Considération de l'Incertitude d'Imputation pour l'Apprentissage des Réseaux de Neurones

Thomas Ranvier, Haytham Elghazel, Emmanuel Coquery, Khalid Benabdeslem

Considération de l'Incertitude d'Imputation pour l'Apprentissage des Réseaux de Neurones

Thomas Ranvier, Haytham Elghazel, Emmanuel Coquery, Khalid Benabdeslem

Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205
43 bd du 11 Novembre 1918, 69622 Villeurbanne, France

{thomas.ranvier,haytham.elghazel,emmanuel.coquery,khalid.benabdeslem}@univ-lyon1.fr

Résumé

Dans cet article, nous nous intéressons à l'entraînement de réseaux de neurones dans un contexte de données incomplètes Nous cherchons à améliorer l'entraînement des réseaux de neurones en réduisant les biais potentiels pouvant survenir durant la phase d'apprentissage sur des jeux de données artificiellement complétés. Nous proposons deux frameworks d'imputation, S-HOT et M-HOT, pouvant être utilisés pour entraîner des réseaux neuronaux sur des données complétées de manière moins biaisée. Nous réalisons des expérimentations comparatives approfondies et évaluons statistiquement les résultats. Nous montrons que les frameworks proposés sont compétitifs et même plus performants que d'autres frameworks d'imputation existants.

Mots-clés

Imputation, Incertitude d'Imputation, Variance Inter, Optimisation, Réseaux de Neurones.

Abstract

In this paper we are interested in dealing with missing values when training a neural network.We focus on improving neural network training by reducing the potential biases that can occur during the training phase on artificially imputed datasets. We propose two new imputation frameworks, S-HOT and M-HOT, that can be used to train neural networks on completed data in a less biased way.We perform extensive comparative experiments and statistically assess the results.We show that our frameworks compete against and even outperform existing imputation frameworks.

Keywords

Data Imputation, Imputation Uncertainty, Between-variance, Optimization, Neural Networks.

1 Introduction

Deux types de variances apparaissent lorsque l'on impute plusieurs fois un jeu de données incomplet : la variance intra et la variance inter. La variance intra correspond à la variance au sein de chaque jeu de données complété. La variance inter correspond à la variance entre chaque jeu de données complété. Dans la réalité, il n'est généralement pas possible de connaître la variance intra, puisque la plupart des méthodes d'imputation estiment une valeur fixe à la place des valeurs manquantes sans fournir de mesure de probabilité. Cependant, la variance inter peut facilement être calculée entre deux jeux de données complétés. Dans la suite, nous appelons "incertitude d'imputation" cette variance inter.

Dans cet article, nous différencions explicitement les méthodes d'imputation, visant à imputer les valeurs manquantes dans un jeu de données, et les frameworks d'imputation tels que l'imputation simple (*SI*) ou l'imputation multiple (*MI*), qui reposent sur n'importe quelle méthode d'imputation pour traiter les données manquantes selon une méthodologie définie. Nos contributions sont des frameworks d'imputation, visant à entraîner des réseaux de neurones en tenant compte de l'incertitude d'imputation pouvant s'appuyer sur n'importe quelle méthode d'imputation. Il est à noter que nous ne sommes pas intéressés par la comparaison des performances des-dites méthodes d'imputation.

Il a été démontré que, lors de l'utilisation de modèles d'inférence forts tels que des réseaux de neurones, quasiment n'importe quelles imputations conduisent asymptotiquement à une prédiction optimale [5]. C'est probablement l'une des principales raisons pour lesquelles la prise en compte de l'incertitude d'imputation n'a jamais fait l'objet de recherches approfondies. Cependant, même si des modèles forts obtiennent de bons résultats dans de telles situations, ils n'en restent pas moins biaisés par l'incertitude d'imputation. Nous montrons que la prise en compte de cette incertitude pendant la phase d'apprentissage permet d'obtenir de meilleurs résultats de prédiction. Dans cet article, nous proposons deux frameworks d'imputation, *S-HOT* et *M-HOT*. Ils visent à entraîner des réseaux de neurones sur des jeux de données imputées en tenant compte de l'incertitude d'imputation de manière à réduire le biais naturel apparaissant lors de l'entraînement sur des jeux de données complétés. Ces frameworks sont destinés à être utilisés dans des situations différentes : *S-HOT* est adapté à l'entraînement d'un large réseau de neurones unique, *M-HOT* entraîne plusieurs modèles de manière ensembliste et permet d'obtenir d'excellents résultats de prédiction au prix d'un coût de calcul plus élevé. Nous menons des expé-

rimentations approfondies pour comparer nos deux frameworks avec les frameworks existants : l'imputation unique et l'imputation multiple. Nous effectuons une analyse statistique afin d'évaluer les résultats obtenus sur différents jeux de données. Nous montrons que les frameworks proposés rivalisent avec les frameworks d'imputation existants, voire les surpassent. Cet article est un premier pas vers la recherche de meilleurs moyens de gérer les valeurs manquantes dans le domaine de l'apprentissage automatique. Nous espérons qu'il suscitera l'intérêt d'autres chercheurs en apprentissage automatique sur cette question importante et pourtant largement négligée dans la littérature.

Dans la suite de cet article, nous présentons d'abord les travaux connexes sur les frameworks et méthodes d'imputation. Nous présentons et décrivons ensuite nos propositions dans la section 3. La section 4 présente nos expérimentations et les résultats obtenus. Enfin, nous concluons par un résumé de nos contributions.

2 Travaux Connexes

L'imputation simple (*SI*) est probablement le framework le plus couramment utilisé pour gérer les valeurs manquantes en pratique [10]. La méthodologie est simple : une méthode d'imputation est choisie et appliquée au jeu de données incomplet, permettant d'obtenir un jeu de données artificiellement complété où les valeurs manquantes ont été remplacées par de nouvelles valeurs, qui peut être utilisé et exploité comme tout autre jeu de données complet. L'obtention d'un jeu de données complet est un avantage considérable, car il est alors possible d'intégrer *SI* dans n'importe quel pipeline ou logiciel existant afin de les rendre utilisables en présence de valeurs manquantes [4]. Cependant, il est problématique de traiter les valeurs imputées comme de vraies valeurs [10], la variabilité due aux valeurs manquantes inconnues ne peut pas être prise en compte, les inférences en se basant sur ces données imputées surestimeront la précision [8].

L'imputation multiple (*MI*) a été originellement proposée par Rubin [10]. Ce framework consiste à remplacer chaque valeur manquante par au moins deux valeurs de substitution représentant une distribution de possibilités, ce qui représente l'incertitude quant à la valeur à imputer [13]. Dans un contexte d'apprentissage automatique, un modèle est formé sur chaque jeu de données complété et les résultats de tous les modèles sont ensuite regroupés de manière ensembliste. Un avantage de *MI* par rapport à *SI* est que chaque valeur manquante est représentée par un échantillon de valeurs d'imputation possibles, ce qui donne lieu à des inférences qui reflètent mieux le niveau d'incertitude associé à chaque valeur manquante [13]. Par conséquent, les résultats obtenus par l'ensemble des modèles seront moins biaisés que ceux de chaque modèle pris indépendamment. Un inconvénient évident est le coût de calcul d'une telle méthodologie, l'entraînement de multiples modèles multiplie le temps de calcul nécessaire. Bien qu'il s'agisse d'un framework assez ancien, *MI* est rarement utilisé en pratique, les scientifiques et utilisateurs de méthodes d'imputation se reposent encore généralement sur l'imputation simple. Cela peut s'expliquer en partie par le coût de calcul de *MI*, qui est élevé en raison du paradigme ensembliste.

Il existe de nombreuses méthodes permettant de traiter les valeurs manquantes en les remplaçant par des valeurs plausibles. Une méthode très simple qui peut être utilisée pour traiter les valeurs manquantes est la substitution par la moyenne, où les valeurs manquantes de chaque caractéristique sont remplacées par la valeur moyenne pour cette caractéristique. Cette méthode a l'avantage d'être facile à mettre en œuvre et à utiliser, tout en conservant toutes les informations non manquantes. Des méthodes d'imputation plus avancées peuvent être utilisées pour obtenir de meilleurs résultats d'inférence sur les données imputées. Une méthode populaire est l'algorithme SOFTIMPUTE, introduit par Mazumder et al. [6], qui fonctionne de manière itérative, à chaque étape les valeurs manquantes sont remplacées en utilisant une décomposition en valeur singulières. En 2012, Stekhoven et Bühlmann ont présenté l'algorithme MISSFOREST, une méthode d'imputation itérative basée sur les forêts aléatoires [11]. Ils ont montré que MISSFOREST peut traiter avec succès les valeurs manquantes, en particulier dans les jeux de données comprenant des types de variables mixtes. En 2018, Gondara et Wang ont présenté MIDA : Multiple-Imputation Using Denoising Autoencoders (DAEs) [3]. Cette méthode est basée sur les modèles d'autoencodeurs, un modèle de réseau neuronal utilisé pour reconstruire sa propre entrée à partir d'une représentation latente réduite. Plus récemment, Yoon et al. ont introduit GAIN : Generative Adversarial Imputation Nets [12], qui s'appuie sur des modèles adversaires pour imputer les valeurs manquantes. En 2020, Muzellec et al. ont présenté SINKHORN OT, une méthode basée sur le transport optimal pour l'imputation des données [7]. Toutes ces méthodes d'imputation peuvent être utilisées différemment en fonction du framework d'imputation que l'on souhaite appliquer.

L'algorithme MICE, pour Multivariate Imputation by Chained Equations [1], traite l'incertitude d'imputation en imputant de manière itérative l'ensemble de données au sein de son propre algorithme. Les valeurs manquantes sont d'abord imputées par une méthode de substitution par la moyenne, puis MICE entraîne itérativement plusieurs modèles de régression linéaire pour imputer chaque valeur manquante de chaque caractéristique en utilisant toutes les autres caractéristiques pour entraîner les régressions jusqu'à convergence. En ce sens, MICE produit un jeu de données complété unique mais qui prend en compte l'incertitude d'imputation.

3 Contributions

MI est un premier pas vers la prise en compte de l'incertitude d'imputation. Il prend naturellement en compte l'incertitude des valeurs imputées grâce à sa nature ensembliste, mais chaque modèle d'inférence prit à part est tout de même biaisé du fait qu'il a été entraîné sur un jeu de données complété arbitrairement [10].

Nous proposons deux frameworks qui prennent en compte cette incertitude d'imputation et montrons que les réseaux de neurones entraînés à l'aide de ces frameworks ont une meilleure capacité de généralisation. Ces frameworks sont basés sur le calcul de l'incertitude entre les imputations, qui correspond à l'écart type entre les valeurs imputées de tous les jeux de données complétés. Cette incertitude est utilisée comme échelle pour ajouter de la stochasticité à l'imputation des valeurs manquantes directement lors de l'extraction des batchs pendant l'apprentissage. Il en résulte une sorte de régularisation par le bruit qui prend en compte l'incertitude d'imputation, ce qui améliore la capacité de généralisation et, par conséquent, les résultats d'inférence sur des données de test.

3.1 Single-Hotpatching

Notre framework Single-Hotpatching (*S-HOT*) est similaire à *MI*, mais présente l'avantage de n'entraîner qu'un seul réseau de neurones. les valeurs manquantes sont imputées dynamiquement lors de l'extraction de batches lors de la phase d'entraînement du modèle.

L'entraînement d'un large réseau de neurones nécessite beaucoup de temps et de ressources. L'entraînement de plusieurs modèles de ce type de manière ensembliste n'est pas toujours une option viable. *S-HOT* vise à former un seul modèle tout en s'appuyant sur des imputations multiples telles que *MI*. Il forme un modèle moins biaisé que s'il avait été entraîné à l'aide de *SI* sans nécessiter autant de temps de calcul que *MI*. Nous montrons expérimentalement que *S-HOT* obtient des résultats nettement meilleurs que *SI* pour des temps d'exécutions identiques. Il est donc intéressant d'utiliser *S-HOT* dans toutes les situations où l'on cherche à former un modèle large et unique sur des données complétées.

Lorsque l'on entraîne un réseau de neurones sur un jeu de données complété dans lequel les valeurs imputées sont très certainement non optimales, le modèle apprend de manière répétée sur des données erronées et imprécises, conduisant à un modèle biaisé manquant de capacité de généralisation. Au lieu de cela, *S-HOT* effectue des imputations multiples et calcule l'écart type entre les imputations de chaque valeur manquante, que nous appelons le niveau d'incertitude. Ce niveau d'incertitude est utilisé pour tirer des valeurs aléatoires d'une distribution normale paramétrée à l'aide de la moyenne et de l'écart type calculés entre les imputations. En entraînant le modèle sur des batches dans lesquelles les valeurs manquantes sont tirées dynamiquement sur les distributions d'imputation, nous nous assurons que le modèle apprend sur une multitude d'imputations plausibles. Le modèle est entraîné sur un éventail de valeurs possibles à la place de chaque valeur manquante, conduisant à un modèle moins biaisé avec une plus grande capacité de généralisation. La figure 1 illustre les phases d'entraînement et de test d'un réseau de neurones lors de l'utilisation de *S-HOT*. Une fois le modèle entraîné, les résultats de prédiction sont obtenus en tirant aléatoirement les valeurs manquantes dans le jeu de test de la même manière pour p itérations, aboutissant à p prédictions. La moyenne des p probabilités de sortie du modèle est utilisée comme prédiction finale, ce qui est plus robuste qu'une itération unique de ce processus.

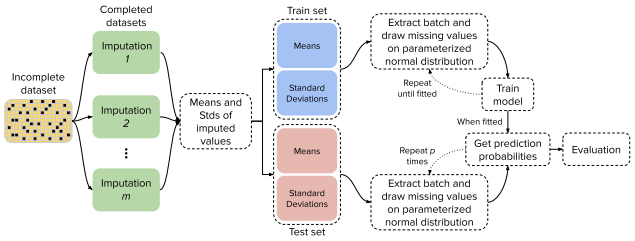


FIGURE 1 – Single-Hotpatching. Nous effectuons m imputations et calculons les moyennes et les écarts types des valeurs imputées. Celles-ci sont séparées entre les jeux d'entraînement et de test. Pendant l'entraînement, chaque fois qu'une batch est extraite, les valeurs manquantes sont tirées d'une distribution normale paramétrée à l'aide des moyennes et des écarts-types calculés précédemment. Une fois le modèle entraîné, nous obtenons les probabilités de prédiction en appliquant le même processus p fois pour chaque instance de test, menant à p prédictions. La prédiction finale est calculée comme la moyenne des p probabilités de prédiction. Le paramètre p est fixé à quelques dizaines pour obtenir des résultats de prédiction robustes.

Mathématiquement, on note $X \in \mathbb{R}^{n \times d}$ le jeu de données originel, où chaque valeur X_{ij} est soit observée ou manquante, avec n et d le nombre d'instances et de variables dans X. On effectue m imputations, menant à m différents jeux de données complétés $\tilde{X}^{1\dots m}$, avec $\tilde{X}^k \in \mathbb{R}^{n \times d}$ le k-ième jeu complété. Donc, une valeur manquante X_{ij} est imputée avec m valeurs différentes $\tilde{X}_{ij}^{1\dots m}$. Ensuite, on calcule les moyennes μ et écarts types σ de chaque valeur des m jeux complétés, avec $\mu_{ij} = \frac{1}{m}\sum_{k=1}^{m} \tilde{X}_{ij}^{k}$ (1) et $\sigma_{ij} = \sqrt{\frac{1}{m}\sum_{k=1}^{m}(\tilde{X}_{ij}^{k} - \mu_{ij})^2}$ (2). On note que les valeurs de $\tilde{X}^{1\dots m}$ qui sont non manquantes dans X ont une moyenne de $\mu_{ij} = X_{ij}$ et un écart type de $\sigma_{ij} = 0$, seul les valeurs manquantes dans X ont une valeur $\sigma_{ij} > 0$. Le réseau de neurones est ensuite entraîné avec des batches calculées à partir de μ et σ. Pour extraire une batch $B \in \mathbb{R}^{b \times d}$, avec b le nombre d'éléments dans la batch, chaque valeur de la batch est tirée à partir d'une distribution normale paramétrisée avec la moyenne μ_{ij} et l'écart type $\alpha \cdot \sigma_{ij}$, tel que $B_{ij} \sim \mathcal{N}(\mu_{ij}, \alpha \cdot \sigma_{ij})$. Où α est un hyper-paramètre d'échelle pouvant être défini à 1 dans la plupart des cas et peut être fixé à une valeur plus basse selon le taux d'incertitude moyen observé. Si la méthode d'imputation utilisée produit des valeurs imputées avec une grande incertitude, l'échelle α doit être empiriquement fixée à une valeur inférieure à 1 afin de limiter l'impact stochastique induit par l'approche. Nous n'avons jamais trouvé de situation où l'augmentation de la valeur de α était bénéfique. Lorsqu'une instance est présenté au réseau de neurones, les valeurs observées sont définies par X_{ij} et les valeurs manquantes par une valeur aléatoire suivant les distributions normales des m imputations. Ainsi, le réseau de neurones n'est pas entraîné de manière répétitive sur des imputations

arbitrairement fixées (et très probablement non optimales), comme ce serait le cas avec *SI* ou *MI*. Ce processus fonctionne comme une régularisation par le bruit prenant en compte l'incertitude entre les imputations, ce qui permet d'obtenir un réseau neuronal moins biaisé et plus généralisé.

3.2 Multiple-Hotpatching

Multiple-Hotpatching (*M-HOT*) étend *S-HOT* à l'aide du paradigme ensembliste. Ce framework entraîne autant de modèles d'inférence que d'imputations effectuées, ces modèles sont entraînés en tenant compte de l'incertitude entre les imputations, ce qui permet d'obtenir des modèles individuellement moins biaisés. Nous montrons empiriquement que *M-HOT* donne des résultats systématiquement meilleurs que *MI* sans pour autant être plus coûteux en termes de calcul. Il est donc avantageux d'utiliser *M-HOT* dans les situations où il est possible de se permettre d'entraîner plusieurs réseaux de neurones de manière ensembliste.

Le framework est similaire à celui de *S-HOT*, à la différence que *M-HOT* repose sur le paradigme ensembliste comme pour *MI*. Nous utilisons les notations mathématiques définies précédemment, $X \in \mathbb{R}^{n \times d}$ est le jeu de données initial incomplet, où chaque valeur X_{ij} est soit observée, soit manquante. Nous effectuons m imputations qui conduisent à m différents jeux de données complétés $\tilde{X}^{1 \dots m}$, avec $\tilde{X}^k \; dans \mathbb{R}^{n \times d}$ le k-ième jeu de données complété. Un modèle est défini par jeu de données complété, conduisant à m modèles. Les écarts types σ sont calculé de la même manière que dans l'équation 2, il n'est pas utile de calculer les moyennes. L'ensemble des modèles est ensuite entraîné. Pour extraire une batch $B^k \; dans \mathbb{R}^{b \times d}$ qui sera donnée en entrée au k-ième modèle, chaque valeur de la batch est tiré d'une distribution normale paramétrée avec une moyenne $\tilde{X}^k_{ij}$ et un écart type $\alpha \cdot \sigma_{ij}$, tel que $B^k_{ij} \sim \mathcal{N}(\tilde{X}^k_{ij}, \alpha \cdot \sigma_{ij})$. Ainsi, tous les modèles sont entraînés en parallèle, les valeurs manquantes présentées au k-ième modèle sont remplacées par des valeurs imputées tirées d'une distribution normale centrée sur la k-ième imputation calculée, menant à une diversité accrue des données d'entraînement utilisées. Il en résulte un ensemble de modèles moins biaisés et capables d'une plus grande généralisation que dans *MI*, puisqu'ils prennent en compte l'incertitude entre les imputations. *M-HOT* peut être utilisé comme substitut à *MI* dans toutes les situations où *MI* est viable.

4 Expérimentations

4.1 Protocole Expérimental

Dans nos expérimentations, nous ne cherchons pas à comparer les performances des méthodes d'imputation utilisées. Nous nous intéressons plutôt aux résultats que nous pouvons observer pour une même méthode d'imputation lorsque nous utilisons chacun des frameworks d'imputation comparés.

Nous avons mené nos expérimentations sur cinq jeux de données tabulaires de classification : le célébre jeu de données IRIS [1], le jeu de données STATLOG [2], le jeu de données WINE [2], le jeu de données PIMA [3] et le jeu de données ABALONE [2].

Les jeux de données utilisés ne contenant pas de valeurs manquantes, nous avons donc ajouter artificiellement différents taux de valeurs manquantes. Nous simulons trois mécanismes de valeurs manquantes différents au sein des jeux de données, MCAR, MAR et MNAR [9]. Avec le mécanisme MCAR, nous introduisons des valeurs manquantes de manière totalement aléatoire en masquant au hasard un certain taux de valeurs. Pour le mécanisme MAR, un sous-ensemble aléatoire de caractéristiques est choisi pour ne pas être masqué, ce sous-ensemble est utilisé comme entrée d'un modèle logistique et la sortie du modèle est utilisée pour masquer les valeurs des caractéristiques restantes. Pour le mécanisme MNAR, la mise en oeuvre est proche de celle de MAR, mais l'entrée du modèle logistique est masquée à l'aide d'un mécanisme MCAR. Ainsi, la sortie du modèle dépend de valeurs qui sont indifféremment connues ou masquées.

Nous comparons les frameworks à l'aide des méthodes d'imputation décrites plus tôt : MISSFOREST, SOFTIMPUTE, GAIN, MIDA et SINKHORN. Nous appliquons chaque framework d'imputation en utilisant chaque méthode d'imputation et nous nous intéressons uniquement à la comparaison des résultats obtenus entre chaque framework d'imputation. Dans nos expérimentations, nous considérons MICE comme un framework d'imputation auquel nous comparons les résultats de nos frameworks.

Nos expériences visent à évaluer les performances des réseaux neuronaux dans un contexte d'apprentissage supervisé afin de mesurer le biais et la capacité de généralisation du modèle. Nous utilisons un réseau de neurone simple sous la bibliothèque scikit-learn [4], paramétré à l'aide d'hyperparamètres menant à de bons résultats et identiques pour chaque jeu de données afin de garantir une comparaison juste et impartiale. Dans tous les cas nous utilisons l'optimiseur Adam avec un taux d'apprentissage fixé à 0.001. Les modèles sont composés de deux couches cachées : pour les jeux de données IRIS, STATLOG, PIMA et ABALONE les deux couches sont de dimensions 32, pour le jeu de données WINE les couches sont de dimensions 64 et 32 respectivement. Les performances de prédiction sont évaluées à l'aide de la métrique AUC (aire sous la courbe ROC), l'accuracy équilibrée et le score F1.

Afin de mieux évaluer les résultats obtenus, nous basons nos comparaisons sur les tests statistiques de Friedman et de Nemenyi tels que décrits dans [2]. Le test de Friedman est d'abord utilisé pour vérifier si l'hypothèse nulle selon laquelle tous les frameworks comparés sont statistiquement équivalents pour une p-value donnée est rejetée ou non. Si

1. https://scikit-learn.org/stable/modules/generated/sklearets.load_iris.html
2. https://archive.ics.uci.edu
3. https://rioultf.users.greyc.fr/uci/files/pima-indians-diabetes
4. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

l'hypothèse nulle est rejeté, le test de Nemenyi est utilisé pour comparer les frameworks par paire. Le test de Nemenyi peut être facilement visualisé à l'aide d'un simple diagramme, ce qui facilite l'analyse des résultats.

4.2 Résultats

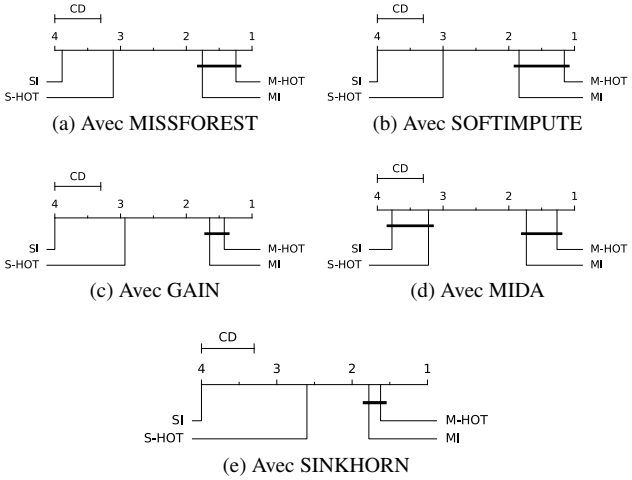
4.2.1 Étude Comparative Entre les Frameworks d'Imputation.

Les m imputation sont calculées, à la suite de quoi nous exécutons les quatre frameworks comparés *SI*, *MI*, *S-HOT* et *M-HOT*, en suivant le protocole expérimental décrit précédemment. Le tableau 1 présente une partie des résultats obtenus avec les quatre frameworks comparés lors de l'utilisation de la méthode d'imputation MISSFOREST, pour des raisons de places nous ne pouvons ajouter les résultats complets. Nous n'observons aucune influence du mécanisme ou du taux de valeurs manquantes sur les résultats obtenus, ce qui semble montrer que nos frameworks ne sont pas sensibles à ces paramètres et peuvent être utilisés en toutes circonstances. Nous constatons que le frameworks *M-HOT* obtient les meilleurs résultats dans la grande majorité des cas, tandis que *S-HOT* obtient systématiquement de meilleurs résultats que *SI*.

Dataset	Pattern		*SI*	*MI*	*S-HOT*	*M-HOT*
WINE	MCAR	10%	0.9987736 (4)	0.9989672 (2)	0.9988008 (3)	**0.9989811** (1)
		15%	0.9955454 (4)	0.9960062 (2)	0.9956407 (3)	**0.9960307** (1)
		25%	0.9910498 (4)	0.9919927 (2)	0.9914148 (3)	**0.9923485** (1)
	MAR	10%	0.9961058 (4)	0.9965056 (2)	0.9961142 (3)	**0.9965060** (1)
		15%	0.9977720 (4)	**0.9982010** (1)	0.9978677 (3)	0.9981809 (2)
		25%	0.9952116 (4)	0.9965157 (2)	0.9959576 (3)	**0.9968702** (1)
	MNAR	10%	0.9987205 (3)	**0.9988244** (1)	0.9987058 (4)	0.9988131 (2)
		15%	0.9974746 (4)	0.9976465 (2)	0.9974850 (3)	**0.9976683** (1)
		25%	0.9808498 (4)	0.9841291 (2)	0.9822719 (3)	**0.9844587** (1)
PIMA	MCAR	10%	0.8193054 (4)	**0.8211340** (1)	0.8196586 (3)	0.8211193 (2)
		15%	0.8073739 (4)	0.8095505 (2)	0.8078378 (3)	**0.8095824** (1)
		25%	0.8029002 (4)	0.8060367 (2)	0.8043589 (3)	**0.8065611** (1)
	MAR	10%	0.8238900 (4)	**0.8257089** (1)	0.8242472 (3)	0.8256414 (2)
		15%	0.8045918 (4)	0.8080830 (2)	0.8061503 (3)	**0.8083194** (1)
		25%	0.8017568 (4)	**0.8041203** (1)	0.8025144 (3)	0.8040062 (2)
	MNAR	10%	0.8280685 (4)	0.8302729 (2)	0.8284115 (3)	**0.8303076** (1)
		15%	0.8279577 (4)	0.8298929 (2)	0.8283792 (3)	**0.8300464** (1)
		25%	0.8005008 (4)	0.8043448 (2)	0.8021749 (3)	**0.8047250** (1)
ABAL	MCAR	10%	0.8737739 (4)	0.8748059 (2)	0.8740180 (3)	**0.8749393** (1)
		15%	0.8714861 (4)	0.8725539 (2)	0.8717831 (3)	**0.8726250** (1)
		25%	0.8663833 (4)	0.8674332 (2)	0.8666186 (3)	**0.8675645** (1)
	MAR	10%	0.8742551 (4)	0.8751972 (2)	0.8743399 (3)	**0.8753671** (1)
		15%	0.8720722 (4)	0.8731502 (2)	0.8721856 (3)	**0.8731739** (1)
		25%	0.8697060 (4)	0.8708046 (2)	0.8699619 (3)	**0.8709102** (1)
	MNAR	10%	0.8760444 (4)	0.8768033 (2)	0.8760447 (3)	**0.8769350** (1)
		15%	0.8753217 (4)	0.8763271 (2)	0.8754605 (3)	**0.8764456** (1)
		25%	0.8683507 (4)	0.8696780 (2)	0.8690281 (3)	**0.8697714** (1)
Average rank			3.8889	1.7556	3.1111	1.2444

TABLE 1 – Résultats de prédictions obtenue après l'application de chaque framework, en utilisant la méthode d'imputation MISSFOREST.

Le test de Friedman permet de rejeter l'hypothèse nulle selon laquelle tous les frameworks seraient équivalent. Nous appliquons le test de Nemenyi et présentons les résultats sous forme graphique dans la Figure 2. Deux frameworks peuvent être considérés comme significativement différents s'ils ne sont pas reliés par une barre noire, c'est à dire si leurs rangs moyens diffèrent de plus que la distance critique indiquée en haut de chaque graphique. Dans tous les cas nous constatons que l'ordre des frameworks, du pire au meilleur, est le même : *SI* est le moins performant, suivi de *S-HOT*, puis de *MI* et enfin de *M-HOT*. *S-HOT* obtient des résultats nettement meilleurs que *SI*, ce qui montre qu'il s'agit d'une bonne alternative à *SI* lorsque l'on souhaite entraîner un large et unique modèle. Nous notons que *M-HOT* obtient systématiquement de meilleurs résultats que *MI*, ce qui semble montrer qu'il s'agit toujours d'une bonne alternative viable à *MI*.



(a) Avec MISSFOREST (b) Avec SOFTIMPUTE

(c) Avec GAIN (d) Avec MIDA

(e) Avec SINKHORN

FIGURE 2 – Tests de Nemenyi comparant les quatre frameworks avec chaque méthode d'imputation, $CD \approx 0.6992$.

4.2.2 Étude Comparative Entre MICE et les Frameworks d'Imputation.

Nous comparons *S-HOT* et *M-HOT* à MICE, nous observons que le framework *M-HOT* obtient les meilleurs résultats dans la plupart des cas. Le test de Friedman nous permet à nouveau de rejetter l'hypothèse nulle. Nous appliquons le test de Nemenyi, la Figure 3 présente les résultats. Nous constatons que les performances de MICE sont largement supérieures à celles de *SI*, même si elles ne sont pas significativement meilleures d'un point de vue statistique. *S-HOT* obtient des résultats nettement meilleurs que *SI* et légèrement meilleurs que MICE. Les frameworks *MI* et *M-HOT* sont nettement meilleurs que les autres frameworks et méthodes testés, une fois encore, *M-HOT* obtient de meilleurs résultats que *MI*.

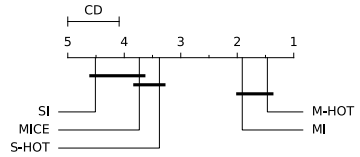


FIGURE 3 – Tests de Nemenyi comparant la méthode d'imputation MICE avec les meilleurs résultats précédemment obtenus sur chaque jeu de données, $CD \approx 0.9093$.

4.2.3 Comparaison des Temps d'Exécution.

Enfin, nous avons comparé le temps de calcul nécessaire à l'exécution de chaque framework dans chaque scénario testé. Dans tous les cas, la majeure partie du temps de calcul provient du calcul des m imputations. *SI* bénéficie largement de ce point, les trois autres frameworks nécessitent tous le même temps pour calculer les multiples imputations.

Nous n'observons aucune différence dans le temps d'entraînement requis entre *SI* et *S-HOT*. Dans le cas de *MI* et de *M-HOT*, *M-HOT* est un peu plus lent de moins d'une seconde à quelques secondes pour les modèles et les jeux de données les plus larges en comparaison à *MI*. La différence globale de temps d'exécution entre *MI* et *M-HOT* est négligeable. Étant donné que nous avons montré que *M-HOT* obtient systématiquement de meilleurs résultats que *MI*, la plupart des scénarios bénéficieraient de l'utilisation de *M-HOT* plutôt que de *MI*.

5 Discussion and Conclusion

La prise en compte de l'incertitude d'imputation lors de l'apprentissage d'un réseau de neurones ne fait pas l'objet de nombreuses recherches. En effet, les réseaux de neurones sont naturellement capables d'une généralisation suffisante pour négliger les conséquences du biais induit par la non prise en compte de l'incertitude d'imputation. Dans cet article, nous avons étudié et proposé deux frameworks d'imputation qui peuvent être utilisés pour entraîner des modèles en tenant compte de ce niveau d'incertitude, permettant d'obtenir des modèles capables d'une plus grande généralisation et de meilleurs résultats d'inférence.

Les deux frameworks proposés, *S-HOT* et *M-HOT*, visent à remplacer respectivement l'imputation simple (*SI*) et l'imputation multiple (*MI*). Nous réalisons différentes expérimentations pour comparer les frameworks d'imputation et nous montrons qu'ils rivalisent avec d'autres frameworks d'imputation, voire les surpassent dans de nombreuses situations. Nous évaluons statistiquement les résultats à l'aide des tests de Friedman et de Nemenyi et montrons que nos frameworks conduisent à des réseaux neuronaux moins biaisés, ce qui améliore les résultats de l'inférence. Nous comparons également les temps d'exécution requis pour chaque framework et concluons que la différence totale de temps d'exécution entre *MI* et *M-HOT* est négligeable, tandis que *SI* est plus rapide que *S-HOT* mais obtient des résultats nettement moins bons que *S-HOT*. Nous avons montré que *S-HOT* obtient des résultats nettement meilleurs que *SI* dans tous les scénarios testés, et nous en concluons qu'il est avantageux d'utiliser *S-HOT* lorsque l'on doit former un réseau de neurones unique et de grande taille. Nos expériences montrent que *M-HOT* obtient systématiquement de meilleurs résultats que *MI* dans tous les scénarios testés pour un temps d'exécution comparable.

Dans ce travail, nous supposons une distribution normale des valeurs imputées. Nous obtenons de bons résultats empiriques, mais une distribution normale n'est pas forcément toujours pertinente selon la nature de la caractéristique manquante ou de la méthode d'imputation utilisée, de futurs travaux se concentreront sur cette question.

Remerciements

This research is supported by the European Union's Horizon 2020 research and innovation program under grant agreement No 875171, project QUALITOP (Monitoring multidimensional aspects of QUAlity of Life after cancer ImmunoTherapy - an Open smart digital Platform for personalized prevention and patient management).

Références

[1] Stef van Buuren and Karin Groothuis-Oudshoorn. mice : Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, 45(3) :1–67, 2011.

[2] Janez Demsar. Statistical Comparisons of Classifiers over Multiple Data Sets. *The Journal of Machine Learning Research*, pages 1–30, 2006.

[3] Lovedeep Gondara and Ke Wang. MIDA : Multiple Imputation Using Denoising Autoencoders. *Pacific-Asia Conference on Knowledge Discovery and Data Mining 2018*, pages 260–272, 2018.

[4] Julie Josse, Nicolas Prost, Erwan Scornet, and Gaël Varoquaux. *On the consistency of supervised learning with missing values*. ArXiv, February 2019.

[5] Marine Le Morvan, Julie Josse, Erwan Scornet, and Gael Varoquaux. What's a good imputation to predict with missing values ? In *Advances in Neural Information Processing Systems*, volume 34, pages 11530–11540. Curran Associates, Inc., 2021.

[6] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral Regularization Algorithms for Learning Large Incomplete Matrices. *Journal of machine learning research : JMLR*, 11 :2287–2322, March 2010.

[7] Boris Muzellec, Julie Josse, Claire Boyer, and Marco Cuturi. Missing Data Imputation using Optimal Transport. In *Proceedings of the 37th International Conference on Machine Learning*, pages 7130–7140. PMLR, November 2020. ISSN : 2640-3498.

[8] D. B. Rubin and N. Schenker. Multiple imputation in health-care databases : an overview and some applications. *Statistics in Medicine*, 10(4) :585–598, April 1991.

[9] Donald B. Rubin. Inference and Missing Data. *Biometrika*, 63(3) :581–592, 1976. Publisher : [Oxford University Press, Biometrika Trust].

[10] Donald B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. John Wiley & Sons, June 2004. Google-Books-ID : bQBtw6rx_mUC.

[11] Daniel J. Stekhoven and Peter Bühlmann. MissForest—Non-Parametric Missing Value Imputation for Mixed-Type Data. *Bioinformatics*, 28(1), January 2012.

[12] Jinsung Yoon, James Jordon, and Mihaela Schaar. GAIN : Missing Data Imputation using Generative Adversarial Nets. In *Proceedings of the 35th International Conference on Machine Learning*, page 5689. PMLR, July 2018. ISSN : 2640-3498.

[13] Yang Yuan. Multiple Imputation for Missing Data : Concepts and New Development. *SAS Institute Inc.*, January 2005.