Journées de méthodologie statistique de l'Insee 2022

Imputation de valeurs manquantes avec des réseaux de neurones : prédiction des salaires dans l'enquête Emploi

Damien BABET (*)

(*) Insee, Direction des études et synthèses économiques

damien.babet@insee.fr

Mots-clés. : enquête Emploi, salaire, non-réponse, imputation, réseau de neurone.

Domaines. Non-réponse, Machine learning.

Résumé

Cet article reprend et développe un travail déjà présenté à l'Unece (Babet, 2020) et dans un document de travail en cours d'achèvement (Babet, Deltour, Faria, et Himpens, 2022).

La variable de salaire de l'enquête Emploi souffre d'une non-réponse importante. Environ 5% des salariés refusent de répondre, et 15% n'acceptent de répondre que par tranche. Ces observations font aujourd'hui l'objet d'une imputation par tirage aléatoire dans une distribution paramétrée par une équation de salaire de type "Mincer" (Mincer, 1974), et bornée par les limites de la tranche lorsqu'elles sont connues. Cela permet de livrer un fichier complet, mais aussi de réduire le biais de non-réponse, dans la mesure où cette non-réponse est ignorable (Rubin, 1976) et où le modèle d'imputation est bien spécifié (Chen et Haziza, 2017, 2019).

Nous expérimentons une autre méthode d'imputation : la prédiction par réseau de neurone (RN). Les méthodes d'apprentissage statistique comme les RN sont particulièrement adaptées pour des usages purement prédictifs. Leur intérêt est de faciliter l'inclusion d'un grand nombre de variables et l'estimation de modèles non linéaires et comprenant des interactions. Si le modèle classique est mal spécifié, l'imputation par RN est susceptible de mieux réduire davantage le biais de non-réponse. On peut de plus espérer un gain de précision de l'imputation. Enfin, un modèle d'imputation plus souple semble moins susceptible de générer des corrélations spécieuses.

Nous utilisons un corpus de 1,2 millions d'observations entre 1993 et 2018, divisé en un échantillon d'entraînement d'environ 1 million d'observation, un échantillon de validation et un échantillon de test d'environ 100000 observations chacun. La reconstitution de séries longues pour les variables explicatives n'a pas besoin d'être complète : les RN tolèrent bien en entrée des valeurs manquantes grossièrement imputées à la moyenne et signalée par une indicatrice. En revanche, les nomenclatures détaillées (profession, secteur, diplôme, etc.) sont difficiles à inclure en raison de leur grand nombre de modalités. Nous adaptons des algorithmes de vectorisation de vocabulaire, issus de l'analyse du langage naturel (Mikolov, Sutskever, Chen, Corrado, et Dean, 2013), pour remplacer le vecteur des indicatrices de modalités par un encodage de plus

faible dimension. Cette méthode, qui peut s'appliquer à d'autres types de variables catégorielles, présente un intérêt en elle-même (Doutreligne, Leduc, Nguyen, et Vuagnat, 2020).

La qualité de l'ajustement est mesurée principalement par le R2 sur l'échantillon test. Il est d'environ 0,64 pour une équation de salaire enrichie d'un grand nombre de variables, et de 0,7 pour la prédiction par RN. Outre ce surcroît de précision, le biais de non-réponse se réduit davantage. D'autres enjeux du choix de la procédure d'imputation sont discutés, en particulier la stabilité des résultats, l'impact de l'imputation sur l'estimation de diverses quantités d'intérêt, et les éventuelles questions éthiques soulevées par le choix des variables explicatives.

1 Les réseaux de neurone

Cet article reprend et développe un travail déjà présenté à l'Unece (Babet, 2020) et dans un document de travail en cours d'achèvement (Babet, Deltour, Faria, et Himpens, 2022) qui vise à fournir aux praticiennes et praticiens de la statistique publique un guide méthodologique et des cas d'usage pour mener leurs propres expérimentations avec des réseaux de neurones (RN).

1.1 Réseaux de neurones et statistique publique

La statistique publique peut-elle mobiliser avec profit les réseaux de neurones ? Sur quels types de données, et comment faire ?

Ils restent très peu utilisés en statistique publique principalement pour les raisons suivantes : La perfomance prédictive des réseaux de neurones se manifeste davantage lorsque la taille des données augmente, et la statistique publique travaille, de ce point de vue, sur des données de taille moyenne : données d'enquête comptant quelques dizaines de milliers d'observations, données administratives en millions, ou un peu plus quand elles croisent des populations avec des événements fréquents ou complexes comme les données de santé. Ces bases comportent moins d'observations, mais aussi moins de variables, typiquement quelques centaines au plus, généralement bien moins que le nombre de pixels d'une petite image par exemple. Les données ne sont pas non plus exactement de même nature. Celles de la statistique publique sont généralement structurées (représentativité, usage de nomenclatures et de définitions normalisées, partage de concepts, etc.), parfois confidentielles et souvent hétérogènes (variables de types différents, valeurs manquantes) tandis que les données typiques des RN sont traitées pour être homogènes (des images de même format) à partir d'un matériel de base non structuré.

Par ailleurs, les réseaux de neurones peuvent apparaître comme des « boîtes noires ». Une fois entraînés, ils peuvent prédire avec une très grande efficacité si une image représente un chien ou un chat, et quel serait le meilleur coup à jouer au go, mais il est difficile de comprendre comment sont faites ces prédictions, même approximativement. L'interprétabilité des algorithmes demande un travail supplémentaire, alors qu'il s'agit souvent d'une exigence des producteurs et des usagers des statistiques publiques. Dernier obstacle enfin, les réseaux de neurones ne sont pas encore fournis clés en main. Il faut d'abord en comprendre le fonctionnement, ils demandent de nombreux réglages, des options, des choix de modélisation. Ils possèdent en effet un très grand nombre « d'hyperparamètres »à déterminer avant l'entraînement. Avant de « laisser parler les données », il faut une longue mise en place qu'on s'épargne parfois en choisissant un modèle classique ou un algorithme plus simple de machine learning comme une forêt aléatoire. Il faut enfin se familiariser avec le fonctionnement de nouveaux packages comme Keras (disponible pour R et python).

Les RN peuvent cependant être efficaces sur des données de petite taille et passent à l'échelle facilement lorsqu'on mobilise ensuite de plus gros fichiers. Ils sont très souples et peuvent ainsi s'adapter à certaines demandes de la statistique publique en particulier la protection de la confidentialité, les appariements, le codage automatique, la détection et la correction d'erreurs, etc, ou encore enrichir les types de données exploitables par la statistique publique (images,

textes,...). En effet et comme d'autres méthodes relevant du machine learning, les réseaux de neurones peuvent se voir comme des modèles statistiques très flexibles pouvant mettre en jeu de nombreuses variables explicatives. Ils sont particulièrement adaptés dès lors qu'on s'intéresse à prédire la valeur d'une variable d'intérêt. On les qualifie alors de prédicteurs. Ils donnent des réponses à des problèmes de type $\hat{y}$, par opposition aux problèmes de type $\hat{\beta}$ qui consistent, eux, à mesurer le lien (ou la causalité) entre deux variables, pour reprendre la distinction utile de Mullainathan et Spiess (2017).

Cet article explore un cas d'usage de ce type, un problème de type $\hat{y}$ habituel pour les statisticiens, l'imputation de valeurs manquante. Peut-on utiliser des RN sur des données classiques (mais tout de même relativement abondantes) ? Quels en sont les avantages et les inconvénients ?

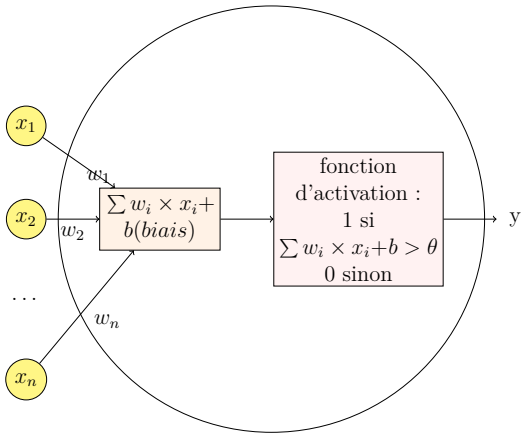
1.2 Qu'est-ce qu'un réseaux de neurones ?

Les RN appartiennent à la famille de l'apprentissage statistique. On peut désigner sous ce terme une « culture »statistique (Breiman, 2001) qui consiste à privilégier des modèles souples et complexes dont la validité est mesurée par les performances sur des échantillons tests, plutôt que démontrée mathématiquement à partir d'hypothèses sur les données. Au sein de l'apprentissage statistique, les RN se situent à un point extrême : ils sont particulièrement souples et complexes, mais aussi encore mal compris théoriquement, bien que leur principe soit simple.

Un réseau de neurones artificiels, ou réseau neuronal artificiel, est un système dont la conception est à l'origine schématiquement inspirée du fonctionnement des neurones biologiques. On les représente communément sous cette forme de « neurones » reliés par des connexions, mais ils peuvent être aussi bien décrits en termes mathématiques, comme un enchaînement de compositions de fonctions. Chaque « neurone »est en réalité une fonction mathématique, qu'on choisit non-linéaire et dérivable, d'une combinaison linéaire des données (ou des sorties des neurones des couches précédentes). Les paramètres du RN, comme les paramètres de beaucoup de modèles statistiques sont les coefficients des combinaisons linéaires.

Brique de base des RN, le « perceptron »est conçu en 1943 (McCulloch et Pitts, 1943) et mis en oeuvre en 1958 (Rosenblatt, 1958), qui propose un algorithme d'apprentissage (figure 1).

Figure 1 – Le perceptron



Le perceptron n'est cependant capable que de résoudre des problèmes linéairement séparables. Pour un modèle plus général, il faut plusieurs couches successives de perceptrons (figure 2).

L'estimation de systèmes multicouches a été rendue possible par la mise au point de l'algorithme de rétropropagation de l'erreur (notamment Le Cun (1985) et Rumelhart, E., et J. (1986)). A la condition que les fonctions d'activation soient dérivables (ou presque partout dérivables) et qu'on se donne une fonction de perte elle aussi dérivable pénalisant la distance entre

Figure 2 – Un RN avec une couche cachée

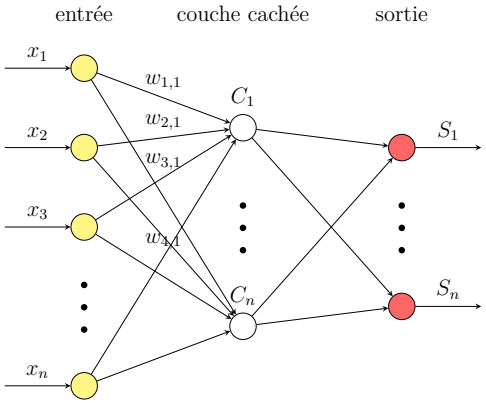


Figure 3 – Choix de l'activation de la dernière couche et de la fonction de perte

Type de problème	Activation	fonction de perte
classification binaire	sigmoid	binary_crossentropy
classification multiclasse	softmax	categorical_crossentropy
regression (dans $\Re$)	linéaire	mse

la prédiction et la vraie valeur (par exemple la perte quadratique), l'algorithme consiste d'abord à calculer le gradient de la perte. Ce gradient peut ensuite être rétropropagé : pour chaque paramètres, on peut calculer le gradient de sa contribution à la fonction de perte en remontant de la sortie du RN vers les inputs.

L'apprentissage se fait ensuite par descente de gradient : à chaque itération, un gradient de la fonction de perte est calculé pour chaque prédiction sur l'échantillon d'entraînement (ou pour un « batch »d'observations), et les paramètres sont légèrement modifiés dans la direction de diminution de la perte. Il n'y a pas de garantie d'atteindre un optimum global

1.3 les RN en pratique

Les RN se caractérisent par un très grand nombre d'hyperparamètres à sélectionner : nombre de couches et de neurones, structure du réseau, fonction de perte, fonction d'activation, optimiseur, taille des batchs et nombre d'époques d'entraînement, méthode de régularisation, etc.. En particulier, il faut déterminer du nombre de couches (la profondeur du RN), du nombre de neurones par couche (qui suit généralement un schéma pyramidal). Plus le réseau est profond, plus il peut exprimer des fonctions complexes, mais plus son entraînement est délicat. La fonction de perte correspond à la nature du problème et on choisit généralement une des trois fonctions classiques (figure 3).

Les fonctions d'activation possibles sont plus nombreuses, et elles peuvent différer pour les neurones des couches cachés et les neurones de la couche de sortie. Les optimiseurs sont les algorithmes de descente de gradient par rétropropagation, qui peuvent avoir des taux d'apprentissage adaptatifs, de l'inertie, etc. Taille de batch et nombre d'époque influent notamment sur la durée de l'entraînement et sa stabilité. Enfin, il existe plusieurs solutions de régularisation (pénalisation $L1$, $L2$ des paramètres, dropout, etc.) qui ont chacune des hyperparamètres. Tous ces hyperparamètres interagissent les uns avec les autres. On peut les tester par cross-validation, mais il n'est pas possible d'explorer l'ensemble des combinaisons possibles, en particulier quand les moyens computationnels sont limités, et le choix s'appuie donc souvent sur les pratiques usuelles et les recommandations de la littérature.

Les librairies les plus populaires sont sans doute PyTorch et TensorFlow (via Keras). Le language python contient les fonctionnalités les plus avancées mais les librairies sont maintenant accessibles en R. A l'Insee, ces environnements sont disponibles sous AUS V3. La plateforme (datalab.sscloud.fr) permet de paralléliser les traitements et donne accès à de la GPU mais n'est pas encore homologuée pour les données confidentielles.

2 Prédire pour imputer

2.1 Pourquoi des réseaux de neurones pour l'imputation de valeurs manquantes ?

L'imputation de valeurs manquantes est une activité classique du statisticien. Il s'agit de remplacer une valeur manquante par une ou des valeurs de remplacement. L'imputation vise à réduire le biais de non-réponse. Elle permet aussi de fournir un fichier complet, généralement directement utilisable par les utilisateurs. Cependant lorsque les données imputées sont utilisées pour l'inférence, le modèle d'imputation peut échouer à réduire voire au contraire amplifier l'éventuel biais de non-réponse. Il peut conduire à sous-estimer l'incertitude des estimations obtenues voire introduire des corrélations fausses entre variables. Cette tâche d'imputation se prête bien à la mise en œuvre d'algorithmes de *machine learning* puisqu'elle est purement prédictive dans son esprit. Le but de l'imputation n'est pas de fournir un modèle facilement explicable et interprétable, et le problème de la boite noire est relégué au second plan. En revanche, la précision de la prédiction est un atout important pour l'imputation. L'avantage *a priori* des réseaux de neurones pour cette tâche est de fournir des modèles très flexibles qui conduiraient à une moindre sous-estimation de la variance de l'échantillon. Nous expérimentons sur des données de salaire l'imputation par RN, en comparant ses performances de prédiction avec la méthode d'imputation actuellement en production et une équation de Mincer.

2.1.1 Prédiction et imputation de la non-réponse partielle

On se situe dans le cadre de l'imputation simple de la non-réponse partielle. Autrement dit, on s'intéresse à des valeurs manquantes pour une seule variable, et on cherche à remplacer chaque valeur manquante par une unique valeur imputée. Outre l'intérêt de proposer un fichier complet et facile d'utilisation, le but principal de l'imputation est de corriger le biais de non-réponse, découlant du fait que les valeurs manquantes ne sont pas distribuées au hasard. Pour espérer corriger le biais de non-réponse, on fait l'hypothèse que la non-réponse est ignorable (MAR ou *missing at random*, Rubin (1976)), c'est-à-dire que la probabilité de répondre d'un individu est indépendante de la valeur de la variable y qu'on impute, conditionnellement aux variables auxiliaires x utilisées pour imputer :

$$\mathbb{P}(r_i = 1|x_i, y_i) = \mathbb{P}(r_i = 1|x_i)$$

Cette hypothèse est agnostique sur le type de modèle utilisé pour prédire la non-réponse ou la valeur manquante. Elle ne suffit pas à assurer la validité de l'imputation si le modèle d'imputation est mal spécifié, mais un modèle bien spécifié peut en revanche corriger le biais de non-réponse sous cette hypothèse.

La question de l'imputation est cependant plus large que celle de la seule recherche d'un bon modèle prédictif, Chen et Haziza (2019) en présentent un panorama récent à partir du lien entre la méthode d'imputation et la quantité d'intérêt formalisée de manière très générale. Si les quantités d'intérêt sont des statistiques de distribution (par exemple, un rapport interquantile), il peut être préférable de recourir à une imputation aléatoire, au prix d'une variance additionnelle, plutôt qu'à une imputation déterministe, qui concentre les valeurs imputées autour

de l'espérance (conditionnelle). De même, les méthodes avec donneurs (plus proche voisin, *hot deck*) ont l'avantage de fournir comme valeurs imputées des valeurs effectivement prises dans la réalité (l'échantillon). C'est la raison pour laquelle elles sont particulièrement recommandées pour des variables catégorielles, mais peuvent aussi avoir du sens pour des variables a priori continues mais souvent arrondis par les répondants comme les salaires. Enfin, l'imputation est source de biais si les données sont ensuite analysées avec des modèles différents et surtout plus riches que les modèles d'imputation (voir Chen et Haziza (2019)).

Dans tous les cas cependant, un bon prédicteur reste utile. L'imputation aléatoire revient généralement à ajouter du bruit à un modèle prédictif déterministe, et bénéficie de la qualité de ce prédicteur. L'imputation par donneur s'appuie souvent sur la définition d'une distance entre observations, et un bon prédicteur peut aussi fournir une distance efficace.

Les réseaux de neurones sont de bons candidats pour leurs performances prédictives. Leur complexité est un atout : lorsque les valeurs imputées sont ensuite utilisées, par exemple pour inférer des grandeurs ou estimer des modèles économétriques, il est souhaitable que le modèle d'imputation soit plus riche que le modèle estimé, pour limiter le risque d'y transférer des relations artificielles entre variables. Par ailleurs, il est possible que les RN intègrent en partie certains des objectifs poursuivis par les différentes méthodes d'imputation. Bien que déterministes comme prédicteurs, ils sont aléatoires dans leur estimation (principalement via le principe de la descente de gradient stochastique et ses dérivées), ce qui peut les rapprocher d'une imputation aléatoire. Ils ont aussi naturellement un comportement proche des méthodes d'ensemble (en particulier avec l'utilisation du dropout), ce qui peut les rapprocher par exemple des méthodes « multiplement robustes »(Chen et Haziza, 2017) qui accumulent les modèles d'imputation. Dans la suite cependant, nous nous limitons à décrire et comparer les résultats des RN comme prédicteurs.

2.1.2 Prédiction de la non-réponse totale

Un autre usage possible des RN est la prédiction du comportement de non-réponse totale dans le but de redresser le biais de non-réponse totale, par exemple par repondération. Il est ainsi possible d'entraîner un modèle à prédire le comportement de non-réponse totale (le refus de participer à l'enquête par exemple) lorsqu'on dispose malgré tout de variables auxiliaires (généralement issues de la base de sondage). Dans cette situation en effet, à la différence de la non-réponse partielle, le recours à l'imputation n'est pas possible, puisqu'il faudrait imputer toutes les variables de l'enquête, avec autant de modèles, et à partir des seules variables de la base de sondage. On procède donc par repondération.

On fait à nouveau l'hypothèse que la non-réponse (totale) est ignorable, c'est-à-dire que la probabilité de réponse est indépendante des variables non observées, conditionnellement aux variables auxilliaires. Sous cette hypothèse, la prédiction d'une probabilité de réponse permet de corriger le biais de non-réponse, en l'utilisant dans un estimateur par double expansion (Haziza, Beaumont, et al. (2017)). En pratique, on pondère les observations (des répondants !) par l'inverse de leur probabilité de réponse estimée, comme s'il s'agissait d'un poids de sondage.

L'entraînement ou l'estimation d'un modèle prédicteur du comportement de non-réponse n'est pas seulement utile dans le cadre de la non-réponse totale. Il peut être un complément de l'imputation simple dans une méthode doublement robuste (Bang et Robins, 2005). L'estimation doublement robuste d'une grandeur d'intérêt fonction d'une variable en partie imputée mobilise à la fois les valeurs imputées (donc un modèle d'imputation) et un modèle de non-réponse (par exemple via la pondération), et est doublement robuste car elle reste convergente même lorsqu'un (et un seul) des deux modèles est mal spécifié.

2.2 Imputation des valeurs manquantes de salaire dans l'enquête Emploi

La question sur le salaire dans l'enquête Emploi est celle qui entraîne le plus grand taux de non-réponse. Elle est posée depuis 1982 (d'abord en tranche, puis en valeur à partir de 1989) aux salariés interrogés pour la première fois puis lors de leur sixième et dernière interrogation. Le salaire mensuel déclaré dans l'enquête Emploi correspond au salaire net tel que déclaré par les répondants, ou corrigé (en multipliant par 0,8) si les répondants disent donner le montant brut, additionné des bonus tels qu'ils sont déclarés par les répondants dans les questions suivantes de l'enquête. La variable résultante est nommée SALRED.

Déclarative, la variable de salaire dans l'enquête Emploi n'est pas la source des statistiques salariales publiées par l'Insee [1] mais elle est très souvent exploitée dans les études économétriques mobilisant l'EEC. Environ 19% des enquêtés interrogés refusent de donner leur salaire en valeur (variable vide, refus ou « ne sait pas »). On leur demande alors d'indiquer leur salaire par tranche : 15% des répondants le font et 4% refusent à nouveau. Les réponses par tranche permettent notamment de constater que la non-réponse est plus concentrée parmi les plus hauts salaires.

Nous expérimentons l'usage d'un RN pour imputer les valeurs manquantes de SALRED. Le développement de l'algorithme est réalisé avec Keras en Python. Certaines parties du code sont présentées ci-dessous dans un but illustratif et pédagogique. Nous comparons les performances d'un réseau à celles de modèles économétriques classiques : notamment un modèle s'appuyant sur une traditionnelle équation de Mincer (Mincer, 1974). Celle-ci relie le log du salaire horaire aux proxies des composantes du capital humain que sont l'expérience et le niveau d'éducation :

$$log(y) = log(y_0) + rs + \beta_1 x + \beta_2 x^2$$

Avec y le salaire horaire, s le nombre d'années d'études et x l'expérience potentielle (la différence entre l'âge au moment de l'observation et l'âge à la fin des études). L'équation prédit plutôt correctement le salaire dans de nombreux fichiers de données, même sous sa forme la plus simple (Lemieux, 2006). Elle sert naturellement de référence (Boelaert et Ollion, 2018).

Nous comparons aussi les performances de ce RN à la méthode d'imputation de valeurs manquantes actuellement en production pour la variable SALRED. Cette méthode d'imputation s'appuie sur des régressions linéaires mais s'éloigne du modèle de Mincer dans la mesure où c'est le salaire mensuel et non horaire qui est mobilisé et où les variables explicatives ne sont pas les mêmes. La méthode en production a recours à cinq équations de salaire mensuel en log, réestimées chaque trimestre pour cinq catégories de salariés (selon le sexe croisé avec les groupes ouvriers/employés et professions intermédiaires/cadres, plus un groupe pour les contrats atypiques, précaires, et les mineurs). Ces équations mobilisent comme variables explicatives : le nombre d'heures habituellement travaillées par semaine (en niveau), la catégorie socio-professionnelle, le secteur d'activité, le nombre d'années d'expérience dans l'emploi occupé (ancienneté), le type d'employeur, le nombre de salariés dans l'entreprise, l'âge, le diplôme, ainsi que des indicatrices d'emploi multiple ou occasionnel, de bonus, de secteur public et de résidence en agglomération parisienne, et enfin le sexe pour le groupe des contrats atypiques. Les valeurs avec des résidus extrêmes sont mises à blancs avant une nouvelle estimation, et les valeurs imputées sont ensuite tirées dans une distribution déduite des équations estimées, de la variance des résidus, et contrainte par la réponse en tranche le cas échéant. L'imputation des valeurs manquantes du salaire en production dans l'enquête Emploi est donc une imputation aléatoire avec une modélisation paramétrique du salaire mais aussi de l'aléa.

1. Elles sont principalement issues de données administratives

2.3 Préparation des données

2.3.1 Sélection et retraitement des variables

Pour entraîner un RN sur ce même problème d'imputation, nous utilisons 1 220 000 observations de salaires, issues de l'EEC entre 1993 et le premier trimestre 2018. La variable de salaire à prédire est construite de la même manière que SALRED, hors imputation évidemment. Il s'agit donc d'un salaire mensuel dont on prendra le log.

Les variables explicatives, les *features*, prises en considération sont plus nombreuses que dans le modèle de SALRED. Pour mieux exploiter les possibilités des RN, nous mobilisons le maximum de variables possible, lorsqu'elles sont disponibles. L'absence de sélection de variable ne pose pas de problème d'estimation ni d'interprétation puisque l'objectif est purement prédictif. En revanche, certaines variables peuvent soulever des questions déontologiques. Faut-il, par exemple, mobiliser la variable de pays de naissance, ou même de sexe, pour imputer le salaire ? On peut choisir de les exclure, notamment pour ne pas courir le risque que des corrélations fallacieuses les impliquant soient ancrées dans les valeurs imputées. Même dans l'hypothèse où le modèle identifie correctement l'impact de ces variables, on peut vouloir les mettre de côté en fonction de l'usage qui sera fait ensuite des valeurs imputée. D'un autre côté, quand les valeurs imputées alimentent des travaux descriptifs, on peut souhaiter qu'elles reflètent au mieux les régularités capturées dans les données d'enquête, et souhaiter intégrer ces variables. Ces questions ne sont pas spécifiques au *machine learning* et se posent aussi pour des modèles d'imputation plus classiques, avec quelques différences cependant. Les modèles classiques permettent au moins en théorie de connaître facilement le poids accordé par un prédicteur à telle ou telle variable. Reposant sur moins de variables, ils pourraient être davantage sujets à générer des corrélations fallacieuses.

Ici l'objectif étant uniquement de montrer les performances prédictives, nous incluons les variables de sexe, année et trimestre d'enquête, date et lieu de naissance, lieu de naissance des parents, statut d'immigration, niveau d'éducation et année du diplôme le plus élevé, expérience dans l'emploi actuel, profession et professions des parents, secteur d'activité, heures travaillées par semaine (habituellement, et spécifiquement pendant la semaine de référence), temps partiel, type de contrat, type d'employeur, existence de bonus. Les variables catégorielles sont remplacées par des indicatrices des modalités (*one-hot encoding*) où nous considérons les valeurs manquantes comme une modalité supplémentaire de la variable. Pour les variables numériques (comme l'âge ou l'année de diplôme) nous imputons les valeurs manquantes par la moyenne et ajoutons une indicatrice de valeur manquante (cette méthode d'imputation brutale est efficace en input d'algorithmes de *machine learning*, car l'algorithme peut exploiter l'indicatrice de valeur manquante si elle est informative, et n'est pas gêné par l'imputation à la moyenne sinon, voir Josse, Prost, Scornet, et Varoquaux (2019)).

Contrairement au modèle d'imputation de SALRED actuellement en production et réestimé chaque trimestre uniquement sur les données du trimestre, nous entraînons le RN sur les données d'une longue période d'enquête (1993-2018), au cours de laquelle le module "salaire" du questionnaire de l'enquête Emploi a évolué, et nous sommes passés du franc à l'euro. Nous avons mobilisé des variables relativement stables sur toute la période, ou reconstruites dans le cadre d'une autre étude (Picart, 2020), travail coûteux mais fréquent lorsqu'on veut mobiliser les enquêtes dans le temps long. Une seule correction supplémentaire a été faite pour le RN : jusqu'à ces dernières années, il était possible pour les enquêtrices et enquêteurs de l'enquête Emploi de coder les "Refus" et "Ne Sait Pas" à la question du salaire avec les raccourcis "9999999" et "9999998". De simples fautes de frappe ont parfois introduit par erreur des salaires très élevés. Nous enlevons tous les salaires du type 999999, 99999, 999998 et 99998 euros (ou francs). Le RN, tel que nous l'avons entraîné, n'était pas capable de repérer et corriger de lui-même ces erreurs qui ont cependant un impact très fort sur les estimations.

2.3.2 Sélection des échantillons

L'échantillon test, d'environ 10%, est composé de deux parties : l'ensemble du premier trimestre 2018, et l'ensemble des individus nés en mars. L'échantillon n'est donc pas tiré aléatoirement, mais en sacrifiant légèrement la représentativité, on s'assure d'une part un échantillonnage très facile à reproduire, et d'autre part un test robuste au problème de dérive de la distribution dans le temps. En testant sur le premier trimestre 2018, on vérifie en effet qu'un prédicteur entraîné sur longue période resterait pertinent sur le temps présent s'il devait être mis en production. On peut ainsi le mettre en situation dans des conditions proches de celles dans lesquelles les performances du prédicteur de SALRED sont mesurées [2]. Enfin, lors de chaque entraînement, les lignes restantes sont aléatoirement divisées en 995000 lignes d'entraînement (90%) et 110000 lignes de validation (10%). Cet échantillon de validation permet de guider l'optimisation des hyper-paramètres du RN.

2.3.3 Réduire la dimension d'une nomenclature par plongement lexical : l'exemple des professions

Certaines variables catégorielles de l'enquête emploi ont de très nombreuses modalités, en particulier celles codées selon les nomenclatures françaises et européennes des professions, des secteurs d'activité ou des diplômes. L'encodage *one-hot* (qui crée des variables indicatrices pour chaque modalité) de ces variables de grande cardinalité génère autant de colonnes que de modalités (plusieurs centaines) et rend le fichier d'entrées peu maniable pour des questions de mémoire et de temps de calcul. Plus fondamentalement, il est difficile pour le RN de s'entraîner sur ces nombreuses variables indicatrices dont beaucoup signalent des modalités rares (ce sont essentiellement des colonnes de 0). Afin de réduire la dimension de cet ensemble de *features*, on va chercher à représenter chaque modalité fine par un vecteur de nombres réels de faible dimension. On va donc projeter dans un espace vectoriel de plus faible dimension ces variables en tenant compte de la proximité sémantique entre les valeurs de ces variables. Cette approche, classique en analyse textuelle, s'appelle *plongement lexical*, **vectorisation des mots**, ou encore *word embeddings* en anglais. La proximité sémantique entre deux modalités, deux « mots », peut s'obtenir de différentes manières, selon les éléments de contexte dont on dispose. Pour la suite de l'exercice on va faire l'hypothèse que sont proches les modalités d'un même individu à différents moments de sa vie (connues grâce aux questions rétrospectives et aux interrogations interrogation successives), les modalités des membres d'un même ménage, d'une même famille (ascendants et descendants), d'une même grappe de logements, etc.. Cette hypothèse est forte et est retenue ici pour les besoins de l'exercice. Plus généralement, une telle étape d'*embeddings* dépend de l'objectif poursuivi et d'éventuelles considérations théoriques et éthiques.

Parmi d'autres algorithmes similaires, *word2vec* a beaucoup fait progresser l'analyse du langage en permettant d'encoder les mots d'un vocabulaire à partir d'un corpus de phrases, voir Mikolov, Chen, Corrado, et Dean (2013). C'est cet algorithme que l'on va implémenter ici. On fait l'hypothèse que des mots qui partagent souvent un même contexte, c'est-à-dire qui apparaissent fréquemment dans des phrases similaires, ont un sens proche. Il s'agit d'apprentissage non supervisé ou auto-supervisé, chaque mot est d'abord traduit en un (très grand) vecteur des co-occurences avec tout le reste du vocabulaire, puis un réseau peu profond avec une unique couche cachée de faible dimension est entraîné à prédire, par exemple, un mot donné à partir des mots voisins dans une phrase, ou au contraire à prédire un contexte à partir d'un mot. A l'issue

2. Cependant, les répondants à l'enquête emploi sont censés déclarer leur salaire deux fois en première et dernière interrogations. Au premier trimestre 2018, environ la moitié des salaires viennent de personnes qui ont déjà pu déclarer un salaire dans la base d'entraînement. Lorsqu'on contrôle cette éventuelle contamination en testant uniquement sur les répondants en première interrogation au premier trimestre 2018, les performances prédictives des RN se maintiennent (elles sont même un peu plus élevées). Il semble donc que les RN ne surapprennent pas en identifiant les individus.

de l'entraînement, la couche cachée fournit un encodage (dont la faible dimension d est fixée à l'avance) de la totalité du vocabulaire. Contrairement aux vecteurs de co-occurences, qui sont de très grande dimension mais essentiellement constitués de zéros (*sparse*), le vecteur encodé est dense, il contient une grande partie de l'information, concentrée dans une petite dimension. L'algorithme est rapide, léger et efficace. Ce principe d'*embedding* est maintenant utilisé pour des problèmes spécialisés comme la vectorisation de concepts médicaux (Beam, Kompa, Schmaltz, Fried, Weber, Palmer, Shi, Cai, et Kohane (2020) ou, sur données françaises, Doutreligne, Leduc, Nguyen, et Vuagnat (2020)). On peut se procurer des modèles pré-entraînés sur de larges corpus (par exemple, les articles de Wikipedia), ou bien entraîner son propre modèle, ou ré-entraîner un modèle préexistant (*transfert learning*) lorsque le problème est très spécifique.

Nous appliquons ici *word2vec* à la vectorisation des nomenclatures statistiques, un problème spécifique sur lequel il est difficile de mobiliser des modèles génériques préentraînés. Nous entraînons donc un nouveau modèle.

En pratique, chaque modalité (par exemple la modalité « 211d : Artisans plombiers, chauffagistes »de la variable P :profession de l'enquête Emploi) est considérée comme un « mot »dans un vocabulaire. Les phrases sont construites comme des suites de tels « mots »à partir des relations suivantes : cohabitation dans un ménage, transition professionnelle d'un individu au cours des 6 interrogations d'un ménage, professions actuelle et ancienne, couple des parents. Si, par exemple, une personne enquêtée est barman lors de sa première interrogation, puis militaire professionnel lors de sa seconde interrogation, la phrase suivante (constituée de deux mots séparés par un point-virgule) :

`[561a : Serveurs, commis de restaurant, garçons (bar, brasserie, café ou restaurant) ; 532c: Hommes du rang (sauf pompiers militaires)]`

apparaîtra dans le corpus. De plus, s'il a déclaré dans l'enquête que son père était gendarme et que sa mère tenait un petit café, la phrase :

`[532a: Gendarmes (de grade inférieur à adjudant) ; 224a: Exploitants de petit restaurant, café-restaurant, de 0 à 2 salariés]`

sera également dans le corpus. Si on prend en compte les liens entre ascendants et descendants, on ajoutera dans le corpus les quatre phrases reprenant chacune des combinaisons entre les professions du père, de la mère, et les deux professions du fils. On fait donc l'hypothèse, que les professions qui apparaissent dans des contextes proches (même ménage, couples, différents emplois d'un même individu au cours du temps, etc.) sont des professions proches. Dans l'exemple des profession, un encodage de dimension $d = 4$ semble capturer une information substantielle (figure 4).

Il est difficile de juger directement de la qualité de l'encodage réalisé pour une telle vectorisation. Cette démarche relevant de l'apprentissage non supervisé, à la manière d'une classification, ou de la hiérarchie d'une nomenclature, il est difficile de prouver quantitativement son intérêt, faute de point de comparaison [3]. C'est à l'usage qu'on éprouve son efficacité. On notera par la suite que l'utilisation des nomenclatures vectorisées améliore les performances du RN par rapport à l'omission totale de ces variables de grande cardinalité et même par rapport à l'intégration d'un niveau plus agrégé des nomenclatures (par exemple, les professions regroupées en 32 postes dans la variable CSE), ce qui tend à valider *a posteriori* la démarche.

De manière plus descriptive, on peut cependant regarder si les résultats semblent conformes à l'intuition. Pour l'encodage des professions, on visualise assez facilement dans différentes projections des dimensions intuitives de l'espace social (par exemple sur le graphique 4), telles que l'opposition salariat / indépendance (axe vertical sur le graphique), le statut social associé à une profession (axe horizontal), ou encore, sur d'autres axes de projection, l'opposition privé / public, les professions plus ou moins féminisées, ou même la proximité, par exemple, de professions

3. Dans son article initial, l'algorithme *word2vec* a rencontré la même difficulté : il est délicat de trouver ou de construire des métriques de performance, des corpus et des *benchmarks* adaptés pour cette procédure non supervisée.

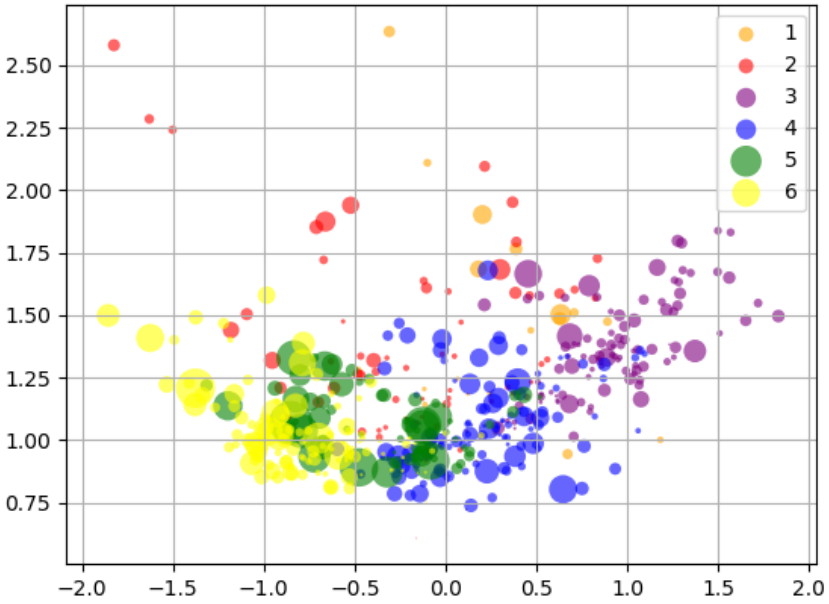


Figure 4 – Professions projetées sur deux axes d'un encodage

Note : Encodage en 4 dimensions projeté sur deux dimensions choisies arbitrairement. Les couleurs correspondent aux groupes sociaux (1. Agriculteurs 2. Artisans, commerçants et chefs d'entreprise 3. Cadres et professions intellectuelles supérieures 4. Professions intermédiaires 5. Employés 6. Ouvriers) et la surface des cercles est proportionnelle aux effectifs.

typiques des littoraux (marins, militaires, restaurateurs...). Certaines proximités sembleront évidentes et banales, certaines de ces dimensions de l'espace social sont déjà constitutives de la construction des nomenclatures et sont donc intégrées dans les niveaux les plus agrégés de la hiérarchie. Cependant, la vectorisation permet une description à la fois automatique et plus fine de la position de chaque profession dans cet espace. Elle contient davantage d'information. La méthode peut facilement être généralisée à d'autres enquêtes ou données administratives et d'autres types de variables (elle a par exemple été testée sur un fichier de prénoms). L'estimation elle-même est réalisée grâce à la fonction `Word2Vec` du package « gensim [4] ».

De la même manière qu'on peut utiliser pour des tâches spécialisées diverses des *embeddings* estimés sur un corpus généraliste comme Wikipedia (*transfert learning*), l'encodage d'une nomenclature à partir de l'enquête Emploi peut être sauvegardé et réutilisé à chaque fois qu'on souhaite utiliser un encodage de cette même nomenclature, quel que soit le fichier, à condition cependant que le contexte utilisé pour entraîner l'encodeur soit proche de la proximité recherchée dans le cadre de la nouvelle utilisation.

Suite à cette vectorisation, chaque variable vectorisée est traduite par huit colonnes. A la suite de toutes les étapes de préparation des données (*one-hot encoding* des variables qualitatives, imputation et indicatrices pour les valeurs manquantes, et vectorisation des nomenclatures), l'input final du RN est un vecteur de dimension 215.

2.4 L'imputation par RN : une meilleure performance prédictive au prix d'une mise en oeuvre plus complexe

2.4.1 Un réseau de neurones simple

On peut maintenant chercher à prédire le salaire à l'aide d'un réseau perceptron multicouche. La variable d'intérêt utilisée pour l'entraînement est le *logarithme* du salaire mensuel, l'utilisation du logarithme améliorant sensiblement les performances, comme souvent dans l'économétrie du salaire. Le type de réseau est un perceptron multicouche ("*feed-forward neural network*"), c'est-à-dire l'architecture standard détaillée au chapitre précédent, et toutes les couches sont denses (tous les neurones d'une couche sont connectés aux neurones de la couche suivante).

Dans l'échantillon d'entraînement, 10% des observations sont extraites à chaque entraînement comme échantillon de validation, et ont servi à sélectionner les hyperparamètres. Le travail préparatoire a été mené sur un poste individuel, sans parallélisation, avec un temps de calcul de l'ordre de la minute pour chaque époque (ou *epoch*). L'exploration de l'espace des hyperparamètres a donc été faite "à la main" sur l'échantillon de validation tiré aléatoirement lors de chaque entraînement. Avec des ressources plus importantes, cette exploration peut être automatisée et optimisée par validation croisée.

Le RN finalement adopté a sept couches cachées [500, (dropout), 100, 50, 20, 8, 4, 2] et une couche de *dropout* avec un taux de 0.5 (une technique de régularisation consistant à supprimer temporairement et aléatoirement, une certaine proportion de neurones à chaque *batch* de données pendant l'apprentissage). Nous utilisons la fonction d'activation SELU : $f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1) & si\ x \leq 0 \\ x & si\ x > 0 \end{cases}$. Elle aide à stabiliser l'entraînement (Klambauer, Unterthiner, Mayr, et Hochreiter, 2017). La dernière couche est linéaire. La fonction de perte est l'erreur quadratique moyenne, l'optimiseur est Adam (Kingma et Ba, 2014). L'entraînement se fait sur 100 époques, avec une taille de *batch* de 200. Sous Keras, le modèle peut se décrire simplement couche par couche.

```
from keras.layers import Input, Dense, AlphaDropout
from keras.models import Model
```

4. https ://radimrehurek.com/gensim/

```
inputs = Input(shape=(215,))
x = Dense(500, activation='selu', kernel_initializer='lecun_uniform',
          bias_initializer='lecun_uniform')(inputs)
x = AlphaDropout(0.5)(x)
x = Dense(100, activation='selu', kernel_initializer='lecun_uniform',
          bias_initializer='lecun_uniform')(inputs)
x = Dense(50, activation='selu', kernel_initializer='lecun_uniform',
          bias_initializer='lecun_uniform')(x)
x = Dense(20, activation='selu', kernel_initializer='lecun_uniform',
          bias_initializer='lecun_uniform')(x)
x = Dense(8, activation='selu', kernel_initializer='lecun_uniform',
          bias_initializer='lecun_uniform')(x)
x = Dense(4, activation='selu', kernel_initializer='lecun_uniform',
          bias_initializer='lecun_uniform')(x)
x = Dense(2, activation='selu', kernel_initializer='lecun_uniform',
          bias_initializer='lecun_uniform')(x)
predictions = Dense(1, activation='linear')(x)

model= Model(inputs=inputs, outputs=predictions)
model.compile(optimizer='adagrad', loss='mean_squared_error',
              metrics=['mse', 'mae'])

history = model.fit(x_train_scaled, ylog_train, epochs=100,
                    batch_size=100, validation_split=0.1,
                    shuffle=True)  # starts training
```

Dans une version plus élaborée, cette construction du RN peut être intégrée au sein d'une fonction python qui prend en entrée un dictionnaire des hyperparamètres. Cette méthode facilite ensuite l'optimisation de ces hyperparamètres :

```
def create_model(input_size=500, hparam_dict=init_param):
    optimizer = hparam_dict['optimizer']
    activation = hparam_dict['activation']
    dropout = hparam_dict['dropout']
    L1_regularization = hparam_dict['L1_regularization']
    L2_regularization = hparam_dict['L2_regularization']
    metrics = hparam_dict['metrics']

    from keras.layers import Input, Dense, AlphaDropout, BatchNormalization
    from keras.models import Model

    inputs = Input(shape=(input_size,))
    from keras import regularizers
    x = BatchNormalization()(inputs)
    for i in range(hparam_dict['number_of_layers']):
      if hparam_dict['layers_size'][i] == 0:
          x = AlphaDropout(dropout)(x)
      else:
        n = hparam_dict['layers_size'][i]
        x = Dense(n, activation=activation,
                  kernel_initializer="lecun_uniform",
```

```
                bias_initializer='lecun_uniform',
                kernel_regularizer=regularizers.l1_l2(l1=L1_regularization,
                                                      l2=L2_regularization))(x)

    predictions = Dense(1, activation='linear')(x)

    model1 = Model(inputs=inputs, outputs=predictions)
    model1.compile(optimizer=optimizer,
                   loss='mse',
                   metrics=metrics)
    return model1
```

2.4.2 Stabilité des performances

L'entraînement des RN est stochastique, et la fonction de perte connaît a priori des minima locaux sur l'espace des paramètres. Autrement dit, il n'est jamais garanti que l'entraînement d'un RN aboutisse à la meilleure performance possible, ou même à une performance satisfaisante, et la stabilité, la fiabilité de l'entraînement est l'un des enjeux importants des RN. Dans notre configuration, les entraînements ont été particulièrement instables. Après quelques explorations, la correction la plus simple a consisté à sélectionner dès la fin de la première époque d'entraînement uniquement les modèles ayant atteint une performance raisonnable. Même ainsi la perte sur l'échantillon d'entraînement et surtout sur l'échantillon de validation reste instable au cours de l'entraînement ou entre deux modèles. La figure 5 illustre cette instabilité pour un modèle, entraîné pour 100 époques après avoir été sélectionné pour sa bonne performance après la première époque d'entraînement. La perte calculée sur l'échantillon d'entraînement est raisonnablement décroissante, mais la perte calculée sur l'échantillon de validation connaît des fluctuations importantes et sans tendance claire pendant les 60 premières époques. L'échantillon de validation restant le même au cours de l'entraînement d'un modèle donné, ces fluctuations sont entièrement dues aux évolutions du RN, par entraînement, d'une époque à l'autre.

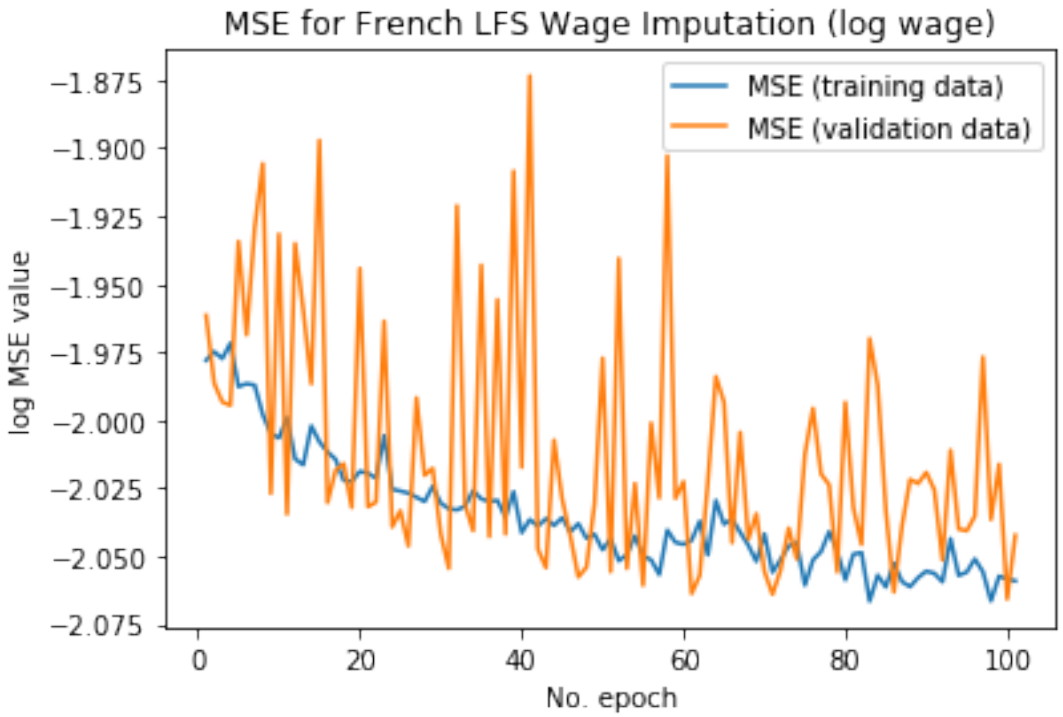


Figure 5 – entraînements de RN

Notes : log perte sur les échantillons d'entraînement et de validation selon le nombre d'époques d'entraînement. Une différence de 0,1 en log MSE correspond à environ 3 points de R^2 sur le log-salaire. Le R^2 final de ce prédicteur sur l'échantillon test du premier trimestre 2018 est 0,727

Cette instabilité tient notamment à la nature du problème : la difficulté à prédire aisément le salaire à partir des inputs, et la distribution particulière de cette variable. La distribution des salaires est non seulement contrainte et concentrée à gauche (salaires positifs et souvent supérieurs au smic) mais très étalée à droite. Les très hauts salaires sont rares mais peuvent être très élevés, et même la transformation logarithmique ne suffit pas à diminuer suffisamment leur grande influence dans la variance totale. Tous les modèles, RN ou non, peinent à prédire les très hauts salaires, et il y a une instabilité de la mesure des performances selon la présence ou non de ces hauts salaires dans les échantillons de validation ou de test, et la performance du modèle sur ce très petit nombre d'observations extrêmes.

Quelques éléments heuristiques sur la stabilité de l'apprentissage se dégagent cependant de nos expériences. Les grands batchs sont plus stables, et si les petits batchs permettent d'apprendre en moins d'époques, alterner les taille de batch semble intéressant pour bénéficier des deux avantages. La figure 6 illustre ce point. Elle diffère également de la précédente par une réduction du paramètre du dropout : le surapprentissage est plus fort, mais l'apprentissage est plus stable.

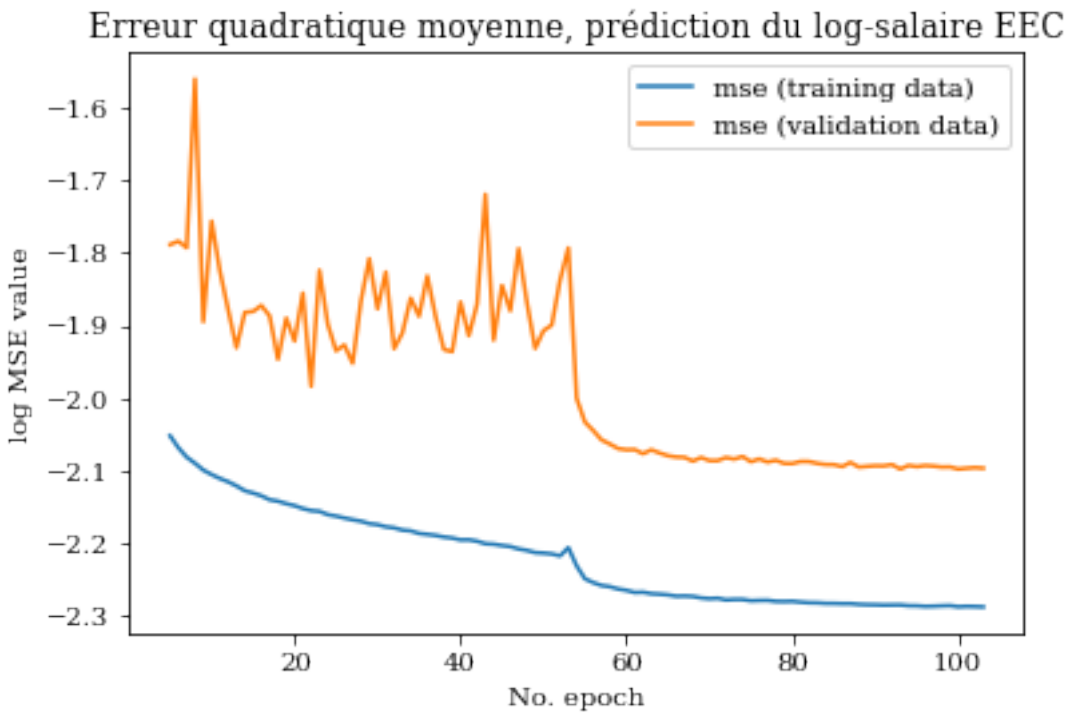


Figure 6 – entraînements de RN

Entraînement avec changement de taille de batch : 64 puis 10000 pour les 4 premières époques (non montrées), 64 pour les époques 5 à 54, 10000 pour les époques 55 à 104

Nous avons également chercher à construire des fonction de perte adaptées au problème, par exemple en pénalisant à la fois la distance au log-salaire et la distance au salaire, ou bien de manière à introduire la prédiction par tranche et une pénalisation de l'erreur de tranche. Ces modèles de prédiction mixtes sont encore plus sensible, et il est facile de générer une fonction de perte particulièrement instable.

Une autre piste pour limiter ces problèmes de stabilité consiste à fournir directement au RN l'output d'un modèle linéaire de salaire, ou, encore plus simple, à donner un double input à la dernière couche du réseau : d'une part, la sortie des couches précédentes, et d'autre part à nouveau les données en entrée. On permet ainsi au RN de retrouver rapidement un modèle linéaire efficace (l'avantage des RN sans couches cachées) sans sacrifier sa capacité à apprendre des non-linéarités compliquées (l'avantage d'un RN profond).

2.4.3 Comparaison des performances

Pour juger de la performance du RN d'imputation, on la compare à la procédure d'imputation en production pour SALRED, puis à la performance prédictive de différentes versions de l'équation de Mincer.

On peut tester la précision de la méthode actuelle d'imputation de SALRED sur des observations dont on connaît le salaire, mais que l'on met à blanc (ni salaire, ni tranche) avant de faire tourner le code d'imputation. En pratique, on partitionne aléatoirement en cinq l'échantillon total (entraînement et test), on met à blanc un cinquième des salaires connus, et on fait tourner le code d'imputation de SALRED. On répète l'opération pour chacun des quatre autres cinquièmes [5].

5. Le code de SALRED fonctionne par trimestre de données, il n'est donc pas possible de l'entraîner sur l'échantillon d'entraînement 1993-2017, puis d'imputer la totalité du premier trimestre 2018

Calculé sur le premier trimestre 2018, le R^2 de la régression du log-salaire observé sur le log-salaire prédit n'est que de $0,43$. Calculé sur les salaires (et non plus les log-salaires) il descend à $0,19$ [6]. La performance purement prédictive est bien entendu diminuée par la nature aléatoire de la procédure.

Nous utilisons également l'équation de Mincer comme référence de prédicteur déterministe, sans alea donc, contrairement à SALRED. Estimé sur l'échantillon test du premier trimestre 2018, le R^2 du salaire observé atteint $0,64$ pour une équation de Mincer incluant les variables de SALRED ainsi que le logarithme des heures travaillées et le carré de l'expérience potentielle. En revanche, l'ajout de toutes les variables utilisées par le RN (y compris les nomenclatures vectorisées) n'améliore pas notablement les performances de l'équation de type Mincer : le R^2 atteint 0,65.

Le RN atteint un R^2 de 0,68 sur cet échantillon, 0.72 sur l'échantillon des personnes nées en mars. L'écart entre Mincer et le réseau de neurone peut suggérer que le RN capte des non-linéarités ou interactions que le modèle linéaire supposé dans l'équation de Mincer n'est pas capable de prendre en compte. Calculé sur le salaire et non plus sur le salaire réel, ces R^2 sont respectivement de $0,19$ pour la méthode d'imputation actuelle de SALRED, $0,29$ pour l'équation de Mincer, et $0,33$ pour le RN (figure 7).

Table 1 – R^2 des modèles sur différents échantillons tests

	Echantillon test			
	log-salaire		salaire	
Modèle :	2018	Nés en mars	2018	Nés en mars
RN	68,3	71,9	33,0	27,6
OLS sur l'input RN	64,8	62,8	29,1	23,4
Mincer enrichi	63,4	62,9	28,7	27,6
Mincer secteur détaillé	63,7	63,3	29,1	27,9
Mincer basique	51,9	45,0	21,5	20,5
SALRED	43,2		18,6	

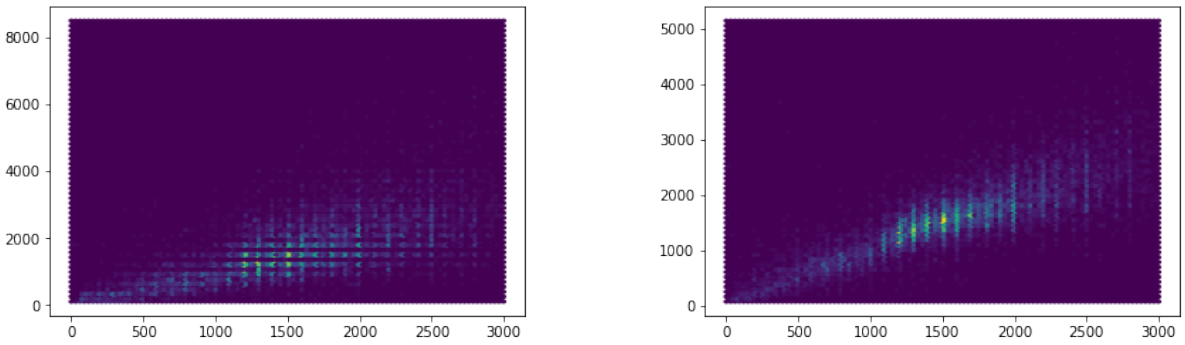
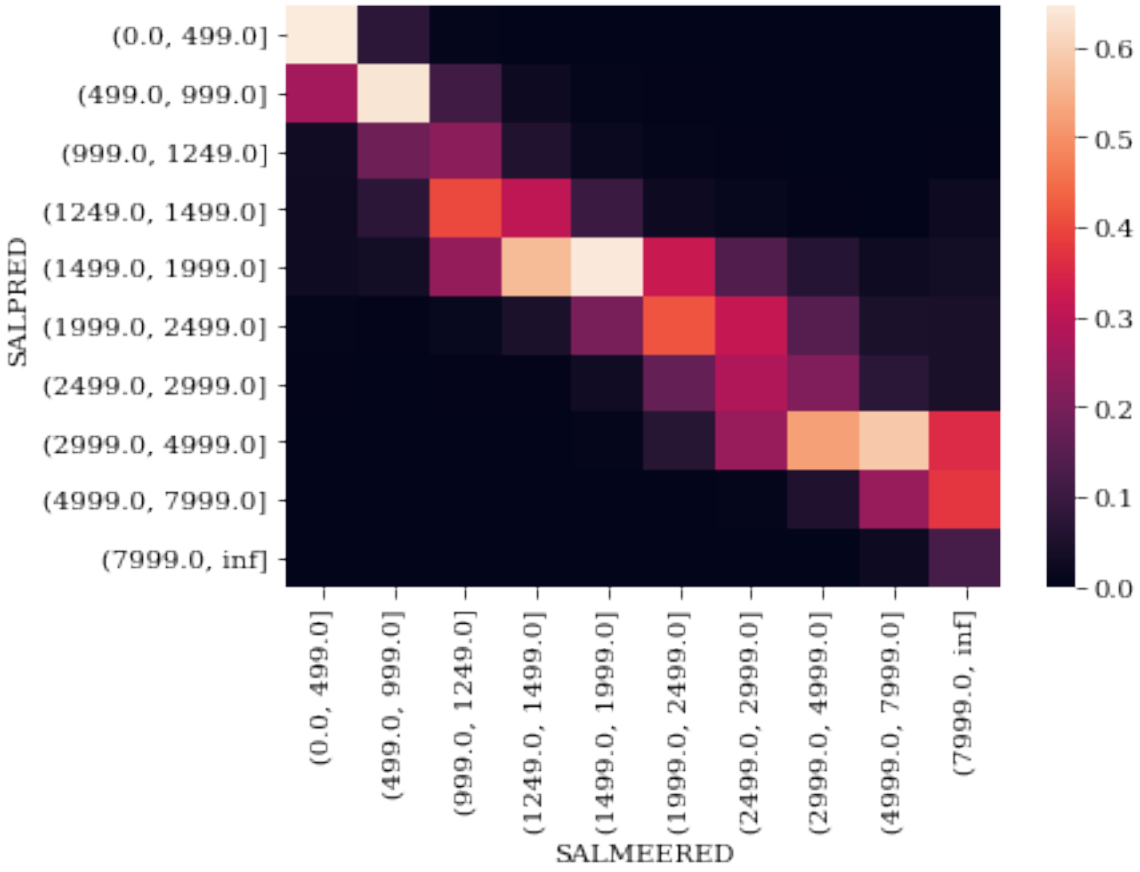


Figure 7 – Salaire déclaré et prédit par imputation par SALRED et RN
Les salaires prédits par imputation sont en ordonnée, les salaires déclarés par les enquêtés sont en abscisse. Pour l'imputation par SALRED, à gauche, le nuage de point est plus diffus que pour l'imputation par RN, à droite. On observe des lignes verticales correspondant au comportement d'arrondis des personnes enquêtées, et, pour SALRED, des lignes horizontales peut-être liées à la méthode de prise en compte des réponses par tranche. Dans les deux cas, les hauts salaires tendent à être sous-estimés par les prédicteurs.

6. on réserve parfois la notion de R^2 au contexte de la régression. Nous l'utilisons ici plus généralement pour désigner la réduction de l'erreur quadratique moyenne permise par le prédicteur. Dans cet usage, le R^2 peut être négatif si le prédicteur est moins bon que la prédiction par la moyenne.

Figure 8 – Distribution du salaire prédit par tranche, par tranche de salaire déclaré



Finalement sur cet exercice, l'écart de performance prédictive entre le RN et un modèle quasi-linéaire inspiré par la théorie économique est limité mais notable. Le RN réduit de 10 à 20% l'erreur quadratique.

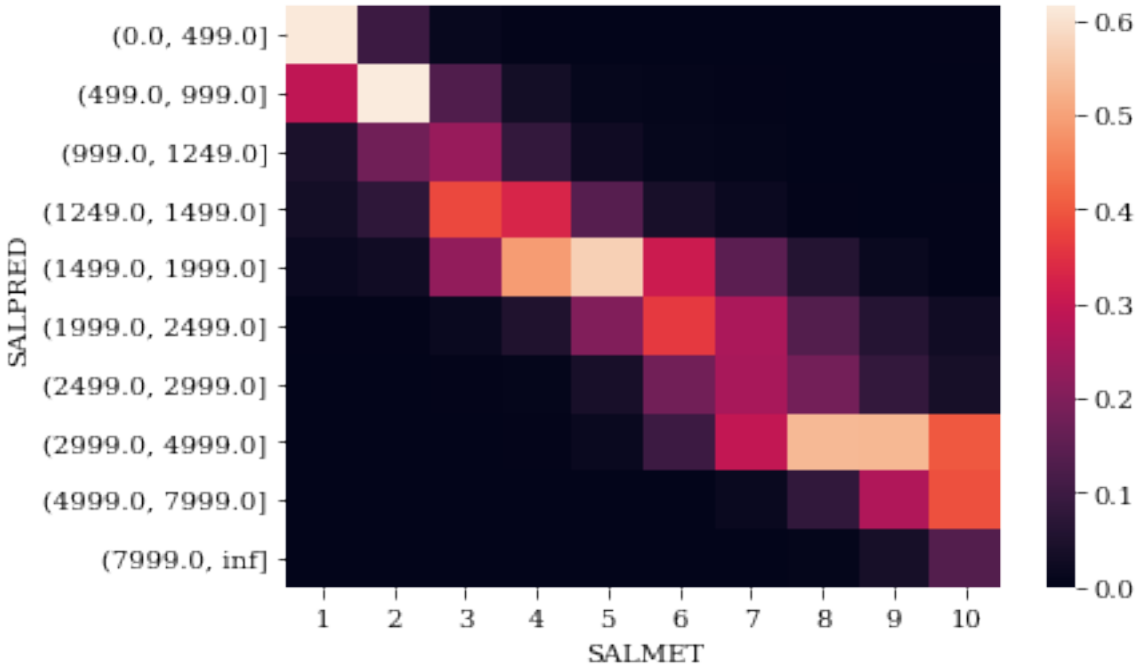
2.4.4 Une réduction du biais de non-réponse

Les questions sur le salaire dans l'enquête Emploi fournissent une opportunité rare d'explorer le biais de non-réponse. Celui-ci est par nature inconnu, et on ne peut pas en général être certain qu'un bon prédicteur d'une variable pour les répondants, nécessairement entraîné sur valeurs observées des répondants, est également un bon prédicteur pour les non-répondants. Mais dans le cas du salaire dans l'enquête Emploi, les non-répondants sont relancés, et on leur propose une réponse par tranche, ce qu'ils acceptent de donner dans trois cas sur quatre. On exploite cette information additionnelle pour etudier le comportement des prédicteurs face à des cas de véritable non-réponse.

On peut classer les salaires observés (SALMEERED) et prédits (SALPRED) selon les tranches de l'enquête. On peut faire de même en cas de non-réponse sur le salaire, lorsque les enquêtés ont tout de même donné leur tranche de salaire (SALMET). Les figures 8 et 9 illustrent les distributions des tranches prédites par le RN, pour les tranches déclarées respectivement par les personnes qui ont donné leur salaire en clair et par les non-répondants pour le salaire en clair. Les deux ensembles de distributions apparaissent très similaires, en particulier pour les tranches élevées, et montrent le même phénomène de sous-estimation des salaires élevés par le prédicteur.

En revanche, lorsqu'on fait la même analyse avec un prédicteur linéaire (un modèle de Mincer enrichi), la sous-estimation s'accentue sur la population des non-répondants. Le phénomène

Figure 9 – Distribution du salaire prédit par tranche, par tranche déclarée, pour les non-répondants sur le salaire en clair



apparaît nettement dans la figure 10 qui représente la différence entre les distributions des prédictions par RN et par modèle linéaire. Parmi les non-répondants dont on sait qu'ils gagnent plus de 8000 euros par mois, les deux modèles se trompent en prédisant fréquemment des salaires dans la tranche 3000 à 5000, mais la fréquence de cette erreur est plus grande de 30 points pour le modèle de Mincer. Autrement dit, par rapport au modèle de Mincer, le RN réduit nettement le biais de non-réponse parmi les hauts-salaires.

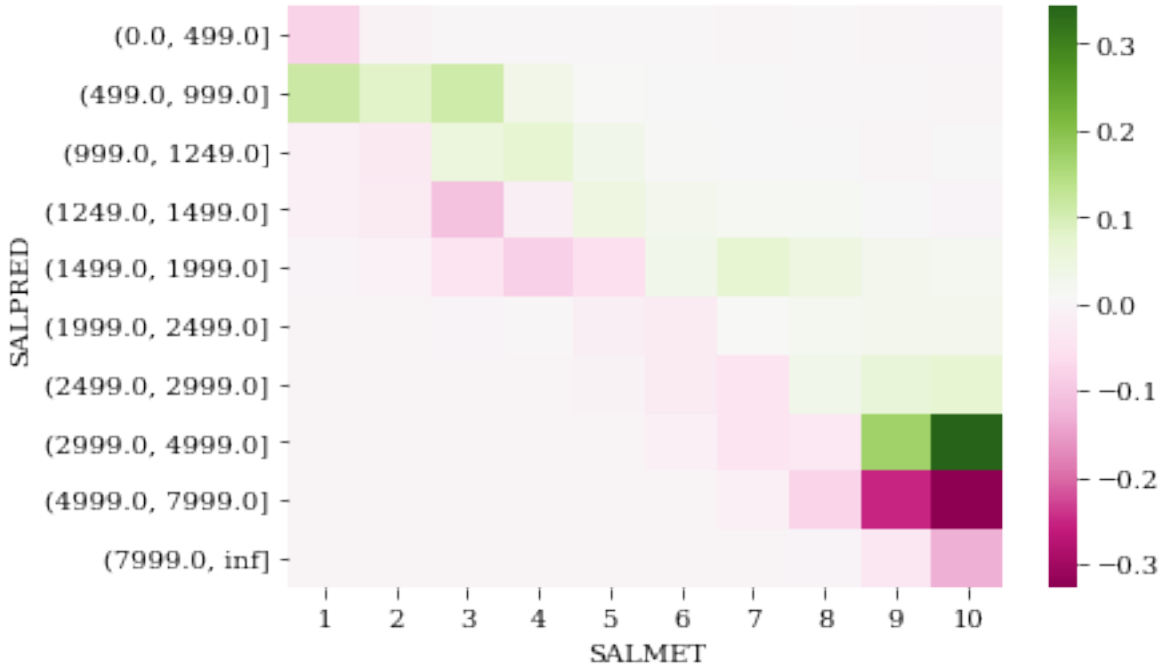
2.5 Des prolongements possibles

Ce travail peut connaître plusieurs prolongements pour améliorer la qualité de l'imputation selon les besoins correspondant aux usages qui seraient fait des données imputées. Un premier prolongement consiste à intégrer l'information fournie par les réponses par tranche. Il est facile d'intégrer cette variable parmi les inputs du RN, et, sans surprise, elle accroît (légèrement) la capacité prédictive du modèle. En sortie, on peut également souhaiter que la valeur imputée appartienne à la tranche de salaire, lorsqu'elle est connue, et ce problème est plus difficile. Les RN présentés jusqu'ici ne prédisent pas systématiquement la tranche correcte. Il est possible d'inciter ou de contraindre un RN à viser la bonne tranche, par exemple avec un modèle à *multiple outputs* entraîné à prédire à la fois le salaire et sa tranche, ou bien avec une fonction de perte construite sur mesure qui prend des valeurs élevées lorsque le salaire imputé sort de la tranche connue. Mais ces contraintes ont en retour un impact sur la qualité de la prédiction du salaire (elles peuvent par exemple encourager le RN à prédire des valeurs éloignées des limites des tranches).

Un autre prolongement consiste à prédire non seulement une valeur, mais les paramètres d'une distribution, reflétant davantage à la fois la variance des données d'origine et la plus ou moins grande incertitude associée par le modèle à chacune de ses prédictions. On entre alors dans le domaine des RN variationnels, qui permettraient d'accomplir la même tâche que l'imputation aléatoire (telle qu'elle est mise en oeuvre actuellement dans le modèle d'imputation de SALRED).

D'autres pistes consistent à développer des idées déjà évoquées au début de ce chapitre. On

Figure 10 – Différence OLS - RN des distribution des prédictions par tranche, pour les non-répondants



peut utiliser les RN dans une procédure d'imputation doublement robuste, en exploitant également l'information communiquée par le comportement de non-réponse lui-même. On pourrait attendre d'une telle démarche une meilleure réduction du biais de non-réponse. Une autre piste relativement facile à mettre en oeuvre est d'utiliser le salaire prédit par un RN non plus pour imputer directement, mais pour déterminer les donneurs potentiels dans une méthode par plus proches voisins ou par *hot deck*. Ces méthodes par donneurs préservent davantage la variance des données d'origine, ainsi que des caractéristiques difficiles à simuler comme les comportements d'arrondis. Il est même possible de combiner double robustesse et imputation par hot-deck, comme dans Boistard, Chauvet, et Haziza (2016) : au sein de cellules d'imputation déterminées grâce à un modèle d'imputation IM, le donneur j est tiré au hasard avec une probabilité proportionnelle à $(1-p_j)/p_j$, avec p_j sa probabilité d'être répondant, déterminée par un modèle de non-réponse NM. Il est aisé d'exploiter des RN pour les modèles IM et NM.

Enfin, il est possible d'envisager l'imputation au sein d'un système plus général de correction ou de débruitage des données. Les auto-encodeurs sont des réseaux de neurones qui peuvent être notamment entraînés pour ces tâches. On peut confronter ces auto-encodeurs à un RN « adversaire », un *discriminateur* entraîné à distinguer les données originales des données imputées ou corrigées. Une telle démarche est susceptible en théorie de produire de manière autonome des imputations très réalistes (reproduisant dans notre exemple la cohérence entre valeur et tranche, les comportements d'arrondis, la variance, etc.) et elle est aujourd'hui très appliquée dans le traitement d'image, mais il est difficile de maintenir un entraînement équilibré entre le générateur et le discriminateur.

Références

Babet, D. (2020) : "Wage Imputation with Deep Learning in the French Labor Force Survey," in *UNECE workshop on statistical data editing, conference of european statisticians.*

Babet, D., Q. Deltour, T. Faria, et S. Himpens (2022) : "Les réseaux de neurones. Méthodes et applications à la statistique publique," à paraître.

Bang, H., et J. M. Robins (2005) : "Doubly robust estimation in missing data and causal inference models," *Biometrics*, 61(4), 962–973.

Beam, A. L., B. Kompa, A. Schmaltz, I. Fried, G. Weber, N. Palmer, X. Shi, T. Cai, et I. S. Kohane (2020) : "Clinical concept embeddings learned from massive sources of multimodal medical data," .

Boelaert, J., et É. Ollion (2018) : "The Great Regression," *Revue française de sociologie*, 59(3), 475–506.

Boistard, H., G. Chauvet, et D. Haziza (2016) : "Doubly robust inference for the distribution function in the presence of missing survey data," *Scandinavian Journal of Statistics*, 43(3), 683–699.

Breiman, L. (2001) : "Statistical Modeling : The Two Cultures," *Statistical Science, Vol. 16, No. 3, 199–231.*

Chen, S., et D. Haziza (2017) : "Multiply robust imputation procedures for the treatment of item nonresponse in surveys," *Biometrika*, 104(2), 439–453.

——— (2019) : "Recent developments in dealing with item non-response in surveys : a critical review," *International Statistical Review*, 87, S192–S218.

Doutreligne, M., A. Leduc, D.-P. Nguyen, et A. Vuagnat (2020) : "Snds2vec, représentations continues pour les concepts médicaux du Système national des données de santé," *Revue d'Épidémiologie et de Santé Publique*, 68, S35.

Haziza, D., J.-F. Beaumont, et al. (2017) : "Construction of weights in surveys : A review," *Statistical Science*, 32(2), 206–226.

Josse, J., N. Prost, E. Scornet, et G. Varoquaux (2019) : "On the consistency of supervised learning with missing values," *arXiv preprint arXiv :1902.06931.*

Kingma, D. P., et J. Ba (2014) : "Adam : A method for stochastic optimization," *arXiv preprint arXiv :1412.6980.*

Klambauer, G., T. Unterthiner, A. Mayr, et S. Hochreiter (2017) : "Self-normalizing neural networks," *arXiv preprint arXiv :1706.02515.*

Le Cun, Y. (1985) : "Une procédure d'apprentissage pour réseau à seuil assymétrique," .

Lemieux, T. (2006) : "The "Mincer equation" thirty years after schooling, experience, and earnings," in *Jacob Mincer a pioneer of modern labor economics*, pp. 127–145. Springer.

McCulloch, W., et W. Pitts (1943) : "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, pp. 115–133.

Mikolov, T., K. Chen, G. Corrado, et J. Dean (2013) : "Efficient estimation of word representations in vector space," *arXiv preprint arXiv :1301.3781.*

Mikolov, T., I. Sutskever, K. Chen, G. Corrado, et J. Dean (2013) : "Distributed representations of words and phrases and their compositionality," *arXiv preprint arXiv :1310.4546.*

Mincer, J. (1974) : “Schooling, Experience, and Earnings. Human Behavior & Social Institutions No. 2.,” .

Mullainathan, S., et J. Spiess (2017) : “Machine learning : an applied econometric approach,” *Journal of Economic Perspectives*, 31(2), 87–106.

Picart, Claude et Babet, D. (2020) : in *Insee Référence, Emploi et revenus des indépendants*Insee.

Rosenblatt, F. (1958) : “The perceptron : a probabilistic model for information storage and organisation in the brain,” *Psychological Review*, 65(6).

Rubin, D. B. (1976) : “Inference and missing data,” *Biometrika*, 63(3), 581–592.

Rumelhart, D. E., H. G. E., et W. R. J. (1986) : “Learning representations by backpropagating errors,” *Nature*, (323), 533–536.