i) = (1-\alpha)P_1(i) + \alpha P_2(i) \end{cases}$$

Dans l'équation, α est un coefficient de pondération aléatoire et peut prendre par exemples des valeurs compris entre [-0.5, 1.5]. Cela engendre des points souvent à l'extérieur ou l'intérieur des deux gènes sélectionnés.

Selon [135], « L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours d'espace ». Le but de la propriété est de s'assurer que l'algorithme est susceptible d'atteindre tous points de l'espace d'état, sans les parcourir tous. Cette propriété permet à l'algorithme génétique de converger sans forcément passer par des croisements. Pour un problème discret, la mutation consiste à choisir par hasard un gène dans le chromosome et à le remplacer par une valeur aléatoire. Pour un problème continu, on choisit aléatoirement un gène dans le chromosome, et on l'ajoute un bruit généralement gaussien. Le problème majeur est que l'écart-type de ce bruit est difficile à choisir en amont.

La sélection consiste à choisir les individus les plus importants d'une population et d'éliminer les moins importants. Il existe plusieurs principes de sélection mais les plus utilisés sont : *Roulette wheel selection* et *stochastic remainder without replacement selection [139].* Le premier principe consiste à associer à chaque individu de la population un segment dont la longueur est relative à sa *fitness*. C'est le principe qu'utilise les roulettes de casinos avec une structure linéaire. Ces segments sont par la suite connectés sur un axe normalisé entre 0 et 1. Un nombre au hasard compris entre [0,1] est sélectionner et on regarde quel est le segment correspondant. Cette manière permet de choisir le plus souvent les meilleurs individus. La

dimension de la population peut introduire une subtilité de sélection. Par conséquent si la dimension est réduite, il est compliqué d'utiliser en pratique l'espérance mathématique de sélection en raison du peu de tirages effectués. Quant au deuxième principe qui est la *stochastic remainder without replacement selection,* elle évite ces problèmes et fourni de bons résultats. Ce principe est :

- Pour chaque élément *i*, on calcule le rapport r_i de sa fitness sur la moyenne des fitness.
- Soit $e(r_i)$ la partie entière de *ri*, chaque élément est reproduit exactement $e(ri)$ fois.
- La roulette wheel selection précédemment décrite est appliquée sur les individus affectés des fitness $r_i - e(ri)$.

Selon la littérature, ce principe de sélection est meilleur pour des populations de petite taille. Il y'a aussi des améliorations pour ce qui concerne les processus de sélection.

En effet le processus de sélection rencontre souvent des problèmes d'homogénéité des individus. Cela se produit quand la population est constituée d'un seul type d'individu. L'origine de ce problème est que les processus présentés en amont son très sensibles aux écarts de fitness. D'autres méthodes de sélection comme le *ranking*, le *scaling* et le *sharing* permet d'éviter ce comportement. Ils empêchent l'élimination totale des individus faibles. Enfin, des notions de dominance lors de la sélection peuvent être introduite pour prendre en compte des recherches à plusieurs objectifs.

3.5.3 GREY WOLF OPTIMIZATION

L'algorithme *Grey Wolf optimization* (GWO) est un algorithme d'optimisation de la famille des métaheuristiques. L'algorithme repose sur une population qui simule la hiérarchie de *leadership* et le mécanisme de chasse des loups gris dans la nature [133]. Cet algorithme a déjà été utilisé pour résoudre des problèmes d'extraction de caractéristiques [140] ou encore d'initialisation des poids dans les réseaux de neurones convolutif (CNN) [141]. Par ailleurs, l'algorithme *Grey Wolf optimization* a aussi été utilisé pour optimiser des hyperparamètres

d'algorithmes d'apprentissage automatique et possède des performances semblables ou supérieures aux algorithmes comme PSO, ACO, GA [133, 142].

PRINCIPES GENERAUX

Les loups gris sont des animaux qui vivent en groupe d'une moyenne de 5 à 12 loups. Leur force est qu'ils ont une hiérarchie au sein de ce groupe. De ce fait, chaque loup joue un rôle précis. Le meilleur pour la gestion de la meute est l'alpha. Son principal rôle est de prendre des décisions par rapport à la chasse. Il est le dominant et ses ordres sont suivis par la meute. Il n'est pas forcément le plus fort. Le deuxième au niveau de la hiérarchie est le bêta. Il aide l'alpha dans des activités de meute comme la prise de décision. Le loup bêta est le meilleur candidat pour remplacer un alpha en cas de besoin. Il commande aussi les autres loups qui ne sont pas à son niveau. En d'autres termes, le loup bêta aide l'alpha à imposer l'ordre dans la meute. Les loups delta composent le troisième niveau dans la hiérarchie. Ils se soumettent à l'alpha et au bêta. Ils jouent le rôle d'éclaireurs, de chasseurs, de sentinelles, de gardiens, etc. Le dernier et quatrième niveau dans la hiérarchie est représenté par les omégas. Ils se soumettent aux autres loups et ne mangent que les restes. La figue 21 illustre la hiérarchie des loups gris.

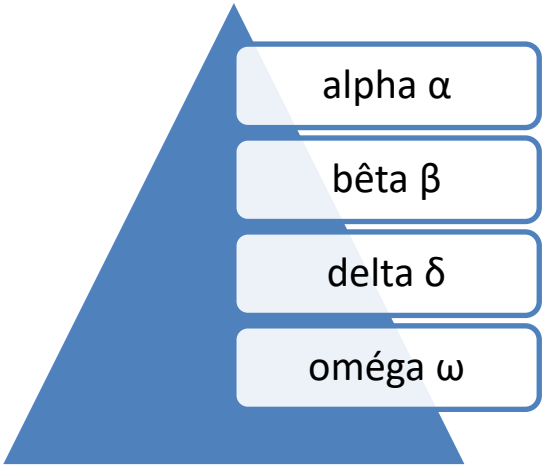

Figure 21: : Hiérarchie sociale des loups gris

En ce qui concerne la chasse, les loups gris ont une stratégie bien précise, constituée de plusieurs phases [143] . Les phases sont :

- traquer, chasser et approche la proie,
- poursuivre, encercler et harceler la proie jusqu'à ce qu'elle cesse de bouger,
- attaque la proie.

DESCRIPTION DETAILLEE

Mathématiquement, le loup alpha (α) est la solution la plus adaptée à un problème d'optimisation. La deuxième meilleure solution est représentée par le loup bêta (β). La troisième meilleure solution est le loup delta (δ) et les autres solutions sont considérées comme des loups omégas (ω).

Encercler une proie peut se formaliser par les équations (2.1) et (2.2). Plus concrètement, le principe est qu'un loup gris à la localisation (X, Y) peut mettre à jour sa position en fonction de la localisation de sa proie (X^*, Y^*).

Pour cela, l'algorithme va ajuster les vecteurs $\vec{A}$ et $\vec{C}$ afin que le loup gris s'approche de la proie en se plaçant dans de nouvelles localisations (des cachettes). Les vecteurs $\vec{r}_1$ et $\vec{r}_2$ permettent au loup gris d'atteindre n'importe quelle localisation autour de la proie.

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \quad (2.1)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \quad (2.2)$$

$$\vec{A} = 2\vec{a} \cdot \vec{rand}_1 - \vec{a} \quad (2.3)$$

$$\vec{C} = 2 \cdot \vec{rand}_2 \quad (2.4)$$

où t indique l'itération actuelle, $\vec{A}$ et $\vec{C}$ sont des vecteurs de coefficients, $\vec{X}_p$ est le vecteur de position de la proie, $\vec{X}$ indique le vecteur de position d'un loup gris, $\vec{a}$ avec ses composants décroissants linéairement de 2 à 0 au cours des itérations, $\overrightarrow{rand}_1$ et $\overrightarrow{rand}_2$ sont des vecteurs aléatoires dans [0, 1].

La chasse : Le procès consiste à localiser la proie et l'encerclé. Il est dirigé par le loup alpha. Le alpha donne les ordres tandis que la bêta et le delta mettent à jour leurs positions en fonction des positions de alpha. Il est supposé que les trois loups α, β et δ ont une meilleure localisation de la proie. Cependant, leurs positions permettent de mettre à jours la position des autres loups. Les équations de la chasse sont représentées de (2.5) à (2.11).

$$\vec{D}_\alpha = \left|\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}\right|, \qquad (2.5)$$

$$\vec{D}_\beta = \left|\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}\right|, \qquad (2.6)$$

$$\vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}| \qquad (2.7)$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \left(\vec{D}_\alpha\right), \qquad (2.8)$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \left(\vec{D}_\beta\right), \qquad (2.9)$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot (\vec{D}_\delta), \qquad (2.10)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \qquad (2.11)$$

Attaquer des proies (exploitation) : La proie est par la suite attaquée après avoir été piégés (encerclé). L'approche de la proie est formalisée par une diminution de la valeur de $\vec{a}$. En ce moment la plage de fluctuation de $\vec{A}$ (une valeur aléatoire ente [*-2a, 2a*]) est diminué de $\vec{a}$. Au cours de chaque itération, la valeur *a* est décroissant de 2 à 0. Les loups attaquent quand au moins une valeur aléatoire de $\vec{A}$ est compris entre [-1, 1] d'où |*A*| <1, c'est le processus d'exploitation.

Recherche de proie (exploration) : Les loups en revanche s'éloigne d'une proie si |A|>1 pour espérer trouver une proie plus proche, c'est le processus d'exploration. Une valeur aléatoire $\vec{C}$ permet de rendre leurs comportements plus aléatoires pendant le processus d'exploration. La valeur aléatoire est comprise entre [0, 2] et C>1 accentue l'attaque tandis que C<1 minimise l'attaque. Cette vecteur C essaie de représenter les obstacles qui peuvent surgir durant l'approche de la proie. Son rôle est comme dans la réalité empêcher les loups d'approcher rapidement la proie dans des cas de figure. Il peut aussi permettre au loup de s'approcher plus vite de la proie.

Voici ci-dessous le pseudocode de l'algorithme GWO [133]:

- **Etape 1** : Initialiser aléatoirement la population de loups gris X_i avec $i = 1, 2, \dots, n$
- **Étape 2** : Initialiser la valeur de a=2, A et C (en utilisant l'équation 2.3, 2.4)
- **Étape 3** : Calculer la fitness de chaque membre de la population
- X_α =membre avec la meilleure valeur de fitness
- X_β = deuxième meilleur membre (en termes de valeur de fitness)
- X_δ = troisième meilleur membre (en termes de valeur de fitness)
- **Étape 4 :** POUR *t* = 1 à *Max_number_of_iterations* :
- Mettre à jour la position de tous les loups oméga par équations : 2.5 ,2.6 et 2.7
- Mettre à jour a, A, C (en utilisant l'équations : 2.3 et 2.4)
- *a = 2(1-t/T)*
- Calculer l'aptitude de tous les agents de recherche
- Mettre à jour X_α, X_β, et X_δ.
- FIN POUR
- **Étape 5** : retourner X α

3.5.4 DIFFERENTIAL EVOLUTION

L'algorithme *Differential Evolution* (DE) est un algorithme métaheuristique conçu pour résoudre des problèmes d'optimisation continus et sans contraintes [144]. Depuis, il a été utilisé pour résoudre des problèmes à variables mixtes et en présence de contraintes non linéaires. Il est à noter que cet algorithme s'inspire fortement des algorithmes génétiques et des stratégies évolutives, il n'est donc pas étonnant que l'algorithme *Differential Evolution* repose sur une méthode de recherche stochastique et une population. De plus, l'algorithme *Differential Evolution* applique des opérations de mutation, de croisement et de sélection. Ces opérations lui permettent d'améliorer sa population à chaque génération et de s'approcher de l'optimum global. L'algorithme DE est considéré comme l'un des plus efficaces [145].

PRINCIPES GENERAUX

L'ensemble des solutions admissibles est une population et chaque individu de la population est un vecteur. Tout comme les algorithmes génétiques, l'algorithme DE utilise la mutation pour générer un vecteur mutant et l'opération de croisement pour générer un vecteur enfant. Il est à noter que le croisement se fait entre les paramètres (valeurs) d'un vecteur muté et ceux d'un vecteur parent issus de la population [146]. Puis, un des vecteurs est choisi pour passer dans la génération suivante (la nouvelle population) en fonction de la valeur de la fonction objectif.

Les stratégies de mutation et les opérateurs de croisement jouent un rôle important sur les performances de l'algorithme (p. ex., vitesse de convergence), tandis que la taille de la population N, le facteur d'échelle F et le taux de croisement C_r permettent de contrôler l'équilibre entre la variété de la population et la vitesse de convergence de l'algorithme.

DESCRIPTION DETAILLEE

La première étape est la génération aléatoire d'une population. Cette population initiale notée P est constitué de N individus. Le vecteur X_i représente un individu tel que $X_i = (x_{1,i}, x_{2,i}, \dots, x_{D,i})$ et où D est le nombre de dimensions dans l'espace de solutions [146]. La population varie avec les générations jusqu'à une génération maximale notée G_{max}.

Le i-ème individu de la population P à la génération G est noté $X_i^G = (x_{1,i}^G, x_{2,i}^G, \dots, x_{D,i}^G)$. X_L et X_U représentent respectivement la borne inférieure et supérieure dans chaque dimension de l'espace de recherche. Nous avons donc :

$$X_L = (x_{1,L}, x_{2,L}, \dots, x_{D,L}) \qquad (2.8)$$
$$X_U = (x_{1,U}, x_{2,U}, \dots, x_{D,U}) \qquad (2.9)$$

La population initiale noté P^0 est générée au hasard et est comprise dans l'intervalle (X_L, X_U). Ensuite, les opérateurs de mutation et de croisement sont appliqués à la population pour donner naissance à des vecteurs enfant. Les meilleurs vecteurs sont sélectionnés pour passer à la génération suivante.

La mutation est formalisée par :

$$v_i^G = x_{r_1}^G + F \times (x_{r_2}^G - x_{r_3}^G), \qquad (2.10)$$

où $r_1 \neq r_2 \neq r_3 \neq i$ tel que $r_1, r_2, r_3 \in \{1,2, \dots N\}$ qui représentent les indices choisis de manière aléatoire et F est un nombre réel pour contrôler l'amplification de $(x_{r_2}^G - x_{r_3}^G)$.

Le croisement est effectué de deux manières, le croisement binomial et exponentiel. Le vecteur enfant dans le croisement binomial est calculé par la formule suivante :

$$u_{j,i}^G = \begin{cases} v_{j,i}^G, if(rand < Crorj = j_{rand}) \\ x_{j,i}^G, otherwise \end{cases} \qquad (2.12)$$

avec *rand* un nombre aléatoire distribué dans [0, 1], et $Cr \in [0,1]$ dénote le taux de croisement. j_{rand} est un entier aléatoire compris entre [1, *D*]. Ce dernier garantit qu'un vecteur enfant hérite au moins d'un composant du vecteur mutant.

La sélection est effectuée à l'aide de la fonction objectif f et permet de passer à la génération suivante. La sélection est formalisée par l'équation (2.13) :

$$x_i^{G+1} = \begin{cases} u_i^G, f(u_i^G) \leq f(x_i^G) \\ x_i^G, otherwise \end{cases} \qquad (2.13)$$

CHAPITRE 4

ÉTUDE ET PREPARATION DES DONNEES

Ce présent chapitre consiste à comprendre et préparer les données afin de pouvoir passer au processus d'apprentissage. Nous décrivons dans les sections ci-dessous les caractéristiques de la base de données brute et les transformations opérées. La première section est consacrée à l'étude descriptive de la base de données regroupant les informations sur les véhicules et les sinistres d'un lot de véhicules. La seconde est dédiée à la préparation des données afin d'être utilisées pour paramétrer des modèles d'apprentissage automatique. Enfin, la dernière section présente les paramétrages du pipeline sur python.

4.1 DESCRIPTION DE LA BASE DE DONNEES

La base de données qui sert de test pour notre étude est une base de données réelle bruit d'un assureur canadien. L'entreprise a anonymisé la base pour des raisons de confidentialité avant de la mettre à notre disposition. A titre d'illustration le numéro de sinistre est anonymisé mais conserve les mêmes caractéristiques (toutes expositions, couvertures, demandeurs, etc. Aussi la ville où le sinistre s'est produit est anonymisé. Le tableau 3 décrit les variables et précise les modifications apportées. Le reste de la description se trouve dans le tableau XX dans l'annexe 1.

Tableau 3: Description des variables de la base de données

Variables	Anonymisé	Avec	Description
CLAIMNUMBER (NUMÉRO DE RÉCLAMATION)	Oui	Fonction	Identifiant de réclamation unique attribué par le Centre de réclamation lors de sa création.

Variables	Anonymisé	Avec	Description
			Le numéro de sinistre est anonymisé mais conserve les mêmes caractéristiques (toutes expositions, couvertures, demandeurs, etc. faisant partie d'un sinistre auront tous le même numéro de sinistre).
CLAIMORDER (ORDRE DE RÉCLAMATION)	Oui		Ordre d'exposition de la réclamation.
EXPOSURE TYPE (TYPE D'EXPOSITION	Non		Le type d'exposition aux réclamations est le niveau le plus élevé de répartition des réclamations qui identifie la raison pour laquelle nous payons une réclamation. Les réclamations sont classées en sous-catégories selon que la police offre une

Variables	Anonymisé	Avec	Description
			couverture ou s'il y a responsabilité. (Par exemple, dommages au véhicule, indemnités d'accident, blessures corporelles, etc.)
COSTCATEGORY_L_EN_CA	Non		Un autre niveau de détail pour catégoriser davantage les transactions financières sous une couverture spécifique. (Par exemple, pour les sinistres automobiles : Entretien ménager, frais d'obsèques, frais d'examens, sont des exemples pour la couverture des indemnités d'accident).
LOSSDATE			La date à laquelle le sinistre est survenu.

Variables	Anonymisé	Avec	Description
			Cet attribut permet de visualiser les réclamations en fonction de la date de la perte plutôt que du moment de l'activité ou de la transaction mesurée.
REPORTEDDATE			Date à laquelle le sinistre a été signalé à l'assureur.
EXPOSURE_ID_M	Oui		ID de l'exposition
COSTTYPE_L_EN_CA	Non		Une composante de la ligne de réserve qui identifie si la transaction est faite pour une perte ou une dépense. Les charges sont ensuite séparées entre les charges légales et les charges d'ajustement.
CLOSEDATE	Non		La date à laquelle le statut de la

Variables	Anonymisé	Avec	Description
			réclamation devient fermé. Cette date de clôture de réclamation peut changer au fil du temps, par ex. si une demande est rouverte, la date de clôture de la demande reviendra à une valeur « vide » jusqu'à ce qu'elle soit à nouveau fermée. Ensuite, la date de clôture de la réclamation sera révisée en conséquence pour refléter la date de clôture la plus récente. Les paiements appliqués à une demande clôturée ne

Variables	Anonymisé	Avec	Description
			déclencheront pas une nouvelle date de clôture de la demande, seul un changement de statut le fera.

L'ensemble de données comptabilise plus de trois millions de contrats entre 2015 et 2021. Pour chaque année, les données sont conservées dans un fichier CSV. Dans chacun des fichiers, nous avons au minimum 29 colonnes (variables). Les variables explicatives sont hétérogènes. Elles sont constituées de variables numériques (ordre de réclamation, date de perte, date de rapport, kilométrage annuel, réserve, frais d'ajustement, récupération, sommes versées pour couvrir les pertes, etc.) et des variables catégoriques (numéro de réclamation, type d'exposition, catégorie de coût, type de coût, type de dommage corporel, etc.). La figure 22 illustre la distribution des variables.

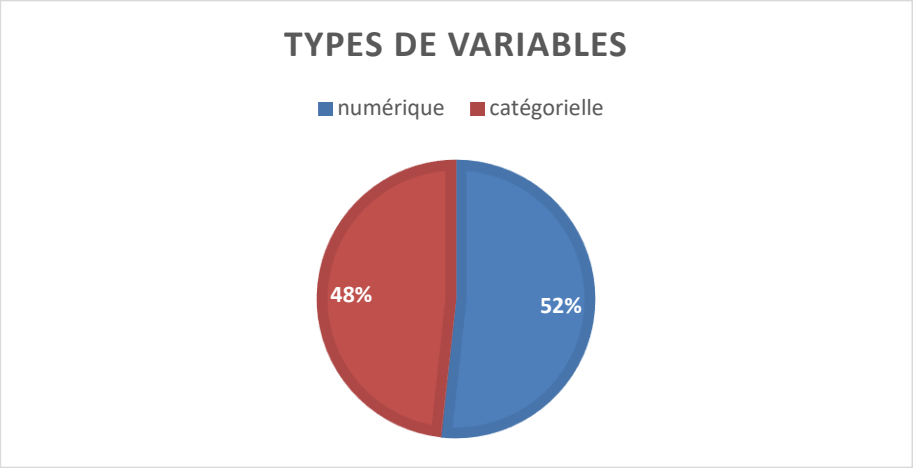

Figure 22: Répartition des types de variable de la base de données

En ce qui concerne le nombre de sinistres, l'ensemble de données comporte 4 593 561 déclarations de sinistre entre 2015 et 2021. Le tableau 4 illustre la répartition du nombre de

sinistre déclarer par année. Pour toutes les déclarations, la réserve est de 730 844 821.1 CAD. Une réserve est un montant qui représente un passif potentiel. Cette métrique mesure la variation, au cours d'une période donnée, de la valeur estimée des réclamations non encore payées. Cette variation est le résultat d'une transaction financière sur une réclamation qui représente une réserve ou un montant de paiement. La réserve moyenne par sinistre est de 159.1 (unité de mesure). Exceptionnellement, les paiements qui n'érodent pas les réserves n'auront aucun impact sur la mesure de la variation des réserves. Le tableau 5 précise le montant de réserve dépensé par année et la moyenne par sinistre.

Pour les paiements, nous avons une valeur moyenne de 533 939 097,65 par an. Cette caractéristique correspond à la somme des montants versées afin de couvrir les pertes et/ou dépenses pour un sinistre spécifique. Le paiement direct est la somme du montant versé à l'assuré, des montants versés à un fournisseur (de services ou de réparations au profit de l'assuré), des montants versés à un tiers ou pour le compte d'un assuré fautif, des frais de règlement des sinistres et des honoraires versés à des avocats, experts et enquêteurs externes pour défendre des réclamations. Le tableau 5 illustre les paiements directs effectués par année.

Tableau 4: Répartition du nombre des sinistres déclarer par année

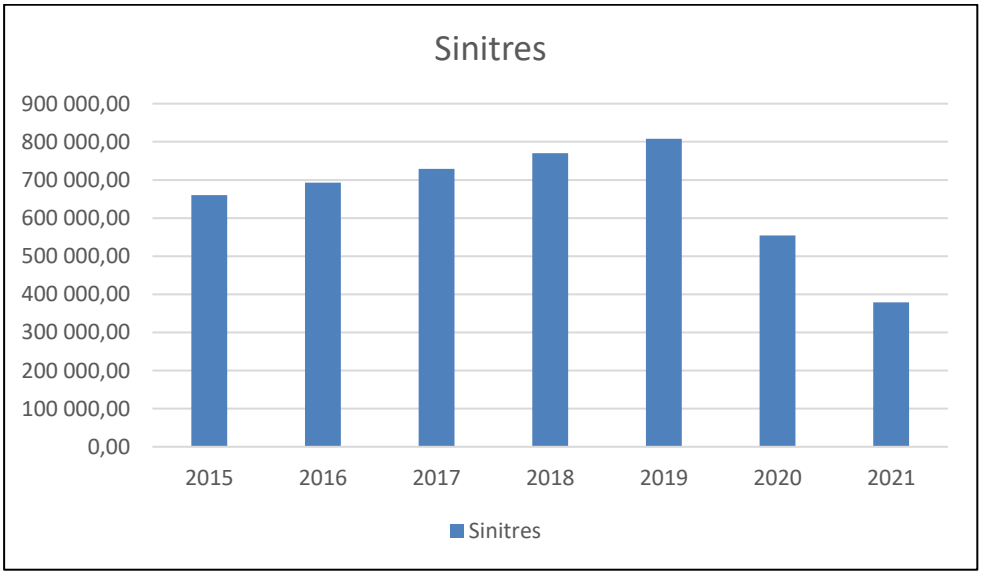

Tableau 5: Sinistre, réserve et paiement du jeu de données

Année	Sinistre déclarer	Réserve	Paiement
2015	660 047,00	35 627 843,90	756 478 323,93
2016	692 717,00	60 031 147,77	813 301 497,94
2017	729 074,00	84 521 065,80	799 031 873,62
2018	770 097,00	133 502 803,20	807 384 073,97
2019	808 163,00	811 554 766,31	199 738 920,56
2020	554 680,00	134 876 203,23	571 175 426, 16
2021	378 783,00	82 546 836,58	361 638 993,52
Moyenne	656 223,00	191 808 666,68	533 939 097,65

En utilisant les données de 2015 à 2021, les statistiques montrent que 37% des réclamations ont une un sinistre compris entre 1 et 4500 CAD. Cependant, ces sinistres jugés de faible montant équivalent à 29,5% du montant totale des sinistres. D'autres part les sinistres extrêmes (>150 000) correspondent à 0.05% des réclamations et contribue à 15,2% du montant total des sinistres. Le montant moyenne de sinistre est de 1 146 unités et le maximum est 2 098 773 unités et un Écart type de 10 325,18. Le tableau 6 donne plus détail sur la répartition des sinistres.

Tableau 6: Répartition de la charge de sinistre.

Charge de sinistre	%nombre	%montant
0	57,06	0
]0, 4500 [	37,63	29.55
]4 500, 30 000 [	04,85	40.66
]30 000, 150 000 [	00,33	16.70
]150 000, max [	00,05	15.22

L’écart entre le montant moyen et la médiane des paiements montre que la distribution des sinistres est asymétrique. Cette observation est confirmée à l’aide du coefficient d’asymétrie *skewness* [96] égal à 60.89.

Une autre variable intéressante est le type de couverture de l’assurance automobile. La couverture d'assurance est l'engagement pris par l'assureur de verser une indemnité à l'assuré (ou à ses ayants droit). Ceci, dans le but de réparer les conséquences d'un sinistre [147]. Elle définit l'ensemble des risques ou possibilités de sinistres qui sont protégés par la police. Ce n'est que lorsque l'un d'eux survient que l'assuré peut demander une indemnisation. Une couverture a une limite qui est établie au moment de la signature du contrat. La limite est appelée capital assuré. Il existe plusieurs types de couverture. Les principales couvertures que l'on trouve en assurance automobile sont *collision*, (l’assurance collision couvre le remplacement ou la réparation de votre voiture sous réserve d’une franchise dans un accident routier, peu importe qui est le fautif), *complète* (une couverture complète prendra en charge les frais pour réparer ou remplacer votre voiture sous réserve d’une franchise déterminée en cas de dégâts ne provenant pas d’une collision), et *responsabilité civile* (l’assureur s’engage à verser des dommages aux personnes qui ont été blessées ou dont la propriété a été endommagée dans un accident automobile). Il existe aussi les types de couvertures comme la *protection contre les blessures corporelles*, couverture de *responsabilité pour automobilistes sous-assurés ou non-assurés*, etc. Un ou plusieurs types de couvertures sont proposés par les assureurs sous forme de formule. Les formules les plus populaires sont *assurance auto au tiers*, *formule intermédiaire*, *assurance tous risques*, etc. Dans notre ensemble de données, la couverture *collision* est la plus utilisée par les individus de sexe masculin. Quant aux individus de sexe féminin, ils choisissent en majorité *accidents benefits*. Les autres individus choisissent le plus souvent *garage specified perils.* La figure 23 illustre la distribution en pourcentage du type de couverture en fonction du genre de l’individu.

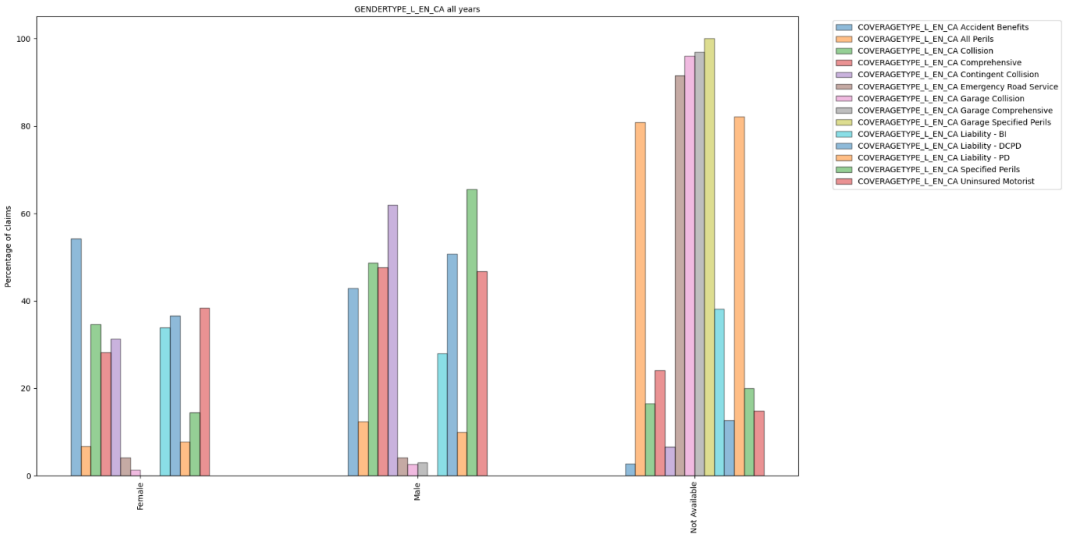

Figure 23: Les couvertures en fonction du genre de l'individu

4.2 PREPARATION DES DONNEES

La préparation des données est la première étape lorsqu'il s'agit de créer un modèle d'apprentissage automatique. Les données peuvent contenir des incohérentes, être incomplètes et/ou inexactes. Ainsi, il faut s'assurer que les données soient nettoyées, formatées et organisées afin d'être adéquatement exploitable par les algorithmes d'apprentissage automatique. L'ensemble de données de ce projet de recherche contient des valeurs non cohérentes, des valeurs manquantes, des variables discrètes avec un très grand nombre de modalités et des valeurs aberrantes. Dans cette section, nous présentons les opérations importantes de traitement des données.

4.2.1 LES VALEURS NON-COHERENTES ET EXTREMES

Il est très important d'identifier et de gérer parfaitement les valeurs manquantes. En effet, s'il reste des valeurs manquantes, cela peut causer des erreurs dans les scripts python car une majorité des librairies ne les supporte pas. Si les valeurs manquantes sont inadéquatement gérées (p. ex., remplacement avec une valeur inappropriée), cela peut créer

des biais dans les résultats de classification, conduisant ultimement à des observations et interprétations inexactes et/ou erronées.

Il existe plusieurs méthodes classiques afin de gérer les données manquantes. Par exemple, nous pourrions supprimer une ligne particulière (où il y a la présence d'une valeur manquante) ou remplacer la valeur manquante par une valeur (p. ex., la moyenne de la colonne où se trouve celle-ci). Le remplacement d'une valeur manquante n'est pas conseillé pour des données catégorielles. D'autres méthodes plus avancées comme *K-Nearest Neighbor* ou *Gray k-Nearest Neighbored* permettent de remplacer les données manquantes par la valeur du plus proche voisin [148].

Dans notre projet de recherche, en fonction du type de données (p. ex., variable numérique, variable catégorielle) et de la signification de la variable, un choix adéquat de gestion des valeurs manquantes a été réalisé. A titre d'exemple, voici les instances qui ont été supprimées :

- celles avec des valeurs manquantes dans la colonne ANNUALMILEAGE_CG,
- celles avec des valeurs de kilométrage inférieures à 1,
- celles où l'année de naissance est non disponible (valeur -1),
- celles où le type d'utilisation du véhicule n'est pas précisé,
- celles où le sexe du demandeur n'est pas identifié,
- celles qui ne précisent pas si l'assuré est responsable de l'accident (valeur -1),
- celles avec la valeur NaN dans la colonne de la variable cible,
    - Si une valeur NaN est détectée dans la colonne, la ligne entière est supprimée. Ainsi, la variables cible n'aura aucun NaN. Néanmoins les autres colonnes de l'ensemble de données peuvent avoir des valeurs NaN.

- celles où la colonne COVERAGETYPE_L_EN_CA possède une des valeurs suivantes : *Emergency Road Service, Garage Collision, Garage Comprehensive, Garage Specified Perils*.

4.2.2 REGROUPEMENTS

Les regroupements des valeurs des variables qualitatives sont importants car ils permettent de mieux généraliser les règles de classification. Cependant, si cela n'est pas effectué, nous pourrons observer une augmentation des temps de calculs et une difficulté plus grande à détecter ou à modéliser les interactions entre les variables. Le regroupement peut être effectué soit en utilisant des outils statistiques, soit en consultant un expert. Les outils statistiques ont pour but de créer une nouvelle variable à partir d'un regroupement de plusieurs valeurs. Par exemple, dans notre projet de recherche, la variable pour les types de couverture a les valeurs Indemnités d'accident 2010 et Indemnités d'accident 2016 qui ont été regroupées en Indemnités d'accident.

4.2.3 REMPLACEMENTS

Ils permettent de remplacer chaque valeur d'une série par une autre valeur, qui peut être dérivée d'une fonction. Ils sont réalisés le plus souvent par un expert. Ils permettent d'identifier des valeurs qui seront simplement remplacer par des valeurs plus adéquates. Cela permet d'éviter à l'algorithmes d'utiliser des valeurs non cohérentes. Dans notre étude, les valeurs comme *40- ,50s, 60s, 2000+* de la variable *année de naissance* sont remplacées respectivement par 40, 50, 60 et 100. Cela permettra de donnée plus de sens et de lisibilité des données.

4.2.4 ENCODAGE DES DONNEES CATEGORIELLES

Les donnée catégorielles (ou qualitatives) sont des variables avec des catégories spécifiques. En d'autres termes, les données sont constituées de valeurs possibles finies [55]. Par exemple, une liste de plusieurs personnes avec leur groupe sanguin : A+, A-, B+, B-, AB+,

AB-, O+, O- etc. Le groupe sanguin d'une liste de personnes est une variable catégorielle. Dans notre projet de recherche, les colonnes décrivant le sexe, le type d'exposition, la catégorie de coûts, le type de couverture, la province ou encore la perte sont des variables catégorielles.

Il existe deux types de données catégorielles : les données nominales et ordinales. Par exemple, dans une entreprise, le nom des différents départements (département informatique, département des ressources humaines, département des comptes et de la facturation, etc.) constituerait une variable nominale. Aucune relation d'ordre existe. Dans notre projet de recherche, le sexe est un exemple de donnée nominale qui peut prendre les valeurs homme, femme ou entreprise. Les données ordinales illustrent une relation d'ordre ou d'échelle entre les valeurs. Par exemple, une variable décrivant le degré ressenti pour un sentiment spécifique peut prendre les valeurs très malheureux, malheureux, ok, content et très content.

L'encodage des données catégorielles est un processus de conversion. Ce processus transforme des données catégorielles en entiers. Il est, en général, obligatoire car une majorité des algorithmes d'apprentissage automatique ne gère pas les données dans le format string. Enfin, pour certains ensembles de données et certains algorithmes, l'encodage permet d'améliorer les prédictions [55]. Dans notre projet de recherche, nous avons remplacé femme par 0, puis homme par 1 et enfin Non disponible par -1. Les variables VEHICULE_USE et LOSS_LOC_PROV_M sont aussi converties en entier. Bien évidemment, l'encodage est inversible.

4.2.5 ECHANTILLONNAGE

Pour traiter le problème du déséquilibre des classes et les problèmes de grandes quantités de données. Nous avons opté pour une approche de d'échantillonnage. Notre but est de parvenir à identifier un échantillon de sorte que le modèle ne soit pas sur-ajusté ou sous-ajusté sur des données. Nous avons besoin d'avoir 6000 déclarations (instances_qty = 6000) de sinistre pour constituer notre échantillon. Puisque le but est d'avoir le même nombre

d’instance pour chaque classe, alors nous listons les différentes classes que comporte la base de données avant de passer à la sélection des instances. Par exemple si la base de données est constituée de 6 classes alors l’idéale est d’avoir 1000 instances par classe. Le problème qui se pose est que toutes les classes n’ont pas forcément 1000 instances. Dans ce cas, nous prenons le nombre existant et essayons de prendre quelques instances d’autres classes pour atteindre les 6000. Cela implique que des classes dépasserons légèrement les 1000 instances. Nous avons une fonction qui permet de retourner le nombre maximal et minimal d’instances d’une classe (majoritaire et minoritaire, illustré dans le tableau 7). Cette fonction permet de savoir quelle classe est sous représenter. Après avoir eu une idée du nombre d’instance maximale et minimale de chaque classe, nous passons à la sélection des instances. Nous sélectionnons de manière aléatoire les instances pour constituer la base. Pour nous faciliter la tâche nous utilisons la classe *RandomUnderSampler* pour la sélection. *RandomUnderSampler* est une fonction qui permet de sous-échantillonner la ou les classes majoritaires en choisissant des échantillons au hasard avec ou sans remise. La fonction permet de définir le rapport souhaité du nombre d'échantillons dans la classe minoritaire sur le nombre d'échantillons dans la classe majoritaire après rééchantillonnage. Ce rapport est calculé en amont avec l’algorithme de la figure 8 avant d’être fourni à la fonction *RandomUnderSampler.* Nous séparons après les données en données d’entrainement et de test en respectant respectivement les proportions 66% et 33%.

Tableau 7: Pseudocode pour déterminer la quantité d'instances par classe

BEGIN
    min_value : nombre d'échantillons dans la classe minoritaire
    max_value : nombre d'échantillons dans la classe majoritaire
    $ratio = \frac{min\_value}{max\_value}$
    IF ratio < 0.20:
        instances_qty = min_value * 10
    ELSE
        instances_qty = math.ceil((max_value + min_value) / 2)
END
/* *math.ceil* permet d'arrondir un nombre vers le haut à son entier le plus proche*/

4.2.6 SELECTION DES VARIABLES

La sélection de caractéristiques est un processus qui permet de réduire le nombre de variables d'entrée lors du développement d'un modèle prédictif. Le processus de réduction du nombre de variables à plusieurs avantages tels que réduire le coût de calcul de la modélisation et, dans certains cas, améliorer les performances du modèle.

Prenons par exemple un ensemble de données sur des réclamations. Les fonctionnalités comme le nom du client, l'identifiant de réclamation ou encore l'ordre d'exposition de la réclamation n'ont aucun rapport avec la colonne paiement. En revanche, d'autres colonnes comme le type de couverture, le type de véhicule ont un rapport avec la colonne paiement. Alors supprimer ces trois colonnes permet de réduire le surajustement, améliorer la précision, réduire le temps de formation et généraliser mieux le modèle. Aussi, les deux colonnes restantes sont faciles à interpréter et à expliquer.

Dans notre cas, les paramètres utilisateurs permettrons de faire une première sélection de caractéristiques, c'est-à-dire que l'utilisateur précisera une liste des colonnes à garder avant de passer au processus de sélection automatiques.

Quant à la sélection automatique nous utiliserons les librairies VarianceThreshold, SelectFromModel et SelectKBest. Chacun de ses librairies à sa manière de réduire les variables (par exemple, le *VarianceThreshold* se base sur un seuil pour faire la suppression des variables). Chaque librairie à un ou plusieurs paramètres qui permet de tendre plus ou moins vers le nombre de caractéristique idéale pour concevoir le modèle. Ces algorithmes seront tous utilisé dans l'algorithme d'optimisation afin de choisir le meilleur en fonction du problème.

4.3 PARAMETRAGE DE L'ALGORITHME SOUS PYTHON

Le langage python est le langage le plus utilisé par les data scientistes et les programmeurs en intelligence artificielle [149]. Il est le plus utilisé pour les projets de Big Data et de Machine Learning. Il est parmi les meilleurs parce que d'une part, il est simple à maîtriser du point de vue de la syntaxe. D'autre part il possède une communauté active et un vaste choix de bibliothèques (pandas, sklearn, numpy, matplotlib) et de ressources. Python est le langage de programmation préféré de 88% des étudiants en data science [150]. La figure 24 présentes les langages les plus populaires en data science.

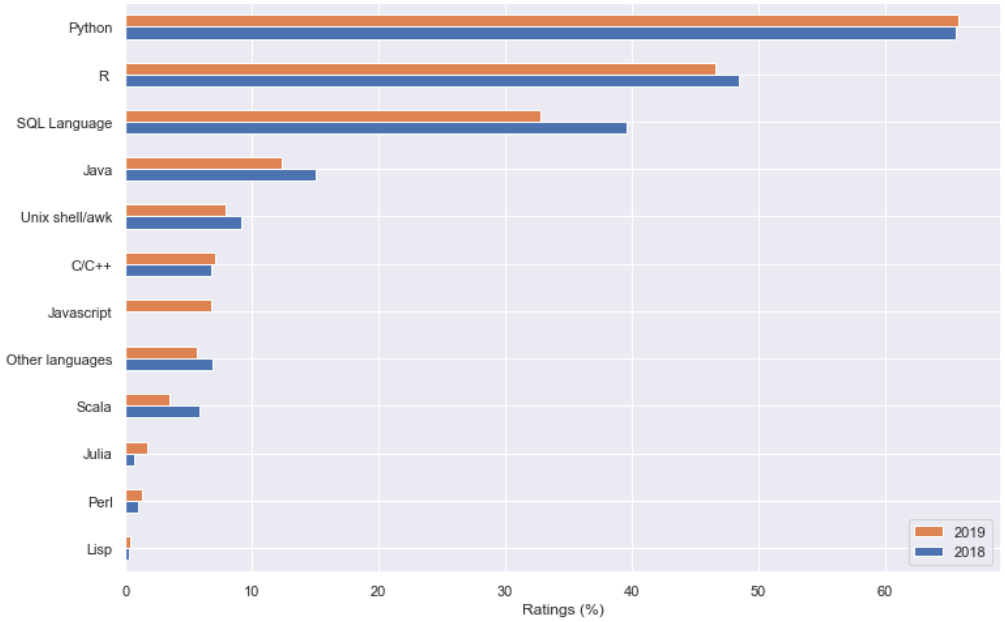

Figure 24: langages de programmation les plus populaires en sciences des données en 2019,

[151]

4.3.1 SELECTION DE VARIABLES ET CLASSIFICATION

Python offre plusieurs bibliothèques ou algorithmes pour faire la sélection, la classification et l'optimisation. Dans notre étude nous utilisons pour la sélection les algorithmes de la bibliothèque *sklearn* : *VarianceThreshold*, *SelectFromModel, SelectKBest*. Le tableau 8 ci-dessous présente les valeurs que prennent les paramètres de chaque algorithme de sélection ainsi que la valeur par défaut.

Tableau 8: Algorithmes de sélection de variable

Algorithmes de sélection	Paramètres optimisé	Valeurs par défaut
VarianceThreshold	threshold	0
SelectFromModel	threshold	None
	estimator	-
SelectKBest	K	10

Le paramètre *threshold* permet de supprimer les fonctionnalités dont la variance de l'ensemble d'apprentissage est inférieure à ce seuil. La valeur par défaut est de conserver toutes les caractéristiques avec une variance non nulle, c'est-à-dire de supprimer les caractéristiques qui ont la même valeur dans tous les échantillons.

Le paramètre *estimator* est l'estimateur de base à partir duquel le transformateur est construit. Il peut s'agir d'un estimateur ajusté ou d'un estimateur non ajusté. L'estimateur est dit ajusté si prefit est défini sur True. L'estimateur doit avoir un attribut feature_importances_ou après l'ajustement.

Le paramètre *k* défini le nombre de fonctionnalités principales à sélectionner. L'option "all" contourne la sélection, pour une utilisation dans une recherche de paramètre. La valeur par défaut de k est 10.

Pour la classification, les bibliothèques suivantes de *sklearn, catboost* sont utilisées : *DecisionTreeClassifier, RandomForestClassifier, CatBoostClassifier.* En rappel, un algorithme de classification est un processus de reconnaissance, de compréhension et de regroupement d'objets et d'idées dans des catégories prédéfinies appelées « sous-populations ». Le tableau 9 présente les paramètres utilisés par les algorithmes de classification.

Tableau 9: Paramètres des algorithmes DT, RF et CatBoost

Algorithmes	Paramètres	Valeurs par défaut
DecisionTreeClassifier	*criterion*	gini
	splitter	best
	max_depth	None
	max_features	None
	min_samples_split	2
RandomForestClassifier	*criterion*	gini
	max_depth	None
	max_features	sqrt

	min_samples_split	2
CatBoostClassifier	*iterations*	1000
	learning_rate	0.03
	depth	6 pour GPU 16 pour CPU
	l2_leaf_reg	3.0

Criterion est Le paramètre permet de choisir la fonction pour mesurer la qualité d'une division. Les critères pris en charge sont « gini » pour l'impureté Gini et « log_loss » et « entropy » pour le gain d'informations de Shannon. La valeur par défaut de *criterion* est gini.

Splitter est le paramètre qui permet de sélectionner la stratégie utilisée pour choisir le fractionnement à chaque nœud. Les stratégies prises en charge sont « best » pour choisir la meilleure répartition et « random » pour choisir la meilleure répartition aléatoire. La valeur par défaut est best.

max_depth permet de définir la profondeur maximale de l'arbre. Si sa valeur est « None », les nœuds sont développés jusqu'à ce que toutes les feuilles soient pures ou jusqu'à ce que toutes les feuilles contiennent moins de min_samples_split échantillons.

min_samples_split défini le nombre minimum d'échantillons requis pour diviser un nœud interne. Sa valeur peut être un entier ou un nombre réel. Quand elle est un entier, alors considérez min_samples_split comme le nombre minimum. Dans le cas contraire, min_samples_split est une fraction et représente le nombre minimum d'échantillons pour chaque fractionnement. La fraction est ceil(min_samples_split * n_samples).

max_features permet de préciser le nombre de fonctionnalités à prendre en compte lors de la recherche de la meilleure répartition : Il peut prendre une valeur entière, réel, ou encore *auto, sqrt*, *log2* et *none*. Si la valeur est un entier, alors il considère max_*features* comme le nombre maximum de fonctionnalités à chaque fractionnement. Dans les autres cas, c'est une

fraction et les caractéristiques sont prises en compte à chaque résultat tourné par la fraction. Comme exemple pour sqrt elle retourne *max_features=sqrt(n_features).*

4.3.2 OPTIMISATION PAR VALIDATION CROISEE

GridSearchCV et *RandomizedSearchCV* sont les deux bibliothèques python d'optimisation par validation croisé que nous utilisons. Le tableau 10 ci-dessous présente les paramètres utilisés ainsi que les valeurs par défauts.

Tableau 10: Paramètres des algorithmes de validation croisées

Algorithmes	Paramètres	Description/ valeur par défaut
GridSearchCV	estimator	Le pipeline mis en place
	Cv	Détermine la stratégie de fractionnement de la validation croisée. La valeur par défaut est None et utilise 5-fold cross validation
	return_train_score	Il est Faux par défaut. Il permet d'inclure ou Non le score d'entraînement
	param_grid	Un dictionnaire avec des noms de paramètres comme clés et des listes de réglages de paramètres à essayer comme valeurs, ou une liste de ces dictionnaires, auquel cas les grilles couvertes par chaque dictionnaire de la liste sont explorées. Elle servira à générer les valeurs possibles pour former le modèle
	scoring	La stratégie utilisée pour évaluer les performances du modèle par validation croisée sur l'ensemble de test.

	n_jobs	None par défaut, Il désigne le nombre de tâches à exécuter en parallèle. -1 permet d'utiliser tous les processeurs
	verbose	Permet de fournir des informations pendant le processus d'entrainement.
RandomizedSearchCV	estimator	Le pipeline mis en place
	n_iter	Permet de régler le nombre de paramètres échantillonnés
	Scoring	-
	jobs	-
	verbose	-

Les deux algorithmes ont presque les mêmes paramètres. La différence majeure est que *GridSearchCV* fait toutes les combinaisons tandis que *RandomizedSearchCV* permet à l'utilisateur de contrôler explicitement le nombre de combinaisons de paramètres qui sont tentées. Les deux seront utilisés pour comparer le compromis entre le temps d'exécution et la qualité de la solution.

4.3.2 OPTIMISATION METAHEURISTIQUES

Pour notre étude nous implémentons 03 métaheuristiques : L'algorithmes Génétique (GA), l'algorithme Évolution différentielle (DE) et le Grey Wolf Optimizer (GWO). La librairies MEALPY est utilisés pour implémenter l'algorithmes génétique. Il existe aussi les librairies **Py GAD** et **TPOT** qui sont utilisés pour implémenter cet algorithme. Quant aux deux autres, nous utilisons toujours la librairie **MEALPY** qui nous fournit une implémentation des deux algorithmes. Elle contient aussi une implémentation d'autres métaheuristiques. Les paramètres utiliser par ces algorithmes sont présenter dans le tableau 11.

Tableau 11: Paramètre utilisé par la bibliothèque Py GAD

Métaheuristiques	Paramètres	Description	Valeur par défaut
MEALPY.GA	epoch	Nombre de générations	1 000
	mutation	est le type d'opération de mutation effectué	-
	pop_size	La taille de la population	50
	pc	probabilité de croisement	0,95
	pm	Probabilité de mutation	0.025
	selection	le type de sélection du parent	tournament
	k_way	il est défini lors de l'utilisation de la sélection tournament	0,2
	crossover	est le type d'opération de croisement mis en place	uniform
	mutation_multipoints	effet sur le processus de mutation	Vrai
	Function fitness		fonction définie
MEALPY.DE	pop_size	La taille de la population	50
	weighting factor	le facteur de pondération	wf = 0.7
	Crossover rate	taux de croisement utiliser	cr = 0.9
	strategy	la stratégie utilisée pour la mutation	0 = DE/current-to-rand/1/bin
MEALPY.GWO	epoch	Le nombre d'époque	1000
	pop_size	La taille de la population	50

Description des paramètres de la librairies MEALPY.GA :

- **Epoch**: nombre de générations

- **Mutation** : est le type d'opération de mutation effectué. Les types pris en charge sont swap (pour la mutation d'échange), inversion (pour la mutation d'inversion), scramble (pour la mutation de brouillage) et flip (pour la mutation adaptative). Les valeurs dépendent du type de mutation. SI le chromosome est muté sur un seul point alors les valeurs possibles sont : flip (devrait définir le pm grand tel que : [0.5 -> 0.9]), swap (identique à flip : pm dans la plage [0,5 -> 0,9]), scramble (devrait définir le pm suffisamment petit tel que : [0.4 -> 0.6]) et inversion (comme scramble [0.4 -> 0.6]) et si le chromosome est muté sur plusieurs points alors les valeurs possibles sont flip et swap.
- ***selection*** : est le type de sélection du parent. Les types pris en charge sont : *sss*(pour la sélection en régime permanent), *rws*(pour la sélection à la roulette), *sus*(pour la sélection universelle stochastique), *rank*(pour la sélection par rang), *random*(pour la sélection aléatoire) et *tournament*(pour la sélection en tournoi).
- **Crossover :** est le type d'opération de croisement mis en place. Les types pris en charge sont : *one*_point (pour le croisement à point unique), multi_points (pour le croisement à plusieurs points), *uniform* (pour le croisement uniforme) et *arithmetic* (pour le croisement arithmetique).

Description des paramètres la librairie MEALPY.DE :

- ***pop_size*** permet de définir une taille de la population
- ***weighting** factor* est le facteur de pondération
- ***Crossover rate :*** taux de croisement utiliser.
- ***Strategy*** est la stratégie utilisée pour la mutation : les valeurs utilisées sont *:*
    - *DE/courant-à-rand/1/bin*
    - *DE/meilleur/1/casier*
    - *DE/meilleur/2/casier*
    - *DE/rand/2/bac*
    - *DE/actuel-au-meilleur/1/bin*
    - *DE/courant-à-rand/1/bin*

Description des paramètres la librairie GWO :

- **Epoque** : le nombre d'époques est un hyperparamètre qui définit le nombre de fois que l'algorithme d'apprentissage fonctionnera sur l'ensemble de données d'apprentissage.
- **Pop size** permet de définir la taille de la population de départ.

Tous les algorithmes utilisés utilisent la même fonction de fitness pour le processus d'optimisation. Le paramètre fitness accepte une fonction qui doit prend un paramètre (une solution ou un chromosome) et retourner la valeur de fitness de la solution. Elle est aussi passée en paramètre ainsi que les autres avant le lancement du processus d'apprentissage. Dans la section suivante nous présenterons les paramètres ainsi que les conditions utilisées pour tester nos algorithmes d'une part et d'autres part nous présenterons les résultats.

CHAPITRE 5

ÉVALUATION DES RESULTATS

Dans cette section est présenté une comparaison des différents algorithmes d'optimisations. D'une part est présenter les résultats des algorithmes d'optimisation standard tels que *Grid Search* et *Random Search* et d'autre part les algorithmes d'optimisation métaheuristique comme *Genetic Algorithm*, *Differential Evolution* et *Grey Wolf Optimization Algorithm*. Le temps d'exécution moyen et le nombre d'évaluation moyen pour trouver l'optimum globale de chaque algorithme est noté.

5.1 RESULTATS

Les expériences sont menées en utilisant le même environnement pour cinq algorithmes. Quelques informations sur l'environnement de développement.

- Bibliothèque python commun : *Scikit-learn, Pandas, Numpy, Matplotlib, os, mealpy*
- Une machine virtuelle : Processeur Intel(R) Xeon(R) de 8GB de ram. Plus de détail dans le tableau 12.
- Les données de déclaration de sinistre de l'assureur canadien COOPERATORS est utilisés comme ensemble de données.
- Les mêmes techniques de prétraitement est effectué sur les données : à savoir le nettoyage des valeurs aberrantes, la conversion des chaines de caractères en entiers.
- Un même échantillon de la base de données est utilisé. Le tableau 13 décrit les détails du jeu de données utilisées en test. Nous récupérons 6 000 instances parmi les millions de données. Ce sous-ensemble est équilibré. Une simple règle est appliquée pour déterminer le nombre d'instance pour chaque classe. Après avoir appliqué la formule, la figure 14 présente la répartition des instances par classe. Ces instances sélectionnées sont par la suite divisées en deux ensembles (0.66 % en données d'entrainement et 0.33% en données de tests). L'ensemble d'entrainement est utilisé pour former les modèles et les données de tests pour évaluer chaque modèle. Chaque

algorithme d'optimisation utilise ses deux ensembles. Chaque ensemble est formé de 16 colonnes sur les 29 au départ. Les 13 colonnes sont supprimées de façon manuelle car sont des données qui n'ont aucun intérêt dans le modèle. Les données sont bien détaillées dans le chapitre précèdent.

Tableau 12: Caractéristique de l'ordinateur utilisé pour les tests

Processor	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz 2.29 GHz
Type	Virtuelle
Random Access Memory (RAM)	8.00 GB
Système d'exploitation	64-bit operating system, x64-based processor
Edition	Windows 10 Enterprise
Version	21H2

Tableau 13: Jeu de donnée de test

Année utilisée	**Nombre d'instances**	**Variable cible**	**Nombre de catégorie contenu dans la class**	**Nombre d'instance pour l'entrainement (66%)**	**Nombre d'instance pour le test (33%)**
2015-2021	6 000	Coverage type	10	4020	1980

Tableau 14: répartition des instances par catégorie

Nom de la catégorie	Nombre d'instances
Accident Benefits	80
All Perils	80
Collision	80
Comprehensive	80
Contingent Collision	72
Liability - BI	80
Liability - DCPD	80
Liability - PD	80
Specified Perils	8
Uninsured Motorist	80

Une fois que les données sont divisées en deux, nous avons deux possibilités. La première est de sauvegarder d'abord les deux ensembles et par la suite charger les deux ensembles avant d'appliquer une technique d'optimisation. La deuxième est de faire une boucle qui utilisera à tour de rôle les cinq algorithmes d'optimisation. Nous utilisons la deuxième technique car elle nous évite de relancer le programme pour chaque algorithme d'optimisation. Il est évident qu'elle prend plus temps parce qu'elle va tester tous les algorithmes une seule fois. La technique fait alors cinq itérations et sauvegarde le temps d'exécution de chaque algorithme ainsi que le meilleur modèle. Chaque algorithme fera un certain nombre de traitement en fonction de son paramétrage. Notre pipeline de base doit optimiser les techniques de sélection de caractéristique et les hyperparamètres. Le tableau ci-dessous présente les éléments à optimisés. Le tableau 15 présente les valeurs à optimiser avec les valeurs possibles et le tableau 16 présente les résultats des algorithmes.

Tableau 15: Les éléments à optimiser

Technique de sélection				Hyperparamètre			
Selector	paramètre	Valeurs possibles	NbS	classifier	paramètre	Valeurs possibles	NbH
VarianceThreshold	threshold	0.04, 0.02, 0.01, 0.008, 0.004, 0.001, 0	7	DecisionTreeClassifier (dt= 252 combinaisons)	criterion	gini, entropy, log_loss	3
					splitter	best, random	2
					max_depth	10, 20, 30, 50, 70, 90, None	7
					max_features	sqrt, log2	2
					min_samples_split	2, 4, 6	3
SelectFromModel	threshold	0.04, 0.02, 0.01, 0.008, 0.004, 0.001, 0	7	RandomForestClassifier (rf = 567 combinaisons)	criterion	gini, entropy, log_loss	3
					n_estimators	100, 200, 300	3
					max_depth	10, 20, 30, 50, 70, 90, None	7
					max_features	sqrt, log2, None	3
					min_samples_split	2, 4, 6	3
SelectKBest	k	8, 9, 10, 11, 12, 13, 14, 15, 16	9	CatBoostClassifier (ct= 150 combinaisons)	iterations	10, 30, 50, 70, 100	5
					learning_rate	0.03, 0.1	2
Aucun selector	Tous les paramètres sont sélectionnés		1		depth	6, 8, 10	3
					l2_leaf_reg	1, 3, 5, 7, 9	5

Les techniques de sélection de caractéristiques à optimiser sont : VarianceThreshold, SelectFromModel et SelectKBest. Les algorithmes de classifications à optimiser sont : DecisionTreeClassifier, RandomForestClassifier et CatBoostClassifier. Nous avons limité notre test à ces choix nous pourrons ajouter autant de *selector*, de *classifier* et de paramètre.

Tableau 16: Résultats de l'évaluation des cinq algorithmes d'optimisation

Classe	Optimiseur	Itération	Nombre d'évaluation	Fitness (accuracy)	Durée Exécution
Validation croisée	GridSearchCV	1	2016	0.741	0:15:05.777
		2		0.737	0:15:16.943
		3		0.735	0:12:09.025
		4		0.737	0:13:59.403
		5		0.730	0:14:39.933
	RandomizedSearchCV	1	1000	0.736	0:10:45.777
		2		0.735	0:10:08.157
		3		0.728	0:12:00.117
		4		0.736	0:10:20.555
		5		0.738	0:10:03.191
Métaheuristique	GA	1	120	0,717	0:04:48.047
		2		0,710	0:02:27.018
		3		0,712	0:03:39.117
		4		0,726	0:06:23.600
		5		0,720	0:02:37.454
	GWO	1	120	0,727	0:04:13.078
		2		0,740	0:06:17.791
		3		0,731	0:03:37.512
		4		0,737	0:11:57.618
		5		0,717	0:10:16.257
	DE	1	120	0,720	0:05:27.660
		2		0,740	0:10:00.075
		3		0,731	0:06:43.951
		4		0,735	0:04:41.148
		5		0,722	0:10:38.027

Les résultats sont détaillés dans les points suivants. Chaque point présente les résultats d'un algorithme.

5.2 ANALYSES DETAILLÉES

5.2.1 GRID SEARCH

La librairies GridSearchCV de la librairies *Scikit-Learn* est utilisé pour implémenter cet optimiseur. L'algorithme *grid search* fait une 2-crossvalidation sur 6000 instances. Les éléments à optimiser sont : 3 algorithmes de sélection de caractéristiques avec chacun un paramètre, 4 algorithmes de classification avec 5, 5, 3 et 4 paramètres respectivement. La technique calcule alors 252 combinaisons possibles pour DecisionTree, 567 combinaisons pour randomForest et 150 combinaisons pour catboost. La validation croisée donne respectivement 7, 7 et 9 combinaisons possibles pour les 3 *Selectors VarianceThreshold, SelectFromModel et SelectKbest*. Une dernière possibilité pour le *Selector* est le fait d'utiliser toutes les données. Les combinaisons possibles sont au total 9 451 776 600 = 252*567*150*7*7*9*1. En appliquant une validation croisée de 2 nous aurons alors 18 903 553 200 modèles à évaluer. Cette technique est rigoureuse et prend plus de temps car utilise toutes les combinaisons possibles. Le meilleur modèle est sans doute présente dans cet espace de recherche. Le temps moyenne mis pour évaluer ces modèles est de 14min. Le tableau 18 donne plus de détaille sur chaque itération le résultat de l'algorithme d'optimisation *grid search*.

Tableau 17: Paramétrage de GridSearchCV

Paramètre	Valeurs	Défaut
Cv	2	5
n_jobs	-1	None
verbose	1	0

Tableau 18: Résultats par itération de GridSearchCV

Itération	Selector			Classifier		
	Intitulé	Paramètre	Valeur	Classifier	Paramètre	Valeur
1	VarianceThreshold	threshold	0.004	DecisionTreeClassifier	max_depth	50
					max_features	sqrt
					min_samples_split	6
					criterion	gini
					splitter	best

2	VarianceThreshold	threshold	0.04	DecisionTreeClassifier	max_depth	10
					max_features	sqrt
					min_samples_split	4
					criterion	gini
					splitter	best
3	VarianceThreshold	threshold	0.004	DecisionTreeClassifier	max_depth	20
					max_features	sqrt
					min_samples_split	4
					criterion	gini
					splitter	best
4	VarianceThreshold	threshold	0.004	DecisionTreeClassifier	max_depth	20
					max_features	sqrt
					min_samples_split	6
					criterion	gini
					splitter	best
5	VarianceThreshold	threshold	0.008	DecisionTreeClassifier	max_depth	20
					max_features	sqrt
					min_samples_split	6
					criterion	gini
					splitter	random

Tableau 19: Résultats de GridSearchCV

Temps mis en moyen	15 min
Nombre de modèle	1008
Nombre d'évaluation	2016
Meilleur score	0.740

L'avantage de la solution est que toutes les combinaisons sont évaluées. La solution prend énormément de temps et s'empire si le nombre de données augmente ou si la dimensionnalité aussi augmente. Le temps mis est long, Plus de la moitié des modèles évalués n'ont pas un bon score. L'algorithme a pris le temps pour évaluer des modèles qui n'ont pas de bon résultat d'évaluation. Nous remarquons que sur les cinq itérations, le *Selector* le plus fréquent est le *VarianceThreshold* tandis que DecisionTreeClassifier est le plus proposé comme solution optimum.

5.2.2 RANDOM SEARCH

La librairie *RandomizedSearchCV* est utilisé pour l 'implémentation. Les mêmes conditions seront respectées à savoir : les 6 000 instances sauvegardées seront utilisées pour entrainer et évaluer le modèle. Une 2-cross validation est effectué pour la validation du modèle. Les mêmes algorithmes de sélection et de classifications sont utilisés. Le nombre de modèle évalué est égale au nombre de candidat multiplié par le nombre de cross-validation soit **nombre de modèle = n_iter * cv**. Sachant que le nombre de modèle possible de *GridSearchCV* est de 1008, nous allons diviser le nombre par 2 soit 504 modèles.

Tableau 20: Paramétrage de RandomizedSearchCV

Paramètre	**Valeurs**	**Défaut**
Cv	2	None
n_jobs	-1	None
verbose	3	0
n_iter	504	10

Tableau 21:Résultats par itération de RandomizedSearchCV

Itération	**Selector**			**Classifier**		
	Intitulé	**Paramètre**	**Valeur**	**Classifier**	**Paramètre**	**Valeur**
1	VarianceThreshold	threshold	0.008	DecisionTreeClassifier	max_depth	20
					max_features	sqrt
					min_samples_split	6
					criterion	gini
					splitter	best
2	VarianceThreshold	threshold	0.004	DecisionTreeClassifier	max_depth	20
					max_features	sqrt
					min_samples_split	6
					criterion	entropy
					splitter	best
3	VarianceThreshold	threshold	0.004	DecisionTreeClassifier	max_depth	20
					max_features	sqrt
					min_samples_split	4

					criterion	gini
					splitter	best
4	VarianceThreshold	threshold	0.004	DecisionTreeClassifier	max_depth	20
					max_features	log2
					min_samples_split	6
					criterion	gini
					splitter	random
5	VarianceThreshold	threshold	0	DecisionTreeClassifier	max_depth	20
					max_features	sqrt
					min_samples_split	2
					criterion	gini
					splitter	best

Tableau 22: Résultat de RandomizedSearchCV

Temps mis en moyenne	10 min
Nombre de modèle	504
Nombre d'évaluation	1 008
Meilleur score	0.738

L'algorithme permet de choisir au hasard 500 des modèles parmi les 1008 modèles. Cela évite d'évaluer tous les modèles. Le principe de random donne la possibilité de sélectionner le meilleur modèle dans la petite partie qui sera évaluer. Le temps d'exécution passe alors de 15 min à 10 min en moyenne. Le principe de random ne permet pas de maitriser les modèles évalués. Tant que tous les modèles ne seront pas pris en charge lors de l'évaluation, rien ne garantit que le meilleur modèle soit dans le lot. Pour les 504 modèles qui sont choisis au hasard, chacun est évalués 2 fois donc au total 1008 évaluations. Le selector le plus utilisé est *Variance Threshold* et le classifier est *Decision tree*.

5.2.3 GENETIC ALGORITHM (GA)

Nous utilisons la libraire PYGAD pour la mise en place de l'algorithme. Dans cette expérience, nous avons fixé la taille de la population à 2, le nombre de générations à 8 et la longueur du génome à 8. Le génome de chaque individu est représenté comme une variante

aléatoire de la distribution de Bernoulli avec un état aléatoire de 0,5. distribution de Bernoulli avec un état aléatoire de 0,5. Les 4 premiers bits représentent le nombre de blocs convolutifs et le reste pour le nombre de blocs denses. Nous utilisons la libraires PYGAD pour nos tests. Le tableau 23 représente les paramètres utilisés pour exécuter l'algorithme. Le résultat de chaque itération est détaillé sur le tableau 24.

Tableau 23: Paramétrage de GA

Paramètre	**Valeurs**	**Défaut**
Nombre de génération	2	-
Taille population	20	50
Longueur du génome	8	-
Cross validation	2	-

Tableau 24: Résultats par itération de GA

Itération	**Selector**			**Classifier**		
	Intitulé	**Paramètre**	**Valeur**	**Classifier**	**Paramètre**	**Valeur**
1	VarianceThreshold	threshold	0.008	RandomForestClassifier	criterion	entropy
					n_estimators	200
					min_samples_split	6
					max_features	None
					min_samples_split	6
2	SelectFromModel	threshold	0	RandomForestClassifier	max_depth	20
					max_features	log2
					min_samples_split	4
					criterion	entropy
					n_estimators	200
					min_samples_split	6
3	VarianceThreshold VarianceThreshold	threshold threshold	0	RandomForestClassifier RandomForestClassifier	max_depth	30
					max_features	log2
					min_samples_split	6
					criterion	entropy
					n_estimators	100

					max_depth	30
4	VarianceThreshold	threshold	0	RandomForestClassifier	max_depth	30
					max_features	log2
					min_samples_split	4
					criterion	gini
					n_estimators	300
5	VarianceThreshold	threshold	0.01	DecisionTreeClassifier	max_depth	70
					max_features	log2
					min_samples_split	4
					criterion	entropy

Tableau 25: Résultats de GA

Temps mis en moyenne	3,4 min
Nombre d'évaluation	120
Meilleur score	0,726

L'algorithme évolutionnaire a mis environ ≈3,4 min à s'exécuter. Le meilleur modèle affiche une précision d'environ 72,6 %. C'est presque l'optimum globale trouvé avec la recherche par grille qui est 74%. La solution optimum n'est pas garantie, elle peut être atteint à chaque moment. Avoir plus de combinaisons n'assure pas d'atteindre le maximum. Cet algorithme nous permet d'avoir une historique sur les modèles évalués. Ainsi, elle permet de lister des meilleures solutions globales trouvées jusqu'à présent dans toutes les générations précédentes, lister des meilleures solutions actuelles dans chacune des générations précédentes, lister des pires solutions globales trouvées jusqu'à présent dans toutes les générations précédentes, lister des pires solutions actuelles dans chacune des générations précédentes, lister la population dans chaque génération.

5.2.4 DIFFERENTIAL EVOLUTIONNAIRES (DE)

Les conditions de tests sont les mêmes que les deux précédents algorithmes. Les deux paramètres qui influencent le nombre de model à évaluer sont : *epoch* et *pop_size*. La formule du nombre de modèle est :

nombre de model = *epoch* * pop_size * cv = 3 * 20 * 2 = 120

Les tests sont lancés plusieurs fois en fixant la taille de la population à 20. Une cross validation est aussi utilisée.

Tableau 26: Paramétrage de DE

Paramètre	Valeurs	Défaut
Epoch	3	-
Pop size	20	50
Cv	2	5

Tableau 27: Résultats par itération de DE

Itération	Selector			Classifier		
	Intitulé	Paramètre	Valeur	Classifier	Paramètre	Valeur
1	VarianceThreshold	threshold	0.001	RandomForestClassifier	criterion	entropy
					max_depth	90
					n_estimators	200
					min_samples_split	6
					max_features	sqrt
					min_samples_split	6
2	SelectFromModel	threshold	0.001	RandomForestClassifier	criterion	gini
					max_depth	20
					n_estimators	300
					min_samples_split	6
					max_features	sqrt
					min_samples_split	6
3	VarianceThreshold	threshold	0.02	RandomForestClassifier	criterion	gini
					max_depth	70
					n_estimators	200
					min_samples_split	6
					max_features	sqrt
					min_samples_split	6
4	SelectFromModel	threshold	0.004	RandomForestClassifier	criterion	gini
					max_depth	30
					n_estimators	100
					min_samples_split	6
					max_features	sqrt

					min_samples_split	6
5	SelectFromModel	threshold	0	RandomForestClassifier	max_depth	90
					n_estimators	100
					max_features	sqrt
					min_samples_split	6
					criterion	gini
					splitter	random

Tableau 28: Résultat de DE

Temps mis en moyenne	7 min
Nombre d'évaluation	120
Meilleur score	0,74

Cet algorithme est simple à implémenter et à comprendre. Il prend seulement 7min pour l'exécution et trouve le maximum global tandis que la recherche par grille prend 15min. Ce qui permet de réduire à moitié le temps d'exécution. Nous remarquons que sur les cinq itérations, le *Selector* le plus fréquent est le *VarianceThreshold* tandis que *RandomForestClassifier* est le plus proposé comme solution optimum.

5.2.5 GREY WOLF OPTIMIZATION ALGORITHM (GWO)

La libraire PYGAD est utilisée avec la classe *DE* pour la mise en place de l'algorithme. Dans cette expérience, nous avons fixé la taille de la population à 20, le nombre d'époques est 2 nous utilisons les valeurs par défaut pour le croisement, les mutations et la sélection. Le résultat de chaque itération est détaillé sur le tableau 30.

Nombre d'évaluation = (epoch+1) * pop size * 2, (epoch+1) * pop_size = 3 * 20*2 = 120

Tableau 29: Paramétrage de GWO

Paramètre	**Valeurs**	**Défaut**
epoch	2	-

Taille population	20	50
Pc	0.9	0.9
pm	0.05	0.05
selection	tournament	tournament
crossover	arithmetic	arithmetic
Mutation	flip	flip

Tableau 30: Résultats par itération de GWO

Itération	Selector			Classifier		
	Intitulé	Paramètre	Valeur	Classifier	Paramètre	Valeur
1	VarianceThreshold	threshold	0.008	RandomForestClassifier	criterion	gini
					max_depth	90
					n_estimators	200
					min_samples_split	6
					max_features	log2
					min_samples_split	6
2	VarianceThreshold	threshold	0.01	RandomForestClassifier	criterion	entropy
					max_depth	20
					n_estimators	300
					min_samples_split	6
					max_features	None
					min_samples_split	6
3	SelectFromModel	threshold	0	RandomForestClassifier	criterion	gini
					max_depth	90
					n_estimators	200
					min_samples_split	6
					max_features	Log2
					min_samples_split	4
4	VarianceThreshold	threshold	0.004	RandomForestClassifier	criterion	gini
					max_depth	90
					n_estimators	200
					min_samples_split	4
					max_features	sqrt
					min_samples_split	4
5	VarianceThreshold	threshold	0.008	RandomForestClassifier	criterion	gini
					max_depth	90

					n_estimators	200
					min_samples_split	4
					max_features	sqrt
					min_samples_split	4

Tableau 31: Résultats de GWO

Temps mis	6.8 min
Nombre de modèle évaluer	120
Meilleur score	0,740

Cet algorithme est simple à implémenter et à comprendre. Il prend seulement 6.8 min pour l'exécution et trouve le maximum global tandis que la recherche par grille prend 15min. Ce qui permet de réduire à plus de la moitié le temps d'exécution. Il est le plus rapide des trois algorithmes.

5.2.6 COMPARAISON DES 5 ALGORITHMES D'OPTIMISATION

Dans ce chapitre est présenté les résultats des cinq algorithmes *Grid search, Randomized search, GA, GWO* et *DE*. Le tableau 32 illustre la comparaison des cinq en temps d'exécution et accuracy. Il est important de comprendre que chaque algorithme à ses forces et faibles. Il était question de voire l'avantages des algorithmes métaheuristiques pour la résolution des problèmes d'optimisation. Nous retenons que d'après notre expérience, ils ont permis de réduire de 30%, 27% et 20% le temps d'exécution comparer à GridSearchCV. Aussi une figure qui illustre la précision de chaque algorithme en 5 itérations.

Tableau 32: Comparaison des résultats des algorithmes d'optimisation

Technique d'Optimization	% de Fitness	Durée d'exécution	% de réduction
GRID SEARCH	74,1%	15 min	-
RANDOM SEARCH	73,8%	11 min	27
GA	72,6%	3.2 min	79

GWO	74,0%	7min	54
DE	74,0%	6.8 min	55

Figure 25: comparaison de la réduction du temps d'exécution

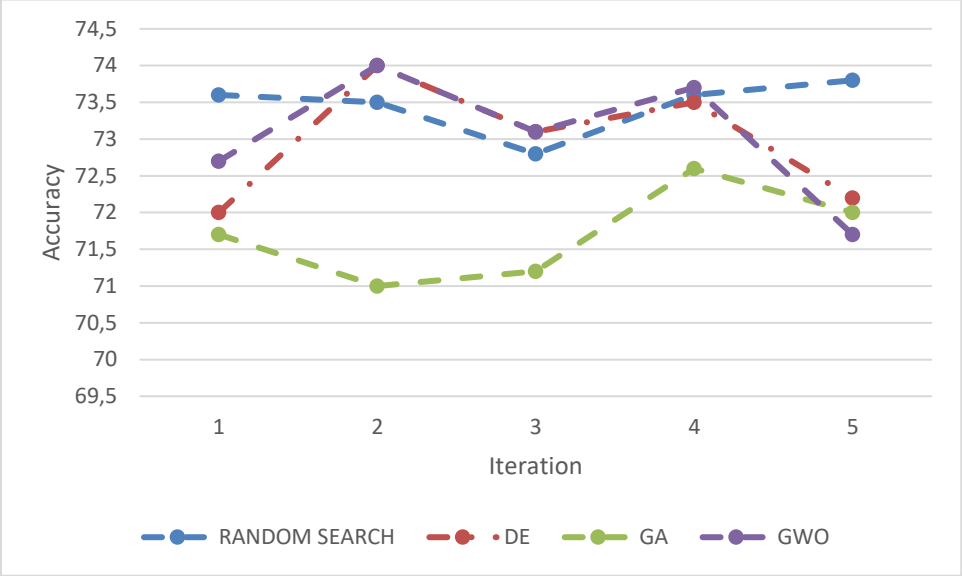

Figure 26: Itération des algorithmes optimisations

CONCLUSION

Dans cette étude, nous avons proposé une solution reposant sur une intelligence artificielle afin d'automatiser les opérations de prétraitement des données de CO-OPERATORS. La solution proposer automatise le processus de pré-traitement. Cette automatisation permet de minimiser le temps d'analyse des données pour les non-experts et experts en science des données. Pour réaliser une solution robuste et performante, nous avons présenté une étude comparative de cinq algorithmes d'optimisation (02 utilisent le brut force et les 03 sont des métaheuristiques). Le modèle d'apprentissage proposer optimise l'algorithme de sélection de variable ainsi que l'algorithme de classification utiliser. La solution permet à l'utilisateur de définir une variable cible, une liste d'algorithme de sélection et de classification avant d'entrainer le modèle. La solution proposera le meilleur modèle en fonction de la variable cible.

Les résultats de notre étude montrent que l'approche proposé à de l'avenir. Aussi, nous retenons que pour ce qui concerne la technique d'optimisation, chaque algorithme a ses propres forces et faiblesses, et le choix dépendra du problème spécifique, des ressources de calcul et des compromis souhaités entre les performances et le coût de calcul.

En somme, Grid Search est une méthode simple et directe pour le réglage des hyperparamètres qui implique de spécifier une plage de valeurs possibles pour chaque hyperparamètre, puis de former et d'évaluer un modèle pour chaque combinaison de valeurs d'hyperparamètres. Il est moins coûteux en calcul mais ses résultats dépendent de la grille spécifiée et il peut être moins efficace pour trouver l'optimum global. La recherche aléatoire est similaire à la recherche sur grille, mais elle échantillonne de manière aléatoire l'espace des hyperparamètres, ce qui peut la rendre plus efficace que la recherche sur grille. L'algorithme génétique est une technique d'optimisation plus sophistiquée et flexible basée sur les principes de l'évolution naturelle. Il peut être plus efficace que la recherche par grille et la recherche aléatoire pour trouver l'optimum global, en particulier lorsque l'espace de recherche est vaste

et complexe. Cependant, cela peut être plus coûteux en calcul et le choix de la fonction de fitness et des opérateurs évolutifs peut avoir un impact significatif sur les performances de l'algorithme. *Green Wolf Optimizer (GWO)* et *Differential evolution (DE)* sont deux techniques d'optimisation qui ont été proposées récemment et qui ont montré des résultats prometteurs dans différents problèmes d'optimisation. Cependant, ils ne sont pas aussi largement utilisés et étudiés que *Grid Search, Random Search* et *Genetic Algorithm*.

Pour finir, la recherche de grille est un moyen simple, efficace et moins coûteux en calcul de régler les hyperparamètres, mais ses résultats dépendent de la grille spécifiée et il peut être moins efficace pour trouver l'optimum global. Alors que l'algorithme génétique est une technique d'optimisation plus sophistiquée et flexible qui peut être plus efficace pour trouver l'optimum global, mais il ne garantit pas et le choix de la fonction de fitness et des opérateurs évolutifs peut avoir un impact significatif sur les performances de l'algorithme. Les 2 autres métaheuristiques sont aussi efficaces en fonctions de l’objectif visé et du problème.

BIBLIOGRAPHIE OU LISTE DE RÉFÉRENCES

[1] B. Bilalli, A. Abelló, T. Aluja-Banet, et R. Wrembel, "PRESISTANT: Learning based assistant for data pre-processing," *Data & Knowledge Engineering*, vol. 123, p. 101727, 2019.

[2] D. T. Larose, *Data mining and predictive analytics*: John Wiley & Sons, 2015.

[3] C. Blier-Wong, H. Cossette, L. Lamontagne, et E. Marceau, "Machine Learning in P&C Insurance: A Review for Pricing and Reserving," *Risks*, vol. 9, no. 1, p. 4, 2021/01// 2021.

[4] S. Gu, R. Cheng, et Y. Jin, "Feature selection for high-dimensional classification using a competitive swarm optimizer," *Soft Computing*, vol. 22, no. 3, pp. 811-822, 2018/02/01/ 2018.

[5] I. Boussaid, "Perfectionnement de métaheuristiques pour l'optimisation continue," Paris Est, 2013.

[6] J.-K. Hao, P. Galinier, et M. Habib, "Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes," *Revue d'intelligence artificielle*, vol. 13, no. 2, pp. 283-324, 1999.

[7] K. K. Ghosh, S. Ahmed, P. K. Singh, Z. W. Geem, et R. Sarkar, "Improved Binary Sailfish Optimizer Based on Adaptive β-Hill Climbing for Feature Selection," *IEEE Access*, vol. 8, pp. 83548-83560, 2020 2020.

[8] *UCI Machine Learning Repository: Data Sets*. [En ligne]. Disponible: https://archive.ics.uci.edu/ml/datasets.php
files/134/datasets.html

[9] G. Sahebi, P. Movahedi, M. Ebrahimi, T. Pahikkala, J. Plosila, et H. Tenhunen, "GeFeS: A generalized wrapper feature selection approach for optimizing classification performance," *Computers in Biology and Medicine*, vol. 125, p. 103974, 2020/10/01/ 2020.

[10] Q. Al-Tashi, H. Md Rais, S. J. Abdulkadir, S. Mirjalili, et H. Alhussian, "A Review of Grey Wolf Optimizer-Based Feature Selection Methods for Classification," dans *Evolutionary Machine Learning Techniques: Algorithms and Applications*, S. Mirjalili, H. Faris, et I. Aljarah, éds., Singapore: Springer, 2020, pp. 273-286.

[11] H. Faris, M. A. Hassonah, A. M. Al-Zoubi, S. Mirjalili, et I. Aljarah, "A multi-verse optimizer approach for feature selection and optimizing SVM parameters based on a robust system architecture," *Neural Computing and Applications*, vol. 30, no. 8, pp. 2355-2369, 2018/10/01/ 2018.

[12] U. Hassan 1st, Y. Khourdifi, M. Bahaj, et U. Hassan 1st, "Heart Disease Prediction and Classification Using Machine Learning Algorithms Optimized by Particle Swarm Optimization and Ant Colony Optimization," *International Journal of Intelligent Engineering and Systems*, vol. 12, no. 1, pp. 242-252, 2019/02/28/ 2019.

[13] Xenonstack, "Data Preparation Process, Preprocessing and Data Wrangling," *Medium*, 2020/08/18/T13:31:59.075Z 2020.

[14] B. Suthar, H. Patel, et A. Goswami, "A survey: classification of imputation methods in data mining," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 1, pp. 309-12, 2012.

[15] R. Houari, A. Bounceur, A. K. Tari, et M. T. Kecha, "Handling missing data problems with sampling methods," Dans *2014 International conference on advanced networking distributed systems and applications*, 2014, pp. 99-104: IEEE.

[16] O. F. Ayilara, L. Zhang, T. T. Sajobi, R. Sawatzky, E. Bohm, et L. M. Lix, "Impact of missing data on bias and precision when estimating change in patient-reported outcomes from a clinical registry," *Health and quality of life outcomes*, vol. 17, no. 1, pp. 1-9, 2019.

[17] J. Ludbrook, "Outlying observations and missing values: how should they be handled?," *Clinical and experimental pharmacology & physiology*, vol. 35, no. 5-6, pp. 670-678, 2008.

[18] Z. Zhang, "Missing values in big data research: some basic skills," *Annals of translational medicine*, vol. 3, no. 21 2015.

[19] D. L. Langkamp, A. Lehman, et S. Lemeshow, "Techniques for handling missing data in secondary analyses of large surveys," *Academic pediatrics*, vol. 10, no. 3, pp. 205-210, 2010.

[20] A. R. T. Donders, G. J. Van Der Heijden, T. Stijnen, et K. G. Moons, "A gentle introduction to imputation of missing values," *Journal of clinical epidemiology*, vol. 59, no. 10, pp. 1087-1091, 2006.

[21] J. W. Graham, "Missing data analysis: Making it work in the real world," *Annual review of psychology*, vol. 60, no. 1, pp. 549-576, 2009.

[22] A. N. Baraldi et C. K. Enders, "An introduction to modern missing data analyses," *Journal of school psychology*, vol. 48, no. 1, pp. 5-37, 2010.

[23] I. B. Aydilek et A. Arslan, "A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm," *Information Sciences*, vol. 233, pp. 25-35, 2013.

[24] J. Lin, N. Li, M. A. Alam, et Y. Ma, "Data-driven missing data imputation in cluster monitoring system based on deep neural network," *Applied Intelligence*, vol. 50, no. 3, pp. 860-877, 2020.

[25] B. Al-Helali, Q. Chen, B. Xue, et M. Zhang, "A new imputation method based on genetic programming and weighted KNN for symbolic regression with incomplete data," *Soft Computing*, vol. 25, no. 8, pp. 5993-6012, 2021.

[26] J. Qiu, Q. Wu, G. Ding, Y. Xu, et S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, pp. 1-16, 2016.

[27] J. M. Jerez *et al.*, "Missing data imputation using statistical and machine learning methods in a real breast cancer problem," *Artificial intelligence in medicine*, vol. 50, no. 2, pp. 105-115, 2010.

[28] Q. Song et M. Shepperd, "Missing data imputation techniques," *International journal of business intelligence and data mining*, vol. 2, no. 3, pp. 261-291, 2007.

[29] L. W. Ben Sherwood, Xiao-Hua Zhou, "Weighted quantile regression for analyzing health care cost data with missing covariates - Sherwood - 2013 - Statistics in Medicine - Wiley Online Library," 2013.

[30] V. Vapnik, S. Golowich, et A. Smola, "Support Vector Method for Function Approximation, Regression Estimation and Signal Processing," 1996, vol. 9: MIT Press.

[31] T. Siswantining, S. M. Soemartojo, et D. Sarwinda, "Application of sequential regression multivariate imputation method on multivariate normal missing data," Dans *2019 3rd International Conference on Informatics and Computational Sciences (ICICoS)*, 2019, pp. 1-6: IEEE.

[32] E. S. Nordholt, "Imputation: Methods, Simulation Experiments and Practical Examples," *International Statistical Review*, vol. 66, no. 2, pp. 157-180, 1998 1998.

[33] L. H. Rubin, K. Witkiewitz, J. S. Andre, et S. Reilly, "Methods for handling missing data in the behavioral neurosciences: Don't throw the baby rat out with the bath water," *Journal of Undergraduate Neuroscience Education*, vol. 5, no. 2, p. A71, 2007.

[34] L. Uusitalo, A. Lehikoinen, I. Helle, et K. Myrberg, "An overview of methods to evaluate uncertainty of deterministic models in decision support," *Environmental Modelling & Software*, vol. 63, pp. 24-31, 2015.

[35] Y. Zhao et Q. Long, "Multiple imputation in the presence of high-dimensional data," *Statistical Methods in Medical Research*, vol. 25, no. 5, pp. 2021-2035, 2016.

[36] M. H. Huque, J. B. Carlin, J. A. Simpson, et K. J. Lee, "A comparison of multiple imputation methods for missing data in longitudinal studies," *BMC medical research methodology*, vol. 18, no. 1, pp. 1-16, 2018.

[37] N. J. Horton, S. R. Lipsitz, et M. Parzen, "A potential for bias when rounding in multiple imputation," *The American Statistician*, vol. 57, no. 4, pp. 229-232, 2003.

[38] S. Greenland et W. D. Finkle, "A critical look at methods for handling missing covariates in epidemiologic regression analyses," *American journal of epidemiology*, vol. 142, no. 12, pp. 1255-1264, 1995.

[39] W. Vach, *Logistic regression with missing values in the covariates*, vol. 86: Springer Science & Business Media, 2012.

[40] D. B. Rubin, *Multiple imputation for nonresponse in surveys*, vol. 81: John Wiley & Sons, 2004.

[41] J. Maillo, S. Ramírez, I. Triguero, et F. Herrera, "kNN-IS: An Iterative Spark-based design of the k-Nearest Neighbors classifier for big data," *Knowledge-Based Systems*, vol. 117, pp. 3-15, 2017.

[42] A. Amirteimoori et S. Kordrostami, "A Euclidean distance-based measure of efficiency in data envelopment analysis," *Optimization*, vol. 59, no. 7, pp. 985-996, 2010.

[43] M. Gimpy, "Missing value imputation in multi attribute data set," *Int J Comput Sci Inf Technol*, vol. 5, no. 4, pp. 1-7, 2014.

[44] E. Acuna et C. Rodriguez, "The treatment of missing values and its effect on classifier accuracy," dans *Classification, clustering, and data mining applications*: Springer, 2004, pp. 639-647.

[45] C. Jiang et Z. Yang, "CKNNI: an improved knn-based missing value handling technique," Dans *International Conference on Intelligent Computing*, 2015, pp. 441-452: Springer.

[46] B. Sun, L. Ma, W. Cheng, W. Wen, P. Goswami, et G. Bai, "An improved k-nearest neighbours method for traffic time series imputation," Dans *2017 Chinese Automation Congress (CAC)*, 2017, pp. 7346-7351: IEEE.

[47] Y. He et D. Pi, "Improving KNN method based on reduced relational grade for microarray missing values imputation," *IAENG International Journal of Computer Science*, vol. 43, no. 3, pp. 356-362, 2016.

[48] M. Zhu et X. Cheng, "Iterative KNN imputation based on GRA for missing values in TPLMS," Dans *2015 4th international conference on computer science and network technology (ICCSNT)*, 2015, vol. 1, pp. 94-99: IEEE.

[49] J. Huang *et al.*, "Cross-validation based K nearest neighbor imputation for software quality datasets: an empirical study," *Journal of Systems and Software*, vol. 132, pp. 226-252, 2017.

[50] T. Emmanuel, T. Maupong, D. Mpoeleng, T. Semong, B. Mphago, et O. Tabona, "A survey on missing data in machine learning," *Journal of Big Data*, vol. 8, no. 1, p. 140, 2021/10/27/ 2021.

[51] G. E. Batista et M. C. Monard, "An analysis of four missing data treatment methods for supervised learning," *Applied artificial intelligence*, vol. 17, no. 5-6, pp. 519-533, 2003.

[52] S. Zhang, X. Li, M. Zong, X. Zhu, et D. Cheng, "Learning k for knn classification," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 3, pp. 1-19, 2017.

[53] H. De Silva et A. S. Perera, "Missing data imputation using Evolutionary k-Nearest neighbor algorithm for gene expression data," Dans *2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, 2016, pp. 141-146: IEEE.

[54] C. Eftimoska, "Data preprocessing for Machine Learning in Python," *Medium*, 2020/05/04/T09:09:48.208Z 2020.

[55] Y. Verma. (2021). *A Complete Guide to Categorical Data Encoding*. [En ligne]. Disponible: https://analyticsindiamag.com/a-complete-guide-to-categorical-data-encoding/
files/180/a-complete-guide-to-categorical-data-encoding.html

[56] M. Moran, "Dummy Coding: The how and why," *Statistics Solutions*, 2017/05/31/T20:13:38+00:00 2017.

[57] B. HB, "Dummy Variables in Regression," 2022.

[58] A. Bhandari. (2020). Feature Scaling | Standardization Vs Normalization. Dans *Analytics Vidhya*. [En ligne]. Disponible: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/
files/343/feature-scaling-machine-learning-normalization-standardization.html

[59] D. Team. (2021). Resampling Methods in Machine LearningResampling Methods in Machine Learning. Dans. [En ligne]. Disponible: https://datamites.com/blog/resampling-methods-in-machine-learning/
files/298/resampling-methods-in-machine-learning.html

[60] M. Schumacher, N. Holländer, et W. Sauerbrei, "Resampling and cross-validation techniques: a tool to reduce bias caused by model building?," *Statistics in Medicine*, vol. 16, no. 24, pp. 2813-2827, 1997 1997.

[61] F.-J. Lin, "Solving Multicollinearity in the Process of Fitting Regression Model Using the Nested Estimate Procedure | SpringerLink," 2008.

[62] M. Madaan. (2022). Sélection de fonctionnalités : Tutoriel pour débutants - Naukri Learning. Dans. Consulté le 2022/11/03/19:50:07. [En ligne]. Disponible: https://www.naukri.com/learning/articles/feature-selection-beginners-tutorial/
files/272/feature-selection-beginners-tutorial.html

[63] M. Cherrington, F. Thabtah, J. Lu, et Q. Xu, "Feature Selection: Filter Methods Performance Challenges," Dans *2019 International Conference on Computer and Information Sciences (ICCIS)*, 2019, pp. 1-4.

[64] D. Aksu, S. Üstebay, M. A. Aydin, et T. Atmaca, "Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm," Dans *International symposium on computer and information sciences*, 2018, pp. 141-149: Springer.

[65] J. M. Sutter et J. H. Kalivas, "Comparison of forward selection, backward elimination, and generalized simulated annealing for variable selection," *Microchemical journal*, vol. 47, no. 1-2, pp. 60-66, 1993.

[66] N. Tyagi, "L2 vs L1 Regularization in Machine Learning | Ridge and Lasso Regularization,"

[67] Y.-Y. Nguwi et S.-Y. Cho, "An unsupervised self-organizing learning with support vector ranking for imbalanced datasets," *Expert Systems with Applications*, vol. 37, no. 12, pp. 8303-8312, 2010.

[68] P. K. Chan et S. J. Stolfo, "Learning with non-uniform class and cost distributions: Effects and a distributed multi-classifier approach," Dans *In Workshop Notes KDD-98 Workshop on Distributed Data Mining*, 1998: Citeseer.

[69] M. Kubat, R. C. Holte, et S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine learning*, vol. 30, no. 2, pp. 195-215, 1998.

[70] C. Blake, "UCI repository of machine learning databases," *http://www. ics. uci. edu/~ mlearn/MLRepository. html*, 1998.

[71] N. Japkowicz, "Learning from imbalanced data sets: a comparison of various strategies," Dans *AAAI workshop on learning from imbalanced data sets*, 2000, vol. 68, pp. 10-15: AAAI Press Menlo Park, CA.

[72] N. Chawla, N. Japkowicz, et A. Kolcz, "ICML'2003 Workshop on Learning from Imbalanced Data Sets (II), 2003," Dans *Proceedings available at http://www. site. uottawa. ca/~ nat/Workshop2003/workshop2003. html*, 2000.

[73] N. Chawla et Z. Zhou, "Data mining when classes are imbalanced and errors have costs," Dans *Workshop, 13th Pacific-Asia conference on knowledge discovery and data mining*, 2009.

[74] S. Mishra, P. K. Mallick, L. Jena, et G.-S. Chae, "Optimization of Skewed Data Using Sampling-Based Preprocessing Approach," *Frontiers in Public Health*, vol. 8, 2020 2020.

[75] (22/05/2022). *Le secteur de l'assurance*. [En ligne]. Disponible: https://www.assurance-et-mutuelle.com/assurance/secteur-assurance.html
files/150/secteur-assurance.html

[76] J. Perols, "Financial statement fraud detection: An analysis of statistical and machine learning algorithms," *Auditing: A Journal of Practice & Theory*, vol. 30, no. 2, pp. 19-50, 2011.

[77] J. L. Perols, R. M. Bowen, C. Zimmermann, et B. Samba, "Finding needles in a haystack: Using data analytics to improve fraud prediction," *The Accounting Review*, vol. 92, no. 2, pp. 221-245, 2017.

[78] Y. Bao, B. Ke, B. Li, Y. J. Yu, et J. Zhang, "Detecting accounting fraud in publicly traded US firms using a machine learning approach," *Journal of Accounting Research*, vol. 58, no. 1, pp. 199-235, 2020.

[79] J. Bertomeu, E. Cheynel, E. Floyd, et W. Pan, "Using machine learning to detect misstatements," *Review of Accounting Studies*, vol. 26, no. 2, pp. 468-519, 2021.

[80] A. Gramegna et P. Giudici, "Why to Buy Insurance? An Explainable Artificial Intelligence Approach," *Risks*, vol. 8, no. 4, p. 137, 2020.

[81] H. Paruchuri, "The Impact of Machine Learning on the Future of Insurance Industry," *American Journal of Trade and Policy*, vol. 7, no. 3, pp. 85-90, 2020.

[82] C. Dugas, Y. Bengio, N. Chapados, P. Vincent, G. Denoncourt, et C. Fournier, "Statistical learning algorithms applied to automobile insurance ratemaking," Dans *CAS Forum*, 2003, vol. 1, no. 1, pp. 179-214: Citeseer.

[83] A. Diana, J. E. Griffin, J. S. Oberoi, et J. Yao, "Machine-Learning Methods for Insurance Applications-A Survey," 2019.

[84] S. Lee et K. Antonio, "Why high dimensional modeling in actuarial science?," Dans *Paper presented*, 2015.

[85] Y. LeCun, Y. Bengio, et G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[86] M. Denuit et J. Trufin, "Effective statistical learning methods for actuaries," 2019.

[87] B. So, J.-P. Boucher, et E. A. Valdez, "Cost-sensitive multi-class adaboost for understanding driving behavior with telematics," *arXiv preprint arXiv:2007.03100*, 2020.

[88] L. Guelman, "Gradient boosting trees for auto insurance loss cost modeling and prediction," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3659-3667, 2012/02/15/ 2012.

[89] Z. Quan et E. A. Valdez, "Predictive analytics of insurance claims using multivariate decision trees," *Dependence Modeling*, vol. 6, no. 1, pp. 377-407, 2018.

[90] N. Chapados *et al.*, "Estimating car insurance premia: A case study in high-dimensional data inference," *Advances in Neural Information Processing Systems*, vol. 14, 2001.

[91] K. Sakthivel et C. Rajitha, "Artificial intelligence for estimation of future claim frequency in non-life insurance," *Global Journal of Pure and Applied Mathematics*, vol. 13, no. 6, pp. 1701-1710, 2017.

[92] M. V. Wüthrich et M. Merz, "Yes, we CANN!," *ASTIN Bulletin: The Journal of the IAA*, vol. 49, no. 1, pp. 1-3, 2019.

[93] R. Henckaerts, M.-P. Côté, K. Antonio, et R. Verbelen, "Boosting insights in insurance tariff plans with tree-based machine learning methods," *North American Actuarial Journal*, vol. 25, no. 2, pp. 255-285, 2021.

[94] L. Diao et C. Weng, "Regression tree credibility model," *North American Actuarial Journal*, vol. 23, no. 2, pp. 169-196, 2019.

[95] S. C. Lee et S. Lin, "Delta boosting machine with application to general insurance," *North American Actuarial Journal*, vol. 22, no. 3, pp. 405-425, 2018.

[96] A. Paglia et M. V. Phelippe-Guinvarc'H, "TARIFICATION DES RISQUES EN ASSURANCE NON-VIE, UNE APPROCHE PAR MODELE D'APPRENTISSAGE STATISTIQUE," p. 34, 2011.

[97] M. V. Wüthrich, "Covariate selection from telematics car driving data," *European Actuarial Journal*, vol. 7, no. 1, pp. 89-108, 2017/07/01/ 2017.

[98] G. Gao et M. V. Wüthrich, "Feature extraction from telematics car driving heatmaps," *European Actuarial Journal*, vol. 8, no. 2, pp. 383-406, 2018/12/01/ 2018.

[99] K. Ding, B. Lev, X. Peng, T. Sun, et M. A. Vasarhelyi, "Machine learning improves accounting estimates: Evidence from insurance payments," *Review of accounting studies*, vol. 25, no. 3, pp. 1098-1134, 2020.

[100] A. Joseph, "Shapley regressions: A framework for statistical inference on machine learning models," 2019.

[101] S. Vadlamudi, "What Impact does Internet of Things have on Project Management in Project based Firms?," *Asian Business Review*, vol. 6, no. 3, pp. 179-186, 2016.

[102] K. MISHR, *Fundamentals of life insurance theories and applications*: PHI Learning Pvt. Ltd., 2016.

[103] Y. Yang, W. Qian, et H. Zou, "Insurance premium prediction via gradient tree-boosted Tweedie compound Poisson models," *Journal of Business & Economic Statistics*, vol. 36, no. 3, pp. 456-470, 2018.

[104] J. Schelldorfer et M. V. Wuthrich, "Nesting classical actuarial models into neural networks," *Available at SSRN 3320525*, 2019.

[105] P. Regulation, "Regulation (EU) 2016/679 of the European Parliament and of the Council," *Regulation (eu)*, vol. 679, p. 2016, 2016.

[106] M. Kaminski, "The right to explanation, explained. University of Colorado Law Legal Studies Research Paper No 18-24," *Berkeley Technology Law J*, vol. 34, 2018.

[107] A. Ferrario, A. Noll, et M. V. Wuthrich, "Insights from Inside Neural Networks." Rochester, NY, 2020.

[108] T. Maynard, *WHAT ROLE FOR AI IN INSURANCE PRICING? A PREPRINT*, 2019.

[109] M. V. Wuthrich et C. Buser, "Data Analytics for Non-Life Insurance Pricing." Rochester, NY, 2021.

[110] V. Kašćelan, L. Kašćelan, et M. Novović Burić, "A nonparametric data mining approach for risk prediction in car insurance: a case study from the Montenegrin market," *Economic research-Ekonomska istraživanja*, vol. 29, no. 1, pp. 545-558, 2016.

[111] M. Corlosquet-Habart et J. Janssen, *Big data for insurance companies*: John Wiley & Sons, 2018.

[112] Y.-L. Grize, W. Fischer, et C. Lützelschwab, "Machine learning applications in nonlife insurance," *Applied Stochastic Models in Business and Industry*, vol. 36, no. 4, pp. 523-537, 2020 2020.

[113] S. Jamal *et al.*, "Machine Learning & Traditional Methods Synergy in Non-Life Reserving," *Report of the ASTIN Working Party of the International Actuarial Association. Available online: https://www. actuaries. org/IAA/Documents/ASTIN/ASTIN_MLTMS% 20Report_SJAMAL. pdf (accessed on 19 July 2019)*, 2018.

[114] A. Panlilio, B. Canagaretna, S. Perkins, V. du Preez, et Z. Lim, "Practical application of machine learning within actuarial work," *London: Institute and Faculty of Actuaries*, 2018.

[115] R. Richman, "AI in actuarial science–a review of recent advances–part 1," *Annals of Actuarial Science*, vol. 15, no. 2, pp. 207-229, 2021.

[116] R. Richman, "AI in actuarial science–a review of recent advances–part 2," *Annals of Actuarial Science*, vol. 15, no. 2, pp. 230-258, 2021.

[117] B. Harej, R. Gächter, et S. Jamal, "Individual claim development with machine learning," *Report of the ASTIN Working Party of the International Actuarial Association. Available online: http://www. actuaries. org/ASTIN/Documents/ASTIN_ICDML_WP_Report_final. pdf (accessed on 19 July 2019)*, 2017.

[118] N. v. R. Ronald Richman, Mario V. Wuthrich, "Believing the Bot - Model Risk in the Era of Deep Learning by Ronald Richman, Nicolai von Rummell, Mario V. Wuthrich :: SSRN," 2019.

[119] A. Paglia et M. V. Phelippe-Guinvarc'h, "Tarification des risques en assurance non-vie, une approche par modèle d'apprentissage statistique," *Bulletin français d'Actuariat*, vol. 11, no. 22, pp. 49-81, 2011.

[120] J. Brownlee. (2020). 4 Types of Classification Tasks in Machine Learning. Dans *Machine Learning Mastery*. [En ligne]. Disponible: https://machinelearningmastery.com/types-of-classification-in-machine-learning/
files/196/types-of-classification-in-machine-learning.html

[121] A. Sharma. (2020). Decision Tree vs. Random Forest - Which Algorithm Should you Use? Dans *Analytics Vidhya*. [En ligne]. Disponible: https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/

files/202/decision-tree-vs-random-forest-algorithm.html

[122] T. K. Ho, "Random decision forests," Dans *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 1995, vol. 1, pp. 278-282 vol.1.

[123] P. Latinne, O. Debeir, et C. Decaestecker, "Limiting the number of trees in random forests," 2001, pp. 178-187: Springer.

[124] S. Bernard, S. Adam, et L. Heutte, "Using random forests for handwritten digit recognition," 2007, vol. 2, pp. 1043-1047: IEEE.

[125] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5-32, 2001 2001.

[126] A. Abraich, H. D. Nguyen, et M. Tounsi, "DÉTECTION DE FRAUDE IEEE-CIS," 2020 2020.

[127] M. Joseph. (2020). The Gradient Boosters V: CatBoost. Dans *Deep & Shallow*. [En ligne]. Disponible: https://deep-and-shallow.com/2020/02/29/the-gradient-boosters-v-catboost/
files/215/the-gradient-boosters-v-catboost.html

[128] P. Liashchynskyi et P. Liashchynskyi, "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS," arXiv, 2019.

[129] S. Loussaief et A. Abdelkrim, "Convolutional neural network hyper-parameters optimization based on genetic algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 10 2018 2018.

[130] J.-K. Hao et C. Solnon, *Méta-heuristiques et intelligence artificielle*, 2014.

[131] N. S. Chauhan. (2020). Hyperparameter Optimization for Machine Learning Models. Dans *KDnuggets*. [En ligne]. Disponible: https://www.kdnuggets.com/hyperparameter-optimization-for-machine-learning-models.html

[132] J. Dreo, A. Petrowski, P. Siarry, et E. Taillard, *Métaheuristiques pour l'optimisation difficile*: EYROLLES, 2003.

[133] (2021). Grey wolf optimization - Introduction. Dans *GeeksforGeeks*. [En ligne]. Disponible:https://www.geeksforgeeks.org/grey-wolf-optimization-introduction/
files/264/grey-wolf-optimization-introduction.html

[134] J.-K. Hao, "Algorithmes génétiques et autres outils d'optimisation appliqués à la gestion du trafic aérien," p. 19,

[135] M. J. N. E. Correspondant, "Algorithmes génétiques et autres outils d'optimisation appliqués à la gestion du trafic aérien," Université d'Angers, 2004.

[136] D. E. Goldberg, *Real-coded genetic algorithms, virtual alphabets and blocking*: Citeseer, 1990.

[137] Z. Michalewicz et C. Z. Janikow, "Handling constraints in genetic algorithms," Dans *Icga*, 1991, pp. 151-157.

[138] W. M. Spears et K. A. De Jong, "An analysis of multi-point crossover," dans *Foundations of genetic algorithms*, vol. 1: Elsevier, 1991, pp. 301-315.

[139] D. E. Golberg, "Genetic algorithms in search, optimization, and machine learning," *Addion wesley*, vol. 1989, no. 102, p. 36, 1989.

[140] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, et R. Ranjan, "A hybrid deep learning-based model for anomaly detection in cloud datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 924-935, 2019.

[141] N. Kumaran, A. Vadivel, et S. S. Kumar, "Recognition of human actions using CNN-GWO: a novel modeling of CNN for enhancement of classification performance," *Multimedia Tools and Applications*, vol. 77, no. 18, pp. 23115-23147, 2018.

[142] R. Mohakud et R. Dash, "Designing a grey wolf optimization based hyper-parameter optimized convolutional neural network classifier for skin cancer detection," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, Part B, pp. 6280-6291, 2022/09/01/ 2022.

[143] C. Muro, R. Escobedo, L. Spector, et R. P. Coppinger, "Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations," *Behavioural processes*, vol. 88, no. 3, pp. 192-197, 2011 2011.

[144] R. Storn et K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341-359, 1997 1997.

[145] S. Das, S. S. Mullick, et P. N. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm and evolutionary computation*, vol. 27, pp. 1-30, 2016.

[146] A. W. Mohamed, A. A. Hadi, et K. M. Jambi, "Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization," *Swarm and Evolutionary Computation*, vol. 50, p. 100455, 2019.

[147] *Couverture d'assurance - Qu'est-ce que c'est, définition et concept - 2021 - Économie-Wiki.com*. [En ligne]. Disponible: https://economy-pedia.com/11030193-insurance-coverage
files/175/11030193-insurance-coverage.html

[148] A. Choudhury et M. R. Kosorok, "Missing data imputation for classification problems," *arXiv preprint arXiv:2002.10709*, 2020 2020.

[149] (2019). *Pourquoi Python est populaire en Data Science ?* [En ligne]. Disponible: https://analyticsinsights.io/pourquoi-python-est-populaire-en-data-science/
files/189/pourquoi-python-est-populaire-en-data-science.html

[150] L. Tung, "Python reste le langage le plus utilisé pour la data science, suivi de SQL," *ZDNet France*, 2021.

[151] KDnuggets. (2019). Python leads the 11 top Data Science, Machine Learning platforms: Trends and Analysis. Dans *KDnuggets*. [En ligne]. Disponible: https://www.kdnuggets.com/python-leads-the-11-top-data-science-machine-learning-platforms-trends-and-analysis.html
https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html

ANNEXE 1

CERTIFICATION ÉTHIQUE

Ce mémoire a fait l'objet d'une certification éthique. Le numéro du certificat est 2022-1031.

ANNEXE 1

DESCRIPTIONS DES COLONNES DES DONNEES

Tableau 33: Description des colonnes - suites

Variables	Anonymisé	Avec	Description
NEW_EXP_CNT_DATE	Non		Date à laquelle l'exposition a été comptabilisée comme une nouvelle exposition
TRANSACTIONDATE	Non		Date de la transaction
COVERAGETYPE_L_EN_CA	Non		Répartition de la couverture qui s'applique à la réclamation. Il peut y avoir plusieurs types de couverture pour un même sinistre.
FAULTRATING_L_EN_CA	Non		Indique si l'assuré est responsable de l'accident, et le pourcentage de responsabilité attribué.
FAULT	Non		Indique si l'assuré est responsable de l'accident, et la valeur de la faute attribuée.
LOSS_LOC_PROV_M	Oui	Catégoriser	Province ou État où le sinistre s'est produit. Notez qu'il n'y a pas de validation dans le centre de réclamation entre la province et le pays, il peut donc y avoir une situation telle qu'une province étant liée à un pays de lieu de perte des États-Unis"
LOSS_LOC_CITY_M	Oui	Fonction	Ville où le sinistre s'est produit
VIN	Oui	Fonction	Numéro VIN du véhicule
ANNUALMILEAGE_CG	Non		Kilométrage annuel du véhicule
VEHICLEUSE_L_EN_CA	Oui	Catégoriser	Type d'utilisation du véhicule
BODILYINJURYTYPE_L_EN_CA	OUI	Catégoriser	Il s'agit d'une classification des lésions corporelles qui fournit plus de détails sur la blessure.
ABQUALIFICATION_L_EN_CA	Oui	Catégoriser	Situation professionnelle du demandeur
GENDERTYPE_L_EN_CA	Oui	Catégoriser	Identifie le sexe du demandeur uniquement si le demandeur est une personne. Si le demandeur est une organisation, la

Variables	Anonymisé	Avec	Description
			valeur de l'attribut est “Not available”.
YEAROFBIRTH	Oui	Catégoriser	Année de naissance du demandeur seulement si le demandeur est une personne. Lorsque le demandeur est une organisation, la valeur de l'attribut est vide.
TREATMENTPROTOCOL_L_EN_CA	Non		Protocole du traitement suivi : AAIB = Prestations d'assurance-accidents de l'Alberta DPTR = Règlement sur les protocoles de diagnostic et de traitement MI = Ligne directrice sur les blessures mineures Non-IM = Ligne directrice sur les blessures non mineures Non-PAF = cadre non préapprouvé PAF = cadre préapprouvé
PAYMENT	Non		La somme des montants payés pour couvrir les pertes et / ou les dépenses pour un sinistre spécifique. Le paiement direct de base est la somme de : Des montants versés à l'assuré, Les montants versés à un fournisseur qui fournit des

Variables	Anonymisé	Avec	Description
			services ou des réparations à l'assuré. Les montants versés à un tiers ou pour le compte d'un assuré responsable. Les frais de règlement des sinistres qui sont assignables ou attribuables à des sinistres spécifiques. Les honoraires versés aux avocats, experts et enquêteurs extérieurs utilisés pour défendre les sinistres.
RECOVERY	Non		La somme des montants reçus par l'assureur qui réduit la responsabilité d'un sinistre, provenant de sources telles que la subrogation, le sauvetage et le recouvrement des dépenses.
ADJUSTER_EXPENSE	Non		Combien d'argent a été engagé pour les frais d'ajustement (autres que juridiques) ?
LEGAL_EXPENSE	Non		Le montant payé pour le type de coût Dépense - Juridique
RESERVE	Non		Une réserve est un montant qui représente un passif potentiel. Cet indicateur mesure la variation, pendant une période donnée, de la valeur estimée des sinistres non encore payés. Ce changement est le résultat d'une transaction financière sur une réclamation qui représente un montant de réserve ou de paiement. Exceptionnellement, les paiements qui n'érodent pas les réserves n'auront aucun impact sur la mesure de la variation des réserves.