Techniques alternatives de modélisation de la probabilité de défaut pour une banque de détail

Janvier 2019

Romane Le Tallec

Mémoire d'actuariat

Résumé

L'un des grands enjeux d'une banque de détail est l'évaluation des risques liés à ses activités. Une bonne maîtrise de ces risques doit permettre à la banque de mieux faire face à un choc économique, réglementaire, ... mais assure aussi la pérennité de ses activités et la bonne marche de l'économie, les banques étant l'un de ses acteurs majeurs. Le principal risque auquel est soumise une banque de détail telle que La Banque Postale est le risque de crédit et, plus particulièrement, le risque de défaut de ses clients. Il existe plusieurs façons d'évaluer ce risque et elles dépendent à la fois du contexte réglementaire et de la nature du portefeuille. La mesure qui nous a intéressées ici est la probabilité de défaut (PD) sous plusieurs formes et périmètres (à l'octroi, bâloise et comptable).

La modélisation d'un tel indicateur dépend fortement des caractéristiques du portefeuille étudié, et notamment :

— de sa volumétrie ;
— du critère à évaluer (le défaut) : en effet, sur certains portefeuilles, nous n'observons (presque) aucun défaut (i.e. la donnée que nous cherchons à modéliser n'existe pas en nombre suffisant pour la prévoir) ;
— de la qualité et de la disponibilité des variables explicatives du phénomène (le défaut) : par exemple, à l'entrée en relation, nous possédons très peu de données sur le client.

En fonction de ces caractéristiques, il n'est pas toujours possible d'appliquer une méthodologie classique (telle qu'une régression logistique sur les données internes). Il est alors nécessaire de développer et de tester des outils ou approches alternatives. L'objet de ce document est d'explorer différentes approches d'estimation de la PD, telles que l'optimisation par essaim particulaire, l'utilisation d'un algorithme génétique ou l'ajout de données externes, sur les principaux portefeuilles de La Banque Postale.

Abstract

The risk assessment of a retail bank's activities is one of the major issue it faces. The better a bank understands its risks, the better it can face an economic or regulatory stress but also, a good knowledge of its risks ensure the permanence of its activities and the proper functioning of the economy of which banks are one of the key players. The main risk a retail bank such as *La Banque Postale* faces is credit risk, and more specifically, default risk of its customers. There are multiple ways to assess this risk and they rely both on the regulatory environment and on the portfolio type. In this document, the measure which we were interested in is the probability of default under various forms and scopes (granting products, Basel II framework and accounting norms).

The modelling of such an indicator is highly linked to the characteristics of the studied portfolio, such as :

— the size of the scope ;
— the criterion we wish to evaluate (the default) : indeed, in some cases, no (or almost no) defaults have occurred (i.e. there is not sufficient information to predict the data we want to model) ;
— the quality and the availability of explanatory variables : for example, at the very beginning of a relationship with a client, there is a lack of data about him.

Depending on those characteristics, it is not always possible to apply a standard methodology (such as a logistic regression on the internal data). Therefore, we need to develop and test alternative tools or approaches. This document deals with several approaches to evaluate the probability of default, such as the particle swarm optimization, the use of a genetic algorithm or the input of additional external data, on *La Banque Postale*'s main portfolios.

Remerciements

Avant de débuter, je tiens à adresser mes remerciements à tous ceux qui, en m'apportant leur aide et leur soutien, ont contribué à ce mémoire.

Je remercie Pascal Gérard de m'avoir accueillie au sein de l'équipe Modélisation, Étude et Anticipation de la Direction des Risques de La Banque Postale ainsi que Jerôme Bastien, responsable de l'équipe Modélisation. Merci de m'avoir initié à ta passion pour les algorithmes génétiques et un grand merci à toute l'équipe DREAM d'avoir participé au bon déroulement de mon alternance et de m'avoir apporté leurs conseils et leurs connaissances sur les sujets traités et sur la banque. C'est un réel plaisir de travailler à vos cotés.

Je tiens tout particulièrement à remercier ma tutrice, Ana Popescu, qui m'a accompagnée tout au long de mon alternance et de la rédaction de ce mémoire. Sa confiance, ses conseils, sa pédagogie et sa bienveillance ont grandement contribué à la réalisation de ce mémoire. Merci pour toute l'aide et les conseils que tu m'as apporté pour la rédaction de ce mémoire autant que pour ma vie professionnelle.

Enfin, merci à Lena et Antoine pour le temps qu'ils ont accordé à la relecture de ces pages. Vos corrections et votre regard extérieur m'ont été d'une grande aide.

Sommaire

Liste des abréviations

AG	Algorithme Génétique
IFRS	*International Financial Reporting Standards*
INSEE	Institut National de Statistiques et d'Études Économiques
IRIS	Îlots Regroupés pour l'Information Statistique
LBP	La Banque Postale
LDP	*Low Default Portfolio*
PD	Probabilité de Défaut
PSO	*Particle Swarm Optimization* ou Optimisation par Essaim Particulaire
RFL	Revenus Fiscaux Localisés
SPL	Secteur Public Local

Première partie

Contexte et objet du document

1 Présentation générale des travaux réalisés

Les banques de détail sont des établissements de crédit qui permettent de financer l'économie et sont extrêmement liées à son bon fonctionnement. Une banque de détail est donc un acteur économique dont les principales activités sont :

— la mise à disposition de moyens de paiement aux clients ;

— la collecte et l'administration ou le placement (épargne) des fonds déposés par les clients ;

— l'octroi de crédits (immobilier, à la consommation, ...).

Du fait de ces activités, la banque de détail est soumise à plusieurs types de risques dont les trois principaux sont :

— Le risque de crédit c'est-à-dire le risque qu'un client ne soit plus en mesure de respecter les engagements qu'il a pris envers la banque (ce qui peut par exemple entraîner le non remboursement d'un prêt ou d'un compte débiteur i.e. le défaut).

— Le risque de marché c'est-à-dire le risque lié à la fluctuation des titres financiers en portefeuille.

— Le risque opérationnel c'est-à-dire le risque lié à l'impact d'une erreur, fraude, etc. du personnel, d'une défaillance d'un système informatique, d'événements extérieurs (incendie ou inondation des locaux, ...), etc.

L'évaluation des risques liés à ses activités est nécessaire à la bonne gestion d'un établissement bancaire. Elle répond :

— aux exigences réglementaires des régulateurs (BCE [1], ACPR [2], ...) : elle permet à la banque de se conformer aux normes en vigueur (Bâle II, IFRS 9, ...) qui visent à protéger l'économie et les consommateurs ;

— aux objectifs de gestion interne : notamment, elle permet de renforcer la maîtrise du risque et le pilotage de la performance des métiers (i.e. du couple rentabilité/risques).

Dans ce document, nous allons uniquement nous intéresser à des techniques permettant de quantifier le risque de crédit. La modélisation et l'évaluation de ce risque peut différer suivant la réglementation sous laquelle nous nous plaçons (Bâle, comptable, ...) et suivant la nature du risque (risque à l'octroi, gestion et pilotage du risque, ...). Cependant, il est tout à fait possible d'utiliser la même mesure pour différents croisements entre une réglementation et la nature du risque à quantifier.

La mesure du risque que nous allons utiliser dans ce document pour modéliser le risque de crédit est la probabilité de défaut (PD). La PD est un indicateur synthétique de la probabilité d'occurrence d'un défaut pour une contrepartie donnée. Cet indicateur fait notamment partie [3] des indicateurs à modéliser pour le calcul des provisions sous IFRS 9 et sous Bâle II.

Dans la suite, nous nous sommes intéressées à deux types de probabilité de défaut :

— la PD utilisée pour le pilotage du risque et le provisionnement (notamment la PD « bâloise » et la PD « IFRS 9 » [4]) ;

1. Banque Centrale Européenne.
2. Autorité de Contrôle Prudentiel et de Résolution.
3. Avec la LGD (*Loss Given Default*) et l'EAD (*Exposure At Default*).
4. La PD utilisée pour le provisionnement n'est pas forcement la même suivant la réglementation pour laquelle elle est calculée : la notion de « défaut » peut varier d'une réglementation à l'autre ainsi que les exigences liées à l'estimation (profondeur d'historique, horizon de calcul, ...).

— la PD pour l'octroi qui permet d'attribuer des moyens de paiement aux nouveaux clients en fonction de leur risque.

L'objet de ce document est d'offrir un panorama des approches d'estimation de ces deux types de PD sur différents portefeuilles de La Banque Postale ainsi que des défis rencontrés.

L'approche générale pour quantifier le risque de crédit consiste à affecter une « note de crédit » à chaque client en fonction de sa probabilité de défaut afin d'identifier et de regrouper les clients présentant des profils de risque similaires. La problématique qui se pose est d'affecter aux contreparties les notes qui leurs correspondent le mieux. Afin de s'assurer de la bonne qualité de la probabilité de défaut calculée, la banque doit faire face à des contraintes de modélisation (réglementations, qualité des données, profondeur de l'historique, outils à disposition, suivi et mise en production des modèles statistiques, ...) qui vont fortement influencer ses choix techniques.

Modélisation de la PD « bâloise » ou « IFRS 9 »

Pour les portefeuilles avec un grand nombre de contreparties et un nombre suffisant de défauts observés (généralement, les portefeuilles *retail*[5]), il est possible d'utiliser des méthodes statistiques classiques (telles que la régression logistique, l'arbre de décision, ...) pour construire un modèle prédictif du défaut.

Pour les portefeuilles avec une plus faible volumétrie et avec très peu d'observations de défaut (généralement, les portefeuilles hors *retail*), ces méthodes statistiques ne permettent pas d'élaborer un modèle à partir des taux de défaut observés sur le portefeuille. Ces portefeuilles sont appelés *Low Default Portfolio* (LDP) car aucun défaut (ou presque) n'a pu être observé sur l'historique. Par exemple, sur le portefeuille Souverains, aucun défaut d'État n'a été historiquement observé et sur le portefeuille Établissements Financiers, un seul défaut a été historiquement observé : Lehman Brothers. Pour pallier la rareté du phénomène, d'autres approches doivent être développées afin de pouvoir modéliser une probabilité de défaut pour ces portefeuilles *Low Default*. Nous nous sommes donc tournées vers des méthodes d'optimisation algorithmiques comme les algorithmes génétiques (AG) ou l'algorithme d'optimisation par essaim particulaire (PSO). C'est le développement et l'utilisation de ces deux méthodes qui vont nous intéresser dans un premier temps.

Modélisation de la PD « à l'octroi »

Une des problématiques que nous rencontrons concerne le développement d'outils de *scoring* (i.e. notation) pour l'octroi des moyens de paiement (type de carte, montant de découvert, ...) aux nouveaux clients. Les notes attribuées aux clients dépendant principalement de leur comportement bancaire, il peut être difficile d'évaluer le comportement futur des nouveaux clients de la banque. Nous avons donc tenté d'améliorer la performance du modèle de notation à l'octroi existant à l'aide de données externes (provenant de l'INSEE) et en testant différentes méthodes (telles que la sélection de variables par algorithme génétique). Nous présenterons également les résultats de ces études dans la suite de ce document.

5. Les portefeuilles *retail* sont constitués de la clientèle de détail (notamment les particuliers). Tous les autres types de clientèle constituent le périmètre de la clientèle hors *retail* : portefeuilles Établissements Financiers, Grandes Entreprises, Secteur Public Local, etc.

2 Sujets traités dans ce document

Avant de s'attacher, dans la partie suivante, aux spécificités des techniques de modélisations utilisées, nous présentons plus en détail le contexte des applications proposées dans ce document.

2.1 Gestion du risque d'un portefeuille *Low Default*

Un des sujets qui va nous intéresser dans un premier temps concerne la modélisation de la probabilité de défaut des portefeuilles dits LDP. Il n'existe pas forcément de définition précise caractérisant un LDP mais nous nous intéressons ici aux portefeuilles hors *retail* de La Banque Postale (LBP) pour lesquels le nombre de défauts observés sur l'historique n'est pas suffisant (i.e. significatif) pour élaborer un modèle basé sur le taux de défaut. Dans cette section, nous nous intéressons aux enjeux, réglementaires mais aussi « métiers », qui nous ont amenées à construire des modèles de notation des LDP. Puis, nous présentons rapidement la méthodologie utilisée pour les deux portefeuilles qui seront traités dans ce document.

2.1.1 Enjeux métiers

La Banque Postale est une banque relativement jeune (sa création remonte à 2006) et en pleine expansion. Ses activités de financement pour la clientèle hors *retail* ont débuté à partir de 2010. Ainsi, suite à la diversification de sa clientèle, La Banque Postale a donc du développer des nouveaux outils, adaptés aux particularités des portefeuilles hors *retail*.

La modélisation de ces nouveaux outils doit respecter plusieurs contraintes. En effet, la méthodologie utilisée pour construire des modèles permettant d'évaluer le risque de ces portefeuilles doit prendre en compte :

- Les contraintes réglementaires : par exemple, dans le cadre de l'application de la réglementation IFRS 9 (qui est entrée en vigueur le 1er janvier 2018), le calcul des provisions est basé sur les probabilités de défaut associées aux notes attribuées à chaque entité du portefeuille.
- La nécessité d'audit et de contrôle interne : les résultats obtenus doivent être reproductibles.
- La facilité d'implémentation du modèle.
- La lisibilité des résultats obtenus : les résultats doivent être faciles à utiliser et à comprendre pour les utilisateurs du modèle (conseillers bancaires, ...).
- Le suivi des modèles : le modèle doit pouvoir être facilement suivi par l'équipe en charge du *backtesting*.

De plus, ces modèles doivent se baser sur des éléments statistiques tout en prenant en compte les connaissances métiers du portefeuille.

2.1.2 Méthodologie de construction

Les portefeuilles *Low Default* contiennent, en général, peu d'observations de défaut et relativement peu d'entités (en comparaison des portefeuilles *retail*) et l'approche classique de modélisation par validation, qui se base sur l'analyse du lien entre le comportement bancaire des clients et les

défauts observés sur le portefeuille, n'est pas possible. Nous utilisons donc une **approche par vérification** pour construire un modèle de notation sur ce type de portefeuilles. Cette approche est basée sur des techniques statistiques, des processus d'optimisation et l'expertise métier des analystes crédit. Elle consiste à construire un modèle qui permet de reproduire les notations données par les experts. Nous cherchons donc à optimiser la notation fournie par le modèle afin qu'il reflète au mieux le jugement des experts.

Des algorithmes d'optimisation, tels que les algorithmes génétiques [1] ou l'algorithme d'optimisation par essaim particulaire [2], ont donc été choisis par l'équipe Modélisation de LBP pour pouvoir construire des modèles de notation via cette approche. En effet, de par leur fonctionnement, ces algorithmes représentent des outils statistiques adaptés à la gestion du risque des portefeuilles LDP.

Nous avons appliqué cette méthodologie à deux portefeuilles de La Banque Postale sous deux environnements réglementaires différents.

2.1.3 Applications

Nous nous sommes tout d'abord intéressées à la modélisation de la PD du portefeuille Établissements Financiers sous un environnement bâlois. L'algorithme génétique (AG) a été utilisé pour développer ce modèle à l'aide de l'approche par vérification.

Dans un deuxième temps, nous nous sommes attachées à la modélisation de la PD du portefeuille Secteur Public Local segment Santé sous la réglementation IFRS 9. C'est l'algorithme par essaim particulaire (PSO) qui a été utilisé lors du développement de ce modèle.

2.1.4 Développement d'algorithmes d'optimisation

Le choix des ces deux algorithmes (AG et PSO) se justifie à travers leur mode de fonctionnement. En effet, ces algorithmes présentent plusieurs avantages qui nous permettent de les utiliser pour construire des modèles de notation, notamment :

— Ils s'appliquent à une grande variété de problèmes.

— Ils ne nécessitent pas d'hypothèse sur la régularité de la fonction à optimiser.

— Ils sont adaptés à l'exploration de grands espaces de recherche.

Nous avons choisi d'implémenter nous-même en Python l'algorithme génétique (AG) et l'algorithme d'optimisation par essaim particulaire (PSO). En effet, même s'il existe des packages déjà implémentés sur d'autres langages (R, Matlab), leur usage implique :

— une dépendance aux évolutions du logiciel et de ses packages ;

— un effet « boite noire » ;

— une impossibilité de modifier nous-même le code.

L'approche « interne », de développement de packages en Python, a donc été retenue car elle offre plusieurs avantages, dont :

— la maîtrise du code de l'algorithme ;

— la pérennité et transparence du code ;

— la liberté de changer et modifier facilement le package ;

1. Cf. section 5.4.
2. Cf. section 5.5.

— la possibilité d'adapter le paramétrage de l'algorithme en fonction de l'application.

Le fonctionnement des algorithmes (AG et PSO) implémentés est décrit dans le chapitre 5 et leur application aux portefeuilles Établissements Financiers et Secteur Public Local segment Santé de La Banque Postale est détaillée au chapitre 7.

2.2 Gestion du risque à l'entrée en relation client

Nous nous sommes aussi intéressées à la modélisation de la probabilité de défaut à l'octroi pour les nouveaux clients. Nous voulons évaluer le risque des nouveaux clients souhaitant souscrire le produit « compte bancaire » (aussi appelé compte courant) afin de leur attribuer les moyens de paiement (carte, découvert, ...) adaptés à leur profil de risque. L'une des problématiques qui se pose pour la construction de tels modèles concerne le manque de données sur le comportement bancaire du client qui rend compliqué la prédiction du comportement futur du client. Cela se traduit par une performance du modèle de notation moins élevée que celle observable sur d'autres portefeuilles.

2.2.1 Contraintes métiers et réglementaires

Le développement d'outils de notation doit se conformer :

— Aux réglementations concernant l'utilisation des données sur le client : par exemple, en France, le périmètre des données utilisables dans un modèle de notation pour l'octroi est réglementé par la Cnil [3].

— Aux contraintes internes : les modèles doivent être auditables, reproductibles, pouvoir être implémentés par le service Informatique de La Banque Postale et donner des résultats facilement interprétables pour leur usage mais aussi pour leur suivi (*backtesting*).

De plus, les résultats obtenus doivent être valides non seulement d'un point de vue statistique mais aussi d'un point de vue métier.

2.2.2 Méthodologie de construction du modèle de notation à l'entrée en relation

Le modèle de notation (aussi appelé score) existant a été construit à partir d'une approche par validation (qui est l'approche classique lorsque la volumétrie et le nombre de défauts observés sont suffisamment élevés).

L'approche par validation utilisée se base sur une régression logistique appliquée sur les données disponibles pour le portefeuille des nouveaux clients. Cette technique va permettre de regrouper les clients avec des comportements similaires dans des classes auxquelles seront associés des taux de défaut. La modélisation de la régression logistique est effectuée sur une certaine période d'estimation. Les résultats obtenus font l'objet de deux validations afin de s'assurer qu'ils sont généralisables :

— Une validation structurelle : l'estimation est faite sur une partie de la population et l'autre partie va servir pour l'étude de la stabilité des résultats.

— Une validation hors-temps : les résultats obtenus sont appliqués à différentes périodes (antérieures et ultérieures) afin de s'assurer de la stabilité des résultats sur l'historique.

3. Commission Nationale de l'Informatique et des Libertés.

La méthodologie de construction d'un score que nous avons utilisée[4] est détaillée à la section 6.2.2.

2.2.3 Pistes d'amélioration du modèle

De manière générale, il est assez difficile d'obtenir une performance élevée des modèles de notation à l'octroi pour les nouveaux clients. Nous avons donc étudié plusieurs pistes d'amélioration du modèle tout en gardant en tête les contraintes citées plus haut.

Ajout de données externes

Le site de l'Insee (Institut National de Statistiques et d'Études Économiques) propose des bases de données géocodées au niveau IRIS (Îlots Regroupés pour l'Information Statistique)[5] pour la France[6]. Nous pouvons associer à chaque client des données de l'Insee (taux de chômage, ...) liées à l'IRIS où il réside. Nous avons donc utilisé ces données afin d'étudier leur influence sur la probabilité de défaut des clients de La Banque Postale.

De plus, nous avons construit une nouvelle variable donnant la probabilité de défaut associée à chaque IRIS. Cette PD moyenne par IRIS a été construite à partir de l'ensemble de la clientèle « personnes physiques » de La Banque Postale. Nous avons également ajouté cette variable, qui permet d'associer à chaque nouveau client une probabilité de défaut caractéristique de la clientèle de la Banque Postale habitant l'IRIS, à notre étude afin d'évaluer son influence sur la prédiction du défaut.

Les résultats obtenus sont analysés à la section 8.2.

Sélection de variables par algorithme génétique

Dans un deuxième temps, nous avons testé d'autres méthodes que celle classiquement utilisée. Nous avons notamment appliqué l'algorithme génétique (cf. section 5.4) pour challenger la sélection de variables.

Les résultats de notre étude sont présentés à la section 8.3.2.

4. À noter que, par contrainte de temps et de moyens, nous n'avons pas étudié la validation temporelle du modèle.

5. C'est-à-dire que la France est découpée en plusieurs zones et que la population de chaque zone est statistiquement homogène.

6. Nous pouvons nous contenter de données sur la France uniquement car la grande majorité des clients de La Banque Postale réside en France.

3 Plan du document

Dans cette première partie, nous avons exposé le contexte et les enjeux liés à la modélisation de la probabilité de défaut pour une banque de détail ainsi que les trois applications qui nous ont tout particulièrement intéressées.

Dans une seconde partie, nous détaillons les concepts mathématiques et les méthodes que nous avons été amenées à utiliser. D'une part, nous présentons le fonctionnement des deux algorithmes d'optimisation, l'AG puis le PSO, que nous avons appliqués dans le cadre de la modélisation de la probabilité de défaut sur les portefeuilles *Low Default*. Et, d'autre part, nous exposons la méthodologie de construction d'un score à l'aide d'une régression logistique qui nous a permis de modéliser la probabilité de défaut des nouveaux clients LBP.

Enfin, dans une troisième partie, nous présentons les résultats obtenus suite à l'application des techniques exposées dans la deuxième partie. Dans un premier temps, nous nous intéressons aux portefeuilles *Low Default* pour lesquels nous utilisons nos algorithmes d'optimisation afin de construire un modèle de notation. Et, dans un second temps, nous analysons les résultats obtenus suite aux différentes techniques testées afin d'améliorer la prédiction de la probabilité de défaut à l'entrée en relation client.

Deuxième partie

Concepts mathématiques et méthodes utilisés

Plan de la deuxième partie

Dans cette partie, nous nous attachons à la présentation des concepts et méthodes mathématiques utilisés dans le cadre de la modélisation de la probabilité de défaut des portefeuilles *Low Default*, d'une part et à l'entrée en relation avec une clientèle *retail*, d'autre part.

Dans un premier temps, nous formalisons la structure générale du problème que les techniques présentées doivent nous aider à résoudre. En effet, pour chacun des portefeuilles étudiés, nous cherchons à calibrer une fonction de score permettant d'affecter à chaque entité une note reflétant sa probabilité de défaut.

Dans un second temps, nous présentons le fonctionnement des algorithmes d'optimisation que nous avons implémenté en Python : l'AG puis le PSO. De plus, nous nous intéressons ensuite plus précisément au problème d'optimisation que nos algorithmes doivent nous aider à résoudre dans le cadre de la modélisation de la probabilité de défaut des portefeuilles *Low Default*. Enfin, afin de répondre à la problématique du paramétrage des algorithmes, nous avons implémenté en Python un module permettant d'effectuer un *grid search*.

Dans un troisième temps, nous présentons les fondamentaux théoriques mais aussi métiers de la méthodologie de construction, à l'aide d'une régression logistique, du score sur les nouveaux clients LBP.

Les résultats de l'application de ces différentes techniques aux portefeuilles de La Banque Postale sont présentés dans la troisième partie.

4 Structure générale des problèmes à résoudre

Pour déterminer la probabilité de défaut associée à chaque client de la banque, l'équipe Modélisation de LBP est amenée à construire des modèles de notation des différents portefeuilles. La construction d'un modèle de notation consiste à calibrer une fonction de score dont le concept est décrit ci-dessous. La fonction de score va, en effet, nous permettre d'attribuer à chaque entité du portefeuille une note en fonction de sa probabilité de défaut.

4.1 Définition d'une fonction de score

La fonction de score g d'une entité z à noter est définie par :

$$g_\beta(z) = \sum_{i=1}^{p} \beta_i \cdot X_i(z) \ \in [0\,;g_{max}]$$

Où :

- g_{max}, la note maximum pouvant être attribuée à une entité (par exemple, 100 ou 10000) ;
- p, le nombre de variables explicatives ;
- $\beta = (\beta_1, \ldots, \beta_p)$, le vecteur des poids à attribuer à chaque facteur explicatif ;
- $X_i(z)$, la valeur prise par la variable explicative i $(1 \leq i \leq p)$ pour l'entité z.

Nous cherchons les poids optimaux β_i, à affecter à chacun de nos p facteurs explicatifs du défaut, qui permettront de calibrer au mieux la fonction de score (i.e. d'attribuer à chaque entité la note qui lui correspond le mieux).

Les entités sont ensuite regroupées par intervalle de note et une probabilité de défaut est associée à chaque classe de note en fonction du taux de défaut observé sur les entités du portefeuille étudié appartenant à la dite classe.

4.2 Approches de modélisation de la fonction de score

Dans la suite, nous utilisons deux approches pour modéliser la fonction de score des portefeuilles étudiés :

- L'approche par vérification : les poids de la fonction de score sont optimisés à l'aide d'un algorithme d'optimisation (AG ou PSO) et la pertinence de la fonction calibrée est calculée par le tau-b de Kendall (pour la modélisation des PD utilisées pour le provisionnement sur les portefeuilles Établissements Financiers et Secteur Public Local segment Santé).
- L'approche par validation : les poids de la fonction de score sont optimisés à l'aide d'une régression logistique et la performance de celle-ci est calculée par l'indice de Gini (pour la modélisation des PD à l'octroi pour les nouveaux clients).

5 Algorithmes d'optimisation

Afin de modéliser la probabilité de défaut sur les portefeuilles *Low Default*, nous utilisons deux algorithmes d'optimisation [1] :

— l'algorithme génétique (AG) ;

— l'algorithme d'optimisation par essaim particulaire (*Particle Swarm Optimization* ou PSO).

Dans un premier temps, nous présentons quelques généralités sur les algorithmes d'optimisation et, plus particulièrement, sur les méthodes métaheuristiques qui vont nous intéresser ici. Nous formalisons ensuite la structure des problèmes d'optimisation que nos algorithmes doivent nous aider à résoudre. Puis, nous présentons le fonctionnement des algorithmes que nous avons implémentés en Python : l'AG puis le PSO. Nous détaillons ensuite l'approche utilisée pour calibrer la fonction de score des portefeuilles Établissements Financiers et Secteur Public Local segment Santé à l'aide de nos deux algorithmes. Enfin, nous nous intéressons au *grid search*, une technique qui permet de tester plus facilement plusieurs paramétrages.

5.1 Un peu d'histoire [2]

Les premiers problèmes d'optimisation apparaissent dès l'antiquité mais les algorithmes d'optimisation commencent à se développer seulement à partir du $17^{e}/18^{e}$ siècle avec l'apparition du calcul différentiel (par exemple, avec la méthode de Newton). C'est à partir du 19^{e} siècle que les algorithmes d'optimisation commencent vraiment à se multiplier et à se populariser (notamment dans le domaine de l'économie). Les méthodes stochastiques font leur apparition dans les années 1950. Le terme « métaheuristique » apparaît à partir des années 80.

5.2 Méthodes métaheuristiques [3]

Les deux algorithmes d'optimisation que nous avons utilisés sont, plus précisément, des méthodes métaheuristiques. Les métaheuristiques forment une sous-famille des algorithmes d'optimisation . Ces algorithmes sont, dans la plupart des cas, stochastiques (i.e. ils utilisent des processus aléatoires) et itératifs. Les métaheuristiques possèdent, en général, les caractéristiques suivantes :

— elles demandent peu d'hypothèses sur la fonction objectif ;

— elles peuvent s'appliquer à une grande variété de problèmes [4] ;

— elles fournissent une solution approchée.

Un des désavantages des méthodes métaheuristiques est qu'elles ne garantissent pas que l'optimum trouvé soit un optimum global ; l'algorithme peut converger prématurément dans un optimum local. Ces méthodes permettent en revanche d'obtenir rapidement une solution réalisable.

1. Ces algorithmes doivent nous permettre de calibrer la fonction de score présentée au chapitre 4 précédent.
2. Source : XHONNEUX (2008).
3. Sources : EL DOR (2012), HERTZ (n.d.).
4. À l'opposé des heuristiques qui s'appliquent, en général, à une certaine catégorie de problème uniquement.

Les métaheuristiques sont notamment utilisées pour résoudre des problèmes d'optimisation dits « difficiles », c'est-à-dire des problèmes pour lesquels une optimisation par une méthode classique [5] n'est pas envisageable, par exemple, parce que le temps de calcul est trop important ou parce que les hypothèses (sur la régularité de la fonction à optimiser) inhérentes à l'utilisation de ces méthodes ne sont pas vérifiées (continuité, convexité ou dérivabilité par exemple) ou encore parce qu'il existe un trop grand nombre de solutions locales.

Les deux métaheuristiques que nous avons choisies d'implémenter, le PSO et l'AG, sont inspirées de phénomènes naturels ou biologiques. Elles sont, de plus, basées sur l'exploration de l'espace de recherche à l'aide d'une population de points [6].

Certaines métaheuristiques, comme le PSO, font aussi intervenir la notion de mémoire (plus ou moins évoluée et sur un plus ou moins long terme), c'est-à-dire que l'algorithme garde en mémoire l'historique de certains paramètres.

5.3 Problèmes d'optimisation

5.3.1 Généralités

Un problème d'optimisation est constitué :

— d'un ensemble de points (l'espace de recherche) ;

— d'une fonction objectif ;

— et, éventuellement, d'une ou plusieurs contraintes d'égalité ou d'inégalité.

Les algorithmes d'optimisation permettent de résoudre un problème d'optimisation. L'algorithme va converger pas à pas vers la meilleure solution au problème. En général, le but est de trouver l'optimum [7] global de la fonction objectif à l'aide de méthodes déterministes (i.e. exactes) ou à l'aide de méthodes approchées.

5.3.2 Périmètre d'application de nos algorithmes

Nous avons choisi d'implémenter nos algorithmes afin qu'ils nous permettent d'approcher le minimum d'une fonction quelconque. Les problèmes qu'ils doivent nous aider à résoudre ne comportent pas de contraintes d'égalité ou d'inégalité, la seule contrainte retenue concerne l'espace de recherche (il doit être fini) [8].

Les problèmes d'optimisation auxquels nous nous intéressons sont définis, de manière plus formelle, ci-dessous :

Formulation mathématique

Soit p, le nombre de variables dont nous disposons.

Soit E, l'espace de recherche [9] de l'algorithme i.e. l'ensemble des valeurs que peut prendre chaque variable considérée dans le problème d'optimisation. Tout point x appartenant à E est constitué de p variables. L'ensemble E est fini.

5. Méthodes déterministes comme l'algorithme du gradient ou l'algorithme de Newton par exemple.
6. Ce n'est pas le cas de toutes les métaheuristiques, voir par exemple, la recherche tabou ou le recuit simulé qui sont basées sur le suivi de la trajectoire d'un seul point.
7. Minimum ou maximum.
8. Ces choix sont adaptés à l'utilisation de ces algorithmes à LBP.
9. Aussi appelé espace de calibration ou espace des variables.

Et, soit f, la fonction objectif (i.e. la fonction à optimiser), définie de E dans $\mathbb{R}$.

Nous cherchons une solution approchée $x^* \in E$ au problème suivant :

$$x^* = \arg\min_{x \in E} f(x)\,.$$

Ou, autrement dit, nous cherchons $x^* \in E$ tel que $\forall x \in E,\ f(x^*) \leq f(x)$.

Rappelons qu'il est tout à fait possible, à partir du problème posé ci-dessus, d'approximer le maximum de la fonction f :

Recherche du maximum

Pour trouver $x^* \in E$ tel que $\forall x \in E,\ f(x^*) \geq f(x)$, il suffit de minimiser la fonction $-f$. En effet : $\arg\max f = \arg\min(-f)$ et $\max f = -\min(-f)$.

Certaines méthodes métaheuristiques, dont le PSO fait partie, font intervenir la notion de voisinage :

Voisinage

Le voisinage d'un point $x \in E$ est un sous-ensemble N de E tels que pour tout $y \in N$, y est un voisin de x.

Le minimum y^* dans le voisinage de x est alors définit par :

$$y^* = \arg\min_{y \in N} f(y)\,.$$

La construction de l'ensemble N (et donc la définition d'un « voisin ») varie d'un algorithme à l'autre (cf. section 5.5 pour la définition du voisinage dans le PSO).

5.3.3 Propriétés

L'algorithme PSO est utilisé pour résoudre des problèmes d'optimisation où les variables sont numériques et continues. L'AG peut être utilisé avec des variables numériques continues, discrètes ou les deux ou avec des variables non numériques. Ces algorithmes sont adaptés aux problèmes où la fonction objectif est non-linéaire. Le périmètre d'application de chaque algorithme est récapitulé ci-dessous [10] :

Type de variable	**Espace de recherche E**	**Type de fonction f**	**Algorithmes**	**Exemple d'application**
Continue	$E = \bigotimes_{i=1}^{p} [b_{inf_i}; b_{sup_i}]$	$f : E \to \mathbb{R}$ f quelconque	AG, PSO	Optimisation de la performance d'une fonction de score
Discrète	$E = \{val_1, val_2, \ldots, val_d\}^p$	$f : E \to \mathbb{R}$ f quelconque	AG	Sélection de variables pour effectuer une régression logistique

Tableau 5.1 – Récapitulatif du périmètre d'application de l'AG et du PSO

10. Nous récapitulons ici les problèmes que nos algorithmes, tels que nous les avons implémentés, peuvent nous aider à résoudre. L'AG et le PSO peuvent potentiellement s'appliquer à une plus grande variété de problèmes qui ne nous intéresse pas ici.

5.4 Algorithme génétique (AG)

Dans cette section, nous décrivons le fonctionnement de l'algorithme génétique en définissant le vocabulaire spécifique à celui-ci ainsi que les méthodes utilisées pour le coder en Python. Enfin, nous nous intéressons à l'influence du paramétrage sur l'obtention d'un résultat final optimal. L'utilisation que nous faisons de cet algorithme dans le cadre de la modélisation de la probabilité de défaut des portefeuilles *Low Default* est détaillé à la section 5.6 et les résultats obtenus sont présentés au chapitre 7. Nous avons également utilisé cet algorithme comme méthode de sélection de variables pour la régression logistique et les résultats que nous avons obtenus sont présentés à la section 8.3.2.

Nous nous sommes principalement appuyées sur la documentation Matlab (MathWorks (2018*a*)) pour choisir les méthodes et déterminer le fonctionnement de notre algorithme génétique. Pour appréhender le fonctionnement et les notions liées à la compréhension des algorithmes génétiques et à leur implémentation, nous nous sommes également aidées de la publication suivante : LE RICHE et al. (2007).

5.4.1 Présentation générale des algorithmes génétiques

Les algorithmes génétiques (AG) ont été introduits par John Holland en 1975 [11] et les mécanismes qui y sont utilisés apparaissent dans la littérature dès la fin des années 60. Ils appartiennent à la famille des algorithmes évolutionnaires (ou évolutionnistes) et à la famille des métaheuristiques. Les algorithmes évolutionnaires s'inspirent de la biologie et, plus particulièrement, des mécanismes d'évolutions des espèces (la sélection naturelle). Ils utilisent une population d'individus dont les gènes vont se modifier à travers trois mécanismes principaux :

— la sélection ;

— le croisement (ou reproduction) ;

— la mutation.

Au fil des générations, les individus les mieux adaptés à leur environnement vont survivre et se reproduire tandis que les autres individus moins adaptés vont disparaître progressivement. Les individus sélectionnés pour faire partie de la population des parents vont pouvoir croiser leurs gènes pour former une nouvelle génération. Certains gènes sont également susceptibles de subir une mutation.

Pour chaque étape de l'algorithme, il existe plusieurs méthodes possibles, nous avons choisi celles qui nous paraissaient pouvoir s'adapter à la majorité des problématiques rencontrées par la banque. Le déroulement de l'algorithme que nous présentons dans les parties suivantes reflète ces choix.

5.4.2 Vocabulaire des algorithmes génétiques

Nous détaillons ci-dessous plusieurs notions clés permettant une bonne compréhension des algorithmes génétiques :

Individu

— c'est une solution éventuelle au problème d'optimisation (i.e. c'est un point x de E auquel est associé la valeur de $f(x)$) ;

— il est caractérisé par ses gènes et sa *fitness*.

11. HOLLAND (1975).

Gène

— c'est une des variables du problème d'optimisation ;

— les p gènes d'un individu correspondent donc à un point de l'espace de recherche E et l'ensemble des valeurs possibles pour chaque gène correspond à l'espace de recherche de l'algorithme.

Fitness

La *fitness* correspond à la valeur prise par la fonction objectif[12] pour les gènes d'un individu donné.

Population

— elle est constituée de l'ensemble des individus ;

— la population initiale est (tout ou en partie) choisie par l'utilisateur ou est (tout ou en partie) générée aléatoirement ;

— elle peut-être subdivisée en plusieurs sous populations de tailles différentes lorsque une option de migration est ajoutée à l'algorithme.

Génération

Une génération correspond à une itération de l'algorithme.

Parents

— les parents représentent l'ensemble des individus qui vont se reproduire pour former la génération suivante ;

— ils sont choisis suivant le principe de sélection retenu pour l'algorithme (cf. paragraphe suivant).

Enfants

— l'ensemble des individus issus de la sélection constitue la population des enfants et ceux-ci forment la génération suivante ;

— les enfants héritent leurs gènes de la population des parents ;

— il existe plusieurs façons de construire la population des enfants (la méthode retenue est explicitée dans le paragraphe suivant).

5.4.3 Définition des méthodes utilisées pour notre algorithme génétique

Pour chaque étape de l'algorithme, il existe (en général) plusieurs méthodes dans la littérature[13]. Nous détaillons dans cette partie les méthodes que nous avons choisies d'utiliser lors de notre implémentation en Python.

Élitisme

L'élite est constituée des meilleurs individus c'est-à-dire des individus qui possèdent la meilleure évaluation de la fonction objectif (i.e. les individus ayant les plus petites *fitness*). Les individus appartenant à l'élite sont reproduits sans modification (pas de croisement, ni de mutation) à la génération suivante, ils ont donc une probabilité de survie certaine.

Le taux d'élites dans la population est choisi par l'utilisateur.

12. Aussi appelée fonction d'évaluation ou d'adaptation.

13. Voir, par exemple, la documentation Matlab (MathWorks (2018*a*)) sur les algorithmes génétiques pour des exemples d'autres méthodes. D'autres méthodes sont détaillées en annexe A.1.

Échelle de *fitness* (ou échelle des scores)

L'échelle de *fitness* permet d'attribuer une valeur à chaque individu en fonction de son rang dans la population ; le rang d'un individu étant déterminé par sa *fitness*. L'échelle de *fitness* est ensuite utilisée pour sélectionner la population des parents.

Après avoir trié la population par *fitness* croissante, nous utilisons une méthode de **sélection par rang** pour attribuer un poids à chaque individu :

— chaque individu a un rang k (1 pour l'individu avec la meilleure *fitness*, 2 pour le deuxième meilleur, etc.) ;

— la valeur de mise à l'échelle de cet individu est alors de $1/\sqrt{k}$ (1 pour le premier individu, $1/\sqrt{2}$ pour le deuxième, etc.).

Avec cette méthode, les mauvais individus ont des valeurs proches sur l'échelle de *fitness*.

Sélection des parents

Il faut définir un principe de sélection des parents permettant de choisir les individus qui vont être amenés à se reproduire. La méthode de sélection des parents permet, en général, de favoriser la sélection des meilleurs individus de la population tout en laissant une chance aux plus mauvais individus de pouvoir se reproduire.

Nous utilisons la méthode **stochastique uniforme** suivante :

— nous traçons une ligne où chaque section de la ligne a une longueur égale à une valeur de l'échelle de *fitness* (définie ci-dessus) : la première portion de la ligne est de longueur 1 et correspond à l'individu de rang 1, la deuxième portion de la ligne est de longueur $1/\sqrt{2}$ et correspond à l'individu de rang 2, etc. ;

— nous choisissons un pas d'avancement (par exemple, 0,9) ;

— une position initiale est choisie aléatoirement entre 0 et le pas (à chaque génération) ;

— l'individu correspondant à la portion de la ligne où nous sommes est ajouté à la population des parents ;

— nous avançons sur la ligne jusqu'à ce que la population des parents atteigne la taille voulue [14] (en reprenant au début de la ligne lorsque sa fin est atteinte).

Croisement ou *crossover*

Il existe plusieurs façon d'effectuer un croisement entre deux parents, il faut donc définir la méthode que nous allons choisir pour croiser les individus et ainsi créer la génération suivante.

Nous utilisons une méthode de **croisement par points** :

— tout d'abord, nous tirons au hasard deux parents dans la population des parents ;

— tous les couples de parents ne se reproduisent pas par croisement : la probabilité de croisement d'un couple de parents est à définir par l'utilisateur (elle peut, par exemple, être de 0,8) ;

— si un couple de parents n'effectue pas de croisement, chaque parent devient un enfant ;

— s'il y a croisement, chaque parent a une chance sur deux de transmettre un gène au premier enfant issu du croisement (et le deuxième enfant issu de ce couple de parents est créé à partir du tirage complémentaire du premier enfant).

14. Le nombre de parents à sélectionner est égal au nombre d'individus moins le nombre d'élites.

Le graphique ci-dessous illustre cette méthode (pour des individus avec 4 gènes) :

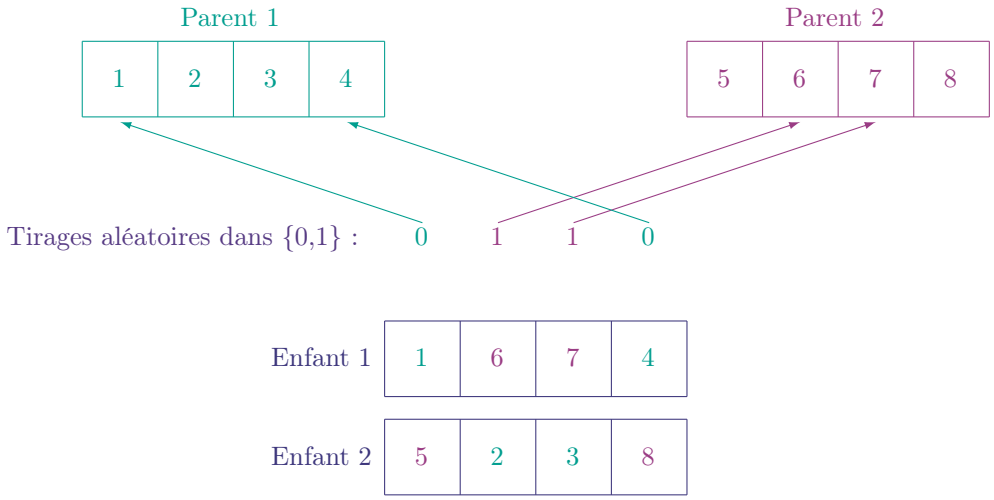

FIGURE 5.1 – Exemple de croisement par points (AG)

Mutation

Il existe de même plusieurs façons d'effectuer une mutation sur un individu i.e. de modifier tout ou une partie de ses gènes. Nous effectuons une mutation **aléatoire uniforme, adaptée aux contraintes de l'espace** :

— chaque gène de chaque individu a une certaine probabilité de mutation (souvent assez faible, entre 0,01 et 0,1) ;

— si un gène est sélectionné pour la mutation, la nouvelle valeur prise par celui-ci est tirée aléatoirement dans l'espace de définition du gène.

En reprenant les enfants obtenus après croisement dans l'exemple précèdent, nous pouvons, par exemple, obtenir les enfants suivants :

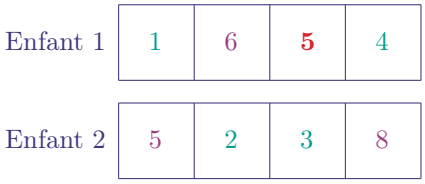

FIGURE 5.2 – Exemple d'enfants après mutation (AG)

5.4.4 Fonctionnement de l'algorithme génétique

Dans ce paragraphe, nous décrivons le fonctionnement de l'algorithme génétique. Pour chaque étape, les méthodes utilisées sont celles décrites au paragraphe précédent.

Pour passer d'une génération à une autre, nous détectons les meilleurs éléments de la population à la génération g. Ces individus forment la population des élites. Puis, nous sélectionnons la population des parents qui vont ensuite former des paires et potentiellement se croiser pour former la population des enfants. Une partie des gènes des enfants va ensuite muter. La population des enfants ainsi obtenue va avec la population des élites former la génération suivante $g + 1$.

Pour illustrer le fonctionnement de cet algorithme, nous prenons l'exemple d'un algorithme génétique avec un taux d'élites à 0,05, un taux de croisement à 0,8 et un taux de mutation à 0,05 [15].

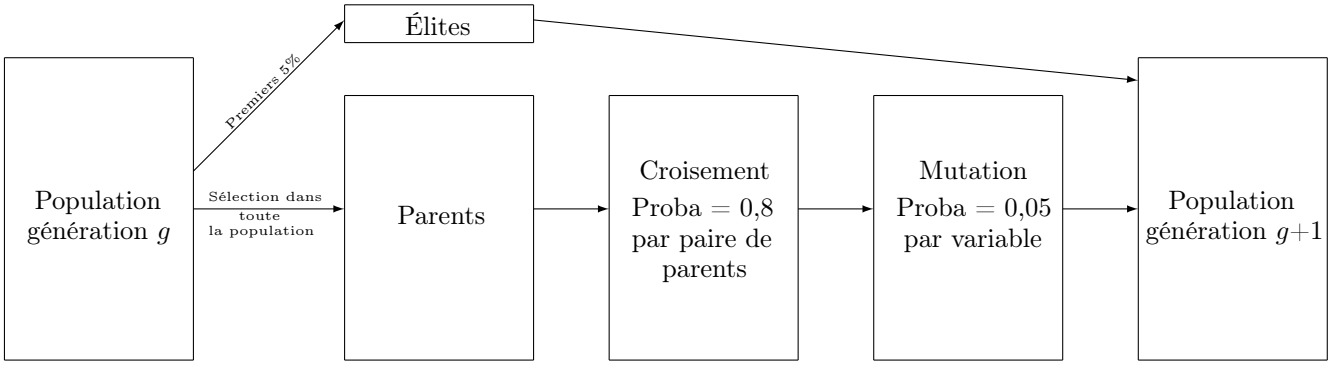

FIGURE 5.3 – Passage d'une génération à une autre (AG)

Le fonctionnement de l'algorithme est récapitulé en annexe A.1 p. 87.

5.4.5 Paramétrage de l'algorithme génétique

Les résultats obtenus grâce à des algorithmes tels que l'AG peuvent grandement varier en fonction du paramétrage choisi. Certains paramétrages ne permettront pas à l'algorithme de converger vers la solution dans le nombre de générations imparti ou alors l'algorithme convergera mais vers une solution locale et non globale. Le paramétrage a aussi une influence sur le temps d'exécution de l'algorithme.

Il n'existe, a priori, pas de paramétrage optimal (i.e. qui donnerait un résultat satisfaisant quelque soit le problème posé), il est donc nécessaire d'en tester plusieurs afin de s'assurer que la solution trouvée est bien optimale.

Les principaux paramètres que doit calibrer un utilisateur de l'algorithme génétique sont récapitulés ci-dessous :

15. Ces taux peuvent être modifiés : cf. paragraphe suivant sur le paramétrage.

Catégorie	Description de la variable	Exemple de valeur
Population	Taille de la population	500
Critère d'arrêt	Nombre de générations maximal	500
Sélection	Taux d'élites	0,05
	Pas de sélection des parents	0,9
Croisement	Probabilité de croisement	0,8
Mutation	Probabilité de mutation	0,05

Tableau 5.2 – Paramètres à calibrer pour utiliser l'AG

Afin d'éviter la perte d'information, il est en général conseillé d'avoir un taux d'élites différent de 0 (de façon à avoir au moins un individu transmis automatiquement à la génération suivante). En effet, en l'absence d'élitisme, la meilleure solution trouvée à la génération g n'est pas forcement transmise à la génération suivante ce qui peut entraîner une perte d'information. À l'inverse, un taux d'élites trop important peut compromettre la convergence de l'algorithme car la sélection s'effectue uniquement sur la population des parents (de taille nombre d'individus moins nombre d'élites).

La façon dont nous avons implémenté notre algorithme en Python permet de continuer l'optimisation en utilisant la population finale obtenue pour initialiser l'algorithme après avoir exécuté l'algorithme une première fois sur un certain nombre de générations $gmax$. Ainsi, l'optimisation peut se poursuivre sans devoir reprendre au début si l'utilisateur souhaite augmenter le nombre de générations. Cette fonctionnalité explique certaines décisions qui ont pu être prises lors du choix des méthodes. En effet, les méthodes choisies ne sont pas dépendantes du pas d'itération.

Pour un même paramétrage, les résultats obtenus peuvent aussi être fortement influencés par les différents processus aléatoires utilisés. Ainsi, pour un paramétrage donné, il peut être intéressant de tester plusieurs graines [16] pour constater les différents résultats qu'il est possible d'obtenir. La population initiale qu'elle soit choisie tout ou en partie par l'utilisateur ou aléatoirement a aussi une influence sur le résultat final et la convergence ou non de l'algorithme.

Plusieurs paramétrages peuvent donner des résultats satisfaisants, il n'y a pas unicité du paramétrage face à un problème donné.

5.5 Optimisation par essaim particulaire (PSO)

Dans cette section, nous décrivons les principes et le fonctionnement de l'algorithme PSO ainsi que les paramètres à prendre en compte dans son utilisation pratique. L'utilisation que nous faisons de cet algorithme dans le cadre de la modélisation de la probabilité de défaut des portefeuilles *Low Default* est détaillé à la section 5.6 et les résultats obtenus sont présentés au chapitre 7.

Pour construire notre algorithme, nous nous sommes principalement inspirées de la documentation Matlab (MathWorks (2018*b*)) ainsi que des publications de CLERC (2012) et SHI & EBERHART (1998).

16. Les processus aléatoires générés par Python sont en réalité des processus pseudo-aléatoires, ils dépendent donc de la graine utilisée pour initialiser le générateur. Lors de l'utilisation de nos algorithmes, nous fixons systématiquement une graine afin de pouvoir reproduire nos résultats.

5.5.1 Présentation générale de l'algorithme PSO

L'algorithme d'optimisation par essaim particulaire (*particle swarm* en anglais) s'inspire des comportements sociaux observés dans la nature, par exemple, dans une nuée d'oiseaux, un banc de poissons ou un essaim d'abeilles. Il a été initialement développé par James Kennedy et Russell Eberhart en 1995 [17] dont le but premier était de modéliser les interactions sociales.

L'algorithme PSO fait partie de la famille des méthodes métaheuristiques et utilise la notion de mémoire. En effet, le PSO fait intervenir le concept d'intelligence distribuée (ou *swarm intelligence* ou intelligence en essaim), c'est-à-dire que l'algorithme contient un système d'interactions entre les individus qui démontre une certaine intelligence au niveau global. Au fil des itérations, les particules vont évoluer dans l'espace de recherche en gardant en mémoire la meilleure position qu'elles ont rencontrée ainsi que la meilleure position rencontrée par leurs voisines. Certaines particules peuvent soit se regrouper dans un minimum local soit continuer à explorer l'espace de recherche mais au moins une partie des particules va finir par converger vers la meilleure solution au problème d'optimisation (i.e. le minimum global de la fonction).

5.5.2 Vocabulaire de l'algorithme PSO

Nous détaillons, ci-dessous, plusieurs notions clés permettant de mieux comprendre l'algorithme PSO :

Essaim

L'essaim est constitué de **l'ensemble des particules**.

Particule

Une particule est caractérisée par :

- sa **position** et ;
- sa **vitesse**.

Position

- c'est un **point de l'espace de recherche** (i.e. c'est une solution potentielle $x \in E$ au problème posé) ;
- elle est constituée des valeurs prises par chacune des variables considérées dans le problème.

Vitesse

- la vitesse de **déplacement d'une particule** dans l'espace de recherche indique la direction que la particule va prendre ;
- la vitesse est représentée par un vecteur de taille p dont les valeurs sont bornées par -2 et 2 [18] ;
- la vitesse d'une particule dépend de sa vitesse à l'itération précédente, de la meilleure position rencontrée par la particule et de celle rencontrée par ses voisines.

17. KENNEDY & EBERHART (1995).

18. Ces bornes de la vitesse peuvent être rencontrées dans la littérature, voir par exemple SHI & EBERHART (1998). Contrôler l'amplitude de la vitesse permet, par exemple, d'éviter l'explosion de la vitesse ce qui pourrait entraîner la divergence de l'algorithme.

Meilleure position

La meilleure position d'une particule ou d'un ensemble de particules désigne la position qui, parmi toutes les positions rencontrées par la ou les particule(s), donne la **plus petite valeur de la fonction objectif** (c'est donc un point de l'espace de recherche).

Voisinage

- les voisines d'une particule sont choisies aléatoirement dans l'essaim à chaque itération et une particule peut être plusieurs fois voisine d'une même particule ;
- chaque particule s'informe elle-même et informe les particules de son voisinage de la meilleure position qu'elle a rencontrée ;
- la taille du voisinage (i.e. le nombre de voisines) peut varier à chaque itération : elle diminue ou augmente en fonction de si la meilleure solution globale a été ou non améliorée (cf. p. 32).

5.5.3 Fonctionnement de l'algorithme PSO

Pour résoudre un problème d'optimisation avec cet algorithme, l'espace de recherche E doit être un sous-ensemble fini de $\mathbb{R}^p$.

Il existe plusieurs versions [19] de l'algorithme notamment en ce qui concerne l'implémentation de la vitesse et l'évolution du coefficient d'inertie. Nous avons donc effectué des choix en fonction de ce qui nous semblait donner les meilleurs résultats et s'adapter à un plus grand nombre de problématiques.

Initialisation

Pour initialiser l'algorithme, un essaim de taille N est généré aléatoirement :

- la position de chaque particule de l'essaim est générée par p tirages uniformes entre les bornes inférieures et supérieures de l'espace de définition de chaque variable ;
- la vitesse de chaque particule est générée par p tirages uniformes entre -2 et 2.

Déplacement des particules

À chaque itération, chaque particule de l'essaim va se déplacer dans l'espace E à la recherche du minimum de la fonction objectif f. La vitesse d'une particule à l'itération k est influencée par trois types de comportements :

- sa tendance à l'exploration (indépendance) i.e. l'influence de sa vitesse à l'itération $k-1$;
- sa tendance à retourner vers la meilleure position qu'elle a rencontrée jusqu'ici (la *self influence* ou influence historique ou cognitive) ;
- sa tendance à suivre le groupe i.e. sa tendance à se diriger vers la meilleure position que les particules dans son voisinage ont rencontrée jusqu'ici (influence du voisinage i.e. influence sociale).

19. Voir par exemple DU & SWAMY (2016) qui recense plusieurs variantes du PSO.

L'influence des différents comportements peut-être paramétrée i.e. elle peut être choisie par l'utilisateur de l'algorithme. De plus, l'importance donnée à chaque influence varie au fil des itérations. Par exemple, au début de l'algorithme, nous allons plutôt favoriser la tendance à l'exploration puis celle-ci va décroître au profit des deux autres influences. Les coefficients à appliquer à chaque tendance sont les suivants :

— w : inertie des particules i.e. importance donnée à l'exploration de la zone de recherche. Généralement comprise entre 0,1 et 1,1, la valeur de w évolue au cours de l'algorithme (cf. p. 32).

— Y_1 : coefficient d'ajustement de l'influence historique.

— Y_2 : coefficient d'ajustement de l'influence du voisinage.

— u_1, u_2 : nombres aléatoires entre 0 et 1 s'appliquant respectivement au coefficient d'ajustement Y_1 et Y_2. Les coefficients d'ajustements des influences historique et du voisinage varient aléatoirement pour chaque variable du vecteur vitesse et pour chaque particule.

La valeur initiale de w est choisie par l'utilisateur (cette valeur peut ensuite varier entre 0,1 et 1,1). Y_1 et Y_2 sont également choisis par l'utilisateur et, généralement, $Y_1 = Y_2$.

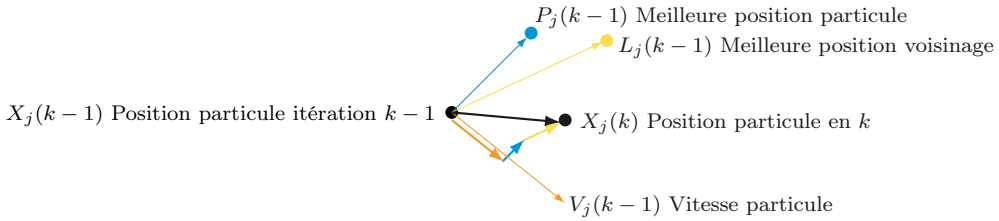

FIGURE 5.4 – Déplacement d'une particule (PSO)

La vitesse V_j de chaque particule j à l'itération k est donc attribuée par l'opération suivante :

$$V_j(k) \quad \leftarrow \quad w \cdot V_j(k-1) + Y_1 \cdot u_1 \cdot \big(P_j(k-1) - X_j(k-1)\big) + Y_2 \cdot u_2 \cdot \big(L_j(k-1) - X_j(k-1)\big)$$

Et, après avoir vérifié que chaque coordonnée du vecteur vitesse appartient bien à $[-2\,;\,2]$ [20], la position X_j de chaque particule j est mise à jour comme suit :

$$X_j(k) \quad \leftarrow \quad V_j(k) + X_j(k-1)$$

Après avoir vérifié que $X_j(k)$ appartient bien à E [21], nous calculons la valeur de f en cette position. Si la position X_j calculée donne un meilleur résultat que la meilleure position rencontrée jusqu'ici P_j [22], cette dernière est mise à jour :

$$P_j(k) \quad \leftarrow \quad X_j(k)$$

20. I.e. pour chaque coordonnée i $(1 \leq i \leq p)$, $V_j(k)[i] \leftarrow \max\big(\min(V_j(k)[i], 2), -2\big)$.
21. Si la valeur d'une des variables de $X_j(k)$ est située en dehors de son espace de définition, la valeur max ou min de son espace de définition lui est attribuée.
22. I.e. $f\big(X_j(k)\big) < f\big(P_j(k-1)\big)$.

De même, la meilleure position globale (i.e. la meilleure position rencontrée jusqu'ici par l'essaim) est mise à jour si nécessaire.

Mise à jour du voisinage

Nous cherchons maintenant à mettre à jour, pour chaque particule j, la meilleure position L_j rencontrée par son voisinage.

Dans notre version [23], le voisinage de la particule est modifié à chaque itération et l'ordre des particules pour la mise à jour du voisinage est choisi aléatoirement. Le nombre de voisines d'une particule varie au cours de l'algorithme : il augmente (dans la limite de la taille de la population) si la meilleure solution globale n'a pas été améliorée et il reprend sa taille initiale S_0 dans le cas contraire. Une valeur initiale S_0 du nombre de voisines peut par exemple être 25 % du nombre total de particules.

Soit S_k, le nombre de voisines (maximum) de chaque particule à l'itération k. La marche à suivre est la suivante :

— le voisinage de chaque particule est défini aléatoirement : il faut effectuer S_k tirages avec remise afin de déterminer les voisines de chacune des particules,

— chaque particule s'informe elle-même et informe entre 1 et S_k voisines de la meilleure position qu'elle a rencontrée jusqu'ici ;

— chaque particule voisine met à jour, si nécessaire, la donnée sur la meilleure position rencontrée dans son voisinage.

Le déplacement des particules et la mise à jour du voisinage sont deux étapes distinctes de chaque itération k de l'algorithme. À chaque itération, il faut effectuer deux itérations successives sur l'essaim : une pour déplacer les particules et une pour mettre à jour le voisinage. Sans ces deux étapes, une particule ne pourrait être influencée que par les particules voisines placées avant elle dans l'essaim (car les autres n'aurait pas encore mis à jour leur position).

Critères d'arrêt de l'algorithme

Deux critères d'arrêt, dont les valeurs maximales sont définies par l'utilisateur, permettent de stopper l'algorithme :

— le critère d'**itération** :

> le compteur d'itération est augmenté de 1 à la fin de chaque itération jusqu'à atteindre sa valeur maximale ;

— le critère de **stagnation** :

> le compteur de stagnation s'itère lorsque la meilleure solution globale n'a pas été améliorée à la fin d'une itération de l'algorithme ; il est, au contraire, diminué lorsque la meilleure solution globale a évolué au cours de l'itération.

Soit c_k, la valeur du compteur de stagnation à l'itération k et soit S_k, la taille du voisinage d'une particule à l'itération k. Ces deux valeurs, ainsi que l'inertie w des particules, sont mises à jour à la fin de l'itération k comme suit :

Si la meilleure solution globale a été améliorée :

23. La définition du voisinage peut différer d'une implémentation du PSO à une autre.

$$
\begin{array}{rcl}
c_{k+1} & \leftarrow & \max(0\,;\, c_k - 1) \\
S_k & \leftarrow & S_0 \\
\left\{ \begin{array}{l} \textbf{Si } c_{k+1} < 2 \textbf{ alors } w \\ \textbf{Si } c_{k+1} > 5 \textbf{ alors } w \end{array} \right. & \leftarrow & \begin{array}{l} \min(2 \cdot w\,;\, 1,1) \\ \max\left(\frac{w}{2}\,;\, 0,1\right) \end{array}
\end{array}
$$

Sinon :

$$
\begin{array}{rcl}
c_{k+1} & \leftarrow & c_k + 1 \\
S_k & \leftarrow & \min(N\,;\, S_k + S_0)
\end{array}
$$

Le fonctionnement de l'algorithme est récapitulé dans le schéma en annexe B.1 p. 89.

5.5.4 Paramétrage du PSO [24]

Les résultats obtenus (solution, temps de calcul, nombre d'itérations, ...) peuvent être très différents en fonction du paramétrage utilisé. Le paramétrage doit donc être adapté en fonction de la complexité du problème à résoudre.

Pour chaque problème d'optimisation, il est important de faire varier le paramétrage de l'algorithme. En effet, il n'existe pas de paramétrage optimal qui conviendrait à tous les problèmes. Dans notre version de l'algorithme d'optimisation par essaim particulaire, l'utilisateur peut faire varier certains paramètres :

Catégorie	Description de la variable	Exemple de valeur
Essaim	Nombre de particules	120
	Pourcentage de voisines initial	0,075
Critère d'arrêt	Nombre d'itérations maximal	200
	Valeur maximale du critère de stagnation	25
Vitesse	Inertie	1,1
	Influence historique	1,49
	Influence sociale	1,49

Tableau 5.3 – Paramètres à calibrer pour utiliser le PSO

En général, plus le problème va être complexe (augmentation du nombre de variables, ...), plus le nombre de particules nécessaires pour arriver à une solution satisfaisante va augmenter. Pour de tels problèmes, un nombre de particules à choisir pour un premier résultat peut, par exemple, être de 2000 particules. Plus le nombre de particules augmente et plus l'espace de recherche est important, plus le temps d'exécution de l'algorithme va augmenter. Il est donc parfois nécessaire d'effectuer des compromis dans la façon de paramétrer la résolution de l'algorithme.

Il est souvent préférable d'éviter de choisir un voisinage initial trop important. En effet, un pourcentage initial de voisines supérieur à 70 % peut compromettre la convergence de l'algorithme vers l'optimum global car les particules auront tendance à converger trop tôt vers le même point sans forcément chercher à explorer l'espace. De même, un pourcentage trop faible peut impacter le

24. Pour des exemples d'analyses de la convergence de l'algorithme PSO (qui dépend du paramétrage et des méthodes choisies), le lecteur peut par exemple se référer à CLERC & KENNEDY (2002) ou TRELEA (2003).

temps nécessaire à la résolution de l'algorithme. Le choix de ce pourcentage se fait, en général, en fonction du nombre de particules choisi.

Nous avons uniquement testé les paramétrages (inertie, influence historique, influence sociale) les plus rencontrés dans la littérature (cf. tableau ci-dessous), mais il est tout à fait possible d'en tester d'autres. Il est néanmoins conseillé de choisir l'inertie initiale w entre 0,1 et 1,1 avec une préférence pour une valeur initiale appartenant à la seconde moitié de cet intervalle. Il est également préconisé de prendre des valeurs relativement proches et dans l'intervalle $[0\,;\,4]$ pour les coefficients d'ajustements des influences historique et sociale.

Paramètre	**Exemple 1** [25]	**Exemple 2** [26]	**Exemple 3** [27]
Inertie	1,1	0,721	0,95
Influence historique	1,49	1,193	2
Influence sociale	1,49	1,193	2

TABLEAU 5.4 – Exemples de paramétrages classiques de la vitesse (PSO)

Le nombre d'itérations maximal et la valeur maximale du compteur de stagnation peuvent être augmentés (en fonction du critère d'arrêt constaté lors d'un premier test) jusqu'à ce que l'utilisateur constate que l'augmentation de la valeur maximale de ces critères n'améliore pas, de manière significative, la solution trouvée. La valeur maximale du compteur de stagnation peut, par exemple, être fixée à 75 ou 25.

5.6 Utilisation pour la gestion du risque des portefeuilles *Low Default*

Dans cette section, nous présentons la méthodologie de modélisation de la probabilité de défaut que nous avons utilisée pour les deux portefeuilles *Low Default* étudiés, puis, nous nous intéressons plus en détail au problème d'optimisation que nos algorithmes doivent nous aider à résoudre et nous faisons le lien entre le vocabulaire de chaque algorithme et les données du problème.

Nous avons utilisé les algorithmes génétique et par essaim particulaire pour les applications aux portefeuilles Établissements financiers et Secteur Public Local segment Santé. Les résultats obtenus ainsi que les différentes spécificités liées à chaque portefeuille étudié sont présentés au chapitre 7.

5.6.1 Méthodologie de modélisation de la probabilité de défaut

La modélisation de la probabilité de défaut des portefeuilles *Low Default* auxquels nous nous sommes intéressées, Établissements Financiers et Secteur Public Local segment Santé, se base sur une approche par vérification. Le fonctionnement général de cette approche est le suivant :

1. Un classement des entités du portefeuille étudié est réalisé par les experts métiers de La Banque Postale.

25. Voir par exemple MathWorks (2018*b*).
26. Voir par exemple CLERC (2012).
27. Voir par exemple SHI & EBERHART (1998).

2. Un modèle de notation est ensuite construit de manière à attribuer aux entités des notes qui permettent de reproduire l'ordre établit par les experts.

Les facteurs explicatifs utilisés par les experts pour effectuer leur classement [28] sont les mêmes que ceux que nous utilisons pour construire le modèle de notation (i.e pour calibrer la fonction de score). En effet, notre but est de construire un modèle qui reproduit la logique utilisée par les experts pour classer les entités afin d'automatiser les processus de notation des portefeuilles.

La méthodologie de modélisation de la probabilité de défaut d'un portefeuille *Low Default* en utilisant cette approche peut être décomposée en 5 grandes étapes [29] :

1. Récupération et traitement des données.
2. Optimisation de la fonction de score à l'aide du tau-b de Kendall.
3. Regroupement des notes données par la fonction de score en classes de note.
4. Définition des correspondances entre ratings externes (notation d'agence) et notation interne.
5. Caractérisation des notes internes par des PD calculées à partir des moyennes historiques des probabilités de défaut issues des agences de notation (par exemple, S&P ou Moody's).

L'utilisation de l'AG ou du PSO intervient lors de la deuxième étape de modélisation de la probabilité de défaut et nous nous sommes donc uniquement intéressées à cette étape pour nos applications.

5.6.2 Optimisation à l'aide du tau-b de Kendall

Tau-b de Kendall

Nous utilisons le **Tau-b de Kendall** qui est un indicateur de corrélation des rangs et qui va donc nous permettre de comparer la différence entre l'ordonnancement des notes données par les experts et celui des notes données par le modèle de notation (i.e. la fonction de score). Les algorithmes d'optimisation développés (l'AG et le PSO) vont nous permettre d'optimiser cet indicateur.

Après avoir calculé la note donnée par la fonction de score [30] pour chaque entité de notre base de données, nous comparons les résultats obtenus au classement donné par les experts à l'aide du tau-b de Kendall.

Nous voulons donc comparer deux vecteurs Z_1 et Z_2 où Z_1 correspond au vecteur du classement donné par les experts et Z_2 au vecteur des notes données par la fonction de score pour un certain vecteur des poids β [31]. L'optimisation du tau-b de Kendall va nous permettre d'ajuster ce vecteur β afin d'obtenir une fonction de score g_β adaptée au portefeuille à modéliser.

Pour calculer le tau-b, nous regardons la différence entre le nombre de paires concordantes et le nombre de paire discordantes. Pour deux entités i et j dont les notes diffèrent, la paire de notes modélisées (z_{2_i}, z_{2_j}) est dite concordante avec la paire de notes expertes (z_{1_i}, z_{1_j}) si : si $z_{2_i} > z_{2_j}$ alors $z_{1_i} > z_{1_j}$ et si $z_{2_i} < z_{2_j}$ alors $z_{1_i} < z_{1_j}$. Sinon elle est dite discordante. Le tau-b de Kendall

28. Pour l'application aux Établissements Financiers, les experts ont attribué un rang (entre 1 et le nombre d'entités) à chaque entité tandis que pour l'application aux Secteur Public Local segment Santé, les experts ont attribué des notes (entre 1 et 7) aux entités.
29. Nous décrivons ici la méthodologie utilisée pour ces portefeuilles au sein de l'équipe Modélisation de LBP, il en existe donc d'autres.
30. Cf. section 4 p. 18 pour la définition générale d'une fonction de score.
31. $Z_2 = \left(g_\beta(z)\right)_{1 \leq z \leq M}$.

τ_b entre Z_1 et Z_2 est alors donné par :

$$\begin{aligned}\tau_b(Z_1, Z_2) &= \frac{N^{bre} \text{ de paires concordantes} - N^{bre} \text{ de paires discordantes}}{\sqrt{(N^{bre} \text{ de paires différenciées de } Z_1)\,(N^{bre} \text{ de paires différenciées de } Z_2)}} \\ &= \frac{\sum_{1\leq i,j\leq M,\, i<j} signe\big((z_{1_i} - z_{1_j})(z_{2_i} - z_{2_j})\big)}{\sqrt{\left(\binom{M}{2} - n_{Z_1}\right)\left(\binom{M}{2} - n_{Z_2}\right)}}\end{aligned}$$

Où :

- M est la dimension des vecteurs Z_1 et Z_2 i.e. le nombre d'entités observées dans notre base de données.
- n_{Z_1} (resp. n_{Z_2}) représente le nombre de paires où les observations de la variable Z_1 (resp. Z_2) sont égales.
- La fonction *signe* vaut 1 si les deux paires de notes, $(z_{1_i},\, z_{1_j})$ et $(z_{2_i},\, z_{2_j})$ sont concordantes, -1 si elles sont discordantes et 0 si $z_{1_i} = z_{1_j}$ ou $z_{2_i} = z_{2_j}$.

Le tau-b est compris entre -1 et 1. Plus le tau-b de Kendall est proche de 1, plus l'ordonnancement des deux vecteurs est proche. Au contraire, un tau-b proche de -1 indique que l'un des vecteur est classé en sens inverse de l'autre.

Données nécessaires à l'optimisation

Les données permettant de calculer le tau-b de Kendall sont donc :

Données	Type	Caractéristiques
$X_z = (X_1(z),\, X_2(z),\, \ldots,\, X_p(z))$ avec $1 \leq z \leq M$	M vecteurs de taille p	Vecteurs des valeurs de chacun des p facteurs explicatifs pour chaque entité z du portefeuille étudié.
$Y = (Y_1,\, Y_2,\, \ldots,\, Y_M)^t$	Vecteur de taille M	Vecteur de la note ou du rang de chaque entité.

Tableau 5.5 – Données nécessaires à l'optimisation

Pour rappel, nous voulons déterminer un vecteur $\beta = (\beta_1,\, \beta_2,\, \ldots,\, \beta_p)^t$ optimal permettant d'affecter un poids à chacun de nos p facteurs explicatifs i.e. nous voulons calibrer la fonction de score du portefeuille étudié grâce à un échantillon de M entités. La fonction de score est définie comme suit :

$$g_\beta(z) = \sum_{i=1}^{p} \beta_i \cdot X_i(z)\,.$$

Pour un vecteur $\beta = (\beta_1,\, \beta_2,\, \ldots,\, \beta_p)^t$ donné, nous pouvons donc calculer le tau-b de Kendall :

$$\tau_b(Y, \big(g_\beta(z)\big)_{1\leq z\leq M})\,.$$

La fonction que nous cherchons à optimiser avec nos algorithmes, aussi appelée fonction *fitness*,

est donc une fonction f du tau-b de Kendall [32] et qui dépend uniquement du vecteur des poids β :

$$\begin{array}{rccc} f & : & E & \to & \mathbb{R} \\ & & \beta & \mapsto & f(\beta)\,. \end{array}$$

Lien entre les variables à optimiser et le vocabulaire des algorithmes

Dans le tableau suivant, nous faisons le lien entre notre problème d'optimisation et les données générées par nos deux algorithmes, l'AG et le PSO.

Vocabulaire AG	Vocabulaire PSO	Type	Utilisation et caractéristiques
Gène	Variable	β_{i_j}	Poids à affecter à la variable explicative i pour calculer le tau-b. Chaque individu/particule S_j peut avoir un gène/une variable avec des valeurs différentes.
Individu	Position d'une particule	$S_j = (\beta_{1_j}, \beta_{2_j}, \ldots, \beta_{p_j})^t$ Vecteur des poids	S_j est un vecteur de taille p (le nombre de variables explicatives). C'est une solution potentielle du problème d'optimisation (i.e. potentiellement $\beta^* = S_j$). L'individu/la particule S_j est tiré aléatoirement dans E à l'initialisation puis évolue selon les méthodes de l'algorithme.
Population	Essaim	$\{S_1, S_2, \ldots, S_N\}$ Ensemble de N vecteurs des poids	Ensemble de N solutions potentielles. Le nombre N d'individus/de particules est un paramètre de l'algorithme, il est choisi par l'utilisateur. Un tau-b est calculé sur toutes les entités du portefeuille pour chaque individu dans la population/particule dans l'essaim (i.e. vecteur des poids S_j).

Tableau 5.6 – Variables de l'algorithme

L'AG et le PSO permettent donc de tester un ensemble de vecteurs β, solutions potentielles de notre problème d'optimisation, afin de déterminer les poids optimaux à affecter à chaque facteur explicatif. Pour rechercher un vecteur β optimal, nous devons paramétrer chacun de nos algorithmes en fonction des caractéristiques de notre problème d'optimisation.

5.7 Paramétrages des algorithmes et *grid search*

Comme nous avons pu le voir dans les sections 5.4.5 et 5.5.4, les paramètres utilisés pour initier l'algorithme peuvent avoir une influence sur la qualité des résultats obtenus. Le paramétrage choisi doit permettre d'approcher la solution optimale en un temps de calcul raisonnable. De plus, du fait du caractère stochastique de ces algorithmes, plusieurs tests doivent être réalisés avec différentes graines [33].

32. f ne correspond pas tout à fait au tau-b de Kendall, nous prenons notamment en compte une pénalité (cf. paragraphes 7.2.4 et 7.3.4).

33. Pour rappel, les processus (pseudo) aléatoires générés par Python dépendent de la graine (*seed* en anglais) choisie. La graine permet également de reproduire les résultats obtenus.

Dans la continuité de l'implémentation des algorithmes génétique et par essaim particulaire, nous avons donc également implémenté un module qui, à la manière d'un *grid search*[34], permet de tester, en parallèle[35], un large éventail de paramètres. Ce module a pour but d'aider l'utilisateur d'un algorithme d'optimisation à déterminer les paramétrages optimaux (i.e. permettant d'obtenir les meilleurs résultats pour un problème d'optimisation donné). En effet, comme il n'existe, généralement, pas de règles[36] quant au choix du paramétrage de ces algorithmes, il est nécessaire d'en tester plusieurs avant de pouvoir établir un résultat.

Le *grid search*, qui consiste donc à tester différentes combinaisons de paramètres simultanément, présente plusieurs avantages. Non seulement l'utilisation de cet outil permet un gain de temps considérable mais il permet aussi de couvrir un grand nombre de solutions potentielles et donc de mieux s'assurer que l'algorithme converge bien vers le résultat voulu (par exemple, le minimum global d'une fonction). Cet outil permet donc de remédier à l'un des désavantages des méthodes métaheuristiques utilisées[37].

De plus, le *grid search* présente un grand intérêt pour l'équipe Modélisation de LBP. En effet, afin de répondre aux exigences de validation et d'audit interne, l'équipe Modélisation doit prouver que la solution finale choisie est la meilleure et donc qu'elle a testé suffisamment de modèles ou paramétrages potentiels.

Les paramétrages et résultats présentés dans ce document ont été obtenus après avoir effectué un *grid search*.

34. I.e recherche par quadrillage.
35. I.e. simultanément.
36. Un paramétrage peut convenir à un problème donné et pas du tout à un autre.
37. Pour rappel, les méthodes métaheuristiques que sont l'AG et le PSO présentent le désavantage de ne pas garantir la convergence vers le minimum global (cf. section 5.2).

6 Construction d'un score à l'aide d'une régression logistique

Afin de modéliser la probabilité de défaut à l'entrée en relation client, nous avons utilisé la régression logistique qui est une des techniques de *scoring* la plus couramment utilisée (avec l'arbre de décision). Les techniques de *scoring* sont adaptées lorsque la variable à prédire est binaire (Oui/-Non). Ces techniques doivent permettre de classer les individus selon la probabilité de la réponse recherchée. Elles peuvent donc être utilisées pour des applications variées : score d'appétence d'un client à un produit (en marketing), score d'octroi, etc. La mesure de la performance d'un score sera différente selon le but recherché (ciblage marketing, évaluation du risque de crédit, ...).

Le score que nous cherchons à construire est un score d'octroi du produit « compte courant » et des moyens de paiement associés. Il doit nous permettre d'évaluer la probabilité de défaut associée à un nouveau client à partir des données à notre disposition. La méthodologie que nous présentons ci-après est donc adaptée à ce type de *scoring*.

Lors de notre étude sur une possible amélioration du modèle de notation (i.e score) sur les nouveaux clients à l'aide de données externes, nous avons suivi la méthodologie de construction des scores déjà utilisée au sein du département de Modélisation des risques de crédit de La Banque Postale. Nous décrivons, ici, uniquement la méthodologie utilisée pour construire le score. Les étapes de récupération et construction des bases de données seront présentées à la section 8.1 lors de l'analyse des résultats obtenus.

Pour effectuer notre étude, nous disposons d'une base de données composée de M observations, D variables explicatives et une variable binaire (0/1) indiquant si un défaut a été observé dans les douze mois suivant la date d'entrée dans le portefeuille.

6.1 Échantillonage

Nous devons effectuer plusieurs étapes préalables avant de pouvoir utiliser la régression logistique sur la base de données. Tout d'abord, nous séparons notre population en deux échantillons, puis, nous discrétisons toutes nos variables après avoir analysé leur qualité.

Nous disposons d'un nombre M d'observations assez élevé des variables étudiées pour pouvoir procéder à un échantillonage de notre base de données. Nous séparons donc celle-ci en deux échantillons :

- Un **échantillon de construction** (ou d'apprentissage) qui est utilisé pour construire le score (sélection des variables, découpages, ...) et qui est constitué de 70 % de la base totale.
- Un **échantillon de validation** qui est utilisé dans un deuxième temps pour contrôler la stabilité des résultats obtenus et qui est constitué des 30 % restants de la base totale. Ces clients ne servent donc pas à la construction du modèle.

L'échantillonnage est réalisé par tirage aléatoire, sans remise et stratifié selon la variable à prédire (le défaut) de manière à avoir la même proportion de sains et de défaillants dans chacune des sous-populations (i.e. le taux de défaut est le même dans la population de construction, la population de validation et la population totale).

Nous nous assurons de la représentativité des échantillons de construction et de validation au moyen d'un test du χ^2 : nous comparons, pour une dizaine de variables explicatives, l'écart entre la distribution empirique constatée sur l'échantillon testé et la distribution « théorique » de la population totale. Les échantillons sont considérés comme représentatifs de la population totale si la statistique du χ^2 est inférieure à un seuil de risque de 5 %.

Nous effectuons les étapes suivantes (préparation des variables et estimation de modèle) sur l'échantillon de construction.

6.2 Préparation des variables

Nous effectuons une étape de préparation des variables afin de nous assurer de la qualité des variables explicatives et de leur capacité à prédire le défaut. Les variables utilisées pour construire le modèle doivent, en effet, pourvoir permettre la généralisation du modèle à toute la population.

D'une façon générale, les variables disponibles ne sont pas toutes de même nature. Il convient donc de les classer suivant leur définition intrinsèque afin de déterminer le traitement qui doit leur être appliqué avant de pouvoir construire le modèle. Nous classons les variables selon deux types :

— variables qualitatives (situation familiale, niveau d'études, ...) ;

— variables quantitatives (montant des avoirs en épargne, taux de chômage des 15-24 ans par IRIS, ...).

Ainsi, nous distinguons le traitement à appliquer à une variable explicative d'un modèle en fonction de sa nature intrinsèque.

6.2.1 Analyse univariée

Dans un premier temps, nous effectuons une analyse univariée des variables explicatives à notre disposition. Cette étape nous permet d'identifier, pour chacune de nos variables, le pourcentage de valeurs :

— **Non applicables** : certaines valeurs non renseignées sont dues au fait que la variable ne s'applique pas à l'individu (par exemple, la variable peut concerner un produit non détenu par le client).

— **Manquantes** c'est à dire les valeurs non renseignées qui n'ont pas d'explication métier.

— **Aberrantes** c'est-à-dire des valeurs qui présentent un écart anormal par rapport aux autres valeurs observées dans un échantillon tiré aléatoirement. Les seuils à partir desquels une variable est considérée comme aberrante sont définis en relation avec les équipes métiers.

— **À exclure** : pour certaines variables, il existe des valeurs non aberrantes mais qu'il est tout de même nécessaire d'exclure car elles nécessitent un traitement spécial.

Nous écartons les variables explicatives avec une qualité jugée insuffisante c'est-à-dire que les variables présentant plus de 16 % de valeurs non applicables [1] ou plus de 10 % de valeurs aberrantes [2] sont exclues de notre analyse. Cette étape a aussi pour but de se familiariser avec nos variables afin de faciliter l'analyse bivariée.

1. La plupart des indicateurs issus de l'Insee ne sont pas disponibles pour 4 ou 15 % de l'échantillon (l'autre valeur rencontrée étant 99 %). Ces valeurs non renseignées ont un sens métier et nous considérons donc qu'elles n'altèrent pas la qualité de la variable en-dessous de 16 %.

2. C'est un choix de l'équipe Modélisation de LBP. Au-delà de 10 % de valeurs manquantes ou aberrantes, l'équipe Modélisation estime que la qualité de la variable n'est pas suffisante pour être représentative du défaut.

6.2.2 Analyse bivariée et découpage des variables

L'analyse bivariée (i.e. l'analyse du lien entre la variable à expliquer (le défaut) et chacune des variables explicatives) ainsi que le découpage de chaque variable sont effectués sur la sous-population des individus ayant une valeur renseignée et cohérente (i.e. les individus possédant pour la variable à discrétiser une valeur manquante, aberrante, à exclure ou non applicable sont mis de coté pour cette étape). Ces individus seront réintégrés à l'étape suivante selon les modalités précisées dans la section 6.2.3.

Variables qualitatives

Certaines variables explicatives qualitatives font l'objet de regroupements de modalités. Ces regroupements doivent permettre de gagner en :

— **Discrimination** : le regroupement choisi doit permettre de séparer au mieux la population des individus défaillants de la population des individus sains.

— **Robustesse** : le regroupement choisi doit être applicable à des populations qui n'ont pas servi à estimer le score et donc rester stable d'un échantillon à l'autre.

Afin de regrouper les modalités des variables qualitatives, deux méthodologies sont utilisées suivant la spécificité de chacune des variables :

— **Regroupement métier** : les modalités correspondant à une même réalité sont regroupées.

— **Regroupement statistique** : les modalités avec des taux de défaut proches au sens du test du χ^2 sont regroupées.

En général, nous utilisons la méthode du regroupement statistique pour effectuer le découpage des variables qualitatives, tout en nous assurant que le regroupement obtenu a un sens métier.

Variables quantitatives

Chaque variable explicative continue est discrétisée, c'est-à-dire que l'ensemble des valeurs prises par la variable est découpé en un nombre fini d'intervalles identifiés par un code, en fonction de sa relation avec la variable du défaut. Ce traitement est effectué afin de :

— prendre en compte la non-monotonie ou non-linéarité du lien existant entre la variable explicative et le défaut ;

— supprimer l'influence des valeurs extrêmes lors de l'estimation du modèle ;

— faciliter l'interprétation métier des grilles de score ;

— faciliter l'insertion informatique des modèles construits.

Les raisons listées ci-dessus expliquent aussi notre choix d'utilisation d'une régression logistique pour modéliser le défaut.

Deux méthodes de découpage implémentées dans des programmes automatisés [3] ont été mises en concurrence pour retenir le découpage offrant les meilleures propriétés :

— Méthode ***split*** : c'est une méthode descendante qui part de l'intervalle complet des valeurs prises par la variable à discrétiser. Lors de la première étape, elle examine tous les points de coupure possibles pour scinder l'intervalle en deux classes et retient le point de coupure

3. Nous avons utilisés des macro-programmes SAS mis en place par l'équipe Modélisation de LBP.

qui maximise le χ^2 d'indépendance avec la variable à expliquer (le défaut). Lors des étapes suivantes, l'algorithme est appliqué itérativement sur les classes déjà constituées sans modifier les points de coupure déjà identifiés.

— Méthode ***merge*** : c'est une méthode ascendante qui part d'intervalles élémentaires et cherche la fusion de deux intervalles adjacents qui sont statistiquement similaires. La similarité est mesurée à l'aide du χ^2 d'indépendance entre la variable constituée des deux intervalles adjacents et le défaut.

Ces deux algorithmes nous donnent chacun les meilleurs découpages en 2 à 10 classes [4] qu'ils ont trouvés. Le choix du découpage utilisé pour discrétiser une variable repose sur des critères statistiques mais aussi sur des raisons métier. Le découpage choisi est celui qui respecte au mieux les critères suivants :

— Le découpage est le plus fin possible i.e. il maximise le nombre de modalités afin de séparer autant que possible les populations. L'inconvénient théorique de cette approche est qu'il est possible d'avoir un découpage qui ne soit pas stable dans le temps.

— La différenciation entre les taux de défaut de deux classes consécutives est suffisante. Pour cela, nous nous basons sur un test de Student qui permet de mesurer la similarité des taux de défaut entre deux modalités. Si la p-value du test de Student est supérieure à 5 %, nous rejetons l'hypothèse de différenciation.

— Aucune modalité n'a un effectif inférieur à 2 % de la population de construction (cette condition, empirique, vise à se prémunir de découpages instables). Notons que exceptionnellement et pour des besoins métier ce critère peut ne pas être respecté.

— Les taux de défaut sont monotones. Ce critère provient du constat empirique que la relation qui unit les variables explicatives au défaut est généralement monotone. Il existe toutefois des exceptions et nous pouvons exceptionnellement accepter un découpage présentant une bosse ou un creux.

6.2.3 Traitement des valeurs non renseignées ou aberrantes

Lors de cette étape, les valeurs non renseignées ou aberrantes sont réintroduites et imputées en fonction de leurs caractéristiques :

— pour les valeurs non applicables, une modalité spécifique est créée ;

— pour les valeurs manquantes :

- si les taux de défaut sont estimés de manière robuste, nous imputons les valeurs manquantes à la modalité avec le taux de défaut le plus proche [5] ;
- sinon, nous imputons les valeurs manquantes à la modalité la plus fréquemment rencontrée dans la population ;

— pour les valeurs aberrantes : nous appliquons les mêmes règles que pour les valeurs manquantes.

4. C'est un choix de l'équipe Modélisation de LBP. Nous nous arrêtons à un découpage à 10 classes car nous considérons qu'au-delà le découpage ne pourrait pas respecter les critères fixés.

5. Nous utilisons une variante de la technique d'imputation dite du « hot deck métrique » où, pour une variable discrétisée, la classe « manquants » est affectée à la classe la plus proche au sens de la distance « taux de défaut ». Pour un panel de traitements des valeurs manquantes se référer par exemple à PÉRIÉ (2014).

Nous considérons que les taux de défaut sont estimés de manière robuste (volumétrie suffisante) pour un groupe de valeurs manquantes si :

— il existe plus de 30 observations manquantes ;

— il existe plus de 5 individus en défaut et plus de 5 individus sains dans la population des manquants ce qui revient à évaluer l'expression suivante : $\min\big(n \cdot t_d\,;\, n \cdot (1 - t_d)\big) > 5$ avec n, le nombre d'observations manquantes et t_d, le taux de défaut de ces individus.

De même pour les valeurs aberrantes.

6.3 Estimation des modèles

Une fois les variables explicatives candidates définies, nous cherchons à estimer un score qui traduit la probabilité de tomber en défaut dans les 12 mois suivants l'entrée en relation. Pour cela, nous utilisons la régression logistique qui est une technique d'apprentissage supervisé. Cette technique appartient à la famille des modèles linéaires généralisés.

6.3.1 Régression logistique [6]

Problématique

Soit $Y = (Y_1, Y_2, \ldots, Y_m)^t$, le vecteur des variables réponses des m individus observés. Y est une variable aléatoire binaire, indiquant si un individu est ou non défaillant (la valeur 1 représente le défaut).

Soit j tel que $1 \leq j \leq m$ et soit $X_j = (X_{j,1}, X_{j,2}, \ldots, X_{j,d})^t$, l'ensemble des caractéristiques des d variables explicatives sélectionnées de chaque individu j.

Nous voulons réussir à prédire l'appétence au défaut d'un individu z dont les caractéristiques sont connues. Nous cherchons donc à estimer la probabilité de défaut (i.e d'avoir $Y = 1$) sachant les valeurs prises par les variables discrètes $X_1, X_2, \ldots, X_d$. Ainsi, nous voulons modéliser, pour un individu z dont les caractéristiques $x_z = (x_1, \ldots, x_d)$ sont connues, la probabilité a posteriori suivante :

$$\pi(z) = \mathbf{P}\big(Y = 1 | X = x_z\big)\,.$$

La variable $Y|X = x_z$ suit une loi de Bernoulli de paramètre $p = \mathbf{P}\big(Y|X = x\big)$ qui est le paramètre que nous cherchons à estimer. Le modèle « logit » est donc adapté à la modélisation de la probabilité π.

Équation

La régression logistique est une technique qui permet d'exprimer le logit de $\pi(z)$ sous la forme d'une combinaison linéaire des variables $X_1, \ldots, X_d$ c'est-à-dire que nous cherchons des constantes, $\beta_0, \beta_1, \ldots, \beta_d$, telles que :

$$\forall z,\, \text{logit}\big(\pi(z)\big) = \ln\left(\frac{\pi(z)}{1 - \pi(z)}\right) = \beta_0 + \beta_1 \cdot X_{z,1} + \beta_2 \cdot X_{z,2} + \cdots + \beta_d \cdot X_{z,d} = X_z\beta$$

6. Cette section ainsi que les 2 suivantes s'appuient en partie sur les documents suivants : RAKOTOMALALA (2017), SAS (2010).

avec $\beta = (\beta_0, \beta_1, \beta_2, \ldots, \beta_d)$ et $X_z = (1, X_{z,1}, \ldots, X_{z,d})^t$. Il est facile de retrouver $\pi(z)$ à partir de l'équation précédente :

$$\pi(z) = \mathbf{P}\big(Y = 1 | X_z\big) = \frac{\exp(X_z\beta)}{1 + \exp(X_z\beta)}\,.$$

Estimation

Les constantes $\beta_0, \beta_1, \ldots, \beta_d$ sont estimées par maximum de vraisemblance. La log-vraisemblance associée à l'échantillon de nos m individus est la suivante :

$$\mathcal{L}(Y, \beta) = \sum_{z=1}^{m} \Big(Y_z \ln\big(\pi(z)\big) + (1 - Y_z) \ln\big(1 - \pi(z)\big) \Big) = \sum_{z=1}^{m} \Big(Y_z \big(X_z\beta\big) + \ln\big(1 - \exp(X_z\beta)\big) \Big)\,.$$

Nous cherchons à maximiser la log-vraisemblance i.e. nous cherchons à optimiser les coefficients β de manière à avoir :

$$\frac{\partial \mathcal{L}}{\partial \beta} = 0\,.$$

Nous obtenons une solution approchée des constantes $\beta_0, \beta_1, \beta_2, \ldots, \beta_d$ grâce à l'algorithme de Fisher [7].

Les coefficients $\beta_0, \beta_1, \beta_2, \ldots, \beta_d$ sont déterminés à partir de l'échantillon de construction. L'échantillon de validation servira ensuite à s'assurer de la stabilité structurelle du modèle construit.

6.3.2 Sélection de variables

Nous voulons sélectionner les p variables qui, parmi les d variables à notre disposition, expliquent le mieux (i.e. les variables les plus discriminantes) le défaut. En effet, la sélection de variables va nous permettre d'éliminer les variables n'ayant aucune ou très peu d'influence sur le défaut ou encore de choisir la variable la plus discriminante parmi plusieurs variables transmettant plus ou moins la même information.

Pour choisir ce nombre p nous appliquons la méthode dite « *stepwise* » de sélection des variables afin de déterminer l'ordre d'apparition dans le modèle des 30 meilleures variables [8].

La méthode *stepwise* est un mélange entre deux méthodes :

— La méthode *forward* : cette méthode consiste à sélectionner les variables explicatives étape par étape en partant d'un modèle à 1 variable jusqu'à obtenir i variables. À chaque étape, la variable explicative ayant la statistique du χ^2 la plus élevée est sélectionnée parmi les variables restantes. Si aucune variable n'a une statistique du χ^2 assez élevée, la sélection s'arrête.

— La méthode *backward* : cette méthode part d'un modèle composé de toutes les variables à disposition et élimine à chaque étape la variable la moins significative par rapport au test de Wald jusqu'à obtenir i variables. Si toutes les variables obtiennent des valeurs au dessus d'un certain seuil de significativité, aucune variable n'est éliminée et la sélection s'arrête.

7. Cet algorithme est implémenté dans la procédure SAS « PROC LOGISTIC » que nous utilisons pour notre estimation de modèle.

8. C'est un choix de l'équipe Modélisation de LBP. Nous nous arrêtons à 30 variables car nous considérons que l'ajout de variables supplémentaires n'améliorerait pas significativement les résultats obtenus.

La méthode *stepwise* consiste à sélectionner à chaque étape une variable à l'aide de la méthode *forward* tout en regardant si une des variables précédemment sélectionnée n'a pas perdu en significativité avec l'ajout d'autres variables (méthode *backward*) ; si c'est le cas la variable est retirée du modèle.

Cette méthode nous indique une combinaison de variables possibles qui va nous permettre de construire un modèle à p variables en combinant les résultats statistiques fournis par la méthode et nos connaissances métiers.

6.3.3 Évaluation de la qualité du modèle

Nous utilisons le D de Somer, aussi appelé indice de Gini, pour comparer la performance des modèles obtenus.

Cet indicateur calcule la corrélation de rang entre les réponses observées Y et les probabilités prédites $\hat{\pi}$ pour chaque individu après estimation des coefficients β. Nous nous intéressons aux paires d'individus (z_1, z_2) présentant des valeurs de la variable réponse Y différentes. Une paire d'individu (z_1, z_2) est dite concordante si l'observation pour laquelle $Y = 0$ a été observé a une plus petite probabilité prédite $\hat{\pi}$ que celle où la réponse $Y = 1$ a été observée. Dans le cas inverse, la paire est dite discordante. Le D de Somer se calcule alors comme suit :

$$D = \frac{N^{bre} \text{ de paires concordantes} - N^{bre} \text{ de paires discordantes}}{N^{bre} \text{ de paires avec des réponses différentes}}.$$

Plus le D de Somer est important, plus le modèle sera considéré comme performant.

6.4 Justification des choix méthodologiques

Le choix de cette méthodologie [9] est motivé par la nature de la variable à expliquer qui possède deux modalités : défaillance/non défaillance. D'autres avantages confortent le choix de cette méthode statistique :

- La réduction du nombre de variables permet de mieux qualifier le risque de défaut et facilite l'interprétation des coefficients obtenus par régression logistique et donc l'interprétation métier.
- La discrétisation des variables facilite l'implémentation dans les systèmes (les intervalles de valeurs pour chaque variable sont facilement implémentables sous forme de règles).
- Les résultats obtenus sont facilement interprétables.
- La sélection de variables diminue le risque de sur-apprentissage qui ne permettrait pas la généralisation du modèle.

L'utilisation de la régression logistique répond donc à la fois aux attentes réglementaires d'appel à des techniques de *scoring* fiables et performantes, mais aussi à la nature du portefeuille modélisé.

9. Discrétisation des variables, méthode de sélection des variables et régression logistique.

Troisième partie

Applications aux portefeuilles de La Banque Postale

Plan de la troisième partie

Dans cette partie, nous présentons les résultats que nous avons obtenus suite à l'application des techniques détaillées dans la partie précédente pour, d'une part, la gestion du risque des portefeuilles *Low Default* et, d'autre part, la gestion du risque des portefeuilles à l'entrée en relation client.

Dans un premier temps, nous nous intéressons à la modélisation de la probabilité de défaut sur les portefeuilles *Low Default*. Après avoir présenté la méthodologie que nous utilisons pour construire un modèle de notation, nous nous attachons à l'application de l'algorithme génétique (dont le fonctionnement est présenté à la section 5.4) aux Établissements Financiers puis à l'application de l'algorithme d'optimisation par essaim particulaire (présenté à la section 5.5) au Secteur Public Local segment Santé. Enfin, nous analysons la performance de nos algorithmes en croisant ces deux applications.

Dans un deuxième temps, nous présentons les techniques alternatives testées afin de modéliser la probabilité de défaut à l'entrée en relation avec une clientèle *retail*. Tout d'abord, nous détaillons les étapes effectuées pour construire notre base de données (issues de sources interne et externe) puis nous appliquons la méthodologie de construction d'un score à l'entrée en relation à l'aide d'une régression logistique (présentée au chapitre 6). Enfin, nous utilisons deux méthodes alternatives : un arbre de décision permettant de réaliser une étude directe des classes de risque et une sélection de variables par algorithme génétique permettant de challenger la sélection de variable *stepwise*.

7 Modélisation de la probabilité de défaut sur des portefeuilles *Low Default*

Dans cette partie, nous présentons les résultats que nous avons obtenus suite à l'application :

— de l'algorithme génétique (AG) pour la construction d'un modèle de notation sur le portefeuille Établissements Financiers ;

— de l'algorithme d'optimisation par essaim particulaire (PSO) pour la construction d'un modèle de notation sur le portefeuille Secteur Public Local segment Santé.

Nous analysons, d'une part, la performance de ces méthodes sur les portefeuilles étudiés et, d'autre part, nous comparons les performance des algorithmes (AG et PSO) que nous avons implémentés en Python ainsi que les performances pouvant être obtenues avec des packages sur R ou Matlab.

L'implémentation et le codage de ces deux algorithmes ne sont pas détaillés dans ce document (les programmes développés étant confidentiels). Un résumé des fonctions et méthodes construites est toutefois donné en annexe C.

Les résultats que nous avons obtenus étant confidentiels, ceux-ci ont été modifiés de façon à obtenir une analyse identique. De plus, le descriptif des variables utilisées n'est pas mis à disposition (les variables explicatives sont appelées $X_1, \ldots, X_p$ sans plus de précision).

Les modèles de notation sur les portefeuilles Établissements Financiers et Secteur Public Local segment Santé ont été développés en lien avec les experts métiers et certains choix de modélisation reflètent leurs avis.

7.1 Rappel de la méthodologie

Nous rappelons ici les grandes lignes des techniques utilisées pour modéliser la probabilité de défaut des deux portefeuilles *low Default* étudiés, l'utilisation que nous faisons de nos algorithmes est détaillée plus amplement à la section 5.6.

7.1.1 Approche par vérification

Comme présenté à la section 5.6.1, nous utilisons une approche par vérification qui consiste à calibrer un modèle de notation qui permet de reproduire le raisonnement effectué par les experts de La Banque Postale pour noter les entités des portefeuilles Établissements Financiers et Secteur Public Local segment Santé.

Pour chaque application, nous calibrons les poids de la fonction de score (dont la forme générale est présentée à la section 4) à l'aide de nos algorithmes d'optimisation, l'AG et le PSO, en estimant la performance du modèle trouvé à l'aide du tau-b de Kendall (cf. section 5.6.2). Le tau-b nous permet en effet de comparer le vecteur des notes expertes au vecteur des notes calculées pour un certain vecteur des poids β.

Pour rappel, les variables explicatives utilisées dans la fonction de score sont les mêmes que celles utilisées par les experts pour effectuer leur classement.

7.1.2 Périmètre de notre application

Dans les applications suivantes, nous nous intéressons uniquement à la calibration de la fonction de score. Les étapes préliminaires de récupération, préparation et sélection des données ne sont pas traitées ainsi que les étapes suivantes de découpage de la fonction de score en classes de notes (auxquelles sont associées des probabilités de défaut) et d'analyse de la stabilité du modèle.

En effet, notre but étant de s'assurer de la bonne performance des algorithmes implémentés en Python, nous avons repris les travaux déjà effectués par l'équipe Modélisation de LBP afin de pouvoir comparer la performance de nos algorithmes « internes » à la performance qu'il est possible d'obtenir avec des packages « externes » déjà implémentés sur R ou Matlab.

7.1.3 Utilisation de *grid search*

Les résultats présentés dans les sections suivantes de ce chapitre ont été obtenus par *grid search* (i.e. recherche par grille). Grâce au module de *grid search* que nous avons construit (cf. section 5.7), nous pouvons en effet tester une grande variété de paramétrages afin de s'assurer de la convergence de nos algorithmes d'optimisation, l'AG et le PSO.

7.2 Application aux Établissements Financiers

La modélisation de la probabilité de défaut sur le portefeuille Établissements Financiers a été réalisée en prenant en compte le contexte réglementaire « Bâle II ». À cette occasion, la performance de la fonction de score avait été optimisée grâce à un algorithme génétique. Celui-ci avait été appliqué en utilisant le solveur *ga* de la *Global Optimization Toolbox* du langage de programmation Matlab. Nous avons donc repris le travail effectué lors de la construction de ce modèle pour tester l'algorithme génétique que nous avons implémenté en Python.

7.2.1 Contexte réglementaire

Le modèle de notation des Établissements Financiers a été construit dans l'optique de se conformer aux exigences réglementaires « Bâle II » dans le cadre du développement d'un modèle interne.

Conformément aux seconds accords bâlois, le dispositif IRB (Internal Rating Based Approach) applicable à l'ensemble des classes d'actifs doit se baser sur des éléments historiques et empiriques, et les résultats qui en découlent doivent être régulièrement comparés aux probabilités de défaut observées. De plus, les travaux réalisés doivent être reproductibles et auditables. En effet, les modèles construits par l'équipe Modélisation de LBP sont ensuite doublement audités (par l'équipe Validation puis par l'Inspection Générale).

L'évaluation du risque de crédit de chaque entité du portefeuille s'appuie sur plusieurs paramètres dont la probabilité de défaut (PD) [1]. Le calcul de ces paramètres doit servir à déterminer les exigences de fonds propres associées à chaque entité de chaque portefeuille de la banque [2].

Les estimations de la PD sont établies pour chaque « échelon de débiteur » de l'échelle de notation. Pour pouvoir estimer la probabilité de défaut des entités du portefeuille Établissements

1. Les modalités de calcul des différents paramètres sont décrites dans le règlement (UE) N° 575/2013 aussi appelé CRR (Capital Requirement Regulation).
2. La formule de calcul des RWA (Risk Weighted Asset) dépend notamment des PD modélisées en interne dans le cadre de l'approche IRB.

Financiers, il faut donc construire un modèle de notation qui permet « l'évaluation du risque de crédit, l'affectation des expositions à un échelon donné [...] et la quantification de la probabilité de défaut [...] » (définition donnée dans le règlement CRR).

7.2.2 Particularité du portefeuille Établissements Financiers

Le portefeuille Établissements Financiers, qui nous intéresse ici, n'a connu qu'un défaut historique (cf. Lehman Brothers) ce qui rend compliqué l'application d'une méthode classique d'approche par validation. D'où la nécessité de développer des techniques alternatives permettant une approche par vérification.

7.2.3 Données à disposition

Nous disposons d'un échantillon de 109 établissements financiers pour calibrer le modèle de notation [3]. Nous avons la valeur prise par chacune de ces entités pour les 14 facteurs retenus après analyse des données [4]. Ces variables peuvent prendre des valeurs entre 0 et 100.

Nous avons également à disposition le rang, attribué par les experts métiers de LBP, de chaque entité de l'échantillon : les établissements financiers sélectionnés pour la calibration du modèle sont classés de 1 à 109 par risque de crédit croissant (i.e. l'entité avec la probabilité de défaut la plus faible a le rang 1).

Nous voulons calibrer une fonction de score permettant d'attribuer une note comprise entre 0 et 100 à chaque établissement financier en portefeuille. Pour cette application, nous considérons que plus la note de l'entité est grande (proche de 100) et plus l'entité sera considérée comme un « bon » établissement (i.e. son risque de crédit sera faible).

Nous voulons calibrer une fonction de score qui permettra de reproduire le mieux possible l'ordonnancement des entités donné par les experts (par exemple, l'entité avec le rang 3 doit avoir une note plus grande que l'entité avec le rang 4).

7.2.4 Fonction à optimiser

Nous voulons calibrer le vecteur $\beta = (\beta_1, \ldots, \beta_{14})$ des poids à affecter à chacun des facteurs. Les valeurs des poids $\beta_1, \ldots, \beta_{14}$ doivent permettre d'identifier les variables les plus explicatives du défaut.

La fonction de score est bornée entre 0 et 100. Les facteurs explicatifs prenant eux-aussi des valeurs entre 0 et 100, le vecteur des poids doit vérifier la contrainte suivante :

$$\sum_{i=1}^{14} \beta_i = 1\,.$$

3. Des établissements financiers ont été rajoutés en plus des établissements en portefeuille afin de calibrer le modèle sur des établissements avec des risques de crédit variés (qualité de crédit dégradée ou à l'inverse particulièrement favorable).

4. Les données utilisées pour construire le modèle de notation proviennent essentiellement des rapports annuels.

Pour satisfaire cette contrainte, nous avons ajouté la pénalité suivante à la fonction à optimiser [5] :

$$10 \cdot \left(\sum_{i=1}^{14} \beta_i - 1 \right)^2 .$$

L'ajout de cette pénalité permet d'affecter une *fitness* plus élevée aux individus qui ne vérifient pas la contrainte sur la somme des poids.

Pour cette application, notre but est de minimiser le tau-b de Kendall entre les notes expertes et les notes modélisées car plus le tau-b de Kendall sera proche de -1, plus l'ordonnancement des notes données par la fonction de score sera proche de celui donné par les experts. En effet, les rangs attribués par les experts vont de 1 à 109, 1 étant le meilleur tandis que les notes données par la fonction de score vont de 0 à 100, 100 étant la meilleure. Nous voulons donc qu'il y ait un désaccord parfait entre les 2 ordres (i.e. que le classement croissant par rang corresponde au classement décroissant par note).

La fonction objectif $f : E \to \mathbb{R}$ que nous voulons minimiser est donc la suivante :

$$f(\beta) = \tau_b\Big(R_e,\, \big(g_\beta(z)\big)_{1 \leq z \leq 109}\Big) + 10 \cdot \left(\sum_{i=1}^{14} \beta_i - 1 \right)^2$$

Où :

— R_e est le vecteur donnant, pour chaque entité, le rang attribué par les experts. R_e est fixe et ne dépend donc pas de β.

— $\big(g_\beta(z)\big)_{1 \leq z \leq 109}$ est le vecteur donnant, pour chaque entité z, la note attribuée par la fonction de score g_β. Ce vecteur dépend de β et doit être calculé à chaque évaluation de la fonction f.

L'espace de recherche E dans lequel f prend ces valeurs est défini par $E = \bigotimes_{i=1}^{14} [b_{inf_i}; b_{sup_i}]$ où les bornes inférieures et supérieures pour chaque poids sont décrites dans le tableau 7.1 suivant [6].

7.2.5 Paramétrage de l'optimisation par algorithme génétique

Pour chaque paramètre de l'algorithme génétique, nous avons testé l'ensemble des valeurs présentées dans le tableau 7.2 suivant.

5. Nous avons repris le choix de fonction de pénalité effectué par l'équipe Modélisation de LBP. L'ajout d'une pénalité pour optimiser sous contrainte est une des méthodes couramment appliquées lors de l'optimisation par AG ou PSO. Cf. MEZURA-MONTES & COELLO COELLO (2011) pour un recensement de techniques utilisées pour l'optimisation sous contraintes.

6. Ces bornes ont été choisies, lors de la modélisation initiale, en accord avec les experts métiers.

Poids	Valeur min	Valeur max
β_1	0,05	0,25
β_2	0,05	0,25
β_3	0,025	0,15
β_4	0,05	0,25
β_5	0,025	0,15
β_6	0,025	0,15
β_7	0,05	0,25
β_8	0	0,1
β_9	0	0,1
β_{10}	0,025	0,15
β_{11}	0	0,1
β_{12}	0	0,1
β_{13}	0,025	0,15
β_{14}	0	0,1

TABLEAU 7.1 – Contraintes d'espace pour l'application de l'AG aux Établissements Financiers

Paramètre	Valeurs testées		
Graine	3	5	
Taille de la population	50	200	500
Nombre de générations	50	200	500
Taux d'élites	0	0,05	0,5
Pas de sélection	0,6	0,9	1,1
Probabilité de croisement	0,5	0,8	1
Probabilité de mutation	0,05	0,1	0,5

TABLEAU 7.2 – Ensemble des paramètres testés pour l'application de l'AG aux Établissements Financiers

7.2.6 Fonctionnement de l'algorithme génétique

Avant de commencer la recherche du meilleur vecteur des poids β, l'algorithme génétique doit générer une population de N individus ($N = 50$, 200 ou 500 en fonction du paramétrage). La population initiale de l'algorithme est générée aléatoirement c'est-à-dire que N individus S_j ($1 \leq j \leq N$) sont tirés au hasard dans l'ensemble E.

Dans cette application, un individu S_j est constitué de 14 gènes $\beta_{1_j}, \beta_{2_j}, \ldots, \beta_{14_j}$ qui ont chacun leur espace de définition. Ces gènes correspondent aux poids que nous associons à chacune de nos variables explicatives $X_1, X_2, \ldots, X_{14}$ pour calculer la valeur de la fonction f. La *fitness* $f(S_j)$ associée à chaque individu permet de caractériser ceux-ci au sein de la population : les bons individus étant ceux possédant la *fitness* la plus faible.

Au fil des générations, les meilleurs individus de la population vont se reproduire entre eux (croisement aléatoire des gènes de 2 individus) afin de former de nouveaux individus. Les gènes

peuvent également subir des mutations (tirage aléatoire d'un nouveau gène) [7]. La solution à notre problème d'optimisation est le meilleur individu de la dernière génération (i.e. l'individu S_j ayant la plus petite *fitness* de la population).

7.2.7 Résultats obtenus

Après avoir testé toutes les combinaisons des paramètres définis dans le tableau ci-dessus, nous obtenons donc 1458 résultats. Les 5 meilleurs résultats obtenus suite à ce panel de tests sont présentés dans le tableau ci-après. Ces résultats sont donnés sous la forme de la différence observée avec les résultats du modèle sur les Établissements Financiers obtenus sur Matlab.

Essai	**1**	**2**	**3**	**4**	**5**
Paramétrage					
Graine	5	5	3	3	3
Taille population	500	500	500	500	500
N^{bre} *générations*	500	200	500	200 (ou 500)	500
Taux élites	0,05	0,05	0,05	0,05	0
Pas sélection	0,6	0,6	1,1	0,6	0,6
Proba croisement	1	1	1	1	0,5
Proba mutation	0,1	0,1	0,05	0,1	0,1
Résultats (%)					
Poids 1	1,80	1,80	0,29	−0,34	0,95
Poids 2	3,19	3,29	2,70	2,72	3,14
Poids 3	−0,04	−0,04	0,12	0,17	0,15
Poids 4	−2,44	−2,44	−1,59	−1,67	−2,22
Poids 5	−0,78	−0,78	−1,37	−1,10	−1,25
Poids 6	0,28	0,28	0,38	0,20	0,05
Poids 7	−3,68	−3,68	−0,54	−0,54	−3,08
Poids 8	−0,06	−0,06	0,17	0,44	0,04
Poids 9	1,41	1,41	−1,82	−1,65	1,51
Poids 10	−0,39	−0,39	3,80	4,16	2,33
Poids 11	−0,05	−0,05	0,22	0,15	−0,09
Poids 12	−0,02	−0,02	−0,04	0,06	0,76
Poids 13	−0,82	−0,81	−2,57	−2,45	−2,31
Poids 14	1,61	1,61	−0,05	−0,16	−0,06
Tau-b	**0,30**	**0,30**	**0,19**	**0,17**	**0,17**
Temps	15 min	5 min	15 min	5 (ou 15) min	15 min

TABLEAU 7.3 – Meilleurs résultats obtenus pour l'application Établissements Financiers après *grid search* des paramètres de l'AG

Exemple de lecture du tableau :

7. Les méthodes de l'algorithme génétique sont décrites plus précisément à la section 5.4.3.

— Le poids 1 obtenu pour l'essai 1 est 1,80 % plus élevé que le poids de référence obtenu avec Matlab : par exemple, si le poids 1 « Matlab » est de 0,15, notre poids 1 sera alors de 0,168.

— Le tau-b obtenu pour l'essai 1 est 0,30 % meilleur que le tau-b de référence obtenu sur Matlab : par exemple, si le tau-b « Matlab » est de -0,95, notre tau-b sera alors de -0,9530.

Les résultats obtenus avec notre AG implémenté en Python sont satisfaisants car ils sont très proches de ceux trouvés sur Matlab. Plusieurs combinaisons très différentes des paramètres testés donnent des résultats concluants.

Par exemple, avec une population de 50 individus, un taux d'élites de 0, un nombre de générations de 500, un pas de sélection de 0,9, un taux de croisement de 1 et un taux de mutation de 0,05, nous obtenons un tau-b à -0,10 % du tau-b de référence avec une graine de 3 et un tau-b à -0,31 % avec une graine de 5 avec un temps d'exécution inférieur à 2 minutes.

De même, il est possible d'obtenir des résultats satisfaisants en un temps inférieur à 2 min, avec un nombre maximal de générations fixé à 50 et ce même avec une population de 500 individus. Les résultats obtenus avec le paramétrage taux d'élites à 5 %, pas de croisement de 0,6, probabilité de croisement de 1 et taux de mutation à 0,1 ont un tau-b à -0,36 % du tau-b de référence pour une graine de 3 et un tau-b à -0,12 % pour une graine de 5.

De façon générale, pour les paramétrages testés, nous avons pu remarquer qu'avec un taux de mutation fixé à 0,5, nous n'obtenons pas des résultats très satisfaisants (le meilleur tau-b obtenu a une différence de -1 % avec le tau-b Matlab de référence). Un taux de mutation assez faible (inférieur à 50 %) semble, pour ce problème au moins, être plus adapté.

7.3 Application au Secteur Public Local segment Santé

L'algorithme PSO, via package *pso* de R, a été utilisé dans le cadre de la notation du Secteur Public Local (SPL) segment Santé et fait l'objet de l'application du package que nous avons développé ultérieurement en Python. Dans un premier temps, nous exposons le contexte qui a amené La Banque Postale a construire un modèle de notation sur ce portefeuille. Puis, nous présentons les résultats obtenus suite à l'application de notre package.

7.3.1 Contexte réglementaire

Le 1er janvier 2018, la réglementation comptable Ifrs 9 a remplacée la norme Ias [8] 39. La norme Ifrs 9 s'applique à toutes les entreprises cotées sur les marchés européens [9] afin d'harmoniser la comptabilisation des instruments financiers.

Dans le cadre de l'application de la réglementation Ifrs 9, La Banque Postale doit calculer des provisions qui prennent en compte le risque de crédit de ses différents portefeuilles. Plus précisément, la banque doit provisionner les pertes futures attendues pour risque de crédit ou *Expected Credit Loss* (ECL). Cette valeur est fonction, pour chaque entité du portefeuille, de 3 paramètres [10] : la PD, l'EAD (*Exposure At Default*) et la LGD (*Loss Given Default*).

8. International Accounting Standards Board.

9. Les normes Ifrs ont également été adoptées par d'autres pays tel que le Canada, le Brésil, l'Australie ou le Japon.

10. Ces paramètres sont les mêmes que dans les accords de Bâle mais leurs conditions de calculs dans le cadre « Ifrs 9 » diffèrent.

Dans le cadre du calcul de la PD « IFRS 9 » sur le portefeuille Secteur Public Local segment Santé, La Banque Postale a donc du élaborer un modèle de notation de ce portefeuille. En effet, auparavant, le risque de crédit de ce portefeuille était évalué par les experts métiers de La Banque Postale. Afin d'octroyer une note, les experts métiers s'appuient sur les comptes financiers des contreparties. L'objectif de la modélisation du score pour le portefeuille SPL Santé est de noter l'ensemble des entités constituant celui-ci en utilisant une méthode d'optimisation par algorithme PSO, appliquée à ces mêmes comptes financiers, afin de reproduire le plus fidèlement possible ces « notes expertes ». Lorsque le modèle est calibré sur l'échantillon noté à dire d'expert, il est alors possible d'appliquer ce modèle à l'ensemble des contreparties du segment.

7.3.2 Particularité du portefeuille Secteur Public Local segment Santé

Les entités appartenant au Secteur Public Local segment Santé [11] forment un portefeuille *Low Default* car les agences considèrent généralement que les centres hospitaliers publics bénéficieraient d'un fort soutien de l'État en cas de besoin (et ce même si le gouvernement français ne garantit pas explicitement la dette des établissements publics de santé). En effet, compte tenu de leur statut [12], l'actif et le passif des centres hospitaliers publics ne peuvent pas être liquidés ou transférés à une autre contrepartie que l'État.

7.3.3 Données à disposition

L'étude des données a déterminé 8 facteurs explicatifs [13] du défaut pour la calibration du modèle de notation sur le portefeuille SPL Santé. Nous souhaitons calibrer les poids optimaux pour chacun de ces facteurs à l'aide de l'algorithme PSO développé en Python.

Nous disposons des notes attribuées par les experts de La Banque Postale et des valeurs prises par chacun des 8 facteurs sur un échantillon de 275 entités.

La note donnée par les experts est comprise entre 1 et 7. Les entités notées 1 étant celles qui présentent un risque de crédit plus faible.

Les 8 variables à notre disposition ont été normalisées et prennent donc des valeurs entre 0 et 1.

La fonction de score que nous souhaitons calibrer doit nous permettre d'obtenir une note comprise entre 0 et 100 pour chacune des 275 entités étudiées. À l'inverse de l'application précédente, une note proche de 0 indique que l'entité a un risque de crédit faible. Nous voulons donc que, par exemple, les entités notées 3 par les experts aient des notes modélisées plus faibles que les entités notées 4 par les experts.

11. Le Secteur Public Local (SPL) contient les portefeuilles « Communes », « Bailleurs Sociaux », « Santé » et « GSFP » (Groupements Sans Fiscalité Propre). Le segment Santé fait référence aux établissements publics de santé (hôpitaux, etc.).

12. Les hôpitaux publics sont des personnes morales de droit public du secteur sanitaire soumises au contrôle de l'État par le biais des agences régionales de santé.

13. Les données utilisées proviennent essentiellement de la DGFiP (Direction Générale des Finances Publiques).

7.3.4 Fonction à optimiser

Pour que la fonction de score g_β du SPL Santé soit bornée entre 0 et 100, le vecteurs des poids $\beta = (\beta_1, \ldots, \beta_8)$ doit vérifier la contrainte :

$$\sum_{i=1}^{8} \beta_i = 100\,.$$

Comme à l'application précédente, nous ajoutons la pénalité suivante à la fonction objectif pour satisfaire la contrainte des poids :

$$10 \cdot \left(\sum_{i=1}^{8} \beta_i - 100 \right)^2 .$$

Dans le cadre de cette application, nous cherchons à maximiser le tau-b de Kendall afin que les notes modélisées soient ordonnées dans le même sens que les notes expertes. Étant donné que nos algorithmes nous permettent de minimiser une fonction, c'est l'opposé du tau-b de Kendall $-\tau_b$ que nous allons chercher à optimiser.

Notre algorithme doit donc nous permettre de minimiser la fonction objectif $f : E \to \mathbb{R}$ suivante :

$$f(\beta) = -\tau_b\Big(N_e,\, \big(g_\beta(z)\big)_{1 \leq z \leq 275}\Big) + 10 \cdot \left(\sum_{i=1}^{8} \beta_i - 100 \right)^2$$

Où :

— N_e est le vecteur des notes attribuées par les experts à chacune des 275 entités.

— $\big(g_\beta(z)\big)_{1 \leq z \leq 275}$ est le vecteur des notes calculées par la fonction de score pour chaque entité avec les poids β.

L'espace de recherche E est défini de manière à ce que chaque poids à calibrer puisse prendre une valeur entre 2 et 60 [14]. Nous avons donc $E = [2\,;60]^8$.

7.3.5 Paramétrage de l'optimisation par essaim particulaire

L'essaim de particules initial est généré aléatoirement. Chaque particule j est constituée d'une liste de valeurs pour chaque poids $\beta_{1_j}, \ldots, \beta_{8_j}$ (qui correspond à sa position) et de la valeur de la fonction objectif $f(\beta_j)$ associée.

Pour chaque paramètre de l'algorithme par essaim particulaire, nous avons testé l'ensemble des valeurs suivantes :

14. Ces valeurs ont été choisies lors de la modélisation initiale afin que tous les facteurs aient une influence significative et qu'aucun facteur ne prépondère complètement sur les autres.

Paramètre	Valeurs testées		
Graine	3	5	
Taille de l'essaim N	1000	2000	3000
Pourcentage de voisines initial S_0	20 %	40 %	60 %
Nombre d'itérations maximal k_{max}	300	500	
Valeur maximale du compteur de stagnation c_{max}	25	75	
Paramétrage de la vitesse (inertie w ; influence historique y_1 ; influence sociale y_2)	(1,1 ; 1,49 ; 1,49)	(0,95 ; 2 ; 2)	(0,721 ; 1,193 ; 1,193)

TABLEAU 7.4 – Ensemble des paramètres testés pour l'application du PSO au SPL Santé

7.3.6 Fonctionnement de l'algorithme par essaim particulaire

Avant de commencer la recherche du meilleur vecteur des poids β, l'algorithme par essaim particulaire doit générer un essaim de N particules ($N = 1000$, 2000 ou 3000 en fonction du paramétrage). L'essaim de particules initial de l'algorithme est généré aléatoirement c'est-à-dire que N particules S_j ($1 \leq j \leq N$) sont tirées au hasard dans l'ensemble E.

Dans cette application, la position d'une particule S_j est constituée de 8 variables β_{1_j}, β_{2_j}, ..., β_{8_j} prenant chacune une valeur entre 2 et 60. Ces variables correspondent aux poids que nous associons à chacune de nos variables explicatives X_1, X_2, ..., X_8 pour calculer la valeur de la fonction f. $f(S_j)$, l'évaluation de la fonction f pour une position S_j, permet de caractériser les particules au sein de l'essaim ; les meilleures particules étant celles possédant la valeur $f(S_j)$ la plus faible.

Au fil des itérations, chaque particule va se déplacer dans l'espace de recherche E en fonction de sa vitesse, la meilleure position qu'elle a rencontré jusqu'ici et la meilleure position rencontrée par son voisinage [15]. La solution à notre problème d'optimisation est la meilleure position trouvée par les particules à travers toutes les itérations.

7.3.7 Résultats obtenus

Après avoir testé les 216 paramétrages pour l'algorithme PSO présentés ci-dessus, les 5 meilleurs résultats que nous avons obtenus sont les suivants [16] :

15. Les méthodes de l'algorithme par essaim particulaire sont décrites plus précisément à la section 5.5.3.

16. Les valeurs indiquées correspondent à la différence entre la valeur de référence (les résultats trouvés en utilisant R lors de la construction du modèle de notation sur le SPL Santé) et la valeur obtenue en utilisant notre algorithme sur Python.

Essai	**1**	**2**	**3**	**4**	**5**
			Paramétrage		
graine	3	3	3	5	5
N	3000	3000	2000	3000	3000
k_{max}	300 (ou 500)	300 (ou 500)	300 (ou 500)	300 (ou 500)	300 (ou 500)
c_{max}	75	25	75	75	25 (ou 75)
S_0	0,6	0,6	0,4	0,4	0,4
w	1,1	1,1	0,95	0,721	1,1
y_1	1,49	1,49	2	1,193	1,49
y_2	1,49	1,49	2	1,193	1,49
			Résultats		
Poids 1	2,94	2,93	2,46	2,46	1,76
Poids 2	−1,07	−1,07	−3,06	−2,18	−0,69
Poids 3	−0,39	−0,39	−0,06	−0,17	1,08
Poids 4	−1,44	−1,44	−1,44	−1,43	−1,37
Poids 5	−0,17	−0,16	0,15	0,85	2,02
Poids 6	−2,1	−2,1	−2,1	−2,1	−2,06
Poids 7	0,88	0,87	0,88	1,48	1,27
Poids 8	1,35	1,35	3,17	1,09	−2,01
Tau-b	**0,14 %**	**0,13 %**	**0,12 %**	**0,10 %**	**0,10 %**
k et c finaux	(152, 75)	(80, 25)	(198, 75)	(189, 75)	(130, 25)
Temps	1 h 45	1 h	2 h	2 h 30	2 h

TABLEAU 7.5 – Meilleurs résultats obtenus pour l'application SPL Santé après *grid search* des paramètres du PSO

Exemple de lecture du tableau :

- Pour obtenir le poids 1 de l'essai 1, il faut ajouter 2,94 au poids 1 de référence obtenu sur R : par exemple, si le poids 1 « R » est de 50, le poids que nous avons obtenu est de 52,94.
- Le tau-b obtenu pour l'essai 1 est 0,14 % meilleur que le tau-b de référence obtenu sur R : par exemple, si le tau-b « R » est de 0,95, notre tau-b sera alors de 0,9514.

Les meilleurs résultats obtenus sur Python sont très proches entre eux et sont, aussi, proches des résultats qui avaient été obtenus avec le package *pso* de R.

En utilisant le même paramétrage que celui utilisé lors de l'optimisation sous R (cf. tableau 7.6 ci-dessous), nous obtenons un tau-b de Kendall à -0,38 %, légèrement plus faible que celui obtenu sous R. Notre programme s'arrête au bout de 166 itérations car le compteur de stagnation a atteint son maximum (25). Le temps d'exécution de notre algorithme PSO en Python est d'environ 1 h 05. Ce temps est du même ordre de grandeur que celui obtenu avec R.

Nous pouvons remarquer qu'en augmentant le nombre de particules, les résultats auxquels nous parvenons sont généralement meilleurs mais le temps d'exécution de l'algorithme augmente.

Nous pouvons également observer qu'il est possible d'obtenir des résultats satisfaisants avec les 3 paramétrages classiques de la vitesse testés (paramètres d'inertie et d'influences historique et sociale). Certains des résultats présentés ici ont le désavantage d'avoir un temps d'exécution assez

Paramètre	Valeurs
graine	3
N	2000
k_{max}	300
c_{max}	25
S_0	20 %
w	1,1
y_1	1,49
y_2	1,49

Tableau 7.6 – Paramétrage de l'application sous R

long. Cependant, il est tout de même possible d'obtenir des résultats satisfaisants en un temps plus proche de 1 h (même en ayant un grand nombre de particules).

Pour la résolution de notre problème le passage de 300 à 500 itérations améliore, en général, très peu, voir pas du tout les résultats et est donc très peu rentable car cette augmentation entraîne un temps plus long d'optimisation (quand la cause d'arrêt de l'algorithme est que le critère d'itération maximal de 300 a été atteint). Pour les essais ci-dessus, le passage de 300 à 500 itérations n'apporte rien parce que l'algorithme finit quand la valeur maximale du compteur de stagnation est atteinte. Le passage de 75 à 25 entre l'essai 1 et l'essai 2 pour la valeur finale du compteur de stagnation a de même très peu amélioré le résultat final.

Il faut noter que le tirage aléatoire initial (et la présence de tirages aléatoires dans l'algorithme) peut aussi fortement influencer les résultats finaux. Par exemple, en prenant une graine de 3 au lieu de 5 pour l'essai numéro 4, nous obtenons un tau-b de Kendall un peu moins bon à -2,35 % du tau-b de référence sous R. La présence de tirages aléatoires dans l'algorithme implique que, parfois, nous pouvons aussi obtenir de moins bon résultats en augmentant le nombre de particule : par exemple, avec le paramétrage ($graine = 3$, $k_{max} = 300$, $c_{max} = 75$, $S_0 = 0{,}4$, $w = 0{,}721$, $y_1 = y_2 = 1{,}193$), nous obtenons un tau-b de -2,35 % avec $N = 3000$, un tau à -1,10 % avec $N = 2000$ et un tau de -0,25 % avec $N = 1000$.

En testant différents paramétrages, nous pouvons observer qu'il existe plusieurs combinaisons de paramètres pouvant donner des résultats satisfaisants en un temps raisonnable (environ 1 h).

À travers l'application de l'algorithme PSO au Secteur Public Local, segment Santé, nous avons donc pu tester le bon fonctionnement de l'algorithme implémenté en Python.

7.4 Comparaison des performances des deux algorithmes

Dans un deuxième temps, nous avons appliqué notre algorithme PSO pour optimiser la fonction de score sur le portefeuille Établissements Financiers et notre algorithme génétique pour l'optimisation du modèle SPL Santé. Nous voulons ainsi observer si les deux algorithmes ont des performances équivalentes sur les problèmes étudiés.

7.4.1 Application du PSO aux Établissements Financiers

Nous avons testé l'ensemble des valeurs de paramètres présentées dans le tableau 7.7 et les 5 meilleurs résultats obtenus suite aux 288 paramétrages testés sont donnés dans le tableau 7.8.

Paramètres	**Valeurs testées**			
Graine	3	5		
Taille de l'essaim N	1000	2000	3000	5000
Pourcentage de voisines initial S_0	20 %	40 %	60 %	
Nombre d'itérations maximal k_{max}	300	500		
Valeur maximale du compteur de stagnation c_{max}	25	75		
Paramétrage de la vitesse (inertie w ; influence historique y_1 ; influence sociale y_2)	(1,1 ; 1,49 ; 1,49)	(0,95 ; 2 ; 2)	(0,721 ; 1,193 ; 1,193)	

Tableau 7.7 – Ensemble des paramètres testés pour l'application du PSO aux Établissements Financiers

Les résultats obtenus avec notre PSO, tant en ce qui concerne le tau-b de Kendall que le temps d'exécution, sont aussi satisfaisants que ceux obtenus avec notre AG.

7.4.2 Application de l'AG aux SPL Santé

Nous avons également testé l'AG pour l'optimisation du modèle de notation du SPL Santé et nous obtenons, de même, un résultat aussi satisfaisant qu'avec le PSO.

Les paramétrages testés ainsi que les résultats obtenus sont présentés dans les deux tableaux 7.9 et 7.10 suivants.

7.5 Conclusion de l'application du PSO et de l'AG à des portefeuilles *Low Default*

Les deux algorithmes que nous avons implémentés en Python offrent des performances satisfaisantes sur les deux applications que nous avons étudiées. Et, ces deux algorithmes, l'AG et le PSO, sont adaptés à l'optimisation d'une fonction de score d'un portefeuille LDP.

L'apport de nos algorithmes réside, d'une part, en la transparence du code utilisé pour la mise en place des algorithmes en eux-même et d'autre part, en leur adaptabilité aux différentes problématiques rencontrées. Les modules créés peuvent facilement être adaptés pour répondre à d'autres besoins de modélisation de la banque.

L'implémentation d'un module de *grid search*, complémentaire à l'optimisation par algorithme génétique ou par essaim particulaire, présente une innovation par rapport à la méthodologie qui avait été utilisée pour construire les deux modèles, Établissements Financiers et SPL Santé, et permet de tester une grande variété de paramètres et de s'assurer de la qualité des résultats obtenus.

Essai	**1**	**2**	**3**	**4**	**5**
Paramétrage					
graine	3	3	3	3	3
N	2000	5000	5000	3000	2000
k_{max}	300	300	300	300	300
c_{max}	25	25	25	25	25
S_0	40 %	40 %	40 %	40 %	40 %
w	1,1	0,721	1,1	1,1	1,1
y_1	1,49	1,193	1,49	1,49	1,49
y_2	1,49	1,193	1,49	1,49	1,49
Résultats (%)					
Poids 1	0,18	−0,71	0,05	0,10	0,24
Poids 2	−3,40	−3,40	−3,40	−3,39	−3,40
Poids 3	0,05	0,08	0,10	0,10	0,10
Poids 4	1,17	2,16	1,64	1,86	1,17
Poids 5	1,40	1,40	1,40	1,38	1,40
Poids 6	−0,01	0,00	0,00	0,00	0,00
Poids 7	−0,42	0,68	0,13	−0,30	−0,53
Poids 8	0,03	0,10	−0,18	−0,50	0,08
Poids 9	2,54	1,31	1,75	2,18	2,60
Poids 10	−4,92	−4,59	−4,34	−4,59	−5,05
Poids 11	0,10	0,10	0,10	0,08	0,10
Poids 12	0,10	0,10	0,10	0,10	0,10
Poids 13	2,99	3,01	2,65	2,66	2,99
Poids 14	0,20	0,20	−0,02	0,20	0,20
Tau-b	**0,44**	**0,39**	**0,37**	**0,37**	**0,30**
k et c finaux	(95, 25)	(51, 25)	(66, 25)	(63, 25)	(84, 25)
Temps	5 min	15 min	20 min	10 min	5 min

TABLEAU 7.8 – Meilleurs résultats obtenus pour l'application Établissements Financiers après *grid search* des paramètres du PSO

Paramètre	**Valeurs testées**		
Graine	3	5	
Taille de la population	200	500	800
Nombre de générations	200	400	
Taux d'élites	0,05	0,1	
Pas de sélection	0,9	1,1	
Probabilité de croisement	0,8	0,9	
Probabilité de mutation	0,05	0,1	

TABLEAU 7.9 – Ensemble des paramètres testés pour l'application de l'AG aux SPL Santé

Essai	**1**	**2**	**3**	**4**	**5**
			Paramétrage		
Graine	3	3	3	5	5
Taille population	500	500	800	500	500
N^{bre} *générations*	400	200	400	400	400
Taux élites	0,05	0,05	0,1	0,05	0,05
Pas sélection	0,9	0,9	1,1	1,1	1,1
Proba croisement	0,9	0,9	0,9	0,9	0,8
Proba mutation	0,1	0,1	0,1	0,1	0,1
			Résultats		
Poids 1	−1,67	−1,67	0,07	2,32	0,05
Poids 2	6,10	6,10	0,72	1,42	5,13
Poids 3	1,03	1,03	0,05	1,75	0,04
Poids 4	−0,91	−0,91	−0,87	−0,15	0,39
Poids 5	−0,05	−0,05	3,14	0,92	−0,08
Poids 6	−0,98	−0,98	2,13	−1,03	−1,70
Poids 7	1,65	1,65	−0,49	0,67	4,63
Poids 8	−5,18	−5,18	−4,75	−5,91	−8,47
Tau-b	**-0,36 %**	**-0,37 %**	**-0,45 %**	**-0,47 %**	**-0,48 %**
Temps	1 h 30	45 min	2 h 15	1 h 30	1 h 30

TABLEAU 7.10 – Meilleurs résultats obtenus pour l'application SPL Santé après *grid search* des paramètres de l'AG

8 Modélisation de la probabilité de défaut à l'entrée en relation avec une clientèle *retail*

Dans une approche classique [1], ce sont notamment les informations sur le comportement bancaire du client et la base conséquente de défauts observés qui seront utilisées pour modéliser la PD. Le score à l'entrée en relation avec une clientèle *retail* existant se base sur un questionnaire qui permet de récolter une dizaine d'informations sur le client (telles que l'âge ou la situation familiale). Le but de ce score est d'évaluer le risque du client à l'entrée en relation afin de lui préconiser les moyens de paiements (carte, découvert, ...) adaptés. Dans la suite, ce score est désigné sous l'appellation « nouveaux clients LBP ».

Du fait du peu de connaissance que nous avons du comportement des nouveaux clients, il est difficile d'atteindre une performance satisfaisante sur ce score (i.e. un bon taux de prédiction du risque du client). Nous avons exploré des pistes d'amélioration du score nouveaux clients LBP. Tout d'abord, en géocodant les clients LBP ce qui nous a permis d'associer des données externes provenant de l'Insee. Nous avons estimé des modèles sur ces nouvelles données à l'aide d'une régression logistique avant de nous tourner vers l'utilisation d'un arbre de décision. Dans un deuxième temps, nous avons testé une méthode alternative à la sélection de variable par méthode stepwise en utilisant l'algorithme génétique que nous avons implémenté en Python.

8.1 Constitution de la base de données

Pour les besoins de notre étude, nous avons utilisé deux types de données : des données internes et des données externes. Dans cette section, nous présentons les étapes de récupération et de traitements effectuées pour construire notre base de données.

8.1.1 Données internes

Les données internes utilisées dans le cadre de notre application sont issues du SID [2] de La Banque Postale. Dans la suite du document, lorsque nous faisons référence à ces données, nous les nommons « données LBP ».

Fenêtre d'observation des données

L'estimation statistique du modèle repose sur l'observation des nouveaux clients qui ont fait l'objet d'une demande de préconisations sur une période donnée. La période de construction constitue une fenêtre d'observation d'un an, entre le *1er mars 2014* et le *28 février 2015*. L'observation d'une année complète permet de capter les phénomènes de saisonnalité.

1. I.e pour construire un score pour les individus déjà clients LBP.
2. Système d'Information Décisionnelle.

Variable à expliquer : le défaut 12 mois

Le défaut est observé sur 12 mois pour chaque génération d'ouverture, soit entre le *1er avril 2014* et le *29 février 2016*.

Un client est considéré en défaut, notamment, si une de ces conditions est vérifiée :

— son compte courant est en dépassement d'autorisation de découvert depuis plus de 90 jours ou à découvert depuis plus de 120 jours ;

— l'un de ses crédits est en impayé depuis plus de 90 jours ;

— il a été déclaré « en surendettement » par la Banque de France.

Périmètre des demandes de préconisations

Le périmètre de la base de modélisation contient les nouveaux clients « personnes physiques » ne détenant pas déjà un compte bancaire à La Banque Postale et qui ont fait l'objet d'une demande d'ouverture de compte courant ayant été finalisée. Les demandes ayant été refusées ne sont pas exploitées car le refus provient du client lorsque les produits proposés ne lui conviennent pas.

Variables explicatives

Dans la cadre de la construction du modèle de notation des nouveaux clients LBP, des variables internes avaient déjà été sélectionnées et traitées. Nous avons donc repris les 23 variables internes qui avaient été retenues pour l'estimation du score sur les nouveaux clients LBP. L'ensemble de ces variables ainsi que les découpages choisis lors de la construction du modèle initial ont été intégrés dans la construction du modèle alternatif. Nous n'avons donc effectué aucun traitement sur ces variables.

8.1.2 Données externes

L'institut national de la statistique et des études économiques (Insee) est en charge de la production, de l'analyse et de la publication des statistiques officielles, dont la quasi-totalité est mise à disposition gratuitement sur son site internet [3]. Une partie de ces données statistiques et démographiques sont collectées et diffusées au niveau IRIS (Îlots Regroupés pour l'Information Statistique). Dans le cadre de l'étude de techniques d'amélioration du score sur les nouveaux clients LBP, il a été décidé de compléter les données LBP par des données publiées par l'Insee au niveau IRIS.

Nous avons utilisé deux publications officielles de base de données issues :

— du recensement de la population ;

— des fichiers de revenus fiscaux localisés (RFL).

Lien avec les données internes LBP

Afin de pouvoir utiliser les données statistiques publiées au niveau IRIS, une étape de géocodage des nouveaux clients LBP est nécessaire. Le géocodage consiste à associer un code IRIS et un identifiant commune à chaque client en fonction de l'adresse communiquée au moment de la demande. Cette étape a été réalisée en collaboration avec la direction Marketing/Data science de LBP qui dispose d'un outil informatique qui permet de récupérer le code IRIS à partir d'une adresse donnée.

3. Cf. `https://www.insee.fr/fr/recherche/recherche-statistiques?q=IRIS&debut=0&categorie=3`.

Choix des bases de données Insee

Les IRIS fournis par l'outil correspondent au découpage géographique de 2008 et les versions des publications de l'Insee sont choisies en conséquence :

— Recensement : les données Insee utilisées proviennent du recensement de 2006 sur la population (Insee (2009*e*)), l'activité (Insee (2009*a*)), la famille (Insee (2009*b*)), la formation (Insee (2009*c*)) et le logement (Insee (2009*d*)). Ces données sont disponibles pour la métropole et les DOM [4] hors Mayotte et pour certains COM [5] (Saint-Pierre et Miquelon, Saint-Barthélemy et Saint Martin).

— Revenus Fiscaux Localisés (RFL) : les données Insee utilisées proviennent du dispositif RFL pour l'année 2008 (Insee (2012)). Les statistiques sur les revenus sont disponibles au niveau ménage, personne et unité de consommation (UC). Nous disposons également de données sur la structure des revenus. Ces données sont mises à disposition pour la métropole, la Guadeloupe et la Réunion.

Base de données externes à fusionner

Après regroupement des différents fichiers sources disponibles sur l'Insee, nous avons à notre disposition les deux bases de données externes suivantes :

— une base regroupant les bases de données du recensement et des revenus fiscaux au niveau IRIS ;

— une base regroupant les bases de données RFL au niveau commune [6].

8.1.3 Fusion des différentes bases

Dans ce paragraphe, nous exposons les traitements effectués pour associer les données externes provenant de l'Insee aux observations de la base de données LBP.

8.1.3.1 Périmètre des données géocodées

Nous disposons, pour chaque observation de la base de données LBP, soit :

— d'un géocodage au niveau IRIS et commune (i.e. un IRIS et un identifiant commune ont été associés à l'observation) ;

— d'un géocodage au niveau commune ;

— d'aucun géocodage.

La base de données LBP n'a pas pu être entièrement géocodée notamment pour l'une des raisons suivantes :

— il n'a pas été possible de récupérer l'adresse du client au moment de la demande ;

— l'outil de géocodage ne permet pas d'attribuer un IRIS aux clients résidant hors France métropolitaine (excepté pour Monaco).

4. Départements d'Outre-Mer.
5. Collectivités d'Outre-Mer.
6. Seules les données RFL sont utilisées au niveau commune car les bases de données RFL contiennent moins d'IRIS que les bases de recensement (13 703 vs 50 881).

De plus, dans le cadre de notre étude, nous n'avons pas pris en compte certains IRIS ou identifiant commune :

— 51 observations géocodées avec IRIS pour Monaco sont considérées comme non géocodées (car hors France) ;

— 13 observations associées à un IRIS non cohérent avec le nom de la commune sont considérées comme non géocodées (car hors France) ;

— 4 observations avec un IRIS non présent dans les bases de recensement et revenus niveau IRIS de l'Insee sont considérées comme géocodées seulement au niveau commune ;

— 16 observations avec un identifiant commune non cohérent avec le nom de la commune sont considérées comme non géocodées.

Nous avons, au final, environ 4 % de la base géocodée seulement au niveau commune, 83 % au niveau IRIS (et commune) et 13 % non géocodée.

8.1.3.2 Méthodologie de construction de la base de travail

La fusion des bases de données est effectuée en fonction des données géocodées à notre disposition.

Les chiffres en gras font référence à la répartition des données présentée dans le tableau 8.1.

Observations géocodées niveau IRIS

Dans un premier temps, nous ajoutons les données de recensement et de revenus disponibles au niveau IRIS aux observations pour lesquelles un IRIS a été associé (**1**, **2** et **3**).

Les données de revenus ne sont pas disponibles pour certains IRIS. Pour ces demandes, nous ajoutons les données de revenus au niveau commune via l'identifiant commune (quand celles-ci sont disponibles) (**2**).

Observations géocodées au niveau commune seulement

Nous ajoutons les données de revenus, disponibles au niveau commune, aux observations auxquelles seul un identifiant commune a été associé (**6**).

Observations non géocodées

Ces observations peuvent être séparées en trois catégories :

— les observations ne possédant aucune donnée géographique (**0**) i.e. les observations pour lesquelles une adresse à la date de l'étude n'est pas disponible ;

— les observations localisées hors France (via un code postal ou un nom de pays/commune) (**9**)

— les observations pour lesquelles un code postal ou un nom de commune localisé en France est disponible (**3**, **5** et **7**).

Nous avons pu associer des données provenant de l'Insee seulement pour cette dernière catégorie.

Dans un premier temps, nous avons ajouté les données de revenus au niveau commune via le nom de la commune et les premiers chiffres du code postal (qui correspondent au numéro du département).

Nous avons ensuite rajouté des données de recensement pour certaines de ces observations via le nom de la commune et les premiers chiffres du code postal correspondant au numéro du département. Les données de recensement qu'il est possible de rajouter sont uniquement celles pour lesquelles la commune est couverte par un seul IRIS (identifiant commune présent une seul fois dans la base de recensement).

Il nous reste tout de même des observations pour lesquelles aucune donnée INSEE n'a été rajoutée (notamment pour les demandes et clients localisés dans certains DOM/COM).

8.1.3.3 Récapitulatif

La répartition des observations suivant la méthode d'ajout des données INSEE est récapitulée ci-dessous :

	Source construction de la base	Pourcentage de la base
1	Données recensement et RFL niveau IRIS (jointure IRIS)	57,11
2	Données recensement (jointure IRIS) et RFL niveau commune (jointure identifiant commune)	25,43
3	Données recensement et RFL niveau commune (jointure nom commune et département)	1,85
4	Données recensement uniquement (jointure IRIS)	0,26
5	Données recensement uniquement (jointure nom commune et département)	0,28
6	Données RFL niveau commune uniquement (jointure identifiant commune)	4,43
7	Données RFL niveau commune uniquement (jointure nom commune et département)	6,91
8	Aucune donnée INSEE	3,21
9	Localisation hors France	0,27
0	Pas de données LBP de localisation	0,24

TABLEAU 8.1 – Récapitulatif source des données INSEE fusionnées

Des données de recensement sont disponibles pour environ 85 % de la base et des données de revenus pour environ 96 % de la base. Nous avons ainsi seulement environ 4 % de la base sans aucune donnée supplémentaire. La répartition des observations selon leurs sources de données externes est donnée dans le tableau ci-dessous.

		RFL		
		Pas de données	**Données RFL IRIS**	**Données RFL commune**
RECENSEMENT	**Pas de données**	3,72	0	11,34
	Données recensement	0,54	57,11	27,23

TABLEAU 8.2 – Répartition des données de recensement et revenus dans la base de données finale (%)

8.1.4 Construction de variables

Dans cette section, nous présentons les variables issues de l'Insee que nous avons utilisées ainsi que la variable PD_MOY_IRIS construite à partir de données internes géocodées.

8.1.4.1 Variables Insee

Notre base de données comporte 411 variables explicatives externes fournies par l'Insee.

Les variables Insee issues du RFL et portant sur les revenus sont utilisables en l'état et ne nécessitent pas de traitement particulier. Dans un premier temps, nous prenons en compte toutes les variables sur les revenus pour notre étude.

Contrairement aux variables de revenus, les variables issues des bases de données du recensement ne sont pas utilisables en l'état car elles ne permettent pas de comparer les IRIS entre-eux [7].

Après s'être assurées de leur bonne qualité [8], nous avons construit une soixantaine d'indicateurs à partir des variables brutes du recensement. Les documents et analyses présents sur l'Insee [9], nous ont aidées à déterminer les indicateurs qu'il est possible de construire. Nous avons ensuite choisi les indicateurs qui nous semblaient les plus pertinents pour expliquer le défaut.

La liste des indicateurs construits (cf. tableau D.1) et la liste des variables de revenus (cf. tableau D.2) sont disponibles en annexe D.

8.1.4.2 Variable PD_MOY_IRIS

La variable PD_MOY_IRIS permet d'associer à nos observations géocodées la probabilité de défaut moyenne des clients de La Banque Postale pour un IRIS donné.

Création de la variable

Dans un premier temps, l'équipe Marketing de LBP nous a fourni une base associant à chaque client du périmètre « personnes physiques » de La Banque Postale son IRIS suivant son adresse connue la plus récente. La probabilité de défaut à la date de l'adresse est accolée à chaque client. Ces PD sont issues du SID de La Banque Postale et sont calculées à partir des modèles de notation construits par l'équipe Modélisation.

Pour chaque IRIS présent dans la base, nous avons ensuite calculé la moyenne des probabilités de défaut de chaque client appartenant à cet IRIS. Pour les clients où aucun IRIS n'a pu être associé [10], c'est la probabilité de défaut moyenne sur ces clients qui a été calculée.

Lien avec notre base de données

La PD moyenne calculée à l'étape précédente est associée à chaque observation de notre base en fonction de la valeur de la variable IRIS. Si l'observation dans notre base n'a pas d'IRIS, nous lui

7. En effet, ces variables sont exprimées en nombre d'habitants. Par exemple, la variable P06_CHOM1564 donne le nombre de chômeurs dans la population des 15-64 ans et, pour comparer les IRIS entre eux, nous rapportons cette valeur au nombre d'habitants de l'IRIS (P06_POP1564) pour construire un indicateur donnant la part des chômeurs dans la population 15-64 ans (en %).

8. Nous avons regardé le pourcentage de valeurs manquantes (qui est, la plupart du temps, égal à 0) et nous nous sommes assurées que toutes les variables étaient bien positives.

9. Cf. REYNARD & VIALETTE (2018) et les fiches synthétiques des bases tableaux détaillés du recensement (cf. `https://www.insee.fr/fr/information/2894421`).

10. Par exemple, il n'a pas été possible d'associer un IRIS pour les clients LBP résidant dans les DOM ou COM.

associons la PD moyenne calculée sur les clients sans IRIS. Si aucune PD moyenne n'est associée à un IRIS présent dans notre base, aucune valeur n'est associée (la valeur est alors manquante).

Nous avons à disposition la variable PD_MOY_IRIS ainsi que les effectifs utilisés pour calculer la PD moyenne de l'IRIS. La variable des effectifs sera utilisée pour s'assurer de la qualité de la variable.

8.2 Application de la régression logistique

Dans cette section, nous présentons les résultats obtenus après application de la méthodologie présentée à la section 6.

8.2.1 Échantillonnage

La base de modélisation contient toutes les demandes finalisées effectuées entre mars 2014 et février 2015.

La base de modélisation est séparée aléatoirement en deux sous populations :

— la population de construction constituée de 70 % de la base, et,

— la population de validation constituée de 30 % de la base.

Avant de commencer la construction du score, nous nous assurons de la représentativité[11] de chacun de nos échantillons :

Variable	p-value validation	p-value construction
X_1	0,18	0,38
X_2	0,28	0,70
X_3	0,10	0,28
X_4	0,43	0,60
X_5	0,46	0,82
Nombre de ménage de l'IRIS	0,13	0,55
Population de l'IRIS	0,28	0,70

Tableau 8.3 – Résultats des tests de représentativité des échantillons

D'après les résultats obtenus, nous concluons que les deux échantillons sont représentatifs.

Pour rappel, les étapes de préparation des variables et d'estimation du score suivantes sont effectuées sur la population de construction.

8.2.2 Préparation des variables

Pour les variables LBP, les traitements de préparation des variables avaient déjà été effectués lors de la construction du score sur les nouveaux clients. Nous avons notamment repris les découpages qui avaient été choisis à cette occasion.

11. Le principe de cette méthode est explicité à la section 6.1.

Les traitements décrits dans cette section concernent donc uniquement les variables issues des bases de données Insee et la variable PD_MOY_IRIS.

8.2.2.1 Analyse univariée

Nous réalisons une étape d'analyse univariée (i.e. détection des valeurs non renseignées ou aberrantes) de nos variables afin de nous assurer de leur bonne qualité avant de les utiliser pour construire un modèle de notation.

Indicateurs recensement

- Les indicateurs issus des variables du recensement présentent 15,04 % de valeurs non applicables [12].
- Nous avons 9 indicateurs qui présentent des valeurs manquantes. Ces valeurs représentant toutes moins de 5 % de l'échantillon, nous n'écartons aucune de ces variables [13].
- Nous n'avons pas remarqué de valeurs aberrantes parmi nos indicateurs (les pourcentages sont supérieurs ou égaux à 0 et inférieurs ou égaux à 100).

Aucune variable n'est donc écartée suite à l'analyse univariée des indicateurs du recensement.

Variables RFL

- Les variables de revenus présentent 4,27 % de valeurs non applicables [14].
- 51 variables de revenus possèdent plus de 5 % de valeurs manquantes. Nous écartons les variables de revenus comportant plus de 5 % de valeurs manquantes à l'exception des variables RFUCQ108 et RFUCQ308 [15].

Suite à ce traitement, il nous reste 5 variables portant sur le revenu :

- la médiane des revenus fiscaux par personne ;
- la médiane des revenus fiscaux par unité de consommation ;
- la médiane des revenus fiscaux par ménage ;
- le premier quartile des revenus par unité de consommation (RFUCQ108) ;
- le troisième quartile des revenus par unité de consommation (RFUCQ308).

Variable PD_MOY_IRIS

- Pour cette variable, nous avons 17 % de valeurs non applicables (i.e. avec un IRIS manquant).
- Cette variable présente 2,3 % de valeurs manquantes (car certains IRIS présents dans notre base n'ont pas été associés à une PD moyenne (i.e. aucun client LBP de la base de construction de la PD moyenne ne réside dans ces IRIS à la date de l'adresse)).

12. Ces valeurs non applicables sont des valeurs non renseignées car il n'a pas été possible de géocoder l'observation ou parce que l'IRIS ou l'identifiant commune affecté à l'observation n'est pas présent dans les bases de recensement. Elles ne proviennent pas d'une donnée manquante dans les fichiers de données de l'Insee.

13. Cf. section 8.2.2.3.

14. Idem pour les bases RFL (cf. note de bas de page 12 ci-dessus).

15. Nous faisons une exception à la règle des 5 % car d'une part, ces variables présentent un pourcentage de valeurs non renseignées (manquante ou non applicable) du même ordre de grandeur que les indicateurs issus du recensement et d'autre part, nous voulons prendre en compte des indicateurs de revenus autre que la médiane tout en limitant le nombre d'indicateur à tester. Nous avons choisi les quartiles de revenus car ils présentent moins de valeurs manquantes que les déciles ou la moyenne (12 % vs 14 %) et le périmètre « unité de consommation » car c'est celui qu'il est conseillé d'utiliser pour comparer des territoires dans la documentation de l'Insee.

— Nous excluons les valeurs de PD moyenne calculée sur un effectif de moins de 30 individus car nous considérons ces valeurs comme non représentatives. Les valeurs exclues représentent 1,6 % de la population.

8.2.2.2 Analyse bivariée et découpage des variables

Nous analysons nos variables explicatives continues en utilisant la population des individus pour lesquels une valeur cohérente est renseignée [16] afin d'effectuer un découpage de ces variables en plusieurs modalités.

8.2.2.3 Traitement des valeurs non renseignées ou aberrantes

Nous réintroduisons les variables manquantes identifiées à la section 8.2.2.1 selon les méthodes explicitées à la section 6.2.3.

Variable	Méthode d'imputation	Volumétrie
MEN_PSEUL	classe la plus fréquente	7.10^{-5} %
MEN_COUPAENF	classe la plus fréquente	7.10^{-5} %
MEN_COUPSENF	classe la plus fréquente	7.10^{-5} %
MEN_FAMMONO	classe la plus fréquente	7.10^{-5} %
MEN_SFAM	classe la plus fréquente	7.10^{-5} %
FAM_NE24F0	classe la plus fréquente	8.10^{-5} %
FAM_NE24F1	classe la plus fréquente	8.10^{-5} %
FAM_NE24F2	classe la plus fréquente	8.10^{-5} %
FAM_NE24F3P	classe la plus fréquente	8.10^{-5} %
PD_MOY_IRIS	classe avec le taux de défaut le plus proche	2,3 %
RFUCQ108	classe avec le taux de défaut le plus proche	12 %
RFUCQ308	classe avec le taux de défaut le plus proche	12 %

Tableau 8.4 – Méthode d'imputation pour les variables contenant des valeurs manquantes

De plus, une modalité spécifique est dédiée aux individus présentant une valeur non-applicable. Les individus avec une valeur exclue (pour la variable PD_MOY_IRIS) sont également affectés à une modalité spécifique.

8.2.3 Estimation des modèles

Nous utilisons la méthodologie d'estimation du score détaillée dans la section 6.3.

La première étape de la méthodologie d'estimation d'un score consiste à estimer un modèle dit « brut » avec pour objectifs :

16. I.e les valeurs non applicables, manquantes, aberrantes ou métier identifiées à l'étape précédente ne sont pas prises en compte pour cette étape.

— d'observer le potentiel de performance du score ;

— d'identifier les variables explicatives qui, combinées aux autres, sont les plus discriminantes.

Dans le cadre de notre étude, nous avons estimé 4 modèles bruts à partir de différents ensembles de variables explicatives. Nous évaluons la performance de notre modèle par rapport à celle obtenue avec les variables LBP uniquement. Le graphique 8.1 ci-dessous illustre l'évolution de l'indice de Gini après ajout de chaque variable sélectionnée au modèle.

Les modèles testés sont les suivants :

Modèle 1

Dans ce premier modèle, nous avons utilisé toutes les variables Insee sélectionnées à l'issue de l'étape de préparation des variables ainsi que les variables LBP qui avaient été sélectionnées lors de la construction du score sur les nouveaux clients LBP. Les variables Insee présentes dans ce modèle sont [17] :

Étape	Variable	Libellé
7	RFUCQ108	1er quartile des revenus (en €) par UC
12	RP_VOIT2P	Part des ménages avec 2 voitures ou plus (%)
16	LOG_RSECOCC	Part des résidences secondaires et logements occasionnels dans les logements (%)
18	TX_EMP_1564	Taux d'emploi des 15-64 ans (%)
23	EMP_1524	Part des 15-24 ans dans l'emploi (%)
25	ACTOCC15P_TCOM	Part des actifs occupés allant en transport en commun au travail (%)
26	ACTOCC15P_ILT2	Part des actifs occupés travaillant dans une autre commune du même départ. de résidence (%)
28	ACTOCC15P_ILT1	Part des actifs occupés travaillant et résidant dans la même commune (%)
29	NSCOL15P_DIPL0	Part de la population de 15 ans et plus non scolarisée sans diplôme (%)

Tableau 8.5 – Variables Insee sélectionnées dans le modèle 1

La performance obtenue pour ce modèle est très proche de celle obtenue avec les données LBP (le Gini obtenu avec 30 variables a une différence de seulement 0,008 avec le Gini des données LBP) ce qui n'est pas très satisfaisant, notre but étant d'améliorer la performance de ce modèle. De plus, la première variable Insee, RFUCQ108 [18] n'apparaît qu'à la 7e position.

17. Cf. tableau D.1 pour la définition de ces variables.
18. RFUCQ108 correspond au premier quartile des revenus par unité de consommation.

Modèle 2

Pour ce deuxième test, nous avons utilisé toutes les variables Insee du précédent modèle saturé mais uniquement les 9 variables LBP qui avaient été initialement sélectionnées (après étude des modèles intermédiaires) lors de la construction du score. Les variables Insee suivantes rentrent dans le modèle[19] :

Étape	Variable	Libellé
6	RFUCQ108	1er quartile des revenus (en €) par UC
10	RP_VOIT2P	Part des ménages avec 2 voitures ou plus (%)
12	ACTOCC15P_TCOM	Part des actifs occupés allant en transport en commun au travail (%)
13	LOG_RSECOCC	Part des résidences secondaires et logements occasionnels dans les logements (%)
14	TX_EMP_1564	Taux d'emploi des 15-64 ans (%)
15	EMP_1524	Part des 15-24 ans dans l'emploi (%)
16	ACTOCC15P_ILT2	Part des actifs occupés travaillant dans une autre commune du même départ. de résidence (%)
17	LOG_LOGVAC	Part des logements vacants dans les logements (%)
18	NSCOL15P_DIPL0	Part de la population de 15 ans et plus non scolarisée sans diplôme (%)
19	ACTOCC15P_ILT1	Part des actifs occupés travaillant et résidant dans la même commune (%)
20	RFMQ208	Médiane des revenus (en €) par ménage
21	RETR_1564	Part des retraités, préretraités dans la population des 15-64 ans (%)
22	RP_VOIT1	Part des ménages avec 1 seule voiture (%)
23	RP_GRAT	Part des résidences principales occupées gratuitement (%)
25	ACT1564_CS1	Part des agriculteurs exploitants dans les actifs de 15-64 ans (%)
26	ACT1564_CS4	Part des professions intermédiaires dans les actifs de 15-64 ans (%)
28	TX_CHOM_2554	Taux de chômage des 25-54 ans (%)
29	TX_CHOM_1524	Taux de chômage des 15-24 ans (%)
30	EMP_SAL15P_CDI	Part des salariés en CDI ou dans la fonction publique dans l'emploi (%)

Tableau 8.6 – Variables Insee sélectionnées dans le modèle 2

Modèle 3

Nous avons également regardé la performance qu'il était possible d'obtenir en utilisant uniquement les variables Insee. La performance obtenue pour ce modèle est de loin inférieure à la performance obtenue pour les autres modèles testés (cf. graphique 8.1). Les variables apparaissent dans le modèle dans l'ordre suivant :

19. La variable RP_LOC rentre dans le modèle en 24e position mais est éliminée lors de l'itération 27.

Étape	Variable	Libellé
1	RP_VOIT2P	Part des ménages avec 2 voitures ou plus (%)
2	RFUCQ108	1er quartile des revenus (en €) par UC
3	EMP_1524	Part des 15-24 ans dans l'emploi (%)
4	TX_CHOM_1524	Taux de chômage des 15-24 ans (%)
5	LOG_RSECOCC	Part des résidences secondaires et logements occasionnels dans les logements (%)
6	ACTOCC15P_TCOM	Part des actifs occupés allant en transport en commun au travail (%)
7	ACTOCC15P_ILT2	Part des actifs occupés travaillant dans une autre commune du même départ. de résidence (%)
8	LOG_LOGVAC	Part des logements vacants dans les logements (%)
9	RP_LOC	Part des résidences principales occupées par locataires (%)
10	NSCOL15P_DIPL0	Part de la population de 15 ans et plus non scolarisée sans diplôme (%)
11	ACTOCC15P_ILT1	Part des actifs occupés travaillant et résidant dans la même commune (%)
12	ACT1564_CS6	Part des ouvriers dans les actifs de 15-64 ans (%)
13	ACT1564_CS1	Part des agriculteurs exploitants dans les actifs de 15-64 ans (%)
14	EMP_SAL15P_CDI	Part des salariés en CDI ou dans la fonction publique dans l'emploi (%)
15	RFMQ208	Médiane des revenus (en €) par ménage
16	RETR_1564	Part des retraités, préretraités dans la population des 15-64 ans (%)
17	RP_VOIT1	Part des ménages avec 1 seule voiture (%)
18	MEN_COUPAENF	Part des ménages composés d'un couple avec enfant (%)
19	RP_GRAT	Part des résidences principales occupées gratuitement (%)
20	ACTOCC15P_VOIT	Part des actifs occupés allant en voiture au travail (%)
21	AINACT_1564	Part des autres inactifs dans la population 15-64 ans (%)
22	ACT1564_CS5	Part des employés dans les actifs de 15-64 ans (%)
23	ACT1564_CS3	Part des cadres, professions intellectuelles supérieures dans les actifs de 15-64 ans (%)
24	RP_PROP	Part des résidences principales occupées par propriétaires (%)
25	RFPQ208	Médiane des revenus (en €) par personne
26	EMP_2554	Part des 25-54 ans dans l'emploi (%)
27	EMP_5564	Part des 55-64 ans dans l'emploi (%)
28	MEN_FAMMONO	Part des ménages composé d'une famille monoparentale (%)
29	ACT1564_CS4	Part des professions intermédiaires dans les actifs de 15-64 ans (%)
30	TX_CHOM_2554	Taux de chômage des 25-54 ans (%)

Tableau 8.7 – Variables Insee sélectionnées dans le modèle 3

Modèle 4

Aux variables utilisées dans le modèle saturé 2, nous avons rajouté la variable donnant la probabilité de défaut moyenne des clients de la banque par IRIS (PD_MOY_IRIS). La variable PD_MOY_IRIS arrive alors en 6e position dans le modèle brut mais la performance globale du modèle reste identique.

Étape	Variable	Libellé
6	PD_MOY_IRIS	PD moyenne des clients de La Banque Postale associée à l'IRIS
13	RFUCQ108	1er quartile des revenus (en €) par UC
15	RP_VOIT2P	Part des ménages avec 2 voitures ou plus (%)
18	ACTOCC15P_TCOM	Part des actifs occupés allant en transport en commun au travail (%)
22	LOG_RSECOCC	Part des résidences secondaires et logements occasionnels dans les logements (%)
23	ACTOCC15P_ILT1	Part des actifs occupés travaillant et résidant dans la même commune (%)
26	ACT1564_CS6	Part des ouvriers dans les actifs de 15-64 ans (%)
28	LOG_LOGVAC	Part des logements vacants dans les logements (%)
29	RFMQ208	Médiane des revenus (en €) par ménage
30	ACTOCC15P_ILT2	Part des actifs occupés travaillant dans une autre commune du même départ. de résidence (%)

Tableau 8.8 – Variables Insee sélectionnées dans le modèle 4

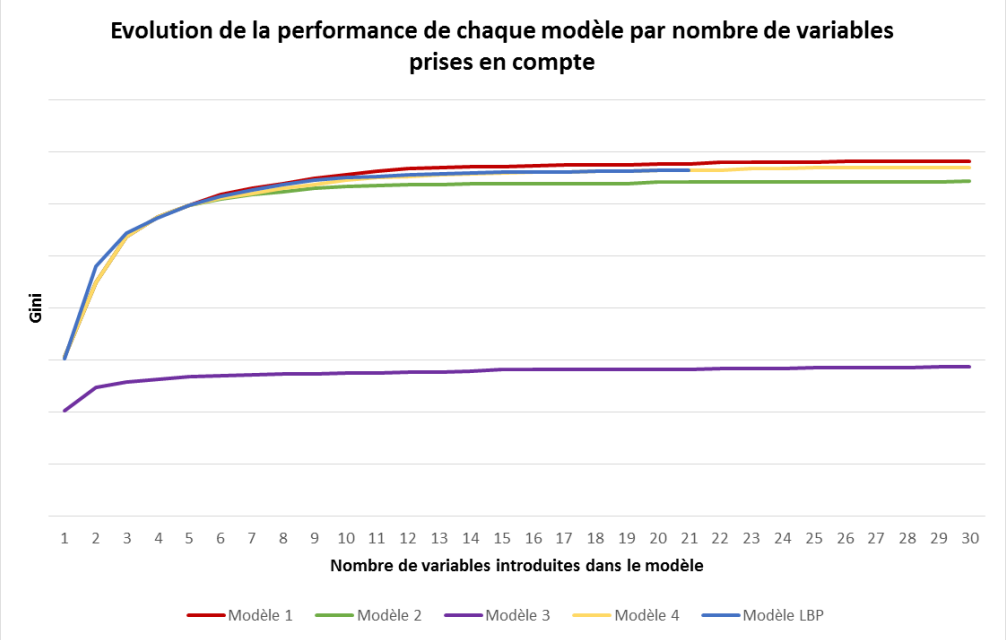

Figure 8.1 – Performance des modèles estimés

L'analyse graphique montre que les indicateurs Insee testés n'ont pas permis d'améliorer significativement la performance du score LBP. De plus, les variables sélectionnées provenant de l'Insee sont présentes assez tardivement dans le modèle.

Nous avons remarqué que dans tous les modèles testés, l'indicateur Insee RP_VOIT2P (donnant le pourcentage de la population de l'IRIS possédant plus de deux voitures) est toujours l'une des premières variables Insee à être sélectionnée dans le modèle.

Face aux résultats obtenus, nous décidons de nous orienter vers d'autres méthodes. Dans la partie

suivante, nous allons essayer de constituer directement des classes de risque afin de les comparer à celles obtenues lors de la construction du score sur les nouveaux clients LBP puis nous exposerons une méthode alternative à la sélection de variables par méthode *stepwise*.

8.3 Utilisation de méthodes alternatives

Dans cette section, nous présentons les résultats obtenus après utilisation d'une méthode alternative à la régression logistique : l'arbre de décision, et d'une méthode alternative à la sélection de variables *stepwise* : l'optimisation par algorithme génétique.

Les variables utilisées pour effectuer ces applications étant confidentielles, nous présentons nos résultats uniquement en terme de différence de performance par rapport au score nouveaux clients LBP.

8.3.1 Arbre de décision

Nous avons construit un arbre de décision à l'aide du logiciel SPAD et nous avons ensuite élagué cet arbre à l'aide d'un macro-programme SAS, mis au point par l'équipe Modélisation de LBP [20]. L'arbre de décision est construit sur la population de construction et sa performance est évaluée sur la population de construction et la population de validation.

Nous obtenons un arbre brut à 6 niveaux composé de 36 feuilles que nous réduisons à 21 feuilles après élagage [21]. Ces 21 feuilles sont ensuite regroupées en fonction de leur taux de défaut afin de former des classes de risque. Nous obtenons ainsi 14 classes de risque (les classes « ARBRE ») que nous comparons aux classes obtenues lors de la construction du score sur les nouveaux clients LBP.

Échantillon	**Construction**	**Validation**
Performance	-0,74	-0,49

TABLEAU 8.9 – Comparaison des indices de Gini entre les classes de risque ARBRE et LBP

Les résultats obtenus avec l'arbre sont de même peu concluants : après élagage, seule une variable INSEE (le 3e quartile des revenus par unité de consommation) intervient dans la segmentation de l'arbre. Bien que l'arbre soit en mesure d'identifier des poches de défaut pas forcément identifiées avec le score actuel (et inversement), cette méthode ne nous permet pas non plus de conclure quant à la pertinence d'une quelconque utilisation des indicateurs INSEE construits [22].

8.3.2 Algorithme génétique

Nous avons également voulu challenger la sélection de variables, qui est actuellement effectuée à l'aide d'une méthode *stepwise*, grâce à notre algorithme génétique. Pour cette application, nous avons uniquement utilisé les 22 variables LBP [23]. Nous prenons, plus précisément, les variables LBP découpées car nous voulons effectuer une régression logistique sur ces variables.

20. La méthodologie de construction de l'arbre est détaillée dans l'annexe E.
21. Ce nombre de feuilles est choisi car il permet d'obtenir 98 % de la performance totale possible observée.
22. Cette analyse est détaillée dans l'annexe E.
23. Nous avons retiré une des 23 variables explicatives initiales car l'équipe en charge de la construction du score avait écarté cette variable après l'avoir étudiée plus en détail car elle présente un comportement instable.

8.3.2.1 Implémentation en Python

Nous utilisons la classe *LogisticRegression* du module *sklearn.linear_model* et sa méthode *.fit* pour effectuer la régression logistique sur nos variables. Comme pour l'estimation des modèles, la performance de la régression logistique est mesurée par l'indice de Gini (D de Somer). Pour cela, nous calculons l'aire sous la courbe ROC à l'aide de la fonction *roc_auc_score* du module *sklearn.metrics*. L'aire sous la courbe ROC (AUC) est liée à l'indice de Gini (D) par la relation suivante :

$$AUC = \frac{D+1}{2} \ .$$

Nous cherchons à obtenir une valeur de D la plus grande possible.

Nous avons adapté l'AG que nous avions implémenté en Python afin d'effectuer une optimisation par algorithme génétique sur des variables de type caractère.

8.3.2.2 Démarche

Nous cherchons les 9 variables LBP [24] qui, parmi les 22 sélectionnées, permettent d'obtenir la meilleure régression logistique. Nous avons choisi ce nombre afin de pouvoir comparer nos résultats avec ceux obtenus lors de la construction du score sur les nouveaux clients LBP.

Nous effectuons nos calculs sur l'échantillon d'apprentissage (qui représente 70 % de la base totale).

La population des individus qui est utilisée pour l'optimisation par l'algorithme génétique est donc définie par :

— 9 gènes (qui prennent leur valeur dans la liste de nos 22 variables) [25] et,

— la performance de la régression logistique effectuée sur ses 9 gènes.

8.3.2.3 Résultats

La performance obtenue grâce à la sélection de variables par algorithme génétique n'est pas significativement plus élevée que celle obtenue avec les variables sélectionnées pour le score LBP. La plupart des variables que nous obtenons font aussi partie du modèle LBP.

L'intérêt de l'algorithme génétique est que nous avons une population de solutions candidates après avoir exécuté l'algorithme. Dans le tableau 8.10, nous présentons les *fitness* des 5 meilleurs individus de la population après avoir exécuté l'algorithme avec le paramétrage suivant :

— 100 individus ;

— un taux d'élites à 5 % ;

— une probabilité de croisement à 80 % ;

— un pas de sélection à 0,9 ;

— un nombre de générations maximal fixé à 200 ;

— un critère d'arrêt supplémentaire : au bout de 25 générations sans évolution de la meilleure performance, l'algorithme s'arrête.

L'algorithme s'arrête au bout de 2 h 15 après 45 itérations.

24. Car 9 variables LBP avaient été choisies suite à l'étape d'estimation de modèle lors de la construction du score sur les nouveaux clients LBP.

25. Il peut y avoir des doublons mais la régression logistique est effectuée sur les variables uniques parmi les 9 gènes de l'individu.

Modèle	1	2	3	4	5
Performance	+0.008	+0.008	+0.007	+0.004	+0.004

TABLEAU 8.10 – Performance des 5 meilleurs modèles obtenus avec la sélection de variables par algorithme génétique par rapport au modèle de référence

8.4 Conclusions de l'application de la régression logistique pour construire un score à l'entrée en relation client

Les différentes méthodes statistiques (régression logistique, arbre de décision, algorithme génétique) testées n'ont pas donné de résultats plus performants que ceux obtenus lors de la construction du score nouveaux clients LBP.

Les résultats obtenus après ajout des données externes sont peu concluants car ils ne permettent pas d'améliorer de manière significative les résultats obtenus lors de la construction du score sur les nouveaux clients LBP : le Gini obtenu avec les données INSEE est quasiment identique à celui obtenu avec les données LBP.

Nous pouvons en déduire que les informations sur le comportement général du client (obtenues grâce au questionnaire actuellement en production) sont plus pertinentes pour expliquer le défaut que les informations sur l'état du niveau de vie de son lieu d'habitation.

Nous avons remarqué que la variable INSEE portant sur le nombre de voitures moyen d'un ménage était présente dans tous nos modèles bruts ainsi que dans l'arbre non élagué. Il pourrait être intéressant d'étudier si une donnée supplémentaire sur le nombre de voitures du client serait en mesure d'améliorer la prédiction de son taux de défaut.

Il est à noter que notre étude comporte certaines limites qui peuvent expliquer en partie nos résultats :

— Il y a une différence de temps entre les données LBP (2014) et les données de recensement (millésimé 2006 [26]) et du RFL (2008).

— Pour les variables de recensement, nous avons effectué un choix d'indicateurs et tous les indicateurs possibles n'ont pas forcément pu être testés.

— Une partie de la base de données LBP n'a pas pu être géocodée (IRIS manquants) notamment les DOM/COM.

De plus, nous n'avons pas pu réaliser d'étude de la stabilité des indicateurs sur l'historique et nous avons ni backtesté ni mis en application un modèle construit à l'aide de variables INSEE.

Cependant, d'autres sources de données externes (à définir) pourraient permettre d'améliorer la performance du score.

La sélection de variables par algorithme génétique donne, quant à elle, de bons résultats et il pourrait être intéressant de l'intégrer dans le processus de sélection des variables et l'équipe Modélisation sera amenée à utiliser le module que nous avons développé en Python dans ces prochains travaux.

26. Les données issues du recensement sont récoltées sur une période de 5 ans de 2004 à 2008.

9 Conclusion générale

Comme nous avons pu le voir, la modélisation de la probabilité de défaut, qui s'inscrit dans la politique d'évaluation des risques d'une banque de détail, est un sujet vaste et complexe. En effet, la probabilité de défaut est un indicateur qui dépend à la fois du contexte réglementaire et des spécificités des portefeuilles étudiés.

La construction de modèles est, de plus, influencée par plusieurs éléments, pas toujours en accord avec les critères statistiques, notamment : les choix métiers, les applications et usages du modèle, les différentes réglementations, les ressources effectives (comme le temps machine, la puissance des ordinateurs à disposition, les données disponibles, ...), etc.

Face aux différentes caractéristiques des portefeuilles à modéliser, il est parfois nécessaire de développer des techniques alternatives à l'utilisation de la régression logistique pour la gestion et le pilotage des portefeuilles *Low Default* et des approches alternatives à l'utilisation des variables disponibles en interne pour les modèles à l'octroi.

Pour la modélisation de la probabilité de défaut sur les portefeuilles *Low Default*, tels que les Établissements Financiers ou le SPL Santé, les algorithmes d'optimisation présentés dans ce document, l'AG et le PSO, sont particulièrement adaptés à l'approche par vérification utilisée. En effet, grâce à ces algorithmes, nous pouvons reproduire le modèle dit « expert ». Cela permet de s'appuyer sur l'expérience de la banque pour construire le modèle, ce qui fait partie des exigences réglementaires à prendre en compte dans l'évaluation des risques d'une banque. De plus, les algorithmes d'optimisation tels que l'AG et le PSO sont des techniques statistiques reproductibles et robustes ce qui représente une nécessité dans le cadre des obligations d'audit notamment interne (Validation et Inspection Générale).

L'implémentation de notre algorithme en Python présente un intérêt métier fort. En effet, l'équipe Modélisation sera amenée à réutiliser ces algorithmes, notamment dans le cadre de la construction de modèles sur de nouveaux portefeuilles ou de la mise à jour des modèles de notation [1].

Pour la construction du score sur les nouveaux clients LBP, nous n'avons, en revanche, pas réussi à construire un modèle permettant d'améliorer la performance du score malgré les diverses approches alternatives testées. Cependant, en dépassant les limites de notre étude ou en utilisant d'autres types de données, il pourrait être possible d'aboutir à une conclusion plus satisfaisante. Face au désir d'amélioration de ce modèle, ce sujet reste ouvert et La Banque Postale sera amenée à tester de nouvelles approches. Néanmoins, les études que nous avons réalisées tendent à confirmer que, pour les seules données internes à disposition, le modèle actuellement en production atteint une performance optimale.

1. En effet, La Banque Postale étant une banque en pleine croissance, ses différents portefeuilles ont été amenés à évoluer ces dernières années, d'où la nécessité d'une mise à jour des modèles ou de construction de nouveaux modèles.

Liste des figures

Liste des tableaux

Bibliographie

CLERC, M. (2012), Standard Particle Swarm Optimisation. https://hal.archives-ouvertes.fr/hal-00764996.

CLERC, M. & KENNEDY, J. (2002), 'The particle swarm - explosion, stability, and convergence in a multidimensional complex space', *IEEE Transactions on Evolutionary Computation* **6**(1), 58–73.

DU, K.-L. & SWAMY, M. N. S. (2016), Particle Swarm Optimization, *in* 'Search and Optimization by Metaheuristics : Techniques and Algorithms Inspired by Nature', Springer International Publishing, pp. 153–173. https://doi.org/10.1007/978-3-319-41192-7_9.

EL DOR, A. (2012), Perfectionnement des algorithmes d'optimisation par essaim particulaire : applications en segmentation d'images et en électronique, Thèse de doctorat, Université Paris-Est. https://tel.archives-ouvertes.fr/tel-00788961.

HERTZ, A. (n.d.), 'Métaheuristiques', Notes du cours d'optimisation combinatoire (section 4.2), GERAD. https://www.gerad.ca/∼alainh/Metaheuristiques.pdf.

HOLLAND, J. (1975), *Adaptation in Natural and Artificial Systems*, MIT press.

Insee (2009*a*), 'Activité des résidents en 2006', Recensement de la population - Base infracommunale (IRIS). https://www.insee.fr/fr/statistiques/2028680.

Insee (2009*b*), 'Couples - Familles - Ménages en 2006', Recensement de la population - Base infracommunale (IRIS). https://www.insee.fr/fr/statistiques/2028565.

Insee (2009*c*), 'Diplômes - Formation en 2006', Recensement de la population - Base infracommunale (IRIS). https://www.insee.fr/fr/statistiques/2028251.

Insee (2009*d*), 'Logement en 2006', Recensement de la population - Base infracommunale (IRIS). https://www.insee.fr/fr/statistiques/2028562.

Insee (2009*e*), 'Population en 2006', Recensement de la population - Base infracommunale (IRIS). https://www.insee.fr/fr/statistiques/2028652.

Insee (2012), 'Indicateurs de structure et de distribution des revenus en 2008', Revenus fiscaux localisés des ménages (RFLM). https://www.insee.fr/fr/statistiques/1893295?sommaire=2389000#consulter.

KENNEDY, J. & EBERHART, R. (1995), Particle swarm optimization, *in* 'Actes de l'IEEE International Conference on Neural Networks de 1995', Vol. 4, pp. 1942–1948.

LE RICHE, R., SCHOENAUER, M. & SEBAG, M. (2007), *Un état des lieux de l'optimisation évolutionnaire et de ses implications en sciences pour l'ingénieur*, Vol. 2 of *Traité Mécanique et Ingénierie des Matériaux*, P. Breitkopf and C. Knopf-Lenoir edn, Hermes, chapter Modélisation Numérique : défis et perspectives, pp. 187–259.

MathWorks (2018*a*), 'Genetic Algorithm', Documentation R2018a. https://fr.mathworks.com/help/gads/genetic-algorithm.html.

MathWorks (2018*b*), 'Particle Swarm', Documentation R2018a. https://fr.mathworks.com/help/gads/particle-swarm.html.

MEZURA-MONTES, E. & COELLO COELLO, C. A. (2011), 'Constraint-handling in nature-inspired numerical optimization : Past, present and future', *Swarm and Evolutionary Computation* **vol.1**(n°4), p.173–194. http://www.sciencedirect.com/science/article/pii/S2210650211000538.

PÉRIÉ, P. (2014), 'STA108 Sondages', Cours au Cnam. http://maths.cnam.fr/IMG/pdf/STA108_-_21NOV2014_cle841d24.pdf.

RAKOTOMALALA, R. (2017), 'Pratique de la régression logistique - régression logistique binaire et polytomique', Cours à l'Université Lumière Lyon 2. http://eric.univ-lyon2.fr/~ricco/cours/cours/pratique_regression_logistique.pdf.

REYNARD, R. & VIALETTE, P. (2018), 'Les dynamiques de la qualité de vie dans les territoires', Insee. Documents de travail N° H2018/02. https://www.insee.fr/fr/statistiques/3545995.

SAS (2010), 'The LOGISTIC Procedure', Documentation SAS/STAT(R) 9.22 User's Guide. http://support.sas.com/documentation/cdl/en/statug/63347/HTML/default/viewer.htm#logistic_toc.htm.

SHI, Y. & EBERHART, R. (1998), A modified particle swarm optimizer, *in* 'Actes de l'IEEE International Conference on Evolutionary Computation de 1998', pp. 69–73.

TRELEA, I. C. (2003), 'The particle swarm optimization algorithm : convergence analysis and parameter selection', *Information Processing Letters* **85**(6), 317 – 325. http://www.sciencedirect.com/science/article/pii/S0020019002004477.

XHONNEUX, S. (2008), 'Perception de l'optimisation en mathématiques et en économie au fil des siècles et l'enseignement du théorème de Lagrange', Présentation donnée aux journées nationales de l'APMEP. http://www.apmep.fr/IMG/pdf/Lu-33-Xhonneux-LaRochelle.pdf.

Annexes

A Algorithmes génétiques

A.1 Exemples d'autres méthodes

Dans cette section, nous présentons d'autres méthodes pouvant être utilisées pour construire un algorithme génétique. Pour rappel, les méthodes que nous avons finalement retenues sont présentées à la section 5.4.3.

Échelle de *fitness*

Il existe une méthode **proportionelle** où la valeur de mise à l'échelle de chaque individu est proportionnelle à sa *fitness*.

Sélection des parents

La sélection des parents peut s'effectuer via la méthode du **tournoi** où :

— Un certain nombre d'individus est sélectionné aléatoirement dans la population.

— Le tirage aléatoire des individus est pondéré par les coefficients de l'échelle de *fitness*.

— Le nombre d'individus (i.e. la taille du tournoi) est un paramètre défini par l'utilisateur.

— L'individu avec la meilleure *fitness* parmi ceux sélectionnés est ajouté à la population des parents.

Croisement

La méthode du **point unique de croisement** peut être utilisée pour croiser deux parents. Pour cette méthode, il faut tout d'abord tirer un nombre aléatoire k entre 1 et le nombre de variables p. Le premier enfant créé par deux parents est alors constitué des k premiers gènes du parent 1 suivi des gènes $k+1$ à p du parent 2.

Il est possible d'appliquer cette méthode avec deux, trois, etc. points de croisement et non un seul point. Avec p points de croisements, la méthode revient à la méthode de croisement par points que nous utilisons.

Une autre méthode de croisement est la méthode du **mélange** où les enfants créés sont une moyenne pondérée des 2 parents. Ainsi, tous les gènes issus d'un croisement prennent des nouvelles valeurs. Les gènes des deux enfants sont calculés par la relation suivante :

$$enfant_1 = \beta \cdot parent_1 + (1-\beta) \cdot parent_2$$
$$enfant_2 = \beta \cdot parent_2 + (1-\beta) \cdot parent_1$$

Où β est un nombre aléatoire entre 0 et une certaine valeur maximale choisie par l'utilisateur.

Mutation

La méthode de mutation peut être **gaussienne** :

— Pour chaque gène de l'individu, nous ajoutons un nombre choisi selon une distribution gaussienne de moyenne 0 et d'écart-type σ_g.

— L'écart-type décroît linéairement via un paramètre de rétrécissement *shrink* à chaque génération g :

$$\sigma_g = \sigma_{g-1}\left(1 - shrink \cdot \frac{g}{g_{max}}\right).$$

— C'est à l'utilisateur de choisir l'écart-type initial σ_0 et le paramètre de rétrécissement *shrink*.

L'avantage de cette méthode est que la mutation est plus forte au début de l'algorithme qu'à la fin (la place donnée à l'exploration de l'espace de recherche décroît au fil des générations). L'inconvénient majeur de cette méthode est qu'elle ne permet pas de respecter les contraintes d'espaces.

Migration

La migration est une méthode qu'il est possible d'ajouter à l'algorithme génétique[1]. Cette méthode consiste à faire interagir plusieurs populations d'individus. L'AG est appliqué à plusieurs sous-populations. Chaque population évolue de son côté selon les méthodes de sélection des parents, de croisement et de mutation définies, et, tout un certain nombre de générations (toutes les 20 générations par exemple), une certaine fraction (par exemple, 10 %) des individus d'une population migrent dans une autre population. Il existe plusieurs types de migration.

Si la migration est de type *forward*, pour chaque population i, les x meilleurs individus de la population i remplacent les x plus mauvais individus de la population $i+1$. Les x meilleurs individus de la population i sont copiés dans la population $i+1$ (i.e. la population i conserve ses meilleurs individus).

Pour une migration de type *backward* et *forward*, le principe est le même que pour la migration de type *forward* excepté que les meilleurs éléments de la population i ne sont pas seulement transmis à la population $i+1$, ils sont aussi transmis à la population $i-1$.

A.2 Schéma de fonctionnement de l'AG

1. Nous avons testé un algorithme génétique prenant en compte cette méthode sans réussir à obtenir des résultats plus performants, c'est pourquoi nous ne l'avons pas retenue dans notre implémentation finale de l'AG.

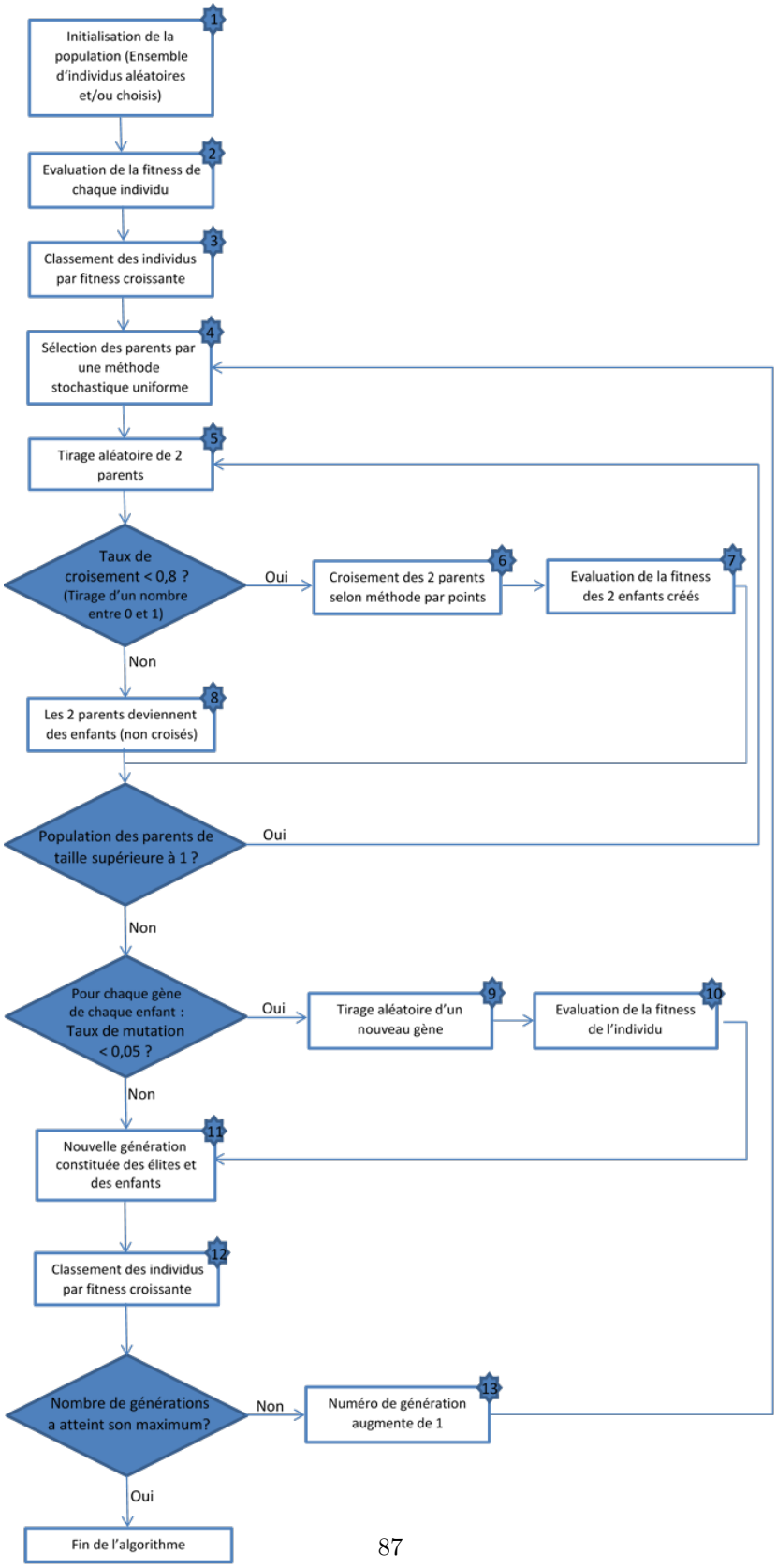

FIGURE A.1 – Fonctionnement de l'AG

B Schéma de fonctionnement du PSO

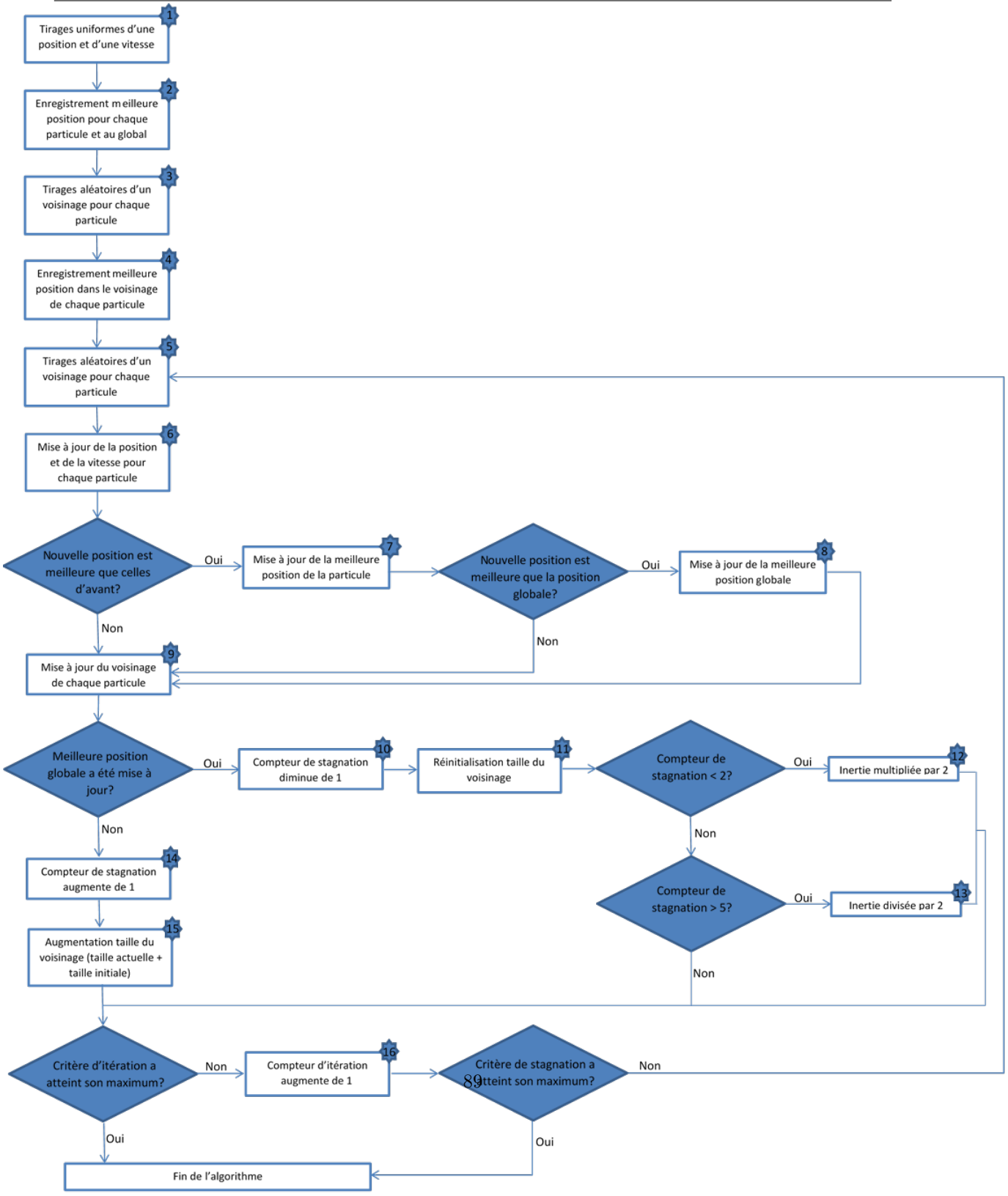

FIGURE B.1 – Fonctionnement du PSO

C Implémentation des algorithmes d'optimisation en Python

Dans cette annexe, nous présentons tout d'abord les fonctions utilisées en Python pour générer de l'aléatoire. Puis, nous expliquons rapidement la notion de classe avant de décrire les classes et fonctions codées en Python pour implémenter l'AG puis le PSO. Enfin, nous exposons une manière d'utiliser l'algorithme à partir des fonctions implémentées dans le package (dans le cadre des applications effectuées sur les Établissements Financiers et le SPL Santé).

C.1 Utilisation du module *random*

Ce module de Python permet de générer des nombres pseudo-aléatoires. Les méthodes du package *random* que nous avons utilisées pour le codage des algorithmes sont :

- *uniform(a,b)* : Tirage uniforme d'un nombre réel entre a et b (si $a \leq b$) ou entre b et a (si $b \leq a$), a et b inclus.
- *random()* : Tirage uniforme d'un nombre réel entre 0 et 1, 0 inclus.
- *choices(x, k)* : Tirage avec remise de k éléments de la liste x.
- *shuffle(x)* : Mélange de la liste x.
- *seed(n)* : Permet la reproductibilité des résultats. Pour une même graine n, l'appel à une fonction du module *random* génère la même série de nombres pseudo-aléatoires.

C.2 Quelques rappels sur les classes

Pour coder nos algorithmes, nous avons utilisé la programmation orientée objet. Chaque algorithme est donc défini par ses classes.

En Python, les classes sont crées par :

```
class Nom_classe (object) :
```

object indique que la classe **Nom_classe** hérite (des fonctions, méthodes, ...) de la classe **object**. Une classe peut hériter d'une autre classe. Le nom d'une classe commence généralement par une majuscule.

Pour initialiser la classe, il faut utiliser la commande :

```
def __init__ (self, param1, param2, ...) :
    self.var1 = param1
    self.var2 = [param1, param2]
    ...
```

Elle permet de définir les paramètres propres à une classe i.e. les attributs d'une classe. Après avoir instancié une classe, il est possible d'accéder à chacun de ses attributs grâce aux méthodes *.var*1, *.var*2, etc.

Pour toutes les fonctions définies dans une classe le premier argument est self. Il fait référence à la classe et permet de prendre en compte tous les attributs de celle-ci. Dans le corps de construction de la classe, pour avoir accès à un paramètre de la classe, il faut utiliser self.*nom_variable*. Les fonctions d'une classe sont des méthodes, elles s'appliquent directement à une instance de la classe en utilisant *.nom_méthode(argument1, ...)* (le mot clé self n'est pas à passer en argument).

C.3 Classes utilisées dans l'implémentation de l'algorithme génétique

Nous avons implémenté deux classes principales :

— La classe **Individu** qui permet de créer un individu (aléatoire ou non).

— La classe **Population** qui permet de créer un ensemble de représentants de la classe Individu et ensuite d'appliquer notre AG à cette population.

C.3.1 Attributs et fonctions propres à la classe Individu

	Nom	Description
Variables nécessaires à l'initialisation	*f_obj*	fonction objectif
	nvar	nombre de variables i.e. nombre de gènes de l'individu
	binf	liste des valeurs minimales prises par chaque variable
	bsup	liste des valeurs maximales prises par chaque variable
	indiv	facultatif, liste des valeurs prises par chaque variable pour cet individu si l'utilisateur souhaite créer un individu spécifique et non un individu aléatoire
Attributs	*indiv*	liste contenant la valeur de chaque variable pour cet individu, cette liste peut soit être rentrée par l'utilisateur, soit être crée aléatoirement
	fitness	valeur prise par la fonction objectif en cet individu

Nom méthode	Argument	Description
new_fitness	*f_obj* fonction objectif	Permet de mettre à jour la *fitness* de l'individu.

Tableau C.1 – Attributs et méthodes de la classe Individu

C.3.2 Attributs et fonctions propres à la classe Population

	Nom	Description
Variables nécessaires à l'initialisation	*f_obj*	fonction objectif
	npop	nombre d'individus/taille de la population
	nvar	nombre de variables
	binf	liste des valeurs minimales prises par chaque variable
	bsup	liste des valeurs maximales prises par chaque variable
	txelite	taux d'élites
	indivs	facultatif, pour chaque individu que l'utilisateur souhaite créer lui-même (individus non aléatoires), liste des valeurs prises par chaque variable pour chacun de ces individus. Si le nombre d'individus renseignés est inférieur à *npop*, le reste des individus est crée aléatoirement.
Attributs	*f_obj*	fonction objectif
	npop	taille de la population
	nvar	nombre de variables
	binf	liste des valeurs minimales prises par chaque variable
	bsup	liste des valeurs maximales prises par chaque variable
	pop	liste des individus (de la classe *Individu*) de la population
	scale	échelle des scores : liste du score de *fitness* attribué à chaque individu (du premier au dernier)
	nbelite	nombre d'élites (égal à la partie entière de $txelite * npop$)
	nbpar	nombre de parents (égal à $npop - nbelite$)

Nom méthode	Argument		Description
selection_fonc	*step_size*	pas de sélection des parents	Fonction sélectionnant une population de parents selon une méthode stochastique uniforme. La population de parents remplace directement les anciens individus de la population.
ga	*gmax*	nombre de générations maximal	Performe l'AG avec la fonction de sélection des parents par méthode stochastique uniforme, la méthode de croisement par points et une mutation aléatoire adaptée aux contraintes. Retourne le meilleur individu de la population après *gmax* itérations de l'algorithme et le temps d'exécution.
	txmut	taux de mutation	
	txcrossover	probabilité de croisement	
	step_size	pas de sélection des parents	

Tableau C.2 – Attributs et méthodes de la classe Population

C.3.3 Exemple d'utilisation de l'AG implémenté dans le cadre de l'application aux Établissements Financiers

Nous présentons ci-dessous un exemple de script Python permettant d'optimiser la fonction objectif utilisée dans le cadre de la modélisation de la fonction de score du portefeuille Établissements Financiers.

```
#Importation des modules Python
import random

from math import sqrt

#Importation de notre module d'AG
import GeneticAlgorithm as AG

#Importation des donnees
#Nombre d'entite
N = 109
#Input RO vecteur dim N rang expert
RO = []
#Input Vx matrice dim N*p donnees des p variables pour les N entites
VX = []

DATA = open("input_data_banks.txt", "r")
DATA.readline()
for i in range(N):
    a = DATA.readline()
    b = a.split()
    RO.append(float(b[0]))
    VX_i = []
    for j in b[1:]:
        VX_i.append(float(j))
    VX.append(VX_i)
DATA.close()

#Construction de la fonction de score
def score_fonc(w):
    """
        Fonction de score pour chaque entite ent.

        :param w: Liste des poids pour chaque variable.
        :type w: list
        :return: Liste de la note donnee par le modele pour chaque entite.
    """
    scores = []
    for ent in VX:
        res = 0
        for var in range(P):
            res = res + w[var] * ent[var]
        scores.append(res)
    return scores

#Construction de la fonction objectif (tau-b de kendall + penalite)
def taub_kendall(w):
    """
        Fonction du tau-b de Kendall entre les rangs experts et
        la note donnee par la fonction score (note modele).
```

```
        Ajout d'une penalite permettant de respecter la contrainte sur la somme
        des poids egale a 1 (penalite = 10*(1 - somme(i=1 a P)(w_i))^2).

        :param w: Liste des poids pour chaque variable.
        :type w: list
        :return: Tau-b de Kendall pour ce modele avec prise en compte de la penalite.

        .. note:: Plus le tau-b est proche de -1 et plus le modele est considere comme performant.
        .. seealso :: score_fonc
    """
    tau = 0
    n_x = 0
    n_y = 0
    score = score_fonc(w)
    for ent_i in range(N-1):
        f_i = score[ent_i]
        for ent_j in range(ent_i+1, N):
            f_j = score[ent_j]
            signe = (RO[ent_i] - RO[ent_j]) * (f_i - f_j)
            if signe < 0:
                tau = tau - 1
            elif signe > 0:
                tau = tau + 1
            else:
                if RO[ent_i] == RO[ent_j]:
                    n_x = n_x + 1
                elif f_i == f_j:
                    n_y = n_y + 1
    tau = tau/sqrt((N * (N-1)/2 - n_x) * (N * (N-1)/2 - n_y))
    tau = tau + 10*(sum(w)-1)**2
    return tau

#Espace des variables
P = 14
BMIN = [0.05, 0.05, 0.025, 0.05, 0.025, 0.025, 0.05, 0, 0, 0.025, 0, 0, 0.025, 0]
BMAX = [0.25, 0.25, 0.15, 0.25, 0.15, 0.15, 0.25, 0.1, 0.1, 0.15, 0.1, 0.1, 0.15, 0.1]

#Definitions des parametres
graine = 3
npop = 200
txelite = 0.1
gmax = 500
txmut = 0.05
txcrossover = 0.8
step_size = 0.9

#Execution de l'algorithme
random.seed(graine)
pop = AG.Population(taub_kendall, npop, P, BMIN, BMAX, txelite)
random.seed(graine)
results = pop.ga(gmax, txmut, txcrossover, step_size)
```

C.4 Classes utilisées dans l'implémentation de l'algorithme d'optimisation par essaim particulaire

Nous avons implémenté deux classes principales :

— La classe **Particule** qui permet de créer ou modifier une particule appartenant à un certain espace de recherche E.

— La classe **ParticuleSwarmOptimizer** qui permet d'appliquer l'algorithme PSO sur un ensemble d'instances de la classe *Particule* (i.e. un essaim de particules).

C.4.1 Attributs et fonctions propres à la classe Particule

	Nom	Description
Variables nécessaires à l'initialisation	*f_obj*	fonction objectif
	nvar	dimension de l'espace des variables
	binf	liste (de taille *nvar*) des valeurs minimales prises par chaque variable
	bsup	liste (de taille *nvar*) des valeurs maximales prises par chaque variable
Attributs de la classe	*position*	liste de *nvar* valeurs choisies aléatoirement entre *binf* et *bsup* pour chaque coordonnée
	vitesse	liste de *nvar* valeurs choisies aléatoirement entre -2 et 2
	nvar	dimension de l'espace des variables
	binf	liste (de taille *nvar*) des valeurs minimales prises par chaque variable
	bsup	liste (de taille *nvar*) des valeurs maximales prises par chaque variable

Nom méthode	Argument		Description
update_vitesse_pos	*bestpart*	meilleure position de la particule	Permet de mettre à jour la vitesse et la position (en respectant les contraintes sur l'espace) selon les formules décrites à la section 5.5.3.
	bestvoisi	meilleure voisine de la particule	
	inertia	inertie w	
	y_1	coefficient d'ajustement y_1	
	y_2	coefficient d'ajustement y_2	

Tableau C.3 – Attributs et méthodes de la classe Particule

C.4.2 Attributs et fonctions propres à la classe ParticuleSwarmOptimizer

	Nom	Description
Variables nécessaires à l'initialisation	*npart*	taille de l'essaim
	nvar	dimension de l'espace des variables
	binf	liste (de taille *nvar*) des valeurs minimales prises par chaque variable
	bsup	liste (de taille *nvar*) des valeurs maximales prises par chaque variable
	f_obj	fonction objectif
Attributs de la classe	*essaim*	liste (de longueur *npart*) contenant toutes les particules de l'essaim
	vitesse	liste de *nvar* valeurs choisies aléatoirement entre -2 et 2
	nvoisi	nombre de voisines d'une particule, est initialisé égal à *npart*
	best_part	liste contenant pour chaque particule une liste composée de la meilleure position de la particule (initialisée à sa position initiale) et de la valeur de la fonction objectif *f_obj* en cette position
	best_voisi	liste contenant, pour chaque particule, une liste composée de la meilleure voisine de la particule (qui est initialisée à sa position initiale) et la valeur de la fonction objectif *f_obj* en cette position
	solution	liste de dimension 2 initialisée à la meilleure position rencontrée par l'ensemble des particules et la valeur de *f_obj* en cette position

Nom méthode	Argument		Description
update_voisinage	*Aucun*		Permet de mettre à jour la liste *best_voisi* de la meilleure voisine de chaque particule. La liste des particules voisines à chaque particule est choisie aléatoirement et l'ensemble des particules est parcouru dans un ordre aléatoire.
optimise	*kmax*	nombre d'itérations max	Permet d'optimiser l'essaim de particules via PSO. Retourne la variable *solution* (qui est une liste contenant la meilleure position et la valeur de la fonction objectif en cette position), les valeurs finales des compteurs k et c et le temps d'exécution (en secondes) de l'algorithme.
	cmax	compteur de stagnation max	
	taille_voisi	pourcentage de voisines	
	w	inertie	
	y_1	coefficient d'ajustement y_1	
	y_2	coefficient d'ajustement y_2	

Tableau C.4 – Attributs et méthodes de la classe ParticuleSwarmOptimizer

D Données utilisées dans le cadre de l'amélioration du score sur les clients à l'entrée en relation

D.1 Variables du recensement

Les indicateurs construits à partir des variables de recensement de l'Insee sont présentés dans le tableau ci-dessous. Les variables utilisées dans le calcul des indicateurs sont les variables brutes des bases de données Insee.

N°	Famille	Nom variable	Libellé	Calcul
1	Activité	TX_EMP_2554	Taux d'emploi des 25-54 ans (%)	100 * P06_ACTOCC2554/P06_POP2554
2	Activité	EMP_1524	Part des 15-24 ans dans l'emploi (%)	100 * P06_ACTOCC1524/P06_ACTOCC1564
3	Activité	EMP_2554	Part des 25-54 ans dans l'emploi (%)	100 * P06_ACTOCC2554/P06_ACTOCC1564
4	Activité	EMP_5564	Part des 55-64 ans dans l'emploi (%)	100 * P06_ACTOCC5524/P06_ACTOCC1564
5	Activité	EMP_SAL15P	Part des salariés dans l'emploi (%)	100 * P06_SAL15P/P06_ACTOCC15P
6	Activité	EMP_NSAL15P	Part des non-salariés dans l'emploi (%)	100 * P06_NSAL15P/P06_ACTOCC15P
7	Activité	EMP_SAL15P_CDI	Part des salariés CDI, Fonction publique dans l'emploi (%)	100 * P06_SAL15P_CDI/P06_ACTOCC15P
8	Activité	EMP_SAL15P_CDD_INTERIM	Part des salariés CDD, intérim dans l'emploi des 15 ans et + (%)	100 * (P06_SAL15P_CDD + P06_SAL15P_INTERIM)/P06_ACTOCC15P
9	Activité	EMP_SAL15P_EMPAID_APPR	Part des salariés 15+ Emplois aidés, Apprentissage - Stage dans l'emploi (%)	100 * (P06_SAL15P_EMPAID + P06_SAL15P_APPR)/P06_ACTOCC15P
10	Activité	EMP_NSAL15P_INDEP_AIDFAM	Part des non-salariés indépendants, aides familiaux dans l'emploi 15+ (%)	100 * (P06_NSAL15P_INDEP + P06_NSAL15P_AIDFAM)/P06_ACTOCC15P
11	Activité	EMP_NSAL15P_EMPLOY	Part des non-salariés employeurs dans l'emploi (%)	100 * P06_NSAL15P_EMPLOY/ P06_ACTOCC15P
12	Activité	TX_EMP_1564	Taux d'emploi des 15-64 ans (%)	100 * P06_ACTOCC1564/P06_POP1564
13	Activité	CHOM_1564	Part des chômeurs dans la population 15-64 ans (%)	100 * P06_CHOM1564/P06_POP1564

14	Activité	ETUD_1564	Part des éleves, étudiants, stagiaires non rémunérés dans la pop. 15-64 ans (%)	100 * P06_ETUD1564/P06_POP1564
15	Activité	RETR_1564	Part des retraités, préretraités dans la pop. 15-64 ans (%)	100 * P06_RETR1564/P06_POP1564
16	Activité	AINACT_1564	Part des autres inactifs dans la population 15-64 ans (%)	100 * P06_AINACT1564/P06_POP1564
17	Activité	TX_CHOM_1564	Taux de chômage des 15-64 ans (%)	100 * P06_CHOM1564/P06_ACT1564
18	Activité	TX_CHOM_1524	Taux de chômage des 15-24 ans (%)	100 * P06_CHOM1524/P06_ACT1524
19	Activité	TX_CHOM_2554	Taux de chômage des 25-54 ans (%)	100 * P06_CHOM2554/P06_ACT2554
20	Activité	TX_CHOM_5564	Taux de chômage des 15-64 ans (%)	100 * P06_CHOM5564/P06_ACT5564
21	Activité	ACT1564_CS1	Part des agriculteurs exploitants dans les actifs de 15-64 ans (%)	100 * C06_ACT1564_CS1/C06_ACT1564
22	Activité	ACT1564_CS2	Part des artisans commer. chefs ent. dans actifs de 15-64 ans (%)	100 * C06_ACT1564_CS2/C06_ACT1564
23	Activité	ACT1564_CS3	Part des Cadres Prof int sup dans les actifs de 15-64 ans (%)	100 * C06_ACT1564_CS3/C06_ACT1564
24	Activité	ACT1564_CS4	Part des Prof intermédiaires dans les actifs de 15-64 ans (%)	100 * C06_ACT1564_CS4/C06_ACT1564
25	Activité	ACT1564_CS5	Part des Employés dans les actifs de 15-64 ans (%)	100 * C06_ACT1564_CS5/C06_ACT1564
26	Activité	ACT1564_CS6	Part des Ouvriers dans les actifs de 15-64 ans (%)	100 * C06_ACT1564_CS6/C06_ACT1564
27	Famille	MEN_PSEUL	Part des ménages composés d'une personne (%)	100 * C06_MENPSEUL/C06_MEN
28	Famille	MEN_COUPAENF	Part des ménages composés d'un couple avec enfant (%)	100 * C06_MENCOUPAENF/C06_MEN
29	Famille	MEN_COUPSENF	Part des ménages composés d'un couple sans enfant (%)	100 * C06_MENCOUPSENF/C06_MEN
30	Famille	MEN_FAMMONO	Part des ménages composé d'une famille monoparentale (%)	100 * C06_MENFAMMONO/C06_MEN

31	Famille	MEN_SFAM	Part des ménages composés autrement sans famille (%)	100 * C06_MENSFAM/C06_MEN
32	Famille	FAM_NE24F0	Part des familles avec 0 enfant de moins de 25 ans (%)	100 * C06_NE24F0/C06_FAM
33	Famille	FAM_NE24F1	Part des familles avec 1 enfant de moins de 25 ans (%)	100 * C06_NE24F1/C06_FAM
34	Famille	FAM_NE24F2	Part des familles avec 2 enfants de moins de 25 ans (%)	100 * C06_NE24F2/C06_FAM
35	Famille	FAM_NE24F3P	Part des familles avec 3 enfants et + de moins de 25 ans (%)	100 * (C06_NE24F3+C06_NE24F4P) /C06_FAM
36	Formation	NSCOL15P_BACP	Part de la population non scolarisée 15+ titulaire au moins du BAC(%)	100 * (P06_NSCOL15P_BAC + P06_NSCOL15P_BACP2 + P06_NSCOL15P_SUP)/P06_NSCOL15P
37	Population	POP_0019	Part des 0-19 ans dans la population (%)	100 * P06_POP0019/P06_POP
38	Population	POP_2064	Part des 20-64 ans dans la population (%)	100 * P06_POP2064/P06_POP
39	Population	POP_65P	Part des 65 ans et + dans la population (%)	100 * P06_POP65P/P06_POP
40	Logement	LOG_RP	Part des rés. principales dans les logements (%)	100 * P06_RP/P06_LOG
41	Logement	LOG_RSECOCC	Part des rés. secondaires et logts occasionnels dans les logts (%)	100 * P06_RSECOCC/P06_LOG
42	Logement	LOG_LOGVAC	Part des logements vacants dans les logts(%)	100 * P06_LOGVAC/P06_LOG
43	Logement	RP_NBPI_MOY	Nombre moyen de pièces par personnes	P06_NBPI_RP/P06_NPER_RP
44	Logement	RP_PROP	Part des résidences principales occupées par propriétaires (%)	100 * P06_RP_PROP/P06_RP
45	Logement	RP_LOC	Part des résidences principales occupées par locataires (%)	100 * P06_RP_LOC/P06_RP
46	Logement	RP_LOCHLMV	Part des résidences principales HLM louée vide (%)	100 * P06_RP_LOCHLMV/P06_RP
47	Logement	RP_GRAT	Part des résidences principales occupées gratuitement (%)	100 * P06_RP_GRAT/P06_RP

48	Logement	RP_ANEM_MOY	Ancienneté moyenne d'emménagement dans la résidence principale	P06_ANEM_RP/P06_RP
49	Logement	RP_VOIT0	Part des ménages avec 0 voiture (%)	100 * (1 - P06_RP_VOIT1P/P06_RP)
50	Logement	RP_VOIT1	Part des ménages avec 1 voiture (%)	100 * P06_RP_VOIT1/P06_RP
51	Logement	RP_VOIT2P	Part des ménages avec 2 voitures ou plus (%)	100 * P06_RP_VOIT2P/P06_RP
52	Logement	LOG_MAISON	Part des maisons dans les logements (%)	100 * P06_MAISON/P06_LOG
53	Logement	LOG_APPART	Part des appartements dans les logements (%)	100 * P06_APPART/P06_LOG
54	Logement	RP_NBPI_MOY	Nombre moyen de pièces par résidence principale	P06_NBPI_RP/P06_RP
55	Formation	NSCOL15P_DIPL0	Part de la population 15+ ans non scolarisée sans diplôme (%)	100 * P06_NSCOL15P_DIPL0 /P06_NSCOL15P
56	Activité	SAL15P_CDI	Part des salariés en CDI ou dans la fonction publique (%)	100 * P06_SAL15P_CDI/P06_SAL15P
57	Activité	ACTOCC15P_ILT1	Part des actifs occupés travaillant et résidant dans la même commune (%)	100 * P06_ACTOCC15P_ILT1 /P06_ACTOCC15P
58	Activité	ACTOCC15P_ILT2	Part des actifs occ. travaillant autre com même dep resid. (%)	100 * P06_ACTOCC15P_ILT2 /P06_ACTOCC15P
59	Activité	ACTOCC15P_MAR	Part des actifs occ. allant au travail à pied (%)	100 * C06_ACTOCC15P_MAR /C06_ACTOCC15P
60	Activité	ACTOCC15P_VOIT	Part des actifs occ. allant en voiture au travail (%)	100 * C06_ACTOCC15P_VOIT /C06_ACTOCC15P
61	Activité	ACTOCC15P_TCOM	Part des actifs occ. allant en transport en commun au travail (%)	100 * C06_ACTOCC15P_TCOM /C06_ACTOCC15P

Tableau D.1 – Ensemble des indicateurs de recensement construits

D.2 Variables de revenus

Les variables de revenus, directement issues des bases de données disponibles sur le site de l'Insee, sont les suivantes :

N°	Nom variable	Libellé
1	NBPERS08	Nombre de personnes des ménages fiscaux
2	RFPQ108	1er quartile (€) par personne
3	RFPQ208	Médiane (€) par personne
4	RFPQ308	3e quartile (€) par personne
5	RFPIQ08	Écart interquartile (€) par personne
6	RFPD108	1er décile (€) par personne
7	RFPD208	2e décile (€) par personne
8	RFPD308	3e décile (€) par personne
9	RFPD408	4e décile (€) par personne
10	RFPD608	6e décile (€) par personne
11	RFPD708	7e décile (€) par personne
12	RFPD808	8e décile (€) par personne
13	RFPD908	9e décile (€) par personne
14	RFPRD08	Rapport interdécile (par personne)
15	RFPET08	Écart-type (€) par personne
16	RFPMO08	Moyenne (€) par personne
17	RFPGI08	Indice de Gini (par personne)
18	NBUC08	Nombre d'UC
19	RFUCQ108	1er quartile (€) par UC
20	RFUCQ208	Médiane (€) par UC
21	RFUCQ308	3e quartile (€) par UC
22	RFUCIQ08	Écart interquartile (€) par UC
23	RFUCD108	1er décile (€) par UC
24	RFUCD208	2e décile (€) par UC
25	RFUCD308	3e décile (€) par UC
26	RFUCD408	4e décile (€) par UC
27	RFUCD608	6e décile (€) par UC
28	RFUCD708	7e décile (€) par UC
29	RFUCD808	8e décile (€) par UC
30	RFUCD908	9e décile (€) par UC
31	RFUCRD08	Rapport interdécile (par UC)
32	RFUCET08	Écart-type (€) par UC
33	RFUCMO08	Moyenne (€) (par UC)
34	RFUCGI08	Indice de Gini (par UC)
35	NBMEN08	Nombre de ménages fiscaux
36	RFMQ108	1er quartile (€) par ménage
37	RFMQ208	Médiane (€) par ménage
38	RFMQ308	3e quartile (€) par ménage
39	RFMIQ08	Écart interquartile (€) par ménage
40	RFMD108	1er décile (€) par ménage
41	RFMD208	2e décile (€) par ménage
42	RFMD308	3e décile (€) par ménage
43	RFMD408	4e décile (€) par ménage

44	RFMD608	6^{e} décile (€) par ménage
45	RFMD708	7^{e} décile (€) par ménage
46	RFMD808	8^{e} décile (€) par ménage
47	RFMD908	9^{e} décile (€) par ménage
48	RFMRD08	Rapport interdécile (par ménage)
49	RFMET08	Écart-type (€) par ménage
50	RFMMO08	Moyenne (€) par ménage
51	RFMGI08	Indice de Gini (par ménage)
52	PMIMP08	Part des ménages imposés (%)
53	PTSA08	Part des traitements salaires (%)
54	PCHO08	dont indemnités de chômage (%)
55	PPEN08	Part des pensions/retraites/rentes (%)
56	PBEN08	Part des bénéfices (%)
57	PAUT08	Part des autres revenus (%)

Tableau D.2 – Liste des variables RFL

E Méthodologie de construction d'un arbre

Les résultats présentés à la section 8.3.1 ont été obtenus en utilisant la méthodologie décrite dans cette annexe.

Nous avons utilisé la méthode des arbres de segmentation pour construire directement des classes homogènes de risque. Le logiciel SPAD permet la construction d'un tel arbre.

E.1 Arbre de segmentation CHAID

La segmentation par arbre consiste à construire un arbre « inversé » à l'aide de divisions successives des unités statistiques (les clients) de la population en segments homogènes (appelés nœuds), par rapport à la variable à expliquer (le défaut à 12 mois) et en utilisant l'information des variables explicatives (indicateurs du recensement, variables RFL et variables LBP).

L'arbre de segmentation est utilisé comme une technique de classement pour identifier les critères permettant de repartir les individus d'une population en n classes prédéfinies (ici, $n = 2$: clients défaillants vs clients sains). Il commence par choisir la variable qui, par ses modalités, sépare le mieux les individus de chaque classe, de façon à avoir des sous-populations, qui constituent des nœuds, contenant chacune le plus possible d'individus d'une seule classe [1]. Ensuite, l'opération est réitérée sur chaque nouveau nœud obtenu jusqu'à ce que la séparation des individus ne soit plus possible ou souhaitable par rapport aux paramètres définis en amont.

Les nœuds terminaux sont appelés des feuilles et sont, par construction, majoritairement constitués d'individus présentant des caractéristiques homogènes. Un individu est affecté à une feuille quand il satisfait l'ensemble des règles permettant d'arriver à celle-ci. L'ensemble des règles associées à chaque feuille permet d'obtenir un modèle de classement. La segmentation de l'arbre s'arrête lorsque l'un des critères suivants est rempli :

- La profondeur de l'arbre a atteint une limite fixée (dans le paramétrage de l'outil SPAD elle correspond au « Nombre de niveaux de l'arbre »). Comme nous possédons beaucoup de variables et afin d'obtenir un arbre assez profond qui reste tout de même lisible, nous avons fixé ce critère à 6.
- La division ultérieure de tout nœud provoquerait la naissance d'un fils d'effectif inférieur à une valeur fixée (dans le paramétrage de l'outil SPAD elle correspond au « Minimum pour diviser un nœud »). Nous avons choisi comme seuil la valeur de 1 % afin de segmenter finement notre population. C'est-à-dire que chaque nœud contient au moins 1 % de la population de construction totale.
- Il n'y a aucun découpage des variables statistiquement significatif (aucun découpage ne rejette l'hypothèse nulle d'indépendance du test de χ^2 avec un risque de 1 %).

La méthode CHAID (CHi-squared Automatic Interaction Detector) a été choisie pour construire l'arbre de segmentation. Cette méthode consiste à fixer une règle d'arrêt qui permet de stopper la construction de l'arbre lors de la phase de modélisation et d'évaluer l'apport informationnel de la

1. Dans notre application, les individus appartenant à la classe défaut étant beaucoup moins représentés dans la population, l'arbre de décision permet d'identifier les individus avec un comportement similaire et de leur associer un taux de défaut.

segmentation que nous allons initier. Le choix de la variable explicative et de ses modalités devant séparer un nœud en plusieurs nœuds-fils est fait de sorte à maximiser la valeur du χ^2 de cette variable croisée avec la variable à expliquer, ici le défaut.

Nous avons choisi la méthode CHAID pour plusieurs raisons :

— elle est capable de traiter de manière indifférenciée les données continues et discrètes et dispose, de plus, d'un mécanisme intégré de sélection de variables ;

— elle est non-linéaire et prend en compte des interactions entre les variables explicatives ;

— elle permet d'obtenir directement des classes homogènes de risque ;

— elle permet d'extraire facilement des règles qui peuvent être implémentées dans les systèmes informatiques.

E.2 Démarche de la création de l'arbre à la constitution des classes de risque

Nous cherchons à construire un arbre de segmentation qui traduit la probabilité de tomber en défaut dans les 12 mois suivant l'ouverture du compte.

E.2.1 Préparation des variables

Les variables utilisées pour construire l'arbre sont les variables (quantitatives et qualitatives) LBP et les variables Insee que nous avons retenues à la suite de l'analyse univariée (cf. section 8.2.2.1) non découpées. En effet, l'arbre procède lui-même à la création de modalités :

— Les variables quantitatives sont discrétisées automatiquement lors de l'élaboration de chaque nœud.

— Les modalités des variables qualitatives peuvent être regroupées lors de la segmentation de la population.

Cependant, avant de créer l'arbre, nous procédons à un traitement pour les variables manquantes et non applicables :

Valeurs non applicables

— Pour les variables quantitatives, nous remplaçons les valeurs aberrantes par -999.

— Pour les variables qualitatives, nous créons une modalité « VNA ».

Valeurs manquantes

— Pour les variables quantitatives, nous remplaçons les valeurs manquantes par la médiane des observations appartenant à la classe de découpage dans laquelle les valeurs manquantes de cette variable avaient été imputées (cf. section 8.2.2.3).

— Pour les variables qualitatives, nous créons une modalité « VM ».

E.2.2 Création d'un arbre brut

Dans un premier temps, nous construisons, à l'aide du logiciel SPAD, un arbre de segmentation brut. L'idée de cette démarche est de, tout d'abord, constituer un arbre de segmentation uniquement basé sur des critères statistiques afin :

— d'observer le potentiel de performance (cf. section suivante) ;

— d'identifier les variables explicatives qui optimisent la qualité de la segmentation.

E.2.3 Mesure de la performance

Cette étape est réalisée sous SAS.

La performance de l'arbre est mesurée avec l'indice de Gini [2]. La démarche consiste à ordonner les feuilles de l'arbre par taux de défaut décroissant et de calculer les éléments de la courbe de performance associée : le pourcentage cumulé de défauts et le pourcentage cumulé d'individus. L'indice de Gini est obtenu à partir de ces données [3].

E.2.4 Élagage de l'arbre

L'arbre brut a comme inconvénient majeur d'être sensible au sur-apprentissage mais aussi d'être « aveugle » au sens métier des branches produites. Pour traiter le sur-apprentissage nous procédons à un élagage progressif (suppression de branches) de l'arbre en contrôlant la perte de performance que cela génère (mesure sur l'échantillon de validation et l'échantillon de construction) [4].

E.2.5 Création de classes homogènes de risque

À cette étape, nous regroupons les feuilles (obtenues après élagage de l'arbre) ayant un taux de défaut proche afin d'obtenir des classes homogènes de risque. Pour cela, nous regardons le taux de défaut constaté pour chaque feuille sur les échantillons de construction et de de validation ainsi que sur la population totale.

Nous avons ensuite comparé les classes obtenues avec l'arbre aux classes construites lors de la modélisation du score sur les nouveaux clients LBP.

E.3 Analyse de l'homogénéité des classes de risque ARBRE et classes de risque LBP

Pour des raisons de confidentialité, les résultats de cette analyse ne sont pas présentés dans ce document.

Après avoir créé nos classes de risque, nous les comparons avec les classes de risque qui avaient été obtenues lors de la construction du score sur les nouveaux clients LBP. Pour cela nous regardons, pour chaque couple de classes de risque, si le taux de défaut de la population appartenant à la fois à la classe de risque LBP j et à la classe de risque ARBRE i est homogène au taux de défaut constaté de la classe de risque LBP j et aussi au taux de défaut de la classe de risque ARBRE i. Nous considérons que la population d'un couple de classes de risque est significative si plus d'1 % de la population totale appartient à la fois à la classe de risque LBP j et à la classe de risque ARBRE i.

2. Cf. section 6.3.3 pour la définition de l'indice de Gini.

3. En utilisant la procédure « FREQ » de SAS avec l'option « MEASURES », nous obtenons l'indice de Gini (Somers D R|C).

4. Cette étape est également réalisée sous SAS via un macro-programme développé par l'équipe Modélisation de LBP qui permet l'élagage de l'arbre brut tout en minimisant la perte de performance de l'arbre obtenu.

Par exemple, pour la classe de risque 2 de l'arbre, nous regardons le taux défaut des classes de risque 1, 2 et 3 de LBP (car plus de 1 % de la population est présente dans chaque couple formé avec ces classes de risque). Nous regardons, d'une part, si le taux de défaut des couples (ARBRE 2, LBP 1), (ARBRE 2, LBP 2) et (ARBRE 2, LBP 3) est homogène (moins de 1 % de différence) avec le taux de défaut de la classe de risque ARBRE 2. Et, d'autre part, nous nous assurons que les couples (ARBRE 2, LBP 1), (ARBRE 2, LBP 2) et (ARBRE 2, LBP 3) ont des taux de défaut croissant.

Les résultats de cette étude n'ont pas permis d'établir qu'une des répartitions par classe serait meilleure que l'autre, chaque répartition étant capable d'identifier des poches de risque.