Université Paris Dauphine

Mémoire de recherche (S13)

Estimation du manque à gagner

9 mai 2022

Gabriel Fiastre
Pierre Fihey
Artus Naar

Supervisé par :
Sylvain Benoit, Maître de conférence en finance, LEDa
Julien Stoehr, Maître de conférence en statistiques, Ceremade

Table des matières

Abstract

En 2020, le syndicat Solidaire Finances Publiques estimait le montant de la fraude fiscale en France à plus de 80 milliards d'euros. Parallèlement, l'Etat n'a récupéré que 7,8 milliards d'euros grâce à des contrôles fiscaux. Ce manque à gagner de près de 70 milliards représente une perte considérable pour l'Etat français qui manque de moyens pour lutter contre ce fléau.

Le terme de fraude fiscale désigne le fait d'agir contre la loi afin d'échapper à l'impôt ou de n'en payer qu'une partie. Cette pratique est évidemment punie par la loi et sanctionnée par les organismes de contrôles lorsqu'elle est détectée. Cependant, au vu des moyens dont ils disposent, il est impossible pour eux de contrôler toutes les entreprises. La fraude fiscale qui n'est pas détectée représente un manque à gagner évident pour ces organismes, c'est à dire une perte sur un bénéfice envisageable.

Un organisme de contrôle de la fraude fiscale agricole s'interroge sur son manque à gagner : Combien aurait-il pu récupérer si toutes les entreprises avaient été contrôlées?
Nous chercherons à proposer une estimation du manque à gagner par différentes approches, que nous comparerons.

Notations

Au long de ce mémoire, nous utiliserons les notations suivantes :

- Soit B la base de données totale comportant n observations et m variables
- En fonction du problème de modélisation on notera $\mathscr{Y} \in \mathbb{R}^n$ la variable cible et $\mathscr{X}$ l'ensemble des features
- On note $X = [\mathbb{1}_n \ \mathscr{X}] \in R^{n\times(m+1)}$
- Soit $i \in \{1,...,n\}$, on désignera par x_i la $i^{ème}$ observation (la $i^{ème}$ entreprise de la base), qui sera donc un vecteur de $\mathbb{R}^{m+1}$ représentant l'ensemble des covariables pour cette observation
- De même, pour $j \in \{1,...,m+1\}$, on notera x^j la $j^{ième}$ covariable de X, donc un vecteur de $\mathbb{R}^n$ représentant l'ensemble des valeurs $(x_i^j)_{i\in 1...n}$ prises pour chaque individu x_i
- De plus, si A est la $j^{ième}$ covariable de X, on notera $A_i = A(x_i) = x_i^j$ la valeur de la $j^{ième}$ covariable pour l'individu x_i
- Si δ est une valeur, on notera $\widehat{\delta}$ son estimation
- Enfin on note $p = m+1$

Introduction

Le but de ce mémoire sera de fournir une estimation de ce manque à gagner. Pour définir théoriquement le manque à gagner (MAG), il convient à présent de s'intéresser au montant de redressement.

Le montant de redressement n'est identifiable que pour certaines entreprises ayant été examinées par l'organisme de contrôle et pour lesquelles on a constaté une fraude. Soit C_i et F_i les variables binaires qui indiquent respectivement si un individu a été contrôlé et s'il est constaté fraudeur. On remarque que $F_i = 1 \implies C_i = 1$
On note M_i le montant du redressement effectivement observé, tel que :

$$\begin{cases} M_i > 0 & \text{si} \quad C_i = 1 \text{ et } F_i = 1 \\ 0 & \text{sinon} \end{cases} \qquad \forall i = 1,...n$$

Soit M_i^* la variable désignant le montant potentiel de redressement de l'entreprise $i \in 1,...n$. Soit F_i^* la variable binaire indiquant pour un individu non contrôlé s'il est potentiellement fraudeur ou non : valeur qu'on cherchera à prédire. On notera $n_{F^*} = \sum_{i=1}^{n} F_i^*$, et on remarque que $F_i^* = 1 \implies C_i = 0$

On suppose l'existence d'une relation linéaire entre le montant potentiel M_i^* et un ensemble de m facteurs explicatifs x_{ij} telle que :

$$M_i^* = \begin{cases} x_i\beta + \epsilon_i & \text{si } F_i^* = 1 \\ 0 & \text{sinon} \end{cases} \qquad \forall i = 1,...n \tag{1}$$

avec β un vecteur de p paramètres, ϵ_i les termes d'erreurs vérifiant $\mathbb{E}(\epsilon_i) = 0$

À partir du redressement potentiel M_i^*, nous pouvons déduire l'expression du MAG pour chacune des entreprises de l'échantillon. Le MAG est positif pour les entreprises ayant fraudé et n'ayant pas été contrôlées ; il est nul pour toutes les autres. Ainsi, si l'on note MAG_i le manque à gagner associé à l'entreprise i, celui-ci vérifie :

$$MAG_i = M_i^* \times \mathbb{1}_{(C_i=0)} \times \mathbb{1}_{(F_i^*=1)} \qquad \forall i = 1,...,n$$

Finalement, nous nous intéresserons plus spécifiquement au MAG agrégé, noté :

$$MAG = \sum_{i=1}^{n} MAG_i$$

Ce dernier est donc défini par la somme des redressements potentiels qui auraient pu être imposés à des entreprises fraudeuses si elles avaient été effectivement contrôlées alors qu'elles ne l'ont pas été en réalité.

Pour répondre à cette problématique, nous avons à notre disposition une base de données contenant des informations sur les entreprises sous le régime de l'organisme. Elle comporte des entreprises non contrôlées par l'organisme, des entreprises contrôlées et non fraudeuses, ainsi que des entreprises contrôlées et redressées pour fraude.

Notre travail va donc s'articuler en plusieurs parties distinctes. Dans un premier temps, il est nécessaire d'effectuer un travail de classification afin de déterminer parmi les entreprises non contrôlées lesquelles sont susceptibles d'être coupables de fraude (estimer F_i^*). Après avoir effectué ce travail de classification, nous allons répondre à un problème de régression afin d'attribuer un montant de redressement aux entreprises détectées comme fraudeuses. Une fois cette première estimation effectuée, il nous sera possible d'envisager d'autres techniques, notamment de Machine Learning, afin d'améliorer au mieux nos modèles.

Le but de ce mémoire est d'utiliser différentes méthodes de classification et de régression et donc de tester différentes associations de ces méthodes afin d'en comparer les résultats et efficacités.

1 Travail sur les données

La première étape de notre mémoire consiste en un travail important sur le jeu de données mis à notre disposition. Le but de cette étape est d'appréhender la base de données dans sa globalité, de comprendre les variables qui la composent et de travailler dessus afin d'en extraire une base sur laquelle il sera possible d'effectuer nos prédictions.

1.1 Appropriation du jeu de données

Il était important, dans un premier temps, de comprendre l'ensemble des 265 variables qui composaient notre base de 187 643 entreprises. Notre travail portant sur la fraude fiscale dans le milieu agricole, il était nécessaire de se renseigner sur les entreprises agricoles et leurs spécificités. Ces données nous ont en effet été transmises par un organisme de collecte des cotisations sociales pour le secteur agricole et comportaient des informations précises et parfois spécifiques à ce secteur sur les entreprises qui y sont affiliées. Cet organisme classe notamment les entreprises selon leurs catégories de risque, leurs secteurs d'activité ou encore leurs catégories juridiques et il nous semblait essentiel de comprendre toutes ces particularités avant de commencer notre travail.

Une fois ce travail de compréhension des variables terminé, il nous a fallu déceler les variables cibles de nos modèles ; c'est à dire identifier les variables sur lesquelles portent les prédictions tout au long de notre travail. En l'espèce, on rappelle que l'on cherchera à estimer :

- La variable binaire qui indique si l'entreprise est potentiellement coupable de fraude : F^*
- La variable qui indique le montant de redressement imposé aux entreprises potentiellement coupables de fraude : M^*

Pour terminer notre travail d'appropriation des données, nous avons décidé de nous pencher sur les variables qui semblaient nécessiter quelques modifications afin de pouvoir les étudier efficacement. Nous avons décidé, par exemple, de dichotomiser plusieurs variables qualitatives, c'est à dire transformer chaque modalité d'une variable en une nouvelle variable binaire indiquant l'appartenance de l'individu à cette modalité. Cette dichotomisation nous permettra d'éviter la présence de variables qualitatives qui ne peuvent être prises en compte lors d'une régression linéaire.

1.2 Gestion des Singularités

A la suite de ce travail d'appropriation du jeu de données, nous avons décidé de nous pencher sur certaines singularités que nous avons pu déceler dans notre base de données. On entend, par singularité, un problème structurel sur les données qui nous empêche de les modéliser. Il existe plusieurs types de singularité et il est donc nécessaire de travailler dessus afin de pouvoir effectuer le travail de régression et de prédiction que nous envisageons.
Dans notre base, nous avons pu déceler plusieurs types de singularités :

- Les variables qui ne possèdent qu'une seule modalité : Dans ce cas, nous pouvons les supprimer dans la mesure où leur présence n'a aucun effet dans le modèle que nous voulons mettre en place ;
- Les variables qui comportent de nombreuses valeurs N/A : Dans ce cas, nous avons décidé de les supprimer lorsque il était impossible de prédire une valeur de remplacement et de les remplacer quand cela nous semblait pertinent ;

- Certaines variables qualitatives qui ont été dichotomisées plusieurs fois selon des regroupements de modalités différents : Nous avons donc étudié les différents découpages afin de ne garder que le plus pertinent pour la réalisation d'un modèle.

Cependant, après avoir exploré plusieurs méthodes de sélection de variables lors la recherche de modèle de régression notamment, nous nous sommes apperçus que notre base de données comportait encore de nombreuses singularités qui rendaient ce travail impossible.
Les méthodes de sélection de modèles que nous allons utiliser ne fonctionnent pas en présence de colinéarités entre les variables. En effet, nous verrons plus tard que la matrice X doit être inversible (et donc de rang plein) pour que l'on puisse effectuer notre régression linéaire [4]. Un jeu de données présente des colinéarités lorsque certaines variables sont redondantes ou peuvent être obtenues par combinaison linéaire d'autres variables présentes dans la base. En plus d'être inutiles, ces variables faussent la sélection de modèle et doivent donc être supprimées.

Pour ce faire, nous avons adopté la même méthode pour chacune des variables. Tout d'abord, l'utilisation de la fonction *regsubsets* du package **leaps** (fonction que nous allons utiliser pour sélectionner notre modèle de régression linéaire) nous renvoie un message d'erreur pour chaque variable considérée comme une singularité car présentant une trop forte colinéarité avec d'autre. Pour chacune de ces variables, nous avons crée un vecteur composé des variables en relation linéaire avec elles (ou qui expliquaient le même phénomène qu'elles) et nous avons étudié leurs corrélations avec la variable cible. Nous avons décidé, pour chacun de ces vecteurs, de supprimer la variable la moins corrélée à notre variable cible. Nous avons répété cette méthode jusqu'à ce que la fonction regsubsets fonctionne, ce qui signifiait que notre base de données ne comportait plus de telles singularités. A la fin de ce travail de gestion des singularités, notre base de données ne comportaient plus que 169 variables.

On constate par ailleurs la présence d'une observation aberrante dans notre base non contrôlée qui menait, lorsqu'elle était considérée comme fraudeuse, à une estimation de montant de redressement au moins 10 fois plus élevée que n'importe quelle autre observation. Nous avons décidé de la supprimer.

2 Propensity Score Matching et régression linéaire : une première estimation du manque à gagner

La résolution de ce problème s'articule donc en deux étapes : détecter chez un individu non contrôlé si il est potentiellement fraudeur ou non, et si c'est le cas, le montant de redressement qui lui serait imposé. Il s'agit donc de la combinaison entre un problème de classification (0 ou 1) et d'un problème de régression (valeurs continues). Dans un premier temps, nous approchons ce problème avec deux méthodes bien distinctes : une technique d'appariement de score de propension (Propensity score matching ou PSM) pour le problème de classification, qui sera combinée à une régression linéaire.

Le but de notre mémoire sera de comparer deux approches différentes pour calculer le manque à gagner : ici on considère une approche paramétrique par l'appariement des scores de propension (PSM) pour le problème de classification, et une régression linéaire pour évaluer les montants potentiels de fraude.

2.1 Détection des entreprises coupables de fraude par appariement

2.1.1 L'appariement par score de propension : contrôle du biais de sélection

L'appariement de score de propension, ou propensity score matching (PSM), est une technique d'apprentissage statistique très populaire dans le cadre du contrôle de biais, et de ce qu'on appelle l'évaluation de traitement. On désigne par traitement l'appartenance à un groupe d'individus, soit possédant une caractéristique différente (généralement représentée par une variable binaire), soit représentant une sélection. Par groupe de contrôle, on désigne les individus qui, à l'inverse, ne possèdent pas cette caractéristique et servent en quelque sorte de témoin. Lorsque cette distinction est faite, on cherche souvent à mesurer l'effet de l'appartenance au groupe des traités sur les covariables, et ce en particulier sur une variable dite d'intérêt, sur laquelle on se concentre en particulier. On parle ainsi d'évaluation de traitement en désignant l'estimation de l'effet moyen du traitement sur la variable d'intérêt, par une comparaison de cette dernière entre des individus semblables de chaque groupe.
Il existe deux cas de figures dans lesquels ce problème est posé :

- Dans le cadre d'une expérience contrôlée où l'assignation au groupe des traités ou des contrôlés est aléatoire, on peut chercher à mesurer l'effet du traitement sur la variable d'intérêt. On peut par exemple mesurer l'efficacité d'un traitement pour une certaine maladie.
- Dans le cadre de données observationnelles, l'assignation au groupe des traités n'est pas nécessairement aléatoire et peut comporter ce qu'on appelle un biais : le groupe des traités est ce qu'on appelle un *échantillon biaisé*, c'est-à-dire qu'il n'est pas représentatif de la population, la sélection ayant introduit un décalage. Dans ce cas, on cherche alors à l'évaluer, ou tout simplement à recréer un échantillon équilibré et non biaisé.

Dans ces deux cas de figure, le propensity score matching est un outil privilégié pour comparer des individus semblables de chaque groupe en les appariant. Cet appariement permet donc de démontrer l'existence ou l'absence, soit d'une causalité, soit d'un biais lors de l'assignation au groupe des traités. [9]
Cet appariement peut être utilisé pour sa capacité à contrebalancer le biais de sélection mais on peut lui trouver d'autres usages : c'est ce à quoi nous allons nous employer, en l'utilisant dans un but de classification. On se doute que les contrôleurs ciblent des entreprises suspectes ou ayant un haut potentiel de fraude. Il est donc très probable que le groupe des entreprises contrôlées soit un échantillon biaisé, et il est par conséquent intéressant de considérer cet appariement dans notre estimation du manque à gagner.

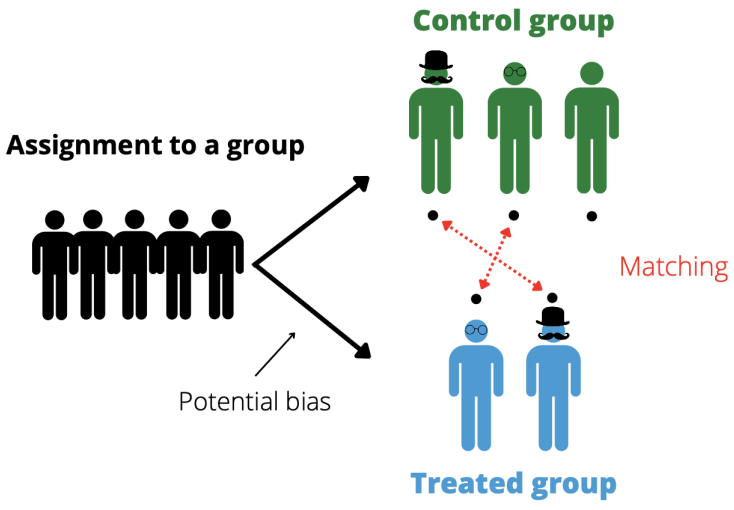


FIGURE 1 – Schéma de l'appariement entre groupes de contrôle et de traitement

Dans notre cas, le groupe de traitement représente les entreprises qui n'ont pas été contrôlées par l'organisme de collecte et dont on ignore si elles sont fraudeuses ou non. Ce sont ces individus que nous voudrons apparier à des individus déjà contrôlés. La base de contrôle représente donc les entreprises qui ont déjà été contrôlées par l'organisme et dont on sait si elles fraudent ou non.
À chaque individu du groupe des contrôlés, nous voulons donc associer un individu du groupe de contrôle qui en soit le plus "proche" possible. Reste à définir la notion de distance entre les individus.

2.1.2 Définition mathématique du score de propension et de l'appariement

L'appariement par score de propension consiste donc à associer chaque individu du groupe de traitement à l'individu du groupe de contrôle le plus proche. Pour se faire, il convient tout d'abord de définir la notion de distance entre les individus : la technique de PSM consiste donc à se baser sur le score de propension comme mesure de distance. Le score de propension représente la probabilité d'appartenance au groupe de traitement. Il est issu d'une régression logistique sur la variable binaire représentant l'appartenance au groupe de traitement (1) ou au groupe de contrôle (0).
Soit B la base des individus, comportant n observations et m variables, B_{Tr} la base de traitement et B_{ctrl} la base de contrôle. On a donc :

$$B_{Tr} = \{x_i \in B,\ x_i \text{ n'a pas été contrôlé}, i \in (1, ..., n)\}$$

$$B_{ctrl} = \{x_i \in B,\ x_i \text{ a déjà été contrôlé}, i \in (1, ..., n)\}$$

On notera n_{Tr} et n_{ctrl} le nombre d'individus des bases de traitement et de contrôle respectivement. Notons C la variable binaire représentant l'appartenance au groupe de traitement. Il s'agit donc de la variable cible lors du calcul du score de propension.
Soit x_i un individu,

$$\begin{cases} C(x_i) = 1 \text{ si } x_i \in B_{Tr} \\ C(x_i) = 0 \text{ si } x_i \in B_{ctrl} \end{cases}$$

Le calcul du score de propension :
On peut calculer le score de propension de l'individu x_i sous certaines hypothèses. [8]

(1) L'hypothèse d'**indépendance conditionnelle** à des caractéristiques observables : la méthode de sélection des individus contrôlés ne s'appuie pas sur des covariables autres que celles du modèle.

(2) L'hypothèse de **support commun** : tout individu peut être à la fois contrôlé ou non contrôlé, les deux groupes se ressemblent assez pour donner sens à la comparaison.

Le **score de propension** d'une observation représente la probabilité que cette observation soit assignée à un certain groupe en fonction d'un ensemble de variables donné :

$$p_{score}(x_i) = \mathbb{P}[C_i = 1 \mid x_i]$$

Ce score de propension peut être calculé selon différentes méthodes qui permettent d'estimer cette probabilité. Ici, nous allons considérer la **régression logistique**.

Cette méthode affirme sous les hypothèses (1) et (2) qu'il existe un certain $\beta^* \in \mathbb{R}^p$ tel que la probabilité $\mathbb{P}[C_i = 1 \mid x_i]$ puisse s'écrire en fonction du produit $x_i\beta^*$. [5]
Il s'agit de la fonction de lien canonique, appelée **lien logit** ou **fonction d'activation sigmoïde**, définie par :

$$g : x \to \frac{1}{1+e^{-x}}$$

On a donc sous les hypothèses (1)&(2) :

$$\mathbb{P}[C_i = 1 \mid X = x_i] = g^{\beta^*}(x_i)$$

C_i suivant une loi de Bernoulli $\mathscr{B}(p_{score}(x_i))$ conditionnellement à x_i, on peut estimer β^* de plusieurs façon, et notamment en **maximisant la vraisemblance** qui s'écrit de la façon suivante :

$$L_n(\beta) = \prod_{i=1}^{n} g^{\beta}(X_i)^{Y_i}(1 - g^{\beta}(x))^{1-Y_i}$$

Et on peut donc écrire l'estimateur $\widehat{\beta^*}$:

$$\widehat{\beta^*} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmax}}(L_n(\beta))$$

Alors, on peut définir le score de propension comme suit :

$$p_{score}(x_i) = g(x_i\beta^*) = \frac{1}{1 + e^{-x_i\beta^*}}$$

On peut donc estimer pour chaque individu x_i son score de propension $p_{score}(x_i)$, en écrivant :

$$\widehat{p_{score}}(x_i) = g(x_i\widehat{\beta^*}) = \frac{1}{1 + e^{-x_i\widehat{\beta^*}}}$$

Enfin, on peut définir la **distance** $d_{PSM} : B_{Tr} \times B_{ctrl} \to \mathbb{R}$ comme la fonction :

$$\forall x \in B_{Tr},\ y \in B_{ctrl},\quad d_{PSM}(x, y) = |p_{score}(x) - p_{score}(y)|$$

La technique d'appariement qui nous intéresse est la technique du *nearest neighbour*. Cette technique minimise, pour chaque appariement, la distance entre les individus. [1]
L'**appariement par score de propension** $\Psi : B_{Tr} \to B_{ctrl}$ est défini comme suit :

$$\forall x \in B_{Tr},\ \Psi(x) = \underset{y \in B_{ctrl}}{\operatorname{argmin}}\ d_{PSM}(x, y)$$

Il s'agit d'une technique avec remise, car plusieurs individus de la base de traitement peuvent avoir le même voisin le plus proche au sein de la base de contrôle. Il est donc possible d'avoir le même appariement pour deux individus différents. Il existe d'autres techniques d'appariement par score de propension, c'est à dire d'autres définitions de distance, cependant nous nous concentrerons sur cette technique qui nous permettra d'utiliser l'appariement à des fins de classification.
En effet, on cherche à apparier les individus de notre base de traitement (comportant les entreprises non contrôlées par l'organisme de collecte), et de notre base de contrôle (les entreprises ayant déjà été contrôlées) dans le but de prédire pour un individu traité s'il est potentiellement fraudeur ou non. En effet, l'appariement nous permet de lui associer une entreprise déjà contrôlée et dont on sait si elle fraude ou non. Le PSM nous permet de définir une entreprise contrôlée comme "la plus semblable" (la moins distante en contrebalançant le biais de contrôle) à une entreprise non contrôlée. Ainsi, on déclarera l'individu en question comme potentiellement fraudeur s'il est associé à une entreprise contrôlée et déclarée fraudeuse.

On a donc l'estimation finale de F^* :

$$\forall x_i \in B_{Tr},\ i \in \{1,...,n_{Tr}\},\quad \widehat{F_i^*} = \begin{cases} 1 & \text{si} \quad F(\Psi(x_i)) = 1 \\ 0 & \text{sinon} \end{cases}$$

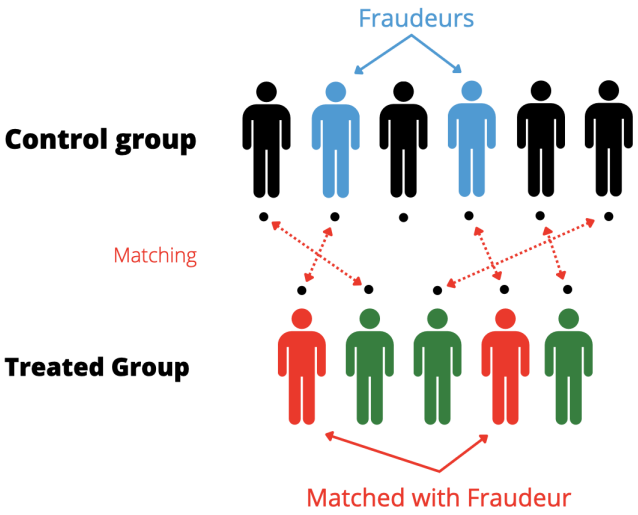


FIGURE 2 – La classification par appariement

2.1.3 Applications et résultats

Nous voulons alors appliquer l'appariement par score de propension à notre jeu de données à des fins de classification et donc de prédiction et il faudra donc évaluer la capacité prédictive du modèle. Or, nous ne pouvons pas avoir recours à la validation croisée, l'information sur la fraude étant inconnue pour la base de traitement. En ce sens, la performance de la classification sera très difficile à évaluer. L'appariement étant fait sur le score de propension, sa précision dépendra donc uniquement de celle de ces scores de propension eux-mêmes et donc de la précision du modèle de régression associé. Il sera donc très important de l'affiner, et d'en évaluer la qualité.
Pour faire cela et ainsi améliorer la précision des scores de propension, nous avons fait appel à un outil de sélection de features : la régression Lasso.

Il s'agit d'un type de régression linéaire qui utilise le système de pénalisation (shrinkage). Cela consiste en la réduction de la valeur des données vers un point central, comme par exemple la moyenne, afin de ramener certains coefficients du modèle à 0. Ce type de régression permet de simplifier le modèle sans perdre en information. Les solutions d'une régression Lasso sont des problèmes de programmation quadratique. On appelle l'estimateur Lasso, noté $\hat{\beta}(L)$, toutes les solutions du problème d'optimisation pénalisée suivant :

$$\min_{\beta \in \mathbb{R}^p} \left\{ \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda \|\beta\|_1 \right\}$$

Avec $\|\beta\|_1 = \sum_{j=1}^{m} |\beta_j|$ et λ un hyperparamètre du modèle. [4]
Après application de cette régression lasso, nous obtenons un modèle final composé de 33 variables.

On cherche désormais à évaluer la qualité du modèle. Pour cela, on s'intéresse d'abord à sa significativité. Il convient alors de s'intéresser à la *déviance* du modèle, afin de la comparer avec celle du modèle réduit à l'intercept, appelé modèle nul et consistant à estimer la variable cible par une constante.

La déviance d'un modèle [M] est définie par rapport au modèle saturé [Msat] comportant l'ensemble des variables de la base, sans sélection. Sa déviance est notée $\mathscr{D}[M]$ et vaut :

$$\mathscr{D}[M] = 2 \times (\mathscr{L}[Msat] - \mathscr{L}[M])$$

Où $\mathscr{L}[Msat]$ et $\mathscr{L}[M]$ sont respectivement les log des vraisemblances du modèle saturé [Msat] et du modèle [M]. [5]

```
    Null deviance: 49427  on 187642  degrees of freedom
Residual deviance: 44844  on 187610  degrees of freedom
AIC: 44910
```

FIGURE 3 – Significativité du modèle : déviance du modèle et déviance du modèle réduit à l'intercept

On constate que la déviance du modèle (*residual deviance*) est bien inférieure à celle du modèle réduit à l'intercept (*null deviance*). On en conclut que le modèle est bien significatif, et on pourra continuer à l'affiner.

Pour aller plus loin, on s'intéresse à la significativité de chaque coefficient. Pour chacun, on regarde la p-valeur du test de de nullité du coefficient j associé, $j = 1, .., p$.
Cet algorithme va tester pour chaque features :

$$H_0 : \beta_j = 0 \qquad \text{VS} \qquad H_1 : \beta_j \neq 0$$

Soit $\widehat{\sigma_j}^2$ l'estimateur non biasié de la variance de la covariable x_j. La statistique de ce test se calcule avec la formule :

$$Z = \sqrt{n}\,\frac{\widehat{\beta_j}}{\widehat{\sigma_j}} \xrightarrow{\mathscr{D}} \mathscr{N}(0,1)$$

Si la p-valeur associé au test est faible, on rejette H_0 et le coefficient est jugé significatif. À l'inverse, si sa p-valeur est élevée, le coefficient n'est pas significatif et on pourra le retirer du modèle. [5]

```
Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)               5.793e+00  2.011e-01  28.799  < 2e-16 ***
ancentr                   3.966e-03  1.380e-03   2.874 0.004052 **
pcadre                    3.177e-01  1.139e-01   2.790 0.005272 **
pbureau                   2.362e-02  1.457e-01   0.162 0.871203
age_ec                   -9.260e-03  3.606e-03  -2.568 0.010225 *
nombre_mensuel_moyen_CDI  6.223e-04  2.552e-03   0.244 0.807386
heu_tec                   1.543e-06  6.153e-07   2.507 0.012167 *
nbviecdi                  3.900e-04  2.506e-03   0.156 0.876318
cotiemp_cotitot          -3.918e-01  6.914e-02  -5.667 1.45e-08 ***
entrep_tec               -1.684e-01  5.575e-02  -3.020 0.002527 **
```

FIGURE 4 – Exemples de tests de nullité des paramètres pour les 10 premier features

Sur la Figure 4, on aperçoit pour les 10 premiers features successivement le coefficient de $\widehat{\beta}$ associé, la *Standardized Error*, la valeur de la statistique du test, notée z, puis la p-valeur associée. On rejette ici l'hypothèse H_0 pour une p-valeur inférieure au niveau $\alpha = 0.05$. Sur l'ensemble de la base, on écarte tous les features qui ne présenteront pas de p-valeur inférieure à ce seuil pour le test de nullité . Cela nous amène finalement à écarter 7 features, jugés non significatifs au sein du modèle.

Enfin, on s'intéresse à la précision du score de propension. Le score de propension est un score d'équilibre : s'il est précis, les distributions des features doivent, conditionnellement à lui, être relativement similaires entre les groupes de traitement et de contrôle. [2] Ainsi, pour évaluer et affiner d'avantage sa qualité, on stratifie le vecteur de scores de propension et pour chacune des strates, on s'intéresse à la répartition des features entre les deux groupes. Un bon équilibre indiquera donc un score de propension précis.
Soit $\widehat{p_{score}} \in \mathbb{R}^m$ le vecteur des estimations des scores de propension. Soit B l'ensemble des individus : on procède à une stratification des scores de propension par quantile. On notera L le nombre de strates, et D_l la $l^{ième}$ strate.
On a alors :

$$\forall l \in \{1,...,L\},\ \ D_l = \left\{x_i,\ \widehat{p_{score}}(x_i) \in [q_{l-1}, q_l[, i \in \{1,...,n\}\right\}$$
$$\text{avec } q_l = \begin{cases} 0 \text{ si } l = 0 \\ 1 \text{ si } l = L \\ \text{le quantile d'ordre } \frac{l}{L} \text{ du vecteur } \widehat{p_{score}} \end{cases}$$

Afin d'évaluer l'équilibre des groupes de traitement et de contrôle, on s'intéresse à la *Strictly standardized mean difference* (SSMD) entre eux, pour chaque variable et ce au sein de chaque strate. On la définit :
Soit $l \in (1,...,L)$ et D_l la $l^{ième}$ strate. Soient $\mu_{Tr}(j)$, $\mu_{ctrl}(j)$, $\sigma^2_{Tr}(j)$ et $\sigma^2_{ctrl}(j)$ respectivement les moyennes empiriques et variance empiriques des bases de traitement et de contrôle de la covariable j.
Alors, pour un certain vecteur $\beta \in \mathbb{R}^p$ issu d'un modèle à $p-1=m$ features, on note $SSMD_l = SSMD_l(\beta)$ le vecteur de $\mathbb{R}^n$ évaluant la *Strictly standardized mean difference* pour ce modèle.
Soit β le vecteur de coefficients issu d'un modèle et $l \in \{1,...,L\}$. $SSMD_l$ s'écrit :
$\forall j \in \{1,...,m\}$,

$$SSMD^j_l = SSMD^j_l(\beta) = \frac{\mu_{Tr}(j) - \mu_{ctrl}(j)}{\sqrt{\sigma^2_{Tr}(j) + \sigma^2_{ctrl}(j)}}$$

On a donc l vecteurs de $\mathbb{R}^n$, que l'on peut pondérer par le nombre d'individus dans chaque Strate afin d'obtenir un vecteur de $\mathbb{R}^n$ qu'on notera $SSMD$:
$\forall j \in \{1,...,m\}$,

$$SSMD^j = \sum_{l=1}^{L} \frac{1}{m_l} SSMD^j_l$$
$$\text{avec } m_l = \sum_{i=1}^{m} \mathbb{1}_{\{x_i \in D_l\}}$$

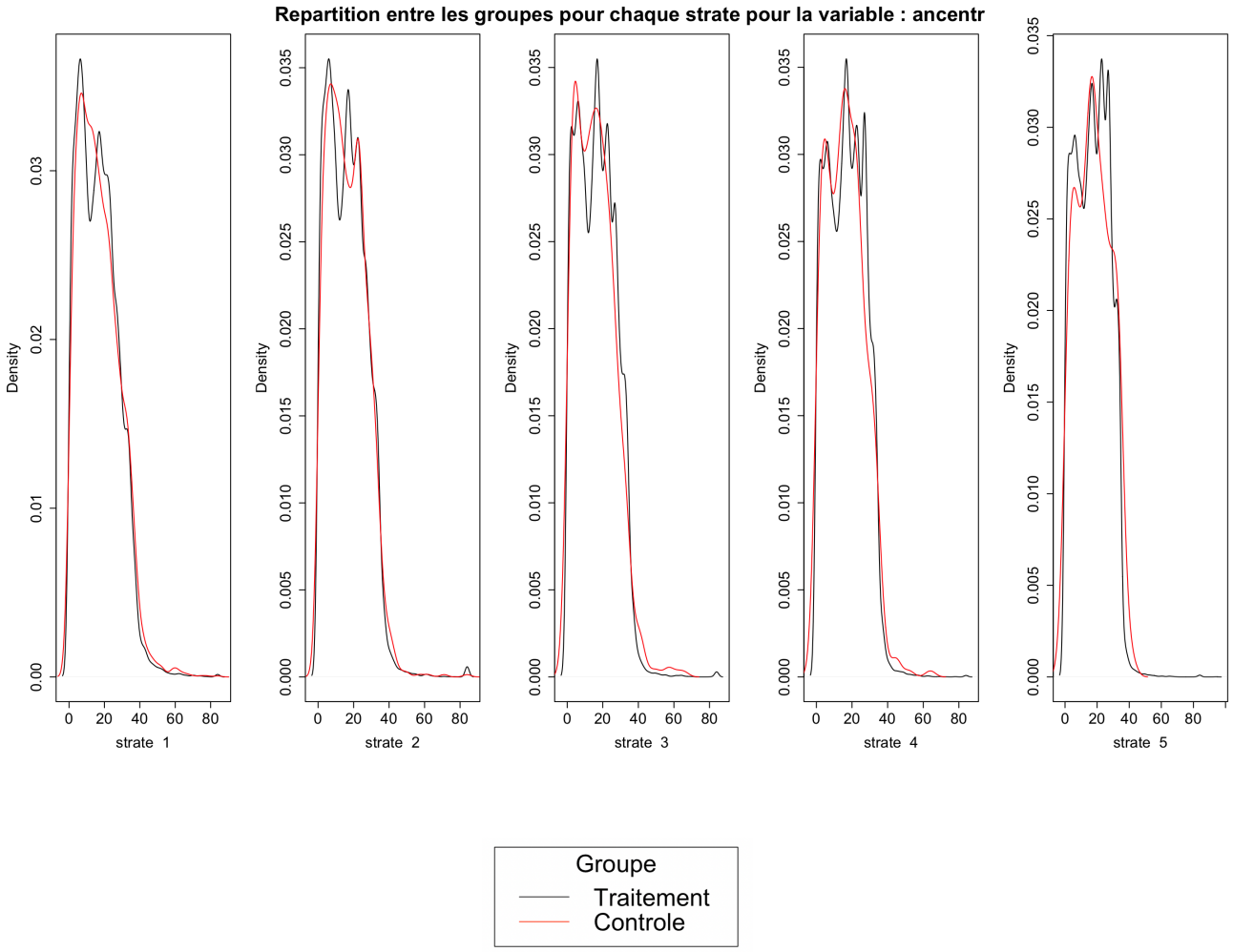
On cherche la combinaison de features qui minimisera la norme ℓ^2 de ce vecteur, c'est à dire :

$$\beta^* = \underset{\beta \in \Theta}{\text{argmin}} \|SSMD\|_2 = \underset{\beta \in \Theta}{\text{argmin}} \sqrt{\sum_{j=1}^{n} (SSMD^j)^2}$$

Nous procèderont à une stratification avec $L = 5$ strates, car cela permet d'éliminer plus de 90% du biais sans que les calculs en deviennent trop coûteux pour autant. [9] Après les sélection par lasso et par significativité on remarque que $\|SSMD\|_2 = 0.956$. Nous remarquons également quatre features relativement peu significatifs au sein du modèle présentant une répartition irrégulière entre les classes et une différence de moyenne standardisée très prononcée quelque soit la strate : nous ne retenons pas ces features dans le modèle final.

En les supprimant, nous obtenons le vecteur final $\widehat{\beta}$ pour lequel : $\|SSMD\|_2 = 0.504$: pour autant, on gagne très peu en déviance. Il s'agit du meilleur compromis entre significativité et équilibre des répartitions.
Le modèle retenu comporte donc 22 features. Les faibles valeurs de la norme de $SSMD$ témoignent d'un score de propension d'une bonne qualité : on conservera ce modèle qui est à la fois simple et pertinent.

FIGURE 5 – Exemple de répartition équilibrée par strate d'un feature entre les groupes de Traitement et de Contrôle



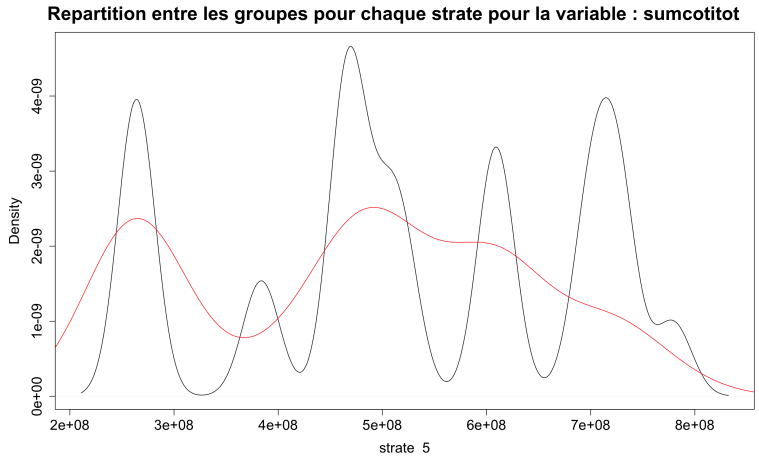


FIGURE 6 – Exemple de répartition non équilibrée pour une strate entre les groupes de Traitement et de Contrôle

On applique ainsi le matching combiné aux scores de propension calculés ci-dessus, par la technique du *nearest neighbour*.

```
#Matching
start_time<-Sys.time()
response<- matchit( invert_ctrlex_2014 ~., data = base_test, method = "nearest",
                    distance = "logit", replace = TRUE )
summary(response)$nn
print(c("Execution time :",Sys.time()-start_time))
```

On va donc appliquer l'appariement défini plus haut à ce modèle sur le jeu de données. Pour ce faire on utilisera le package **Matchit**, très répandu pour l'implémentation de diverses techniques d'appariement sur R. On précise l'argument $replace$ = TRUE car on souhaite un appariement avec remise.
On utilisera ensuite ces informations pour trouver la liste des individus du groupe de Traitement appariés avec des entreprises contrôlées et fraudeuses : on appellera cette liste *idFraud*.
On appellera n_{F^*} le nombre d'individus identifiés comme potentiellement fraudeurs :

$$\forall i \in \{1,...,n\}, \quad n_{F^*} = \sum_{i=1}^{n} \mathbb{1}_{F_i^*=1}$$

```
> #Soit idFraud la liste des individus non contrôlés et potentiellement fraudeurs
> #On notera fraud la variable binaire qui indiquera si un individu fait partie de ce groupe
> base_finale$fraud<-ifelse(base_finale$id %in% idFraud, 1, 0)
>
> # On peut alors calculer le nombre de fraudeurs potentiels
> nb_potential_fraud_2014<-sum(base_finale$fraud)
> nb_potential_fraud_2014
[1] 53174
```

FIGURE 7 – Calcul du nombre de fraudeurs potentiels

Cet appariement nous indique finalement un total de 53,173 entreprises potentiellement fraudeuses dans notre base non contrôlée. La proportion de telles entreprises dans la base totale est donc similaire à celle de la base contrôlée. On s'aperçoit donc qu'il ne semble pas y avoir de biais de contrôle sur la fraude des entreprises.

```
> part_potential_fraud
[1] 0.291881
> part_efffective_fraud
[1] 0.294914
```

FIGURE 8 – Part de la fraude potentielle contre la fraude effective

2.2 Estimation du montant de redressement des fraudeurs : régression linéaire

Une fois ce travail de classification effectué, il faut désormais attribuer un montant de redressement aux entreprises qui ont été identifiées comme fraudeuses. Nous allons, pour cela, effectuer une régression linéaire sur la base qui comporte nos prédictions de fraude.

L'une des principales difficultés que nous allons rencontrer repose sur le fait que nous étudions des données observationnelles et en faible quantité (seulement 1614 observations d'entreprises fraudeuses). Ces données ne sont donc pas forcément distribuées normalement et le fait de les passer au logarithme ne corrige pas totalement ce problème. Il sera donc compliqué d'obtenir un modèle avec des résidus avec une distribution parfaitement normale.

2.2.1 La régression linéaire

Le principe de la régression linéaire est de modéliser la relation entre une variable cible et des variables explicatives comme une relation linéaire, c'est à dire de définir un vecteur β tel que :

$Y = X\beta + \epsilon$

avec (en notant n le nombre d'observations et m le nombre de variables qui composent notre modèle) :

$$\begin{cases} Y \in \mathbb{R}^n \text{ la variable cible,} \\ X \in \mathbb{R}^{n\times(m+1)} \text{ les valeurs des observations pour les p variables choisies dans notre modèle,} \\ \beta \in \mathbb{R}^{m+1} \text{les coefficients à optimiser pour la régression linéaire,} \\ \epsilon \in \mathbb{R}^n \text{ les résidus de la régression} \end{cases}$$

Lorsque l'on effectue une régression linéaire, la première étape est d'établir un modèle. Pour cela, nous devons trouver les variables de notre jeu de donnée qui seront les plus significatives dans l'estimation de notre variable cible. Il est en effet impossible d'effectuer une régression linéaire avec les 180 variables qui constituent notre base de données. Le but est de trouver un modèle qui soit à la fois performant et peu coûteux (donc avec un minimum de variables explicatives). On entraînera ce modèle sur les observations contrôlées et fraudeuses ($C_i = 1, F_i = 1$).

Une fois ce modèle établi, il est important de vérifier que ce dernier vérifie les hypothèses d'un modèle linéaire. On effectue pour cela une analyse des résidus qui doivent satisfaire 4 postulats [5] :

P1 : Les résidus sont centrés : $\mathbb{E}[\epsilon_i] = 0$.
P2 : Les résidus ne sont pas corrélés : $cov_{\epsilon_i,\epsilon_j} = 0 \ \forall i \neq j$.
P3 : Les résidus ont une variance constante : $\mathbb{V}[\epsilon_i] = \sigma^2$.
P4 : Les résidus sont Gaussiens.

Ces 4 postulats peuvent être résumés par la relation :

$$\epsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0;\sigma^2)$$

Si les résidus vérifient bien ces 4 postulats, le modèle de régression linéaire est alors validé. Il est donc possible d'effectuer nos prédictions sur la base de donnée totale grâce aux coefficients qui sont attribués à chaque variable de notre modèle lors de sa création. Ces coefficents sont établis en approchant le vecteur β à l'aide de l'estimateur des moindres carrés ordinaires[4] :

$$\widehat{\beta} = \hat{\beta}^{(MC)} = \underset{\alpha\in\mathbb{R}^m}{\operatorname{argmin}} \|Y - X\alpha\| = \underset{\alpha\in\mathbb{R}^m}{\operatorname{argmin}} \sum_{i=1}^{n} (Y_i - X_i\alpha)^2$$

Sous les postulats [P1] et [P3] ainsi que l'hypothèse de rang plein, cet estimateur se calcule selon la formule :

$$\hat{\beta}^{(MC)} = (X^T X)^{-1} X^T Y$$

L'hypothèse de rang plein est bien nécessaire pour le calcul de la régression linéaire, comme précisé en (1.2).

2.2.2 Choix du modèle

Nous avons donc commencé notre travail de régression linéaire par le choix du modèle que nous allions utiliser. Après avoir séparé aléatoirement notre base d'entreprises fraudeuses en deux parties (Un ensemble d'entraînement comportant 80% de la base et un ensemble de test issu des 20% restants), nous nous sommes penchés sur la variable cible que nous allions choisir pour notre modèle.

Ce travail sur la variable cible est important car la qualité de notre modèle en dépend grandement. Il est en effet préférable de choisir une variable qui possède une densité de loi normale pour que les résidus de notre modèle respectent les postulats énoncés précédemment. Dans notre cas, la variable que nous voulions étudier était bien entendu le montant de redressement imposé aux entreprises. Cependant, cette variable n'avait pas la densité d'une loi normale. Les solutions pour en extraire une densité normale sont nombreuses, on peut par exemple étudier son inverse, sa racine ou son logarithme. Ici, le logarithme de cette variable suivait approximativement bien une loi normale et nous avons donc décidé de le choisir comme variable cible.

Cette variable cible choisie, il nous a fallu trouver les variables explicatives et dans un premier temps, décider du nombre de variables qui composeraient notre modèle. Pour ce faire, nous avons utilisé la fonction *regsubsets* du package **leaps** afin de sélectionner les modèles les plus performants pour un nombre de variables donné. Cet algorithme peut utiliser trois méthodes de sélections de variables pas à pas. Ici, nous avons décidé d'utiliser la méthode forward : L'algorithme part du modèle réduit à l'intercept et le compare à tout les modèles à une variable et sélectionne le meilleur selon une mesure de performance donnée. Il considère alors l'ajout d'une nouvelle variable et fonctionne ainsi jusqu'à ce qu'il arrive au nombre de variables donné. Nous avons décidé d'utiliser cette méthode car elle permet de sélectionner le nombre optimal de variables et s'avère plus économique dans notre cas au vu du nombre conséquent de variables.

Pour choisir le nombre de variables qui allaient composer notre modèle, nous avons donc utilisé cette fonction *regsubsets* avec différentes limites de nombre de variables. En étudiant les critères BIC et AIC ainsi que le R-squared de chaque modèle, nous avons conclu que les modèles les plus efficaces comportaient 8 et 9 variables.

Le R-squared d'un modèle est son coefficient de détermination, il représente la proportion de la variance de la variable cible qui est expliquée par les variables explicatives présentes dans le modèle. Plus le R-squared du modèle s'approche de 1, plus le modèle est efficace. Ce coefficient s'obtient par un calcul sur la somme des carrés totale (SCT) et la somme des carrés résiduels (SCR) définis, sous les hypothèses [P1] et [P3] ainsi que l'hypothèse de rang plein, comme suit :

$$\begin{cases} SCT = \|y - \bar{y}\mathbb{1}_n\| = \sum_{i=1}^{N} (y_i - \bar{y})^2 \\ SCR = \|y - \hat{y}\| = \sum_{i=1}^{N} (y_i - \hat{y})^2 \end{cases}$$

Le R-squared s'obtient alors par la formule :

$$R^2 = 1 - \frac{SCR}{SCT}$$

Les critères AIC (Akaike Information Criterion) et BIC (Bayesian Information Criterion) permettent également d'évaluer l'efficacité d'un modèle mais en y ajoutant le principe de parcimonie [6]. Ils permettent de calculer la vraisemblance du modèle tout en la pénalisant par le nombre de variables qu'il comporte. Plus le critère est faible, plus le modèle est précis. On remarque que ces critères augmentent en fonction du nombre de variables : ces deux critères marquent donc un compromis entre le biais d'un modèle et sa parcimonie. Leur différence repose sur le fait que le BIC pénalise d'avantage le nombre de variable que l'AIC.

Soient m le nombre de features présents dans notre modèle et $\tilde{L}$ le maximum de vraisemblance du modèle. Les critères AIC et BIC sont calculés selon les formules :

$$\begin{cases} AIC = 2m - 2log(\tilde{L}) \\ BIC = mlog(n) - 2log(\tilde{L}) \end{cases}$$

Nous avons travaillé dans un premier temps sur un modèle à 8 variables mais avons décidé d'en rajouter une après avoir réalisé que le postulat d'absence d'autocorrélation n'était pas satisfait dans un tel modèle. Cependant, les modèles générés par notre fonction regsubsets différaient selon les seed que nous prenions en compte (car la base d'entraînement différait) ; c'est à dire le nombre utilisé pour générer une séquence de valeurs aléatoires. C'est ce qu'on appelle l'Overfitting : une modélisation trop adaptée aux spécificités de la base d'individus sur laquelle il s'entraîne, et qui aura donc du mal à prédire de manière fiable de nouvelles observations. Pour rendre notre modèle mois sensible à l'overfitting, nous avons décidé de croiser 15 différents modèles à 9 variables afin de constituer notre modèle final avec les variables qui apparaissaient dans le plus de modèles. Cette pratique a effectivement rendu notre modèle moins sensible à l'overfitting, et donc plus plus robuste à l'aléatoire et plus performant.

En outre, cette sélection de modèle avec la fonction regsubsets ne prend pas en compte un critère important de la régression linéaire : le modèle ne doit pas présenter de multicolinéarité entre les variables. [3] La multicolinéarité entre les variables apparaît lorsque certaines sont trop corrélées ou qu'il existe une relation de causalité entre elles : une variable peut notamment être expliquée par la combinaison linéaire d'autres covariables. Ce phénomène est problématique car il peut rendre les coefficients de prédictions instables (les coefficients peuvent varier grandement en fonction des seed ou paraître non significatifs pour des variables importantes dans le modèle).

Pour mesurer cette colinéarité, nous utilisons l'indicateur VIF, que nous calculeront à l'aide de la fonction *VIF* du package **car**. Les VIF (Variance Inflation Factor) d'un modèle indiquent l'augmentation de la variance des coefficients dûe à la relation linéaire de la variable avec d'autres prédicteurs. Ils se calculent par la formule suivante :

$$VIF_j = \frac{1}{1 - R_j^2}$$

où R_j^2 est le coefficient de détermination obtenu en effectuant une régression sur la $j^{ième}$ variable en fonction des autres variables du modèle.

On estime que la multicolinéarité risque de poser problème concernant une variable lorsqu'elle possède un vif supérieur à 2. Pour chaque modèle fourni par la fonction regsubsets, nous devions donc étudier les variables pour lesquelles c'était le cas. Nous générions un nouveau modèle après avoir supprimé l'une des variables qui augmentaient les scores VIF jusqu'à ce que ceux-ci soient satisfaisants pour chaque variable du modèle.

Cette sélection croisée de modèles nous a finalement menée à choisir le modèle suivant, dont le R-squared ne varie que très peu en fonction des seed générés et dont les VIF sont très proches de 1 :

```
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       6.6486634  0.2880186  23.084  < 2e-16 ***
mensu             0.4158290  0.1016215   4.092 4.49e-05 ***
sal_hom           0.0069636  0.0008448   8.243 3.45e-16 ***
cotiemp_cotitot   1.1612662  0.1892951   6.135 1.07e-09 ***
segment3         -0.9237495  0.2003839  -4.610 4.35e-06 ***
entrep_controleur 0.0007639  0.0001871   4.084 4.65e-05 ***
num_ma_reg_5      0.6188532  0.1176933   5.258 1.65e-07 ***
num_ma_reg_8     -0.6173082  0.1069989  -5.769 9.54e-09 ***
embcdd_nbtot     -0.3865573  0.1228313  -3.147  0.00168 **
num_ma_reg_2      0.4613283  0.1202944   3.835  0.00013 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.614 on 1598 degrees of freedom
Multiple R-squared:  0.2301,    Adjusted R-squared:  0.2257
F-statistic: 53.06 on 9 and 1598 DF,  p-value: < 2.2e-16
```

FIGURE 9 – Modèle de régression linéaire

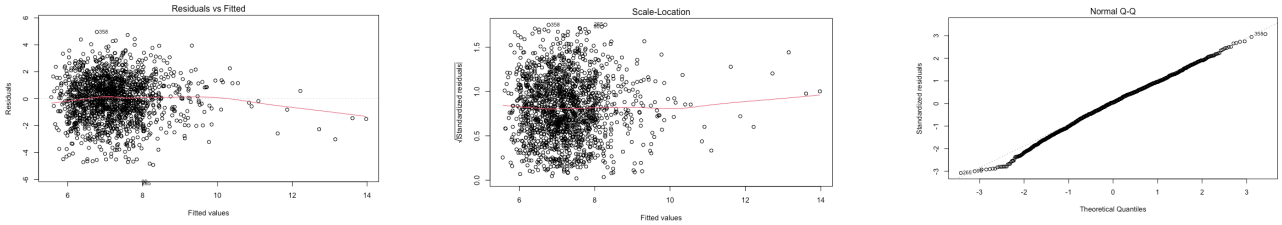
```
> vif(modele1)
           mensu          sal_hom   cotiemp_cotitot         segment3 entrep_controleur
        1.272461         1.247142         1.215575         1.254822         1.067646
     num_ma_reg_5     num_ma_reg_8      embcdd_nbtot     num_ma_reg_2
        1.162666         1.178421         1.191641         1.165265
```

FIGURE 10 – Scores VIF du modèle final

2.2.3 Diagnostic et interprétations

Le travail de modélisation ne s'arrête bien entendu pas là. Après avoir choisi notre modèle, le plus important est d'évaluer son efficacité et de s'assurer qu'il satisfait les postulats d'un modèle linéaire.

La première étape du travail de diagnostic est de vérifier que notre modèle satisfaisait les 4 postulats cités en amont. Pour se faire, nous avons combiné une analyse graphique de notre modèle avec des tests statistiques nécessaires pour certains postulats.



(a) Espérance des résidus **(b)** Variance des résidus **(c)** Normalité des résidus

FIGURE 11 – Analyse graphique des résidus

[P1] : Résidus centrés (linéarité du modèle) : On peut voir sur la figure 11.(a) que les résidus sont globalement symétriques en 0 (ligne rouge) malgré un biais pour les valeurs élevées qui est dû au faible nombre de telles observations dans notre base d'entreprises fraudeuses. Nous pouvons donc valider [P1]

[P2] : Les résidus ont une variance homoscédatique : De même, on s'aperçoit sur la figure 11.(b) que la ligne rouge est approximativement droite avec un léger biais pour les valeurs élevées pour les raisons cités précédemment. Nous pouvons également valider [P2]

[P3] : Non corrélation des résidus : Il n'existe pas d'analyse graphique pour ce postulat qui nécessite la réalisation d'un test de Durbin-Watson. Lorsque nous avions effectué ce test sur notre modèle à 8 variables, la p-valeur était très faible et indiquait la corrélation des résidus. L'une des solutions préconisée étant d'ajouter des variables [3], nous avons décidé d'étudier un modèle à 9 variables (qui augmente donc légèrement le R-squared au détriment de l'AIC) pour lequel le test de Durbin-Watson nous donne :

```
> durbinWatsonTest(modele1)
 lag Autocorrelation D-W Statistic p-value
   1      0.03275862       1.93399   0.142
 Alternative hypothesis: rho != 0
```

L'hypothèse H0 du test de Durbin-Watson est la non corrélation des résidus. Notre test nous renvoie une p-valeur de 0.142 > 0.05. On ne peut pas rejeter H0 et [P3] est donc validée.

[P4] : Normalité des résidus : La première étude de ce postulat nous avait mené à modifier légèrement notre modèle dans la mesure où quelques observations semblaient influer sur sa normalité. Une simple étude graphique nous avait donc poussé à supprimer ces observations de la base d'entraînement afin d'améliorer la normalité. Le graphique de normalité est désormais celui de la Figure 11 (c). Les points sont plutôt bien alignés le long de la première bissectrice et nous pouvons donc valider [P4].

Notre modèle satisfait donc les 4 postulats d'un modèle linéaire, ce modèle est donc bien un modèle linéaire que nous pouvons exploiter pour prédire les montants de redressements sur la base de fraudeurs potentiels donnée par le PSM.

Pour ce qui est de l'efficacité de notre modèle, les critères d'évaluations ont été optimisés le plus possible lors de notre choix. Comme énoncé précédemment, notre sélection de modèle s'est notamment basée sur les critères d'AIC et de BIC dont les valeurs finales sont respectivement de 6114 et 6173. Ces valeurs sont vraiment satisfaisantes au regard des autres modèles que nous avons pu essayer. Mais ces valeur d'AIC et BIC ne sont pas les plus efficaces pour évaluer la performance de notre modèle dans la mesure où il ne s'agit pas de valeur absolues. Ces critères peuvent être utilisés pour comparer des modèles entre eux mais ne peuvent pas nous donner d'indice clair sur la performance du modèle.

En ce sens, d'autres critères peuvent sembler plus intéressants à prendre en compte dans l'évaluation de notre modèle. Tout d'abord, le R-squared ajusté de 0.2257 peut sembler faible mais il s'agit du meilleur R-squared que nous puissions trouver au vu des observations dont nous disposons et sans pénaliser les critères d'AIC et BIC notament. En outre le RMSE (Root mean square error) de notre modèle est de 1.608, ce qui est très satisfaisant pour un modèle comme le notre dont les observations de log-montant varient entre 0 et 14. Le RMSE représente l'erreur quadratique moyenne et donne donc une indication de la concentration des données autour de la droite de régression. Le RMSE nous indique donc à quel point notre modèle représente bien les observations et évalue sa capacité prédictive. Soit $\widehat{\beta} \in \mathbb{R}$ le vecteur de coefficients obtenus dans notre modèle, on le calcule selon la formule :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i\widehat{\beta})^2}$$

```
Residual standard error: 1.614 on 1598 degrees of freedom
Multiple R-squared:  0.2301,    Adjusted R-squared:  0.2257
F-statistic: 53.06 on 9 and 1598 DF,  p-value: < 2.2e-16

> RMSE<-sqrt(mean(modele1$residuals^2))
> RMSE
[1] 1.608906
```

FIGURE 12 – Mesure d'ajustement aux données (R^2, adjusted-R^2) et RMSE du modèle

2.3 Combinaison des deux techniques : estimation du manque à gagner

Une fois ce modèle choisi, il ne nous reste plus qu'à effectuer notre prédiction sur la base des entreprises potentiellement fraudeuses obtenue grâce au propensity score matching. Cette prédiction de redressement s'effectue donc de la sorte :
Soit F^* la variable indiquant pour un individu du groupe de traitement s'il est potentiellement fraudeur où non. On rappelle que :

$$\forall x_i \in B_{Tr},\ i \in \{1,...,n_{Tr}\},\quad \widehat{F_i^*} = \begin{cases} 1 & \text{si} \quad F(\Psi(x_i)) = 1 \\ 0 & \text{sinon} \end{cases}$$

On arrive donc à la formule suivante pour l'**estimation du manque à gagner** :
Soit $\widehat{\beta} \in \mathbb{R}^p$ le vecteur de coefficients issu de notre modèle linéaire.

$$\forall x_i \in B_{Tr},\ i \in \{1,...,n_{Tr}\},\quad \widehat{M_i^*} = x_i\widehat{\beta}$$
$$\Longrightarrow \widehat{MAG_i} = \begin{cases} x_i\widehat{\beta} & \text{si} \quad F(\Psi(x_i)) = 1 \\ 0 & \text{sinon} \end{cases}$$

On calcule alors :

$$\widehat{MAG} = \sum_{i=1}^{n} \widehat{MAG_i}$$

```
> # Montant de redressement effectif :
> redress_2014 <- sum(subset(base_mag_clean,!is.na(base_mag_clean$Mt_redress_tot))$Mt_redress_tot)
> redress_2014
[1] 11525958
>
> # Ajout de la prédiction conditionnelle
> base_finale$predictionFraud <- ifelse(base_finale$fraud==1, exp(predict.lm(modele1, base_finale)),0)
>
> # Estimation du manque a gagner :
> mag_2014<-sum(base_finale$predictionFraud)
> mag_2014
[1] 86258571
```

FIGURE 13 – Estimation du manque à gagner par la combinaison de PSM et régression linéaire

On obtient finalement une première estimation du manque à gagner de 86 millions d'euros. On obtient donc un montant de redressement seulement 8 fois plus élevé que celui de la base contrôlée pour 33 fois plus d'observations. Pourtant la proportion de fraudeurs entre les deux groupes était la même. Nous en déduisons donc que le biais de contrôle semble porter sur le montant de redressement plutôt que sur la probabilité de fraude d'une entreprise. L'organisme contrôlerait alors en priorité les entreprises plus importantes et donc susceptibles de se voir attribuer de grands montants de redressements, plutôt que de contrôler les plus susceptibles d'être fraudeuses dans l'absolu.

2.4 Alternatives

Après avoir fourni cette première estimation du manque à gagner, il nous semble pertinent d'introduire de nouvelles méthodes d'appariement afin d'obtenir un point de comparaison sur les valeurs obtenues.

2.4.1 La distance de Mahalanobis

L'une des principales alternatives au matching par score de propension est le matching par distance de Mahalanobis (MDM), qui représente l'une des méthodes les plus communes d'appariement. Le MDM fonctionne en associant des individus sur la base d'une distance appelée distance de Mahalanobis. A la différence de la distance euclidienne, elle est basée sur la variance et la corrélation des covariables dont elle calcule l'éloignement. Elle pénalise ainsi les variables les plus volatiles. Soient x_i et x_j deux individus d'un échantillon, la distance de Mahalanobis entre x_i et x_j s'écrit :

$$d(x_i, x_l)_{Mahalanobis} = \sqrt{(x_i - x_l)^T \Sigma^{-1} (x_i - x_l)}$$

Où Σ est la matrice de covariance empirique entre x_i et x_j.

Nous avons donc effectué dans un second temps un appariement entre les individus non contrôlés et les individus contrôlés grâce à cette nouvelle technique. Ainsi, chaque entreprise non contrôlées s'est vue appariée à l'entreprise contrôlée la plus proche selon la distance de Mahalanobis. Nous avons à nouveau utilisé la fonction *matchit*, mais cette fois ci avec l'option *distance="Mahalanobis"*.
Après application de ce type de matching, nous obtenons un nombre de potentiels fraudeurs de 52 035. Nous pouvons constater que ce résultat est très similaire à l'estimation par appariement de score de propension, et nous demander si les deux appariements ne sont pas relativement similaires. Pourtant, nous obtenons une estimation du manque à gagner d'environs 128 Millions, soit $\simeq 30\%$ de plus que l'estimation par PSM, alors qu'on applique les mêmes prédictions linéaires. Cela signifie donc que les individus de la base de Traitement qui ont été déclarés fraudeurs potentiels ne sont pas les mêmes, et qu'ils sont en l'occurrence susceptibles d'être associés à de plus gros montants de redressement : cela semble confirmer l'existence d'un biais dans le contrôle des entreprises, contrebalancé par l'appariement par score de propension. Il nous est difficile d'interpréter ce résultat plus en profondeur, mais nous nous attendions à trouver un montant de cet ordre, et cela nous donne une première confirmation de l'ordre de grandeur de nos estimations, qui est le bon.

(a) Calcul du nombre de fraudeurs potentiels

```
> #Nombre de fraudeurs potentiels par appariement de Mahalanobis
> nb_fraud_2014_mahalanobis<-sum(Resume$fraud)
> nb_fraud_2014_mahalanobis
[1] 52035
```

(b) Application du de la régression linéaire

```
> #Estimation du manque a gagner par appariement de Mahalanobis :
> mag_2014_mahalanobis<-sum(Resume$predictionFraud)
> mag_2014_mahalanobis
[1] 128642777
```

FIGURE 14 – Estimation du manque à gagner par appariement de Mahalanobis et régression linéaire

2.4.2 Approche par la classification KNN

Une autre approche concernant la classification par score de propension serait de considérer un ratio d'appariement 1 à k : on peut imaginer améliorer la précision de la classification engendrée par le matching en appariant un individu traité non pas à un, mais à k contrôlés. Nous pourrions alors :

- le déclarer fraudeur si la majorité des k plus proches voisins de la base de contrôle sont fraudeurs.
- s'il est classé fraudeur, lui associer la moyenne des fraudes de ses voisins, pondérée par leur distance.

Soit n_{Tr} et n_{ctrl} le nombre d'individus respectivement de la base de Traitement et de Contrôle. Soit k le ratio (nombre de voisins à apparier pour un individu) choisi arbitrairement. On note alors l'appariement :
$\forall i \in \{1,...,n_{Tr}\}$,

$$\Psi_{k-nn} : \begin{array}{l} B_{Tr} \to B_{ctrl}^{k} \\ x_i \mapsto \left(\Psi_{(1)}(x_i),...,\Psi_{(k)}(x_i)\right) \end{array}$$

avec $\Psi_{(h)}(x_i)$ le *h-nearest neighbour* de x_i pour $h \in \{1,...,k\}$

On peut donc écrire l'estimateur de F^* :

$$\widehat{F_i^*} = \mathbb{1}_{h_k(x_i) > \frac{k}{2}}$$

avec

$$h_k(x_i) : \left\{ \begin{array}{l} B_{Tr} \to \{1,...,k\} \\ x_i \mapsto \sum_{h=1}^{k} \mathbb{1}_{F(\Psi_{(h)}(x_i))=1} \end{array} \right.$$

Deux options s'offrent alors à nous. On peut tout d'abord appliquer la régression linéaire à la sous partie des fraudeurs. On aurait alors :
Soit $\widehat{\beta} \in \mathbb{R}^{m+1}$ le vecteur de coefficients associé à notre modèle, estimation du β vérifiant la relation (1). Alors on a :

$$\forall x_i \in B_{Tr},\ MAG_i^* = \left\{ \begin{array}{ll} x_i\beta + \epsilon_i & si \quad \widehat{F_i^*} = 1 \\ 0 & \text{sinon} \end{array} \right.$$

$$\Leftrightarrow \forall x_i \in B_{Tr},\ MAG_i^* = \left\{ \begin{array}{ll} x_i\beta + \epsilon_i & si \quad h_k(x_i) > \frac{k}{2} \\ 0 & \text{sinon} \end{array} \right.$$

D'où l'estimateur du manque à gagner suivant, obtenu en combinant la classification knn à la regression linéaire :

$$\forall x_i \in B_{Tr},\ \widehat{MAG_i^*} = \left\{ \begin{array}{ll} x_i\widehat{\beta} & si \quad h_k(x_i) > \frac{k}{2} \\ 0 & \text{sinon} \end{array} \right.$$

On estime donc le manque à gagner par cette formule en utilisant la distance d_{PSM} définie auparavant, c'est-à-dire qu'on procède à un appariement de score de propension à k voisins pour un individu de la base de traitement. On effectue ces calculs pour $k = 3,5,7$, k impair pour avoir une majorité nette pour la classification.

On obtient alors les résultats suivants :

```
> ################ Estimation du nombre de fraudeurs :
>
> ######## 3-nn :
> potentialFraud_2014_3nn <- sum(Resume$fraud_3nn)
> potentialFraud_2014_3nn
[1] 40261
> # 40,261
>
> ######## 5-nn :
> potentialFraud_2014_5nn <- sum(Resume$fraud_5nn)
> potentialFraud_2014_5nn
[1] 30980
> # 30,980
>
> ######## 7-nn :
> potentialFraud_2014_7nn <- sum(Resume$fraud_7nn)
> potentialFraud_2014_7nn
[1] 23246
> # 23,246
```

(a) Estimation du nombre de fraudeurs potentiels par la classification knn, avec k=3,5,7

```
> ############# k-nn puis Regression :
>
> ######## 3-nn :
> mag_lin_2014_3nn <- sum(Resume$linear_prediction_3nn)
> mag_lin_2014_3nn
[1] 78135182
> # 78,135,182
>
> ######## 5-nn :
> mag_lin_2014_5nn <- sum(Resume$linear_prediction_5nn)
> mag_lin_2014_5nn
[1] 36245498
> # 36,245,498
>
> ######## 7-nn :
> mag_lin_2014_7nn <- sum(Resume$linear_prediction_7nn)
> mag_lin_2014_7nn
[1] 54933906
> # 54,933,906
```

(b) Estimation du manque à gagner par la combinaison entre classification knn et régression linéaire, avec k=3,5,7

FIGURE 15 – Estimation du mag par algorithme knn et regression linéaire

On constate tout d'abord une très grande variance dans les résultats de classification en fonction du nombre de voisins k. Surtout, on remarque que l'estimation du nombre de fraudeurs potentiels décroît fortement en fonction de k. En s'intéressant à l'appariement, nous avons en effet remarqué que cela était dû au déséquilibre de la base de données, qui comprend très peu d'individus fraudeurs ($\simeq 29\%$). Les individus étant de plus très regroupés, la part de voisins fraudeurs d'un individu tendra rapidement vers cette proportion si k est grand. Nous avons donc considéré que cet algorithme n'est pas très adapté au problème dès lors que $k > 3$. Toutefois l'ordre de grandeur des estimations obtenues après application de la régression linéaire continuent de nous rassurer sur nos résultats.

2.4.3 Approche par la régression KNN

Plutôt que de simplement appliquer la régression linéaire conditionnellement à $\widehat{F^*}$, nous avons aussi considéré un algorithme de régression *k-nearest neighbours*. Il consiste à attribuer, à un individu potentiellement fraudeur, un montant de redressement potentiel de la valeur de la moyenne des voisins fraudeurs parmi les k appariés ([4]). On peut écrire cet estimateur du manque à gagner de la façon suivante :

$$\forall x_i \in B_{Tr},\ \widehat{MAG_i^*} = \begin{cases} \frac{1}{k}\sum_{h=1}^{k} M_i(\Psi_{(h)}(x_i)) \times \mathbb{1}_{F(\Psi_{(h)}(x_i))=1} & si \quad h_k(x_i) > \frac{k}{2} \\ 0 \qquad\qquad\qquad\qquad\quad \text{sinon} \end{cases}$$

Nous avons de même appliqué cette méthode en utilisant l'appariement par score de propension avec ratio $1:k$ (et donc la distance d_{PSM}), $k = 3,5,7$. On obtient alors :

```
> ############## k-nn puis k-nn :
>
> ######## 3-nn :
> mag_2014_3nn <- sum(Resume$fraudPrediction_3nn)
> mag_2014_3nn
[1] 156876663
> # 156,876,663
```

FIGURE 16 – Estimation du manque à gagner par la combinaison entre classification et régression knn, $k = 3$

Comme précisé auparavant, nous avons choisi de ne plus considérer que l'estimation par algorithme $3nn$. On constate une estimation d'environ 156 Millions, donc plus élevée que les estimations précédentes. L'ordre de grandeur semble tout à fait satisfaisant. La différence entre cette valeur et celle prédite par la régression linéaire suggère un mauvais ajustement du modèle linéaire aux données, les montant associés aux petites entreprises semblant sous-estimés et ceux associés aux plus grosses surestimés. Cependant, la méthode de régression *k-nearest neighbours* nécessite un appariement avec ratio 1 : k, qui peut s'avérer coûteux. Nous ne considérerons donc pas de la combiner avec des méthodes de classification n'en faisant pas intervenir, comme par exemple la classification par *forêt aléatoires* que nous implémenterons plus bas.

3 Approche par l'algorithme Random Forest

Afin de parfaire notre modèle, nous avons réétudié notre approche de l'estimation du Manque à gagner, en passant cette fois par la technique de Machine Learning des *Random Forest*. À nouveau, la résolution du problème s'articule en deux étapes : une classification suivie d'une de régression, mais cette fois donc par une approche de modèle non linéaire.

3.1 Théorie de l'algorithme Random Forest

Le *Random forest* est un algorithme de Machine Learning créé par Leo Breiman en 2001 qui se base sur l'assemblage d'*arbres de décision* avec une approche de type *bagging*.

Tout d'abord, rappelons ce qu'est un arbre de décision.
Les arbres de décision sont un type d'apprentissage supervisé où les données sont continuellement divisées en fonction d'un certain paramètre qui sera sujet à l'apprentissage. L'arbre peut être représenté par des assemblages de deux types d'entités, à savoir les nœuds de décision et les feuilles. Les données passent par les nœuds de décisions où elles sont divisées jusqu'à atteindre les feuilles qui représentent la décision finale.

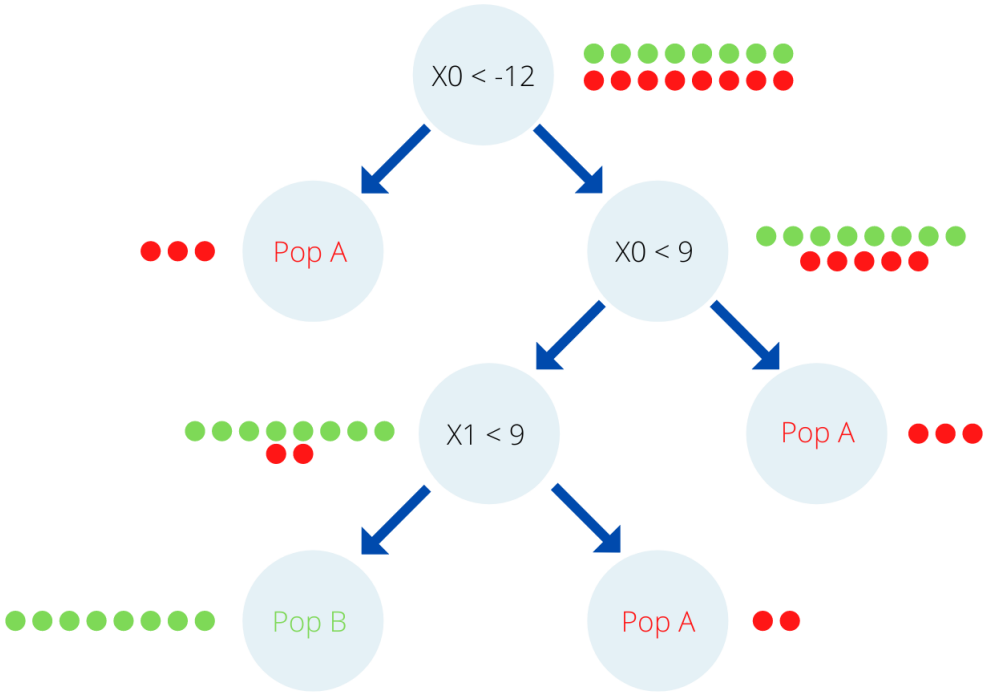


FIGURE 17 – Arbre de décision pour un exemple de classification à 2 features

Le fonctionnement d'un arbre de décisions repose sur le principe de l'incertitude, représentée mathématiquement par l'*Entropie*. L'entropie est une métrique de la *Théorie de l'information*, qui mesure l'impureté ou l'incertitude d'un groupe d'observations, et qui va permettre à un arbre de décision de déterminer comment diviser les données de façon optimale. En effet, l'idée est de minimiser l'incertitude afin de proposer un résultat précis et "sûr".
Soit un problème de classification à N classes, l'entropie se calcule à l'aide de la formule suivante :

$$E = \sum_{i=1}^{N} -p_i log(p_i)$$

Où p_i est la probabilité d'être dans la classe $i \in \{1, ..., N\}$.

On définit ensuite le *gain d'information*, qui une mesure de la quantité d'informations qu'un nœud de décision fournit sur une classe. Ainsi l'arbre de décision va se fractionner pour maximiser ce gain d'information dont la formule est :

$$Gain = E_{parent} - E_{enfant}$$

Où E_{parent} est l'entropie du nœuds de décision parent et E_{enfant} est l'entropie du nœuds de décision enfant. Le gain d'information mesure donc la différence d'incertitude entre chaque nœuds. C'est ainsi, en maximisant le $Gain$ à chaque noeud de décision, qu'un arbre de décision est construit.

Expliquons maintenant le concept des forêts aléatoires, qui repose principalement sur ce que l'on appelle le bagging. Le **bagging** est une technique qui rassemble le **b**oostrap et l'**agg**regat**ing**.
Dans un premier temps, chaque arbre de la forêt aléatoire est entraîné sur un sous ensemble aléatoire de données et de features sélectionné grâce au bootstrap.
Ensuite, une fois l'entraînement de tous les arbres de décision terminé, l'algorithme combine les résultats de ces arbres et va effectuer une agrégation des données afin d'obtenir la prévision la plus solide. Pour un problème de classification, l'algorithme prend sa décision à l'aide d'un système de vote (par défaut, la majorité l'emporte). Pour un problème de régression, celui-ci va calculer une moyenne des valeurs prédites. [7]

Sous R, l'algorithme Random Forest s'implémente à l'aide du package **randomForest**. La fonction *randomForest* associé à ce package prend en principaux arguments les paramètres indispensables pour définir une forêt aléatoire, à savoir les données sur lesquelles entraîner le Random Forest, la variable cible à prédire, le nombre d'arbres et le nombre de features utilisés par chaque arbre.

3.2 Application à la classification

Pour le problème de classification, nous nous sommes dans un premier temps concentré sur la base des entreprises contrôlées pour entraîner notre modèle. Comme pour les appariements, nous avons commencé par appliquer une régression *Lasso* afin de se ramener à un nombre raisonnable de covariables. Après avoir effectué cette sélection des features pertinents, nous nous ramenons à un modèle de 29 variables.
Afin de pouvoir valider notre modèle, nous avons procédé à la courante *Cross-validantion* : nous avons séparé aléatoirement cette base des individus contrôlés en une base de *Train* et une base de *Test* représentant respectivement 80% et 20% des observations.

Nous avons donc entraîné notre Random Forest sur cette base Train grâce à la fonction *randomForest*. Nous lui avons passé les paramètres suivants :

- Les données sur lesquelles nous voulons entraîner notre Random Forest (ici la base de train)
- La variable cible (la variable binaire associée à la fraude, F_i)
- Un nombre d'arbre pris arbitrairement à 500
- Un nombre de feature par arbre égal à 5 (la racine carré du nombre de features, bonne pratique par défaut)

3.2.1 Mesure de performance

Nous avons donc mis en place notre premier modèle de forêts aléatoires, dont les résultats sont affichés ci-dessous.

```
Call:
 randomForest(formula = train$redres_2014 ~ ., data = train, ntree = 500,      mtry = 5)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5

        OOB estimate of  error rate: 27.54%
Confusion matrix:
     0   1 class.error
0 2896 208  0.06701031
1  997 274  0.78442172
```

FIGURE 18 – Résultats de notre premier modèle de classification Random Forest

Pour mesurer la performance d'un randomForest de classification, les indicateurs présentés par défaut sont l'*Erreur Out-of-bag* (OOB) ainsi que la *matrice de confusion*.
L'erreur Out-of-bag (OOB) est une méthode de mesure de l'erreur de prédiction des forêts aléatoires qui utilise le bootstrap agregating (bagging). Il s'agit de l'erreur moyenne calculée à partir des erreurs de prédiction de chaque arbre sur les données qui n'étaient pas contenues dans l'échantillon bootstrap utilisé pour l'entraînement de ceux-ci.
Soit K le nombre d'arbre de la Random Forest, $k \in \{1....K\}$, et a_k le $k^{ième}$arbre de la forêt aléatoire. On note D_k, de cardinal n_{D_k}, la base des observations qui ne sont pas utilisés pour l'entraînement de d'arbre a_k. Alors on peut calculer l'erreur out of bag e_{OOB} :

$$e_{OOB} = \sum_{k=1}^{K} \frac{TP_k + TN_k}{n_{D_k}}$$

Où TP_k et TN_k son respectivement le nombre de vrais positifs et de vrais négatifs obtenus après prédiction de l'arbre a_k sur les données D_k.
La matrice de confusion mesure la qualité d'un modèle de classification en déterminant la différence entre la classe réelle et la classe prédite d'un individu. Pour avoir cette information, nous avons appliqué notre modèle à la base test grâce à la fonction "predict" et nous avons comparé les valeurs obtenues avec les valeurs réelles.

```
> test$predicted <- predict(rf,test)
> table(test$predicted,test$redres_2014)

    0   1
0 712 279
1  38  64
> conf <-confusionMatrix(data = test$predicted, reference = test$redres_2014)
> conf$byClass["Sensitivity"]
Sensitivity
  0.9493333
> conf$byClass["Specificity"]
Specificity
  0.1865889
```

FIGURE 19 – Matrice de confusion

Grâce à cette matrice, nous pouvons obtenir la *Sensibilité* et la *Spécificité* de notre modèle sur la base de test, qui correspondent respectivement à la proportion de vrais positifs et de vrais négatifs parmi l'ensemble des prédictions d'un positif.

$$Sensibilité = \frac{TP}{TP+FN}$$

$$Spécificité = \frac{TN}{TN+FP}$$

TP est le nombre de vrais positifs, TN le nombre de vrais négatifs, FP le nombre de faux positifs et FN le nombre de faux négatifs. Ici, les vrais positifs/négatifs sont les individus prédits fraudeurs/non-fraudeurs et qui le sont réellement.

Dans la Figure 18, on observe que l'erreur Out-of-bag de notre modèle est de 27,54%.
Grâce à Figure 19, nous observons que notre modèle prédit très bien la classe de l'entreprise lorsqu'elle est non-fraudeuse, mais a tendance à échouer à détecter les entreprises fraudeuses. Nous en concluons que notre Random Forest, qui n'est pas assez performant, peut être sujet à l'Overfitting, ce qui peut être dû à plusieurs facteurs distincts.
Tout d'abord, la base de train contenant beaucoup plus d'individus non-fraudeurs que d'individus fraudeurs, elle apparaît comme très déséquilibrée. Dans cette situation, la forêt aléatoire va reproduire ce déséquilibre dans ses résultats, et présenter une très bonne sensibilité, compensant la très mauvaise spécificité du modèle. Nous allons donc envisager par la suite de rééquilibre ces données.
Ensuite, les paramètres du Random Forest ayant été fixé arbitrairement, nous allons devoir effectuer du tuning d'hyperparamètres afin d'optimiser notre modèle, notamment sur le nombre d'arbres et le nombre de features sélectionnés par arbre.
Enfin, par définition, l'algorithme Random Forest maximise l'accuracy. Or dans notre situation, on cherche beaucoup plus à maximiser l'AUC (Aire sous la courbe ROC) afin d'avoir une prédiction beaucoup plus juste notamment sur les individus fraudeurs.

La courbe ROC (Receiving Operator Characteristic) est la courbe $(FPR(s), TPR(s))$ avec :

$$FPR(s) = 1 - \frac{TP(s)}{TP(s) + TN(s)}$$

$$TPR(s) = 1 - \frac{FN(s)}{FP(s) + FN(s)}$$

Où s est le seuil de probabilité à partir du quel un individu est considéré comme de la classe positive, TP le nombre de vrais positifs, TN le nombre de vrais négatifs, FP le nombre de faux positifs, FN le nombre de faux négatifs, $FPR(s)$ le taux de faux positifs en fonction du seuil s et $TPR(s)$ le taux de vrais positifs en fonction du seuil s. Ainsi, l'aire sous la courbe (Area Under the Curve – **AUC**) est un indice calculé pour les courbes ROC. Il représente la probabilité que la classe d'un individu soit justement prédite. Pour choisir notre modèle et en évaluer la performance, nous allons préférer utiliser cette métrique et l'optimiser.
Par la suite, nous avons travaillé sur ces différents sujets afin d'obtenir un modèle plus approprié et plus performant.

3.2.2 Rééquilibrage des données

Afin d'améliorer l'entraînement de notre Random Forest, nous avons décidé d'effectuer un suréchantillonnage sur la base des entreprises contrôlées afin d'avoir un équilibre entre le nombre d'individu fraudeur et non-fraudeur. Pour cela, nous avons utilisé la technique d'*over-sampling* **SMOTE** (Synthetic Minority Over-Sampling TEchnique).
Celle-ci permet de reproduire un certain nombre d'individus de la classe minoritaire afin d'équilibrer les données. Le principe de cette méthode est de générer de nouveaux échantillons en combinant les données d'un individu de la classe minoritaire avec celles de ses voisins les plus proches.
Dans notre cas, l'algorithme SMOTE va s'articuler en 5 étapes.

Etape 1 : Celui-ci va choisir aléatoirement un individu parmi les individus fraudeurs

Etape 2 : Il va déterminer ses K-plus proches voisins en calculant la distance euclidienne entre cet individu et tous les individus fraudeurs

Etape 3 : Il va choisir aléatoirement un individu parmi ses k plus proches voisins

Etape 4 : En considérant que x_1 est l'individu choisi à l'étape 1 et que x_2 est l'individu choisi à l'étape 3 parmi les K-plus proches voisins de x_i. La fonction SMOTE va créer un nouvel individu x grâce à la formule suivante :

$$x = x_1 + Z \times \|x_1 - x_2\|$$

avec Z une variable aléatoire distribuée selon une loi $\mathcal{N}(0,1)$

Etape 5 : Il va répéter ces étapes afin d'obtenir un équilibre entre le nombre d'individu fraudeur et non-fraudeur

Sous R, SMOTE s'utilise grâce au package **smotefamily** et plus particulièrement à la fonction *SMOTE*. Celle-ci prend en arguments principaux : la variable cible pour l'over-sampling, les données non-équilibrées et le nombre de K-plus proches voisins que l'on veut. Dans notre cas, nous avons décidé de prendre K = 5, souvent utilisé par défaut, et nous avons implémenté le code suivant :

```
base_controle_equilibre <- SMOTE(base_controle, base_controle$redres_2014, K = 5)
base_controle_finale <- base_controle_equilibre$data
```

Après application de la fonction SMOTE sur notre base des individus contrôlés, nous obtenons une nouvelle base équilibrée sur laquelle nous pouvons entraîner notre Random Forest.

3.2.3 Éviter le surapprentissage : la K-fold cross validation

Afin d'éviter le surapprentissage que nous avons évoqué, nous avons décidé de mettre en place une technique de validation de modèle nommée *K-fold Cross Validation*. Il s'agit d'une stratégie de partitionnement des données qui permet d'utiliser efficacement un ensemble de données afin de construire un modèle plus généralisé et donc d'éviter l'overfitting. C'est de plus d'une technique utilisée pour le tuning des hyperparamètres. Le principe de cette méthode peut être décrit en plusieurs étapes :

- L'ensemble des données est divisé en une base de train et une base de test.
- La base de train est ensuite divisé en K partitions appelées folds
- Parmi les K-folds, (K-1) folds sont utilisés pour entraîner le modèle avec des hyperparamètres différents et le fold restant est utilisé pour la validation. La performance de chaque modèle est enregistrée.
- L'étape ci-dessus est répétée jusqu'à ce que chacun des k-folds soit utilisé pour la validation.
- Enfin, les hyperparamètres du modèle le plus optimal sont sélectionnés.
- Le modèle est ensuite entraîné en utilisant la base de train totale et la performance du modèle est calculée sur la base test.

Sous R, cette validation croisée s'effectue à l'aide du package **caret** et plus particulièrement à l'aide des fonction *trainControl* et *train*. La fonction *trainControl* est utilisée pour définir les modalités du train, que l'on effectuera grâce à la la fonction *train*.

3.2.4 Optimisation des hyperparamètres

Nous avons donc tout d'abord effectué une K-Fold cross validation afin de déterminer les paramètres optimaux pour notre modèle. Dans notre cas, nous avons choisi de prendre $K = 5$, compromis entre une validation moins précise et trop coûteuse. Nous créerons un modèle qui cherchera à maximiser l'AUC, comme précisé plus tôt.

Avant de mettre en place cet algorithme de validation croisée, nous avons cherché à déterminé le nombre d'arbres optimisant notre modèle, afin d'optimiser la précision des calculs, sans qu'ils soient trop coûteux. Pour cela, nous avons choisi de calculer l'AUC de notre modèle en fonction du nombre d'arbre (Figure 20).

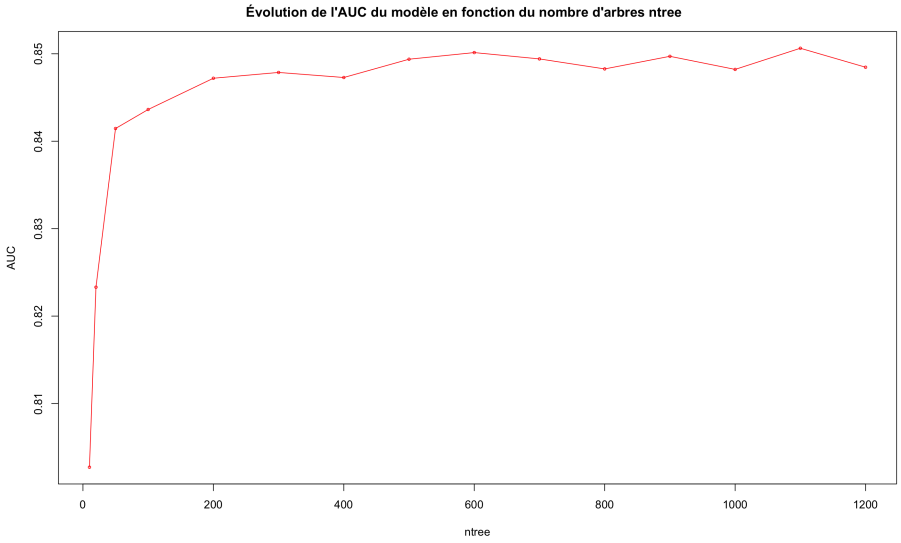


FIGURE 20 – AUC en fonction du nombre d'arbre

Il résulte de cette étude que le nombre d'arbres qui va maximiser notre AUC et réduire au maximum le temps de calcul de notre algorithme est $ntree = 600$.
Nous avons ensuite utilisé la K-fold cross validation pour le tuning d'hyperparamètres en appliquant le code suivant :

```
train.control <- trainControl(method = "cv", number = 5,
verboseIter=TRUE,classProbs = TRUE,
                summaryFunction = twoClassSummary)
mod<-train(redres_2014 ~ ., data = base_rf, method = "rf",
                trControl = train.control, metric ="ROC", maximize = TRUE,
                tuneGrid = data.frame(mtry = c(4,5,6,7)), ntree = 600)
print(mod)
print(mod$finalModel$confusion)
```

On obtient alors les résultats suivants :

```
No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 5666, 5666, 5665, 5665, 5666
Resampling results across tuning parameters:

  mtry  ROC        Sens       Spec
  4     0.8482702  0.8884262  0.6446809
  5     0.8496862  0.8840164  0.6521156
  6     0.8496566  0.8821989  0.6564504
  7     0.8488024  0.8793431  0.6533501

ROC was used to select the optimal model using the largest value.
The final value used for the model was mtry = 5.
> print(mod$finalModel$confusion)
             non_fraudeur fraudeur class.error
non_fraudeur         3402      452   0.1172807
fraudeur             1095     2133   0.3392193
```

FIGURE 21 – K-folds cross validation

Nous observons donc que le mtry optimal (nombre de features utilisé par chaque arbre) est de 5. L'affinement du modèle et le choix de l'AUC comme mesure principale nous a aussi permis d'améliorer la spécificité du modèle, quitte à sacrifier un petit peu de sensibilité. Nous avons un AUC de 0,8496 ce qui est très satisfaisant et indique une très bonne capacité prédictive. Cependant, l'erreur de type I est toujours beaucoup plus faible que l'erreur de type II. Or dans notre situation, nous souhaitons avoir une erreur de type II la plus faible possible. En effet, dans le cadre de notre étude, nous avons fait le choix d'accorder une plus grande importance à la détection de tous les potentiels fraudeurs, quitte à avoir quelques individus considérés commme fraudeurs potentiels à tort.

Or notre modèle Random Forest est capable de prédire un score d'appartenance à une classe, qui doit être interprété avant de pouvoir être associé à une étiquette de classe précise. Le seuil de base de vote pour la classification est de 0,5 : on classera comme fraudeur potentiel toutes les observations pour lesquelles 50% ou plus des arbres ont indiqué que c'était le cas. Ainsi, une approche simple et directe pour améliorer les performances de notre modèle est d'ajuster le seuil utilisé pour faire correspondre les probabilités aux étiquettes de classe, et optimiser l'AUC. Après avoir étudié les différents modèles en fonction de ce seuil, nous avons trouvé un seuil optimal de 0.42, qui maximise l'AUC et minimise notre erreur de type II. Cela reviendrait à classer comme fraudeur potentiels tous les individus que seulement 42% des arbres ou plus ont considéré comme tels.

Nous obtenons donc, grâce à l'option cutoff de la fonction train, qui permet de modifier ce seuil, le code suivant :

```
train.control <- trainControl(method = "cv", number = 5,
verboseIter=TRUE,classProbs = TRUE,
                summaryFunction = twoClassSummary)
mod <- train(redres_2014 ~ ., data = base_rf, method = "rf",
        trControl = train.control, metric ="ROC", maximize = TRUE,
        tuneGrid = data.frame(mtry = 5), ntree = 600,  cutoff=c(0.58,0.42))
print(mod)
print(mod$finalModel$confusion)
```

Nous obtenons alors les résultats suivants :

```
No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 5666, 5665, 5665, 5666, 5666
Resampling results:

  ROC        Sens       Spec
  0.8502965  0.7703691  0.7630163

Tuning parameter 'mtry' was held constant at a value of 5
> print(mod$finalModel$confusion)
             non_fraudeur fraudeur class.error
non_fraudeur         2959      895   0.2322263
fraudeur              776     2452   0.2403965
```

FIGURE 22 – K-folds cross validation

Nous avons donc un AUC final de 0.85, ce qui est encore meilleur que précédemment et justifie notre choix de modifier le seuil. De plus, les erreurs de type I et de type II sont maintenant équilibrées. Ces résultats témoignent donc de la bonne capacité du modèle à prédire si un individu est fraudeur ou non.

3.3 Application au problème de régression

3.3.1 Random Forest de régression

Tout comme nous l'avons fait pour la classification, nous avons implémenté un Random Forest de régression sur la base des individus fraudeurs afin de prédire le montant du Manque à gagner. Afin de réduire le nombre de variables, nous avons à nouveau effectué une régression Lasso pour ne garder que les variables pertinentes, et nous obtenons un modèle final composé de 18 features. De la même façon que pour la classification, nous avons mis en place une validation croisée K-Folds avec $K = 5$, le *Rsquared* comme métrique de référence ainsi qu'un nombre d'arbre égal à 100.

Nous obtenons les résultats suivants :

```
No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 1293, 1290, 1291, 1291, 1291
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
   4    1.617058  0.2554754  1.271837
   5    1.624195  0.2495864  1.275074
   6    1.621027  0.2517440  1.274926
   7    1.626755  0.2480766  1.274577
   8    1.626309  0.2486903  1.276182
   9    1.629224  0.2462271  1.276157
  10    1.629745  0.2458998  1.279452

Rsquared was used to select the optimal model using the largest value.
The final value used for the model was mtry = 4.
```

FIGURE 23 – K-folds cross validation

Le mtry optimal est donc égal à 4 et nous obtenons un RMSE de 1.617 et un Rsquared de 0.255

3.3.2 Applications de forêts aléatoires successives

Maintenant que nous avons créé ces deux forêts aléatoires, nous pouvons les combiner afin d'obtenir notre estimation du Manque à gagner. Tout d'abord, en appliquant le Random Forest de classification à la base des entreprises non contrôlées nous obtenons un total de 49 833 potentiels fraudeurs. (Figure 24)

```
> # Prediction nombre potentiels fraudeurs
> base_non_controle$isFraud<-predict(mod, base_non_controle)
> nb_fraud<-sum(base_non_controle$isFraud=="fraudeur")
> nb_fraud
[1] 49833
```

FIGURE 24 – Résultats du Random Forest de classification

Ensuite, en appliquant le Random Forest de régression à cette base d'individus potentiellement fraudeurs, nous obtenons une estimation du Manque à gagner de 79 millions d'euros (voir Figure 25).

```
> #Estimation du Manque à Gagner
> base_non_controle$potentialFraud<-ifelse(base_non_controle$isFraud=="fraudeur",exp(predict(mod_lin, base_non
_controle)),0)
> mag_2014_RF<-sum(base_non_controle$potentialFraud)
> mag_2014_RF
[1] 79294711
```

FIGURE 25 – Estimation du MAG

À nouveau, nous nous apercevons que le montant de redressement total semble faible proportionnellement à celui de la base des entreprises contrôlées. La proportion de fraudeurs entre les groupes étant la même, nous sommes dans le même cas de figure que (2.3). Ce qui nous permet de consolider l'hypothèse que le biais de contrôle semble porter sur le montant de redressement plutôt que sur la probabilité de fraude d'une entreprise.

3.3.3 Combinaison : algorithme Random Forest et régression linéaire

Nous avons finalement décidé d'appliquer notre modèle de régression linéaire à la base de fraudeurs potentiels que nous avons obtenue à l'aide du Random Forest de classification. En effet, s'il peut s'avérer plus intéressant qu'un appariement classique pour un problème de classification, il présente quelques inconvénients pour un problème de régression comme le notre. Il semble donc pertinent de combiner cette technique de classification avec notre modèle linéaire afin d'affiner notre estimation.

```
> mag_2014_rf_lin<-sum(base_non_controle$linearFraud)
> mag_2014_rf_lin
[1] 141140106
```

FIGURE 26 – Estimation du MAG : classication Random Forest/Regression linéaire

Appliquée à la même base de fraudeurs potentiels, notre régression linéaire nous donne donc une estimation d'environ 140 millions d'euros, soit près de 60 de plus de manque à gagner. Cela peut s'expliquer par les difficultés, pour l'algorithme de Random Forest, à extrapoler les données. En effet, l'algorithme de Random Forest ne peut prédire des montants qu'en fonction des observations effectives de la base d'entraînement, en renvoyant la moyenne de certains d'entre ces montants. Il ne pourra donc pas prédire un montant de redressement supérieur ou inférieur à ceux des observations qu'il a à sa disposition. Cela peut s'avérer problématique dans notre cas dans la mesure où nous ne disposons que de peu d'observations (et en particulier d'observations qui présentent un redressement élevé). L'algorithme est donc dans l'impossibilité de prédire de grands montants de redressements.

Bien que la méthode de Random Forest linéaire présente un R-squared plus élevé que la méthode de régression linéaire, nous sommes donc en droit de penser que cette dernière est plus efficace dans notre cas de figure dans la mesure où elle nous permet de prédire des montants plus cohérents grâce à de meilleurs prédictions sur les valeurs extrêmes et peu semblables aux individus de la base d'entraînement.

Conclusion

Au cours de ce mémoire, nous avons donc eu l'occasion d'estimer le manque à gagner des entreprises sous le régime d'un organisme de collecte. Nous avons pu effectuer ces estimations par différentes méthodes, chacune présentant certains avantages et inconvénients. Concernant dans un premier temps l'estimation du nombre de fraudeurs potentiels $n_{F^*} = \sum_{i=1}^{n} F_i^*$ (problème de classification), nous avons pu tester les méthodes d'appariement par score de propension, distance de Mahalanobis et par l'algorithme des K plus proches voisins. Nous avons enfin essayé de répondre au problème par un modèle non linéaire, en passant par l'algorithme des forêts aléatoires. Ce travail nous a permis de comparer ces différentes approches, et d'évaluer leur efficacité dans la résolution de ce problème de classification.
L'approche de l'appariement par score de propension nous a semblé juste, et nous fournit une estimation qui, théoriquement, contrebalance un éventuel biais sur la variable de contrôle. Cependant, outre le nombre fourni, nous n'avons que très peu de moyens pour évaluer cette qualité. La distance de Mahalanobis nous fournit une estimation similaire de n_{F^*}. Elle présente l'avantage de pénaliser les variables les plus bruitées. Cependant cet appariement est plus coûteux en utilisation, ce qui peut représenter un frein à son utilisation. Nous avons également vu que l'estimation par l'algorithme *knn* semble moins précise, voire n'est pas envisageable dès que $k > 3$. Cela est dû à la répartition des individus au sein de la base, qui est notamment très déséquilibrée. Bien que coûteuse, cette approche permet en revanche d'appliquer pour la suite un algorithme de régression *3nn*, présentant certains avantages. Enfin, la classification par l'algorithme *Random Forest* semble la plus précise. Grâce aux efforts déployés pour équilibrer la base (*Oversampling*) et éviter le sur-apprentissage (*K-Fold Cross Validation*), cet algorithme de classification a été très satisfaisant. Il reste relativement coûteux, mais sa capacité à contrebalancer les observations influentes (*outliers*) permet l'élaboration d'un modèle précis et robuste.

Globalement, l'ensemble de ces méthodes nous ont donné des estimations de n_{F^*} relativement similaires, mais nous avons pu constater que l'identité des entreprises potentiellement fraudeuses variait de façon importante selon la méthode, expliquant les variations de résultats pour l'estimation du MAG. En effet, la classification par PSM avait par exemple tendance à déclarer plus de "petites entreprises" comme fraudeuses potentielles, contrairement à la classification *Random Forest* : la taille de l'entreprise s'est avérée être un facteur très significatif à corrélation positive sur le modèle de régression linéaire notamment. Ces plus petites entreprises se voyaient ainsi attribuer de plus faibles montants de redressement potentiel, expliquant l'estimation du MAG plus importante pour la méthode des forêts aléatoires que pour l'appariement PSM.
Ces différents modèles de classification nous ont permis d'obtenir des bases d'individus potentiellement fraudeurs sur lesquelles il était possible d'appliquer nos prédictions afin d'estimer le manque à gagner. Nous avons également pu effectuer ce travail de régression selon plusieurs méthodes qui présentaient chacune leurs spécificités. La méthode de régression par Random Forest présente en effet un R-squared plus élevé que la régression linéaire et peut donc sembler plus juste. Cependant, son incapacité à extrapoler les données peut s'avérer problématique, notamment dans le cadre de cette étude : on ne dispose que d'un faible nombre d'observations et surtout de données d'entraînement très regroupées, contre des données à prédire plus disparates voire forçant justement l'extrapolation. Ce même problème d'évaluation de données extrêmes se retrouve pour la méthode de régression par les *K-nearest neighbours*, bien qu'elle nous donne également une estimation plus juste que la régression linéaire pour les observations ressemblantes aux individus contrôlés. Toutefois elle ne peut donner une estimation fine pour les individus qui présentent des features susceptibles de mener à un montant de redressement hors de l'intervalle des montants de redressement effectifs. Cette méthode est enfin nettement plus coûteuse que la méthode de régression linéaire classique. Si la méthode de régression linéaire présente un faible R-squared et a tendance à sous évaluer les montants faibles et

surévaluer les montants élevés, sa capacité à fournir des estimations plus réalistes pour les montants qui ne ressemblent pas aux données d'entraînement est un réel avantage dans le cadre de notre travail : ces extrapolations faussées se compensent, nous donnant une estimation du MAG agrégé très satisfaisante et cohérente avec les chiffres que nous pouvions attendre.
Finalement , cette étude nous a mené à conclure sur l'existence certaine d'un biais de contrôle. L'organisme de collecte semble s'intéresser aux entreprises susceptibles de se voir attribuer, si elles fraudent, de grands montants de redressement. Nous pouvons nous questionner sur l'efficacité de cette méthode, l'organisme pourrait-il mieux sanctionner la fraude en contrôlant en priorité les entreprises susceptibles de frauder, peu importe le montant ?

Références

[1] Peter C. AUSTIN. *An Introduction to Propensity Score Methods for Reducing the Effects of Confounding in Observational Studies*. 2011.

[2] Peter C. AUSTIN. *The relative ability of different propensity score methods to balance measured covariates between treated and untreated subjects in observational studies*. 2009.

[3] Alboukadel KASSAMBARA. *Machine learning essentials*. 2018.

[4] Yating LIU. *Cours d'apprentisatge statistiques*. 2021-2022.

[5] Kattia MEZIANI. *Cours de Modèles linéaires et ses généralisation*. 2021-2022.

[6] Frédéric Bertrand et MYRIAM MAUMY. *Cours de Master 1ère année*. 2008.

[7] Rakotomalala RICCO. *Bagging - Random Forest - Boosting*.

[8] Paul R. ROSENBAUM. « Propensity scores ». In : (2005).

[9] Paul R. ROSENBAUM et Donald B. RUBIN. *The central role of the propensity score in observational studies for causal effects*. 1983.