

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

DÉPARTEMENT DE GÉNIE MATHÉMATIQUES ET MODÉLISATION

Stage de Fin d'Etudes - Contrat de professionnalisation -

Spécialité : Mathématiques Appliquées

Déploiement de la Data Science au sein de la
Caisse d'Epargne Midi Pyrénées.

- Février à Septembre 2023 -

Auteur :

Roig Lila

Entreprise :

Caisse d'épargne Midi Pyrénées,
10 Avenue Maxwell, 31100 Toulouse

Référent INSA :

Reveillac Anthony

Responsables de stage :

Martin Jérôme
Michelot Bertrand

Année 2022-2023

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage de fin d'études réalisé sous la forme d'un contrat de professionnalisation (alternance) à la Caisse d'Epargne Midi-Pyrénées.

Tout d'abord, j'adresse mes remerciements à mon tuteur de stage Jérôme Martin, manager du pôle Data & Décisionnel de la Caisse d'Epargne Midi-Pyrénées, pour m'avoir permis de réaliser cette alternance, pour m'avoir très bien intégré dans l'équipe Data et pour m'avoir fait confiance dès les premiers jours de mon alternance.

Je remercie vivement mon second tuteur de stage Bertrand Michelot, chef de projet au pôle Data & Décisionnel de la Caisse d'Epargne Midi-Pyrénées, pour m'avoir également permis de réaliser cette alternance et m'avoir fait confiance, pour sa bienveillance, son encadrement et son aide.

Je tiens également à remercier Mathieu Le Pajolec, prestataire Data Scientist dans le pôle Data, pour l'ensemble de ses encouragements, pour son aide et ses conseils sur mes projets, mais aussi sur le métier de Data Scientist en général. Ceci m'a permis de gagner confiance en moi et d'avoir une meilleure vision du monde de la Data Science.

Je remercie l'ensemble de l'équipe du pôle Data & Décisionnel de la Caisse d'Epargne Midi-Pyrénées, pour leur accueil, pour leur aide et leur bonne humeur. Je les remercie pour m'avoir très bien intégré dans l'équipe dès le début de mon alternance, ce qui m'a permis de me sentir à l'aise et a fortement contribué au succès de cette alternance.

Enfin, je remercie Anthony Réveillac, chef du département des sciences humaines à l'INSA de Toulouse et référent INSA, pour son suivi et pour avoir veillé au bon déroulement de mon alternance. Je remercie également Philippe Villedieu, professeur et responsable des stages au département Génie Mathématiques et Modélisation de l'INSA, pour sa réactivité et sa flexibilité lors de nos échanges sur l'organisation de cette année.

Table des matières

1	Introduction	1
2	Cadre et objectifs du Stage	2
2.1	L'intérêt de la Data Science pour la CEMP	2
2.2	Le département Data & Décisionnel	2
2.3	Organisation et conditions de travail	3
2.4	Chronologie des missions confiées	4
3	Projet Back Office crédit	5
3.1	Introduction	5
3.2	Présentation des données	6
3.3	Analyse Exploratoire	6
3.3.1	Premier pré-traitement des données	6
3.3.2	Variables nulles et manquantes	7
3.3.3	Variables quantitatives	8
3.3.4	Variables qualitatives	8
3.3.5	Dépendances entre les variables	9
3.3.5.1	Variables quantitatives	9
3.3.5.2	Variables qualitatives	10
3.3.6	Sélection de variables	11
3.3.7	Imputation des données manquantes	11
3.4	Méthodes utilisées	12
3.4.1	Mise en forme des jeux de données	12
3.4.2	Débruitage des séries temporelles univariées	14
3.4.3	Modélisation par la méthode SARIMA	14
3.4.3.1	Décomposition STL	15
3.4.3.2	Application du modèle ARIMA et SARIMA.	17
3.4.4	Modélisation par l'algorithme MiniRocket	20
3.4.4.1	Approche multivariée	21
3.4.4.2	Approche univariée	22
3.5	Résultats	23
3.5.1	SARIMA	23
3.5.2	MiniRocket	24
3.5.3	Tableau comparatif	26
3.6	Conclusion	26

4 Projet détection de signature	28
4.1 Introduction	28
4.2 Présentation des données	28
4.3 Méthodes utilisées	29
4.3.1 Premières approches	29
4.3.2 Programme Python	30
4.3.2.1 Pré-traitement de l'image	30
4.3.2.2 Création du modèle	32
4.3.2.3 Détection de la signature	34
4.3.2.4 Limites de la méthode	37
4.3.3 Workflow Alteryx	37
4.4 Résultats	38
4.5 Conclusion	39
5 Projet détection de mots interdits	40
5.1 Introduction	40
5.2 Présentation des données	40
5.3 Méthodes utilisées	40
5.3.1 Programme Python	40
5.3.2 Workflow Alteryx	41
5.4 Résultats	41
6 Projet avis d'imposition	42
7 Conclusions et perspectives	42
Annexes	45
Annexe 1	45
0.1 Le groupe Banque Populaire Caisse d'Epargne (BPCE)	45
0.2 La Caisse d'Epargne de Midi Pyrénées (CEMP)	45
Annexe 2	47
Annexe 3	48
Annexe 4	50
Annexe 5	52
Bibliographie	56

1 Introduction

En 2023, les utilisateurs du monde entier génèrent chaque jour environ 2.5 quintillions (10^{30}) d'octets de données [1]. Cette masse colossale d'information, couplée à la baisse du coût et l'augmentation de la puissance des ordinateurs, a permis l'essor fulgurant de l'Intelligence Artificielle (IA). Les applications de l'IA concernent presque tous les domaines et sont de plus en plus présentes dans nos vies quotidiennes.

La Data Science est une science multidisciplinaire qui englobe le Machine Learning et le Deep-Learning, deux domaines scientifiques derrière la plupart des projets d'IA. Elle utilise notamment des méthodes mathématiques, statistiques et des algorithmes pour extraire de l'information exploitable à partir de données brutes. Utilisée pour optimiser les tâches, réduire les coûts et prendre des décisions, la Data Science est devenue un facteur important pour maintenir la compétitivité d'une entreprise. Ainsi, l'utilisation de l'IA par les entreprises devrait atteindre un taux de croissance annuel moyen de 40% entre 2023 et 2032 [2].

De plus, compte tenu de la volumétrie des données récoltées, le secteur bancaire est particulièrement bien adapté pour mettre en place diverses techniques de Data Science.

C'est donc dans l'optique de développer des solutions de Data Science pour diverses applications, que j'ai réalisé mon alternance (contrat de professionnalisation) à la Caisse d'Epargne Midi-Pyrénées (CEMP), du 19 septembre 2022 au 29 septembre 2023.

La Caisse d'Epargne est une banque française divisée en 15 Caisses d'Epargne régionales, dont la Caisse d'Epargne Midi-Pyrénées dans laquelle j'ai réalisé mon alternance. La Caisse d'Epargne fait partie du groupe bancaire Banque Populaire et Caisse d'Epargne (BPCE), qui détient depuis 2009 les Caisses d'Epargne (CE) et les Banques Populaires (BP). Une présentation plus détaillée de l'entreprise se trouve en Annexe A1 de ce rapport.

Dans un premier temps, nous présenterons le cadre de cette alternance en abordant au chapitre 2 l'organisation et l'environnement de travail. Puis, nous décrirons les missions confiées en commençant au chapitre 3 par le projet *Back Office crédit*, portant sur la prédiction de séries temporelles. Nous poursuivrons en détaillant le projet de *détection de signature* au chapitre 4, concernant le traitement d'images et l'extraction d'information à partir de fichiers PDF. Par la suite, nous présenterons succinctement les deux derniers projets réalisés en commençant par le projet de *détection de mots interdits* au chapitre 5, portant sur du traitement de texte. Nous terminerons en évoquant le projet sur les *avis d'imposition* au chapitre 6 portant également sur l'extraction d'information à partir de documents PDF. La description de chaque mission réalisée sera découpée en deux parties. Une première partie concernant les méthodes utilisées et une deuxième partie présentant les résultats obtenus. Enfin, nous conclurons ce rapport au chapitre 7.

2 Cadre et objectifs du Stage

2.1 L'intérêt de la Data Science pour la CEMP

La Data Science est déployée au niveau du groupe BPCE, auquel appartient la CEMP. Cependant les Caisses d'Epargne et les Banques Populaires développent également une activité importante autour de la Data. En effet, les pôles Data de chaque établissement jouent un rôle de support métier¹ en développant des outils Data pour répondre au besoin des différents services. Mais pourquoi développer des outils Data de façon locale alors que les équipes de BPCE ont bien plus de ressources et de budget ? La CEMP peut solliciter BPCE-SI (filiale Data du groupe BPCE) pour l'aider à réaliser des projets Data. Les équipes de BPCE-SI ont des spécialistes en Data Science et autres domaines, détiennent plus de ressources et de capacités de calcul. Cependant, leurs délais de réponses sont très longs, le coût financier est important et le manque de proximité entre BPCE-SI et les équipes métier locales de la CEMP pose un problème. Par ailleurs, les équipes de BPCE traitent des sujets universels à toutes les Caisses d'Epargne et Banques Populaires. Les problématiques locales spécifiques à la CEMP ne sont généralement pas prises en compte au niveau groupe BPCE et nécessitent donc une équipe Data locale.

2.2 Le département Data & Décisionnel

La CEMP est divisée en 5 pôles : Ressources, Présidence du Directoire, Finances et Moyens Généraux, Banque de Détail et Banque de Développement Régional. Chaque pôle est divisé en départements. J'ai intégré l'équipe du département Data & Décisionnel créé en novembre 2019 et situé dans le pôle Finance et Moyens Généraux.

L'équipe Data & Décisionnel a pour mission de :

- Répondre aux demandes des métiers¹ souhaitant une analyse de leurs données en réalisant et publiant des tableaux de bord.
- D'accompagner les métiers pour l'appropriation des tableaux de bord et pour des études ponctuelles.
- D'assurer la maintenance des logiciels de traitements des données et d'intégrer les données dans le système d'information.

L'équipe est à ce jour composée de 14 personnes dont le manager Jérôme Martin, 6 titulaires, 4 prestataires, 2 alternants et 1 stagiaire. L'ensemble de l'équipe exerce le métier de Data Analyst ou de Chef de Projet Data, à l'exception d'un prestataire en Data Science, présent pour une durée de 6 mois pendant mon stage. J'ai ainsi eu l'occasion d'échanger sur mes projets avec le Data Scientist. Néanmoins, celui-ci travaillait sur des projets différents

1. Différentes équipes représentant les collaborateurs dans des activités diverses.

des miens et j'ai donc mené l'ensemble de mes projets en autonomie. Durant les trois derniers mois de mon stage, je fus la seule à travailler sur des projets de Data Science.

2.3 Organisation et conditions de travail

L'équipe a adopté la méthode Agile, en effectuant notamment tous les matins le *Daily*, une réunion d'une trentaine de minutes où chaque membre de l'équipe explique les tâches réalisées la veille et le travail qu'il va effectuer le jour même. Ceci permet d'échanger de manière rapide et efficace sur les difficultés rencontrées. J'ai également signé un accord de télétravail à hauteur de deux jours par semaine.

Pour réaliser mes projets de Data Science, un ordinateur portable d'entreprise HP muni de 16Go de RAM m'a été prêté. La CEMP ne dispose pas à ce jour, de GPUs ou de capacités de calcul importantes facilement accessibles. J'ai donc développé mes solutions uniquement avec les ressources disponibles sur l'ordinateur portable. Par ailleurs, pour des raisons de sécurité, les pares-feux de la CEMP sont conséquents et bloquent l'accès à de nombreuses pages web ainsi que de nombreux téléchargements.

Pour réaliser mes projets, j'avais à ma disposition deux outils différents. Le premier outil était le langage de programmation Python et le gestionnaire de package Conda que j'ai installés avec Miniforge. En raison des pares-feux, l'installation de cet environnement de travail a été réalisée au début de l'alternance et m'a demandé plusieurs jours. De plus, toujours à cause des pares-feux, certaines librairies Python n'étaient pas disponibles et il m'a fallu installer de nombreuses librairies à la main. En dépit de ces problèmes d'installation, j'ai pu mener à bien l'ensemble de mes projets.

Pour réaliser les projets, j'ai commencé par implémenter une solution en Python, ce qui m'a permis d'explorer et de tester facilement et rapidement les différentes méthodes possibles. Cependant, il est nécessaire de savoir coder en Python pour maintenir les solutions proposées dans ce langage. Or, l'équipe dans laquelle je travaille ne dispose pas, pour le moment, de connaissances approfondies en Python (le prestataire Data Scientist restant seulement 6 mois). Il leur serait donc difficile de maintenir la solution réalisée à la fin de mon alternance. Par ailleurs, dans le cas où la solution Python devrait être mise en production, il serait compliqué d'installer les librairies Python sur le serveur de la CEMP pour des raisons de sécurité.

C'est pour ces raisons qu'après avoir trouvé la meilleure solution en Python, celle-ci devait être au maximum adaptée sur Alteryx. Alteryx est un logiciel de no-code principalement conçu pour réaliser de l'analyse de données (ETL). Il permet également de réaliser des opérations basiques de Data Science. Le module supplémentaire Alteryx Intelligence Suite a été installé sur mon poste et donne accès à quelques outils supplémentaires pour la Data Science comme des outils pour le traitement d'image, le Machine Learning ou le NLP. Bien que les possibilités avec Alteryx soient largement plus restreintes qu'en Python (par exemple : difficile de faire une boucle « for » ou « while »), ce logiciel est préféré par la CEMP car il a pour avantage d'être facile à maintenir et à mettre en production pour l'équipe Data.

2.4 Chronologie des missions confiées

Les missions de mon alternance se regroupent sous l'intitulé « Développement de cas d'usage avancés de la donnée (projets de Data Science) en lien avec le projet stratégique de l'entreprise. » Durant mon alternance, j'ai donc travaillé sur les projets suivants :

- Découverte de l'entreprise, installation du poste de travail (installation de Python et ses diverses librairies, installation de Alteryx) durant les semaines de présence au premier semestre. *Septembre 2022 à Janvier 2023.*
- Prédiction du nombre de crédits immobiliers arrivant en agence. Analyse exploratoire des données et prédiction du nombre de crédits avec des algorithmes de séries temporelles univariées et multivariées. *Février à Mars 2023.* Solution réalisée en Python uniquement. Voir chapitre [3](#).
- Détection de la présence d'une signature sur plusieurs types de documents PDF. Traitement d'image et utilisation d'un OCR pour extraire le texte des images. *Mars à Juin 2023.* Solution d'abord implémentée en Python puis reproduite en Alteryx. Voir chapitre [4](#).
- Détection des mots interdits dans du texte. Utilisation de méthodes basiques de traitement de texte et de NLP. *Juin à Juillet 2023.* Solution d'abord implémentée en Python puis reproduite en Alteryx. Voir chapitre [5](#).
- Extraction de l'information présente sur des avis d'imposition. Traitement d'images et utilisation d'un OCR pour extraire le texte des images. *Juillet à Septembre 2023.* Solution implémentée en Alteryx uniquement. Voir chapitre [6](#).

La suite de ce rapport présente les différents projets menés.

3 Projet Back Office crédit

- De Février à Mars 2023. -

3.1 Introduction

Lors de l'acquisition d'un crédit immobilier, de nombreuses interactions ont lieu entre un client et sa banque. Ses interactions se font par l'intermédiaire d'un conseiller, mais aussi par le biais de services de back-office (saisie des opérations). Parmi les facteurs essentiels au bon déroulement de la souscription d'un crédit immobilier, le délai d'édition de l'offre de crédit peut constituer un élément majeur de satisfaction ou d'insatisfaction pour le client. Diminuer les temps de traitement en mobilisant les ressources nécessaires améliorerait donc la relation client. Par ailleurs, l'activité crédit constitue le second poste en termes de ressources humaines dans l'établissement. Les variations d'activité peuvent engendrer des fluctuations d'effectifs d'environ 20%. Prévoir ces variations permettrait à la CEMP d'adapter son organisation interne pour prévoir une semaine à l'avance, le personnel à mobiliser.

L'objectif de ce projet est donc d'implémenter une méthode pour prédire le nombre de crédits immobiliers qui arrivent au service de back-office par jour ou par semaine.

Ce projet fait suite aux travaux de l'ancien alternant à la CEMP. Son objectif était de prédire le nombre de crédits immobiliers qui arrivent au back-office par semaine ainsi que le délai d'édition d'un crédit. Les méthodes testées comprenaient une méthode statistique simple (ARIMA), des méthodes de Machine Learning (XGBoost et Random Forest) et des méthodes de Deep Learning (Perceptron Multi-Couches et RNN-LSTM). Les résultats précédemment obtenus n'étant pas satisfaisants, nous disposons d'environ un mois pour revenir sur le projet. Dans le temps imparti, nous n'avons pas eu le temps d'aborder le sujet de prédiction du délai d'édition et nous sommes concentrés sur la prédiction du nombre de crédits.

La solution de ce projet a été entièrement implémentée en Python. Le code n'est pas adapté en Alteryx, car il n'existe pas d'équivalence Alteryx pour les méthodes utilisées.

Les données crédit disponibles comprennent une dépendance temporelle. Afin de conserver cette dépendance, nous décidons d'utiliser des méthodes spécifiques aux séries temporelles (ST). Une série temporelle est une suite de points représentant l'évolution d'une variable au cours du temps. Une série temporelle est dite univariée lorsqu'une seule variable évolue au cours du temps. La série est dite multivariée lorsque plusieurs variables varient en parallèle au cours du temps.

Dans un premier temps, nous présenterons les données disponibles en section 3.2 et réaliserons une analyse exploratoire des données à la section 3.3. Puis, nous expliquerons les méthodes utilisées section 3.4 : nous commencerons par implémenter à nouveau une méthode statistique simple avec SARIMA en section 3.4.3 avant de nous tourner vers une méthode de

Deep Learning avec l'algorithme MiniRocket à la section 3.4.4. Les deux approches univariées et multivariées seront testées. Enfin, nous présenterons les résultats obtenus en section 3.5 avant de conclure ce projet en section 3.6.

3.2 Présentation des données

Pour ce projet, des données concernant les dossiers de crédits immobiliers des clients de la CEMP ont été recueillies. Les données disponibles s'étendent de début 2018 à fin 2022. Dans un premier temps, nous décidons d'explorer l'ensemble des données crédit et conservons les 36 variables disponibles. Un résumé des données disponibles est présenté ci-dessous :

- Identifiant et caractéristiques du dossier (type de contrat, taux d'intérêt, durée du prêt, prêt complexe ou non, assurances, etc).
- Détail de l'instruction des dossiers depuis la simulation jusqu'à l'édition (date de création du dossier de vente, date de validation du scénario, date d'accord, date d'édition, etc).
- Délais d'édition du dossier (délai d'instruction, de décision, d'édition).
- Autres données (identifiant des employés chargés du dossier, code postal du bien financé, recours à un prescripteur, etc).

Le tableau 7.1 fournit une liste exhaustive de l'ensemble des données crédit disponibles et est disponible en annexe A2.

3.3 Analyse Exploratoire

3.3.1 Premier pré-traitement des données

Avant de réaliser l'analyse exploratoire, nous effectuons un premier pré-traitement des données. Tout d'abord, nous supprimons les colonnes redondantes, remplaçons les variables qui s'y prêtent par des variables binaires et mettons certaines valeurs manquantes à 0 dans le cas où une donnée manquante indiquerait en réalité une valeur nulle. Enfin, nous sélectionnons uniquement les lignes correspondant à des dossiers qui ont transité par le back-office. A la fin des étapes ci-dessus, nous passons d'un jeu de donnée de taille initiale 61014×36 à un jeu de données de taille 42659×28 .

Par la suite, nous traitons les variables correspondant aux dates. Le type `Timestamp` de la librairie `pandas` leur est affecté, le jeu de données est trié par dates croissantes et les valeurs manquantes (présentées dans la figure 3.3) sont imputées à l'aide d'une interpolation linéaire.

Enfin, nous calculons la variable réponse `NB_DOSS_DAY`. `NB_DOSS_DAY` est le nombre de dossiers qui arrivent chaque jour au back-office. Cette variable est calculée en comptant le nombre de `CODOSB` (identifiant unique du dossier bancaire) par jour où les jours sont données par la variable `DATEDI` (date d'arrivée du dossier au back-office).

Après ce premier pré-traitement, nous pouvons débuter l'analyse exploratoire.

3.3.2 Variables nulles et manquantes

Le graphique figure 3.1 présente les variables quantitatives nulles et manquantes. Nous observons que les variables **DELINS** et **DELDEC** correspondant à des délais en jours sont manquantes aux mêmes lignes. Par la suite, les variables manquantes **NBASSGPE** et **NBASSEXT** sont le nombre d'assurance groupe et le nombre d'assurance externe et sont également manquantes aux mêmes lignes. Comme un client est obligé de posséder une assurance, **NBASSGPE** et **NBASSEXT** ne peuvent pas valoir toutes deux simultanément 0. Ainsi, ces valeurs manquantes ne correspondent pas à des 0.

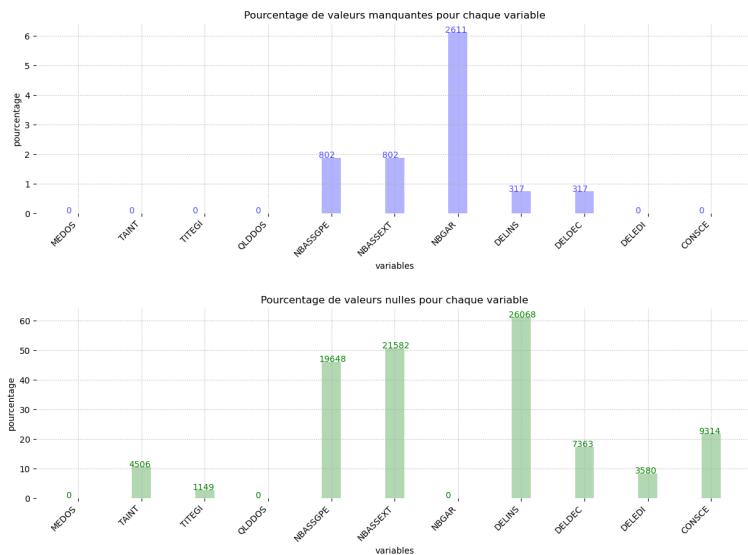


FIGURE 3.1 – Variables quantitatives nulles et manquantes

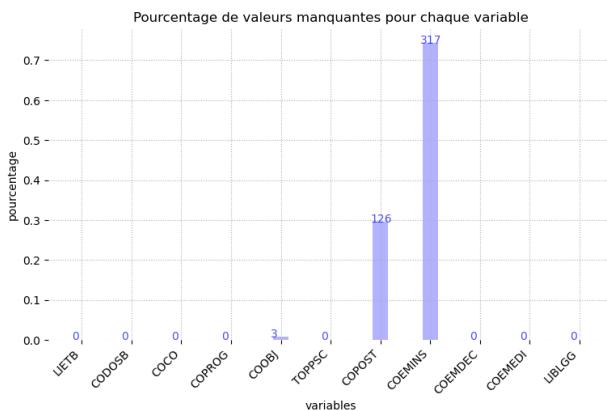


FIGURE 3.2 – Variables qualitatives manquantes

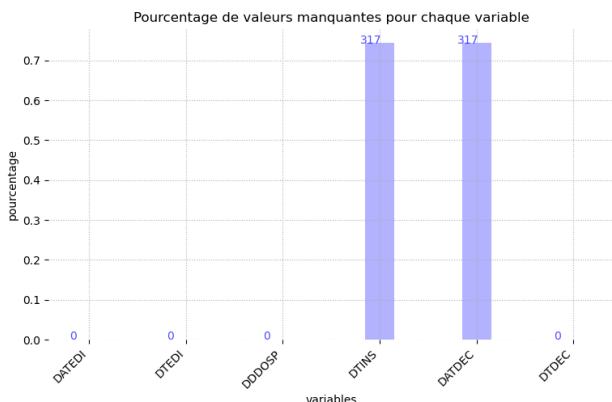


FIGURE 3.3 – Variables dates manquantes

Le graphique 3.2 montre peu de variables qualitatives manquantes : 0.5% pour **COPOST**, 0.7% pour **COEMINS** et moins de 0.1% pour **COEMEDI** et **COOBJ**. Concernant les variables dates présentées par le graphique 3.3, on observe des valeurs manquantes aux mêmes lignes pour les variables **DTINS** et **DATDEC** qui correspondent également aux lignes des valeurs manquantes pour **DELINS** et **DELDEC** (cf graphique 3.1).

3.3.3 Variables quantitatives

La figure 3.4 présente les histogrammes et box-plots des variables quantitatives afin d'observer leur répartition. On n'observe pas de variables normalement distribuées et beaucoup de variables sont des variables de comptage qui prennent des valeurs entières positives.

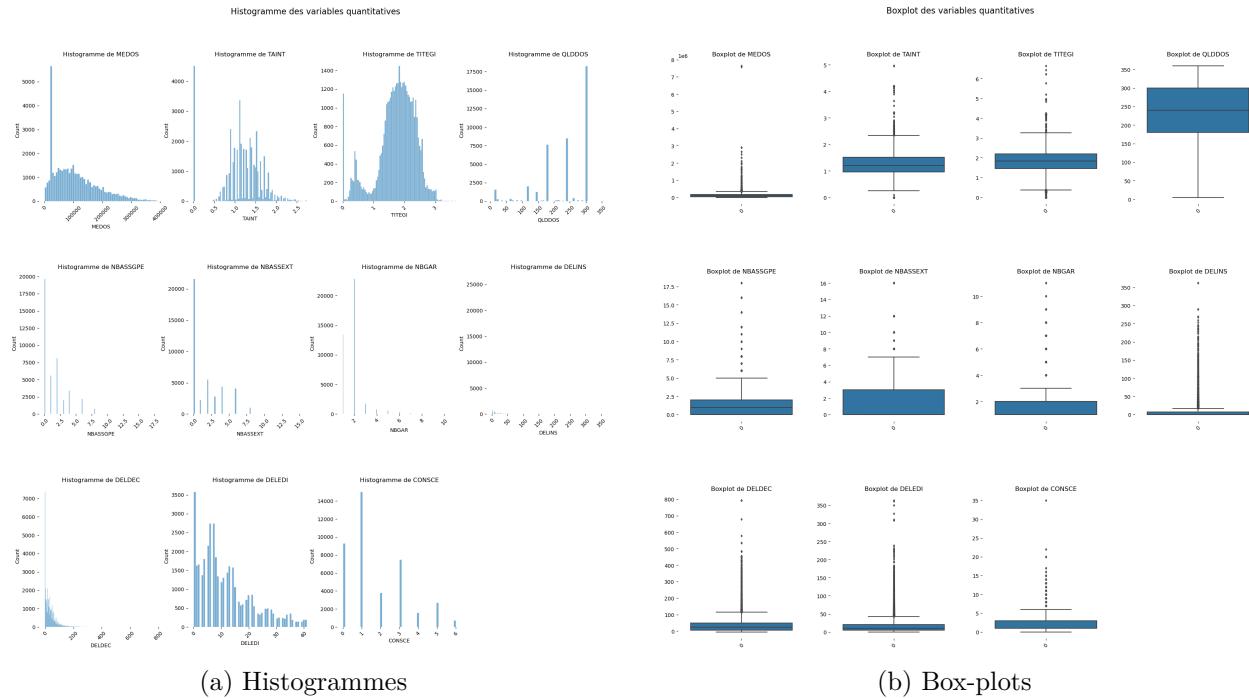


FIGURE 3.4 – Histogrammes et box-plots des variables quantitatives.

3.3.4 Variables qualitatives

Le graphique figure 3.5 montre que de nombreuses variables qualitatives ont plus d'une centaine de modalités. De nombreuses modalités est difficile à encoder et à gérer pour les algorithmes de Machine Learning. La variable CODOSB correspondant à l'identifiant du dossier bancaire possède autant de modalités que de lignes dans la base de données.

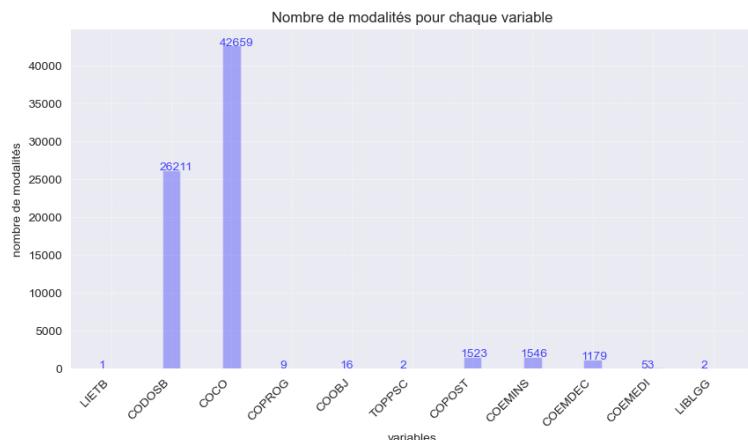


FIGURE 3.5 – Nombre de modalités pour chaque variable qualitative.

3.3.5 Dépendances entre les variables

3.3.5.1 Variables quantitatives

Afin d'observer les relations entre les variables, nous traçons la matrice de corrélation. Il existe plusieurs types de corrélations. La corrélation de Pearson dénotée r suppose une relation linéaire entre les variables et se calcule comme suit pour deux variables aléatoires X et Y : $r_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}$ où σ_X est l'écart-type de X . Cette corrélation est calculée à partir des valeurs de X et de Y et est sensible aux outliers. La corrélation de Spearman φ suppose une relation monotone entre les variables et est donc plus robuste que la corrélation de Pearson. De plus, la corrélation de Spearman est basée sur les rangs et ne suppose pas d'hypothèses sur la distribution des données. Elle se définit comme $\varphi_{R(X),R(Y)} = \frac{\text{cov}(R(X),R(Y))}{\sigma_{R(X)} \sigma_{R(Y)}}$ où $R(X_i)$ est le rang de la variable X_i dans l'échantillon (X_1, \dots, X_n) . Enfin, la corrélation Phik ϕ_k [3] est un nouveau coefficient de corrélation basé sur le test du khi-deux. Phik possède les mêmes caractéristiques que la corrélation de Pearson mais permet de mesurer la corrélation entre les variables qualitatives et quantitatives où les corrélations Spearman et Pearson traitent uniquement les variables quantitatives. Nous favoriserons donc les corrélations Spearman et Phik pour cette étude.

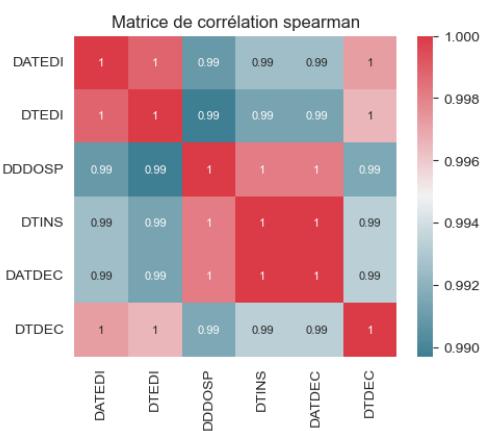


FIGURE 3.6 – Matrice de corrélation Spearman des variables dates.

Afin de pouvoir calculer la corrélation avec les variables dates, nous transformons les dates qui sont au format **Timestamp** en **floats**. De plus, après un premier calcul de corrélation, nous observons avec la figure 3.6 que les dates sont toutes corrélées entre elles avec une corrélation supérieure ou égale à 0.99. De plus, l'information contenue dans ces variables dates peut se retrouver dans les variables correspondants à des délais puisque $DELINS = DTINS - DDDOSP$, $DELDEC = DTDEC - DATDEC$ et $DELEDI = DTEDI - DATEDI$. Ceci nous permet de conserver uniquement la variable **DATEDI** et de supprimer les autres variables dates dans toute la suite du projet, sans perte d'information.

Comme certaines variables qualitatives possèdent beaucoup de modalités, la corrélation Phik ne parvient pas à s'exécuter. Nous calculons alors la matrice de corrélation Spearman et Phik uniquement pour les variables quantitatives, la variable date **DATEDI** et la variable réponse **NB_DOSS_DAY**. Ces deux graphiques sont présentés dans les figures 3.7 et 3.8.

Dans un premier temps, nous regardons les corrélations entre les variables explicatives. Le graphe 3.7 montre que les variables **TAIN** (taux d'intérêt) et **TITEGI** (taux d'intérêt prenant en compte la totalité des frais) sont corrélées positivement à 80%. Ceci semble cohérent car ces deux variables sont calculées à partir d'une même valeur. La variable **TAIN** est également corrélée avec **QLDDOS** (durée du prêt) à 40% et avec **MEDOS** (montant nominal du dossier) à 33%. De même, **NBASSGPE** (nombre d'assurances groupe) et **NBASSEXT** (nombre d'assurances externes) sont corrélées négativement à -85% ce qui semble logique car un client

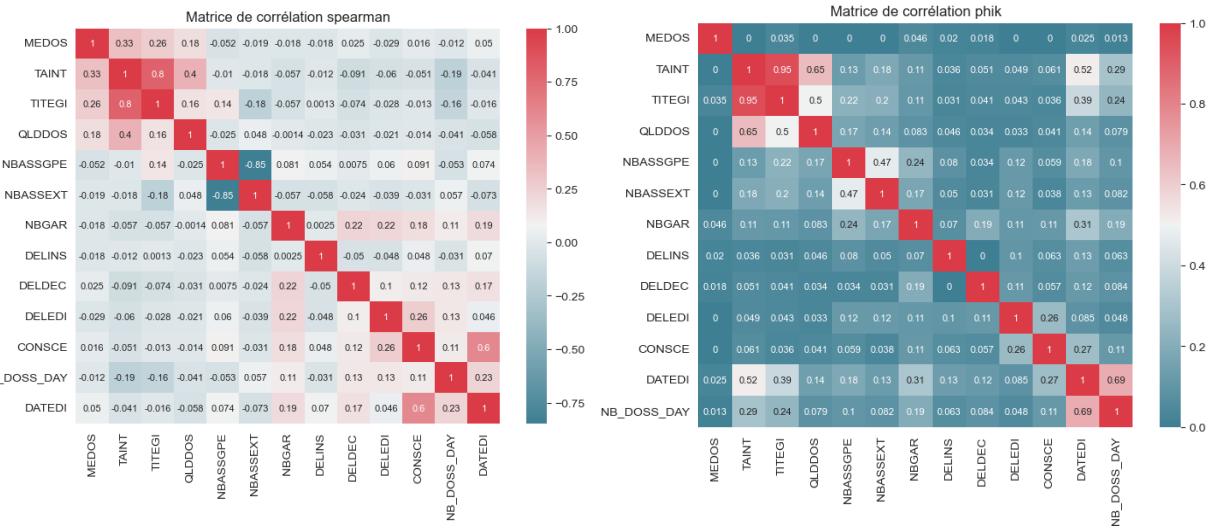


FIGURE 3.7 – Matrice de corrélation Spearman des variables quantitatives et de la variable réponse.

FIGURE 3.8 – Matrice de corrélation Phik des variables quantitatives et de la variable réponse.

possède soit une assurance groupe soit une assurance externe. La variable **CONSCE** (nombre d’interactions entre l’agence et le back-office lorsqu’un dossier n’est pas valide du premier coup) est corrélée avec **DATEDI** (date d’édition) à 60% et avec **DELEDI** (nombre de jours édition) à 26%. Ceci s’explique par le fait que plus un dossier transite entre l’agence et le back-office et plus il met de temps à être édité. Par la suite, on observe des corrélations plus faibles : **NBGAR** (nombre de garanties) est corrélé avec **DELDEC** (nombre de jours décision) et **DELEDI** (nombre de jours édition) à 22%, **DATEDI** à 19% et **CONSCE** (nombre de scénarios) à 25%. Le graphe 3.8 montre des corrélations plus faibles entre les variables.

Concernant la variable réponse **NB_DOSS_DAY**, on constate qu’elle est faiblement corrélée avec toutes les autres variables explicatives à l’exception de **DATEDI** à 23%. Cependant, ceci est cohérent car **NB_DOSS_DAY** est calculée à partir de **DATEDI**.

3.3.5.2 Variables qualitatives

Afin d’illustrer les possibles dépendances entre les variables qualitatives et la variable réponse **NB_DOSS_DAY**, nous traçons les boxplots et la matrice de corrélation Phik présentés dans les figures 3.9 et 3.10 uniquement pour les variables qualitatives possédant peu de modalités.

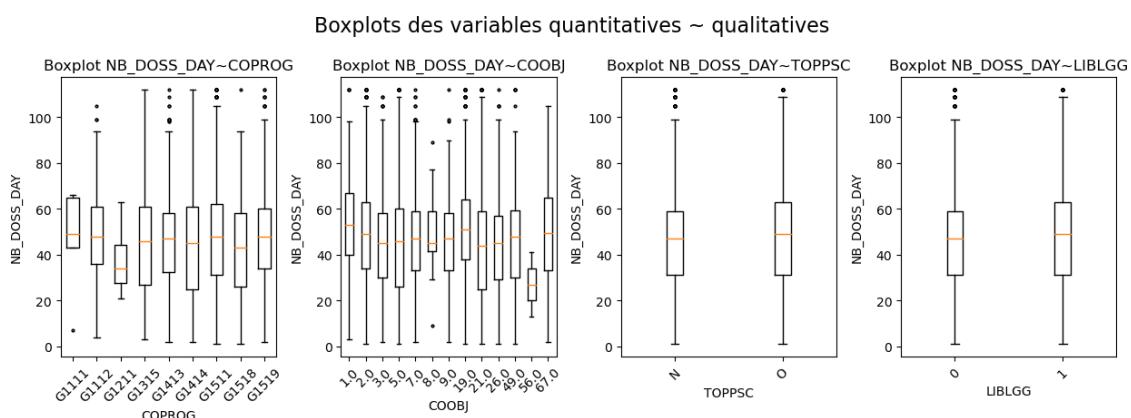


FIGURE 3.9 – Boxplots entre les variables qualitatives avec peu de modalités et la variable réponse.

Concernant les variables qualitatives avec de nombreuses modalités, nous traitons ces variables comme des variables numériques et traçons la matrice de corrélation présentée dans la figure 3.11.

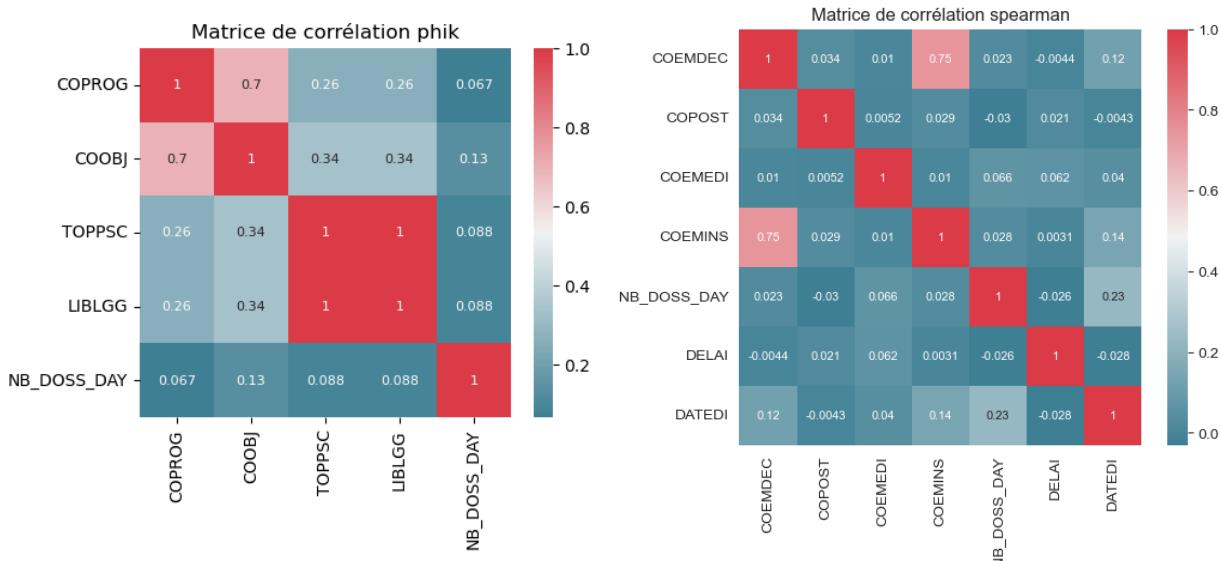


FIGURE 3.10 – Matrice de corrélation Phik des variables qualitatives avec peu de modalités et de la variable réponse.

FIGURE 3.11 – Matrice de corrélation Spearman des variables qualitatives avec de nombreuses modalités et de la variable réponse.

Nous observons ici aussi une faible dépendance entre ces variables qualitatives et la variable réponse NB_DOSS_DAY.

Compte tenu des faibles corrélations entre l'ensemble des variables explicatives et la variable réponse, nous pouvons émettre l'hypothèse qu'une approche multivariée où l'on cherche à prédire NB_DOSS_DAY avec les variables explicatives pourrait ne pas donner de résultats satisfaisants. Nous testons néanmoins l'approche multivariée.

3.3.6 Sélection de variables

A l'issu de cette analyse exploratoire, nous pouvons sélectionner les variables explicatives que nous utiliserons par la suite pour prédire NB_DOSS_DAY dans le cadre d'une approche multivariée. Les variables qualitatives ayant beaucoup de modalités comme COEMDEC, COEMEDI, COEMINS (identifiant employé décision, édition, instruction), COPOST (code postal), COCO (identifiant du contrat) et CODOSB (identifiant du dossier bancaire) ne sont pas très corrélées avec la variable réponse. Comme l'encodage des variables qualitatives avec beaucoup de modalités peut entraîner la création de nombreuses colonnes supplémentaires difficiles à traiter pour les algorithmes, nous supprimons ces variables. Par la suite, comme évoqué avec le graphe 3.6, nous ne conservons que DATEDI parmi toutes les variables dates. Enfin, nous supprimons la variable LIETB qui est constante.

3.3.7 Imputation des données manquantes

Une fois que nous avons sélectionné les variables explicatives d'intérêt, nous pouvons imputer le reste des données manquantes. On observe des valeurs manquantes pour NBGAR

(nombre de garanties) et aucune valeur nulle. On en déduit que lorsqu'un client n'a pas de garanties, la variable est manquante. On met donc toutes les variables manquantes de **NBGAR** à 0. On impute les variables manquantes de **DELINS** et **DELDEC** avec les calculs $\text{DELINS} = \text{DTINS} - \text{DDD0SP}$ et $\text{DELDEC} = \text{DTDEC} - \text{DATDEC}$ (où les variables dates ont déjà été interpolées dans la section 3.3.1). **COOBJ** (code objet prêt) possède très peu de valeurs manquantes que l'on impute par la modalité majoritaire. Enfin, **NBASSGPE** (nombre assurances groupe) et **NBASSEXT** (nombre assurances externes) ont des valeurs manquantes aux même lignes. Or, un client est supposé posséder soit une assurance de groupe, soit une assurance externe. Pour un ligne dont **NBASSGPE** et **NBASSEXT** sont manquantes, on tire aléatoirement avec probabilité $\frac{1}{2}$ une assurance, par exemple **NBASSGPE** dont on impute par la valeur moyenne, puis on met la valeur de **NBASSGPE** à 0.

3.4 Méthodes utilisées

3.4.1 Mise en forme des jeux de données

Pour ce projet, nous allons utiliser deux approches. La première approche par série temporelle univariée est la plus simple et la plus documentée. Il s'agit de prédire l'évolution d'une seule variable (**NB_DOSS_DAY**) dans le temps. La deuxième approche, plus complexe, est l'approche multivariée où l'on prédit l'évolution de la variable **NB_DOSS_DAY** au cours du temps à l'aide de plusieurs variables explicatives interdépendantes. L'approche multivariée considère donc plusieurs séries temporelles univariées interdépendantes.

Pour constituer le jeu de données pour l'approche univariée, nous considérons uniquement la variable **NB_DOSS_DAY**. Nous construisons ainsi une série temporelle nommée **TS_day** contenant pour chaque jour, le nombre de dossiers qui arrivent au back-office. **TS_day** vaut 0 si aucun dossier n'arrive. Le besoin métier étant de savoir le nombre de dossiers qui arrivent à la semaine, nous créons pour comparaison, une autre série temporelle **TS_week** contenant le nombre de dossiers qui arrivent au back-office par semaine. Les résultats obtenus sont présentés dans les graphes 3.12 et 3.13 et illustrés par un schéma figure 3.15.

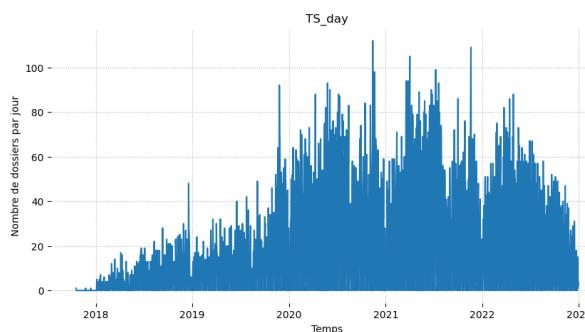


FIGURE 3.12 – Nombre de dossiers de crédits par jour (TS_day).

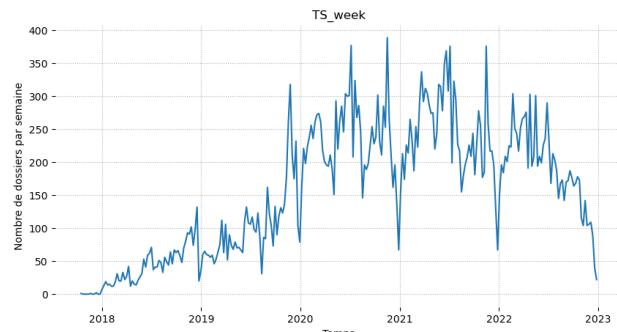


FIGURE 3.13 – Nombre de dossiers de crédits par semaine (TS_week).

Pour constituer le jeu de données pour l'approche multivariée, on observe avec la table 3.1 que plusieurs dossiers de crédits peuvent être édités pour la même date. Or, pour avoir une série temporelle, nous ne pouvons pas avoir de doublons dans les dates. Nous allons

donc agréger les données correspondant à une même date en prenant la classe majoritaire pour les variables qualitatives et la valeur moyenne pour les variables quantitatives. Un autre problème est que les données temporelles n'ont pas un pas de temps constant car nous n'avons pas des dossiers tous les jours. Alors que pour l'approche univariée nous pouvions rajouter des dates avec `NB_DOSS_DAY` à 0 afin d'avoir un pas de temps constant, nous ne pouvons pas ajouter des dates pour l'approche multivariée. En effet, cela supposerait de connaître les valeurs des autres variables explicatives aux dates où il n'y a pas de dossiers. Nous rencontrons le même problème en agrégeant les données à la semaine car il existe certaines semaines où aucun dossier n'est édité. Nous décidons de poursuivre tout de même l'étude multivariée en conservant les données regroupées par jour et en faisant l'hypothèse que les données ont un pas de temps constant. La table créée pour l'approche multivariée s'intitule `credits_MTS` et est illustrée par le schéma figure 3.14.

Index	DATEDI	TAIN	...	NB_DOSS_DAY
0	19/10/2017	2.4266	...	1
1	24/11/2017	1.6625	...	1
2	13/12/2017	1.5104	...	1
3	14/12/2017	1.8146	...	1
4	05/01/2018	0.0000	...	5
...
42654	29/12/2022	2.1305	...	4
42655	29/12/2022	1.8658	...	4
42656	29/12/2022	2.4777	...	4
42657	30/12/2022	2.6312	...	2
42658	30/12/2022	2.2938	...	2

TABLE 3.1 – Jeu de données à l'issu de l'analyse exploratoire.

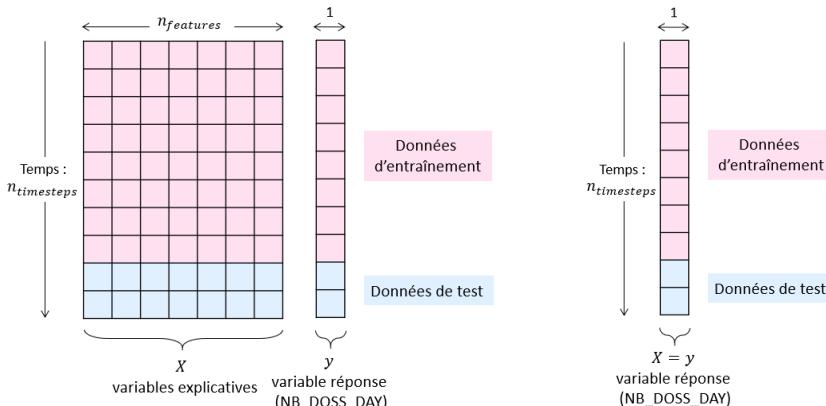


FIGURE 3.14 – `credits_MTS` : une série temporelle multivariée.

FIGURE 3.15 – `TS_week` et `TS_day` : une série temporelle univariée.

Nous pouvons à présent débuter la prédiction de `NB_DOSS_DAY`.

3.4.2 Débruitage des séries temporelles univariées

Les graphiques 3.12 et 3.13 montrent des signaux très bruités, notamment pour la série TS_day. Pour tenter d'améliorer les résultats nous débruitons les signaux univariés TS_day et TS_week à l'aide de la décomposition en Ondelettes.

Nous commençons par estimer l'écart-type du bruit. Pour ce faire, nous nous basons sur le papier [4] et sur le cours [5], et calculons l'écart-type du bruit $\sigma = \frac{\text{median}(|W|)}{m}$ où $m = \sqrt{2}\text{erf}^{-1}(0.5)$, avec **erf** la fonction d'erreur et W le vecteur des coefficients d'Ondelettes du signal d'entrée à l'échelle la plus fine. Le bruit estimé est supposé gaussien.

Par la suite, nous effectuons un seuillage doux en Ondelettes pour débruiter le signal d'entrée. En effet, le bruit d'un signal est généralement concentré sur les coefficients d'ondelettes de détails les plus précis [5]. Nous calculons donc les coefficients d'Ondelettes du signal d'entrée et conservons uniquement les coefficients qui, en valeur absolue, sont supérieurs à un certain seuil T . Nous effectuons ensuite la transformée inverse pour reconstruire le signal débruité. Le seuillage doux en Ondelettes s'écrit : $\hat{S} = \sum_{k \in I_T}^N \langle s, e_k \rangle e_k$ où \hat{S} est une approximation du signal d'entrée s et représente le signal débruité, et les $(e_k)_k$ sont les vecteurs constituant la base d'Ondelettes $\mathcal{B} = \{e_k \in \mathbf{R}^N; k = \llbracket 0, N - 1 \rrbracket\}$ à l'échelle la plus fine. L'ensemble I_T désigne l'ensemble des indices k tels que $|\langle s, e_k \rangle| > T$ où le seuil $T = \sigma \times c$ avec c un paramètre du débruitage.

De plus, la performance du débruitage par seuillage doux en Ondelettes peut être améliorée en utilisant le fait que la transformée en Ondelettes n'est pas invariante par translation [5]. Ainsi, si on effectue une translation circulaire sur les composantes d'un vecteur, on modifie l'amplitude des coefficients d'Ondelettes. On peut exploiter cette propriété pour le débruitage en translatant le signal d'entrée, en appliquant le seuillage doux pour reconstruire le signal débruité translaté puis en effectuant la translation inverse. Le signal débruité final est obtenu en faisant la moyenne des signaux translatés pour différentes valeurs de translations.

Enfin, nous implémentons un Dashboard interactif présenté figure 3.16 avec les librairies Python **panel** et **ipywidgets** pour pouvoir régler les paramètres du débruitage. La transformée en Ondelettes est calculée avec les librairies **pywt** et **scipy**.

Plusieurs bases d'Ondelettes sont disponibles avec **pywt**. En faisant jouer les paramètres, nous conservons pour TS_day la base **db3** (Daubechies avec 3 moments nuls), le seuil $c = 0.5$ et 8 translations. Pour la série TS_week, nous sélectionnons la base **sym11** (Symlet avec 11 moments nuls), le seuil $c = 1.5$ et 8 translations.

Nous débruitons les séries TS_day et TS_week avec les paramètres sélectionnés ci-dessus avant d'appliquer la méthode SARIMA section 3.4.3.

3.4.3 Modélisation par la méthode SARIMA

Nous commençons par appliquer un modèle simple univarié pour la prédiction de NB_DOSS_DAY. Comme le besoin métier est à la semaine et que la série TS_day semble très bruitée et sans motifs reconnaissables, nous décidons de travailler avec TS_week construite dans la section 3.4.1.

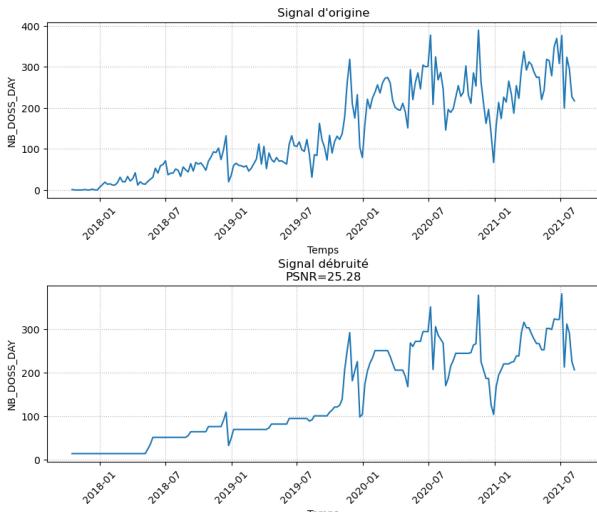


FIGURE 3.16 – Graphiques issus du Dashboard lorsque l’option `TS_week` est sélectionnée.

D’après l’ouvrage [6], l’approche la plus simple pour travailler avec des données hebdomadaires consiste à utiliser une décomposition STL puis une méthode non saisonnière appliquée aux données désaisonnalisées (pour lesquelles on a retiré la saisonnalité). Nous allons donc suivre la démarche proposée et en détailler les étapes ci-dessous.

3.4.3.1 Décomposition STL

L’objectif principal de l’analyse d’une série temporelle est la prévision de ses futures réalisations. Afin de réaliser cet objectif, une première étape consiste à décomposer la série temporelle en plusieurs composantes, où chaque composante représente un modèle sous-jacent. Une série temporelle y_t est communément décomposée en une tendance m_t correspondant à l’évolution à long terme de la série, une saisonnalité s_t correspondant à un phénomène périodique de période identifiée et un résidu ou erreur r_t , idéalement un bruit blanc, qui est la seule partie aléatoire de la décomposition.

Cette décomposition peut être additive $y_t = m_t + s_t + r_t$ ou multiplicative $y_t = m_t \times s_t \times r_t$. D’après [6], une décomposition additive est appropriée si les fluctuations saisonnières ne varient pas avec le niveau de la série temporelle et une décomposition multiplicative est utilisée si la variation des fluctuations saisonnières semble être proportionnelle au niveau de la série temporelle. Avec le graphe 3.13 de la série `TS_week`, on observe que la saisonnalité semble varier avec le niveau de la série : les fluctuations saisonnières sont plus importantes lorsque la tendance est élevée autour des années 2020-2022 tandis qu’elles sont moins importantes lorsque la tendance est faible autour des années 2018-2019 et 2023. Comme recommandé par [6], nous sélectionnons donc une décomposition multiplicative que nous réaliserons avec la méthode STL de la librairie python `statsmodels`.

La méthode STL [7] est une méthode robuste pour la décomposition de séries temporelles et comprend de nombreux avantages par rapport à d’autres méthodes de décomposition plus classiques. Elle traite tout type de saisonnalités qui peuvent changer au cours du temps et est robuste aux valeurs aberrantes. Cependant, la décomposition STL traite uniquement les décompositions additives. Il est possible d’obtenir une décomposition multiplicative en

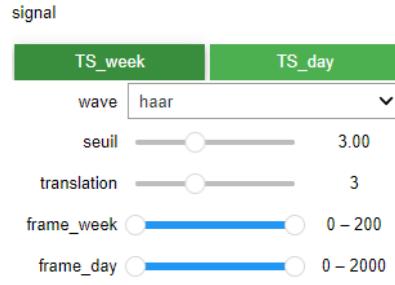


FIGURE 3.17 – Dashboard pour choisir les paramètres du débruitage.

tenant d'abord le log des données puis en faisant une décomposition additive car $y_t = m_t \times s_t \times r_t$ est équivalent à $\log(y_t) = \log(m_t) + \log(s_t) + \log(r_t)$.

La transformation Box-Cox [8] permet d'obtenir une décomposition entre la décomposition multiplicative et additive pour $0 < \lambda < 1$ et supporte les données négatives ou nulles, ce qui n'est pas le cas du log. Elle se définit comme suit :

$$w_t = \begin{cases} \log(y_t) & \text{si } \lambda = 0 \\ \text{sign}(y_t)(|y_t|^\lambda - 1)/\lambda & \text{sinon} \end{cases} \quad (3.1)$$

où w_t est la série transformée. La transformation inverse s'écrit :

$$y_t = \begin{cases} \exp(w_t) & \text{si } \lambda = 0 \\ \text{sign}(\lambda w_t + 1)|\lambda w_t + 1|^{1/\lambda} & \text{sinon} \end{cases} \quad (3.2)$$

Nous utilisons donc la transformation Box-Cox (3.1) sur $y_t = \text{TS_week}$ pour obtenir w_t et présentons les résultats obtenus dans le graphe 3.18. La valeur $\lambda = 0.1$ est choisie telle que la taille de la variation saisonnière soit la plus identique possible dans toute la série. Comme les données autour de l'année 2018 semblent incohérentes et pourraient brouter les prédictions, nous retirons arbitrairement les 110 premières valeurs de TS_week pour la suite de l'étude.

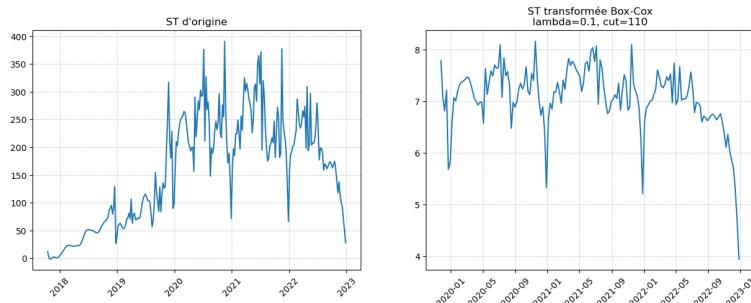


FIGURE 3.18 – Transformation Box-Cox avec $\lambda = 0.1$. A gauche y_t et à droite la série transformée w_t .

Nous pouvons à présent appliquer la décomposition STL additive sur la série transformée $w_t = m_t + s_t + r_t$. Un des paramètres de la fonction STL, est la période (nombre de fois que le cycle saisonnier se répète en un an). Afin d'identifier ce paramètre, nous appliquons la transformée de Fourier de la librairie `scipy` à w_t . Ainsi, nous observons dans la figure 3.19 un pic de fréquence à $f = 0.06$ [1/semaine] indiquant que le signal a une période de $1/f = 16$ semaines. Le résultat de la décomposition additive STL sur w_t est présenté dans la figure 3.20.

Nous observons sur le graphe 3.20 que les résidus r_t semblent centrés autour de 0 et sans motif particulier (bruit blanc). Ceci implique que toute l'information est bien contenue dans m_t et s_t . En effectuant la décomposition STL sans la transformation Box-Cox au préalable, nous obtenions des résidus avec des motifs.

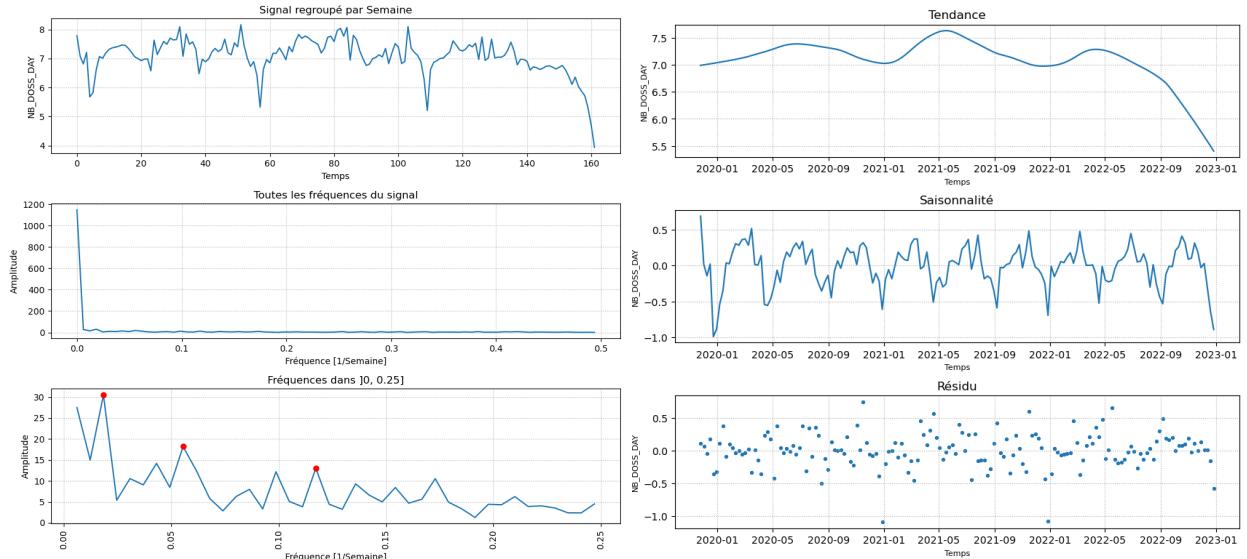


FIGURE 3.19 – Décomposition dans le domaine de Fourier de w_t .

La décomposition STL permet d'extraire la composante saisonnière s_t que nous pouvons ensuite retirer à w_t comme mentionné par [6]. Ainsi, nous appliquerons une méthode prédictive sur les données désaisonnalisées $\tilde{w}_t = w_t - s_t = m_t + r_t$. Le graphe figure 3.21 montre la série temporelle \tilde{w}_t .

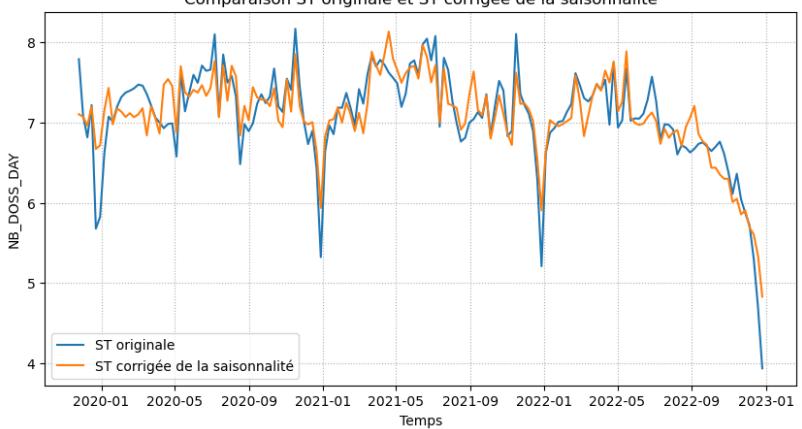


FIGURE 3.20 – Décomposition STL $w_t = m_t + s_t + r_t$ avec une période = 16.

FIGURE 3.21 – Série w_t et série corrigée de la saisonnalité \tilde{w}_t .

3.4.3.2 Application du modèle ARIMA et SARIMA.

Présentation théorique des modèles

Le modèle ARIMA est un modèle non-saisonnier classique et populaire pour la prédiction des séries temporelles univariées que nous utiliserons comme premier modèle de base, comme suggéré par [6]. ARIMA(p, d, q) est une combinaison de 3 modèles :

- La partie Integrated $\mathbf{I}(d)$ indique qu'il faut différencier la série originale d fois afin d'éliminer un éventuel caractère non-stationnaire. Une série temporelle est stationnaire si les propriétés de la série ne dépendent pas du moment où elle est observée. Ainsi, les séries temporelles avec tendance ou saisonnalité ne sont pas stationnaires tandis qu'une série de bruit blanc l'est. Si nous définissons par B l'opérateur de différenciation (ou "lag") tel que $By_t = y_{t-1}$, la série différenciée d fois s'écrit $y'_t = (1 - B)^d y_t$.
- Le modèle Auto Regressive (auto-régressif) $\mathbf{AR}(p)$ prédit la variable d'intérêt en utilisant une combinaison linéaire des valeurs passées de la variable. Un modèle auto-régressif d'ordre p s'écrit : $y'_t = c + \sum_{i=1}^p \phi_i y'_{t-i} + \varepsilon_t$ où ϕ_i sont des coefficients, y'_t est la série différenciée d'ordre

d , ε_t est un bruit blanc gaussien et c une constante.

- Le modèle Moving Average (moyenne mobile) $\text{MA}(q)$ prédit la variable d'intérêt en utilisant les erreurs de prédition passées de la variable. Un modèle moving-average d'ordre q s'écrit : $y'_t = c + \sum_{i=1}^q \theta_i \varepsilon_{t-i} + \varepsilon_t$ où θ_i sont des coefficients, ε_t est un bruit blanc gaussien. Au final, le modèle ARIMA(p, d, q) sur la série différenciée $y'_t = (1 - B)^d y_t$ s'écrit :

$$y'_t = c + \sum_{i=1}^p \phi_i y'_{t-i} + \sum_{i=1}^q \theta_i \varepsilon_{t-i} + \varepsilon_t. \quad (3.3)$$

Cependant, après plusieurs tests, nous avons constaté que le modèle ARIMA ne donne pas des résultats satisfaisants. De plus, on observe avec le graphe 3.21 que la série temporelle corrigée de la saisonnalité \tilde{w}_t comporte toujours des variations saisonnières complexes. Nous appliquons donc le modèle saisonnier SARIMA (Seasonal ARIMA). Il s'agit d'une extension du modèle ARIMA qui permet de prendre en compte la composante saisonnière dans la modélisation. Le modèle SARIMA($p, d, q)(P, D, Q, m$) reprend les mêmes paramètres (p, d, q) que ARIMA et ajoute quatre autres paramètres : P l'ordre de la partie auto-régressive saisonnière, D l'ordre de la différence saisonnière, Q l'ordre de la moyenne mobile saisonnière et m la période de la composante saisonnière. Le modèle SARIMA sur la série différenciée $y'_t = (1 - B)^d y_t$ s'écrit :

$$y'_t = c + \sum_{i=1}^p \phi_i y'_{t-i} + \sum_{i=1}^q \theta_i \varepsilon_{t-i} + \sum_{i=1}^P \alpha_i y'_{t-im} + \sum_{i=1}^Q \beta_i \varepsilon_{t-im} + \varepsilon_t. \quad (3.4)$$

Jeu d'entraînement et jeu de test

La série temporelle \tilde{w}_t est séparée en 2 sous-ensembles : la série temporelle d'apprentissage `TS_train` utilisée pour entraîner le modèle et la série temporelle de test `TS_test` utilisée pour mesurer la précision de la prédition. Contrairement à un problème de Machine Learning classique, la séparation du jeu de données ne se fait pas de façon aléatoire puisque l'on doit conserver la dépendance temporelle des observations. Nous prenons donc 80% des premières valeurs de \tilde{w}_t pour constituer la série d'entraînement `TS_train` et les valeurs restantes formeront la série de test `TS_test`.

Choix des paramètres

Nous nous intéressons à présent à l'identification des 7 paramètres du modèle SARIMA. La figure 3.19 nous aide à déterminer la période de la composante saisonnière $m = 16$. Pour déterminer les autres paramètres, nous traçons figure 3.22 les graphes ACF (auto-correlation function) et PACF (partial ACF) de la série `TS_train` à l'aide la librairie `statsmodels`. Notons que l'identification des paramètres se fait en analysant uniquement la série temporelle d'apprentissage.

Le graphique ACF montre les auto-corrélations qui mesurent la relation entre y_t et y_{t-k} pour différentes valeurs de k . Or si y_t et y_{t-1} sont corrélés, alors y_{t-1} et y_{t-2} sont aussi probablement corrélés. Cependant, y_t et y_{t-2} peuvent être corrélés car ils sont tous deux reliés à y_{t-1} plutôt que parce que y_{t-2} contient une nouvelle information pertinente. Pour pallier ce problème, le graphe PACF mesure la relation y_t et y_{t-k} en enlevant les effets des lags 1, 2, 3, ..., $k - 1$.

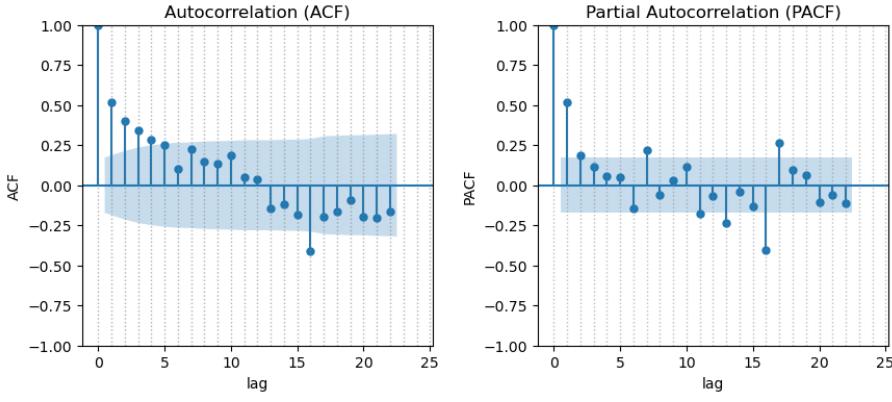


FIGURE 3.22 – ACF et PACF de la série temporelle \tilde{w}_t .

- Choix de d : pour savoir si la série est suffisamment différenciée, nous réalisons le test ADF (Augmented Dickey-Fuller) [9] de `statsmodels` où l'hypothèse nulle est que la série temporelle est stationnaire. Nous obtenons une p-valeur = 0.1 > 0.05 indiquant que l'on ne rejette pas l'hypothèse nulle et donc que la série est non-stationnaire et requiert une différenciation. Cependant, la p-valeur est faible et d'après [10], un pic négatif ou nul au lag-1 du graphe ACF indique que l'on ne doit pas différencier davantage la série. Nous décidons de conserver $d = 0$ (pas de différenciation).
- Choix de p : des termes **AR** doivent être ajoutés si l'on observe une décroissance lente du graphe ACF couplée avec une décroissance rapide du PACF et/ou si l'on voit un pic positif au lag-1 du graphe ACF. La figure 3.22 montre que le graphe ACF décroît lentement tandis que PACF décroît brutalement, et nous ajoutons donc des termes **AR** au modèle. Sur le PACF, on voit un pic significatif (positif ou négatif) en dehors de l'intervalle de confiance au lag 7 et d'autres pics autour des lags $m = 16$ qui sont liés à la période saisonnière. Ceci signifie que l'on doit ajouter $p = 7$ termes **AR**.
- Choix de q : des termes **MA** doivent être ajoutés si l'on observe une décroissance rapide du graphe ACF couplée avec une décroissance lente du PACF et/ou si l'on voit un pic négatif au lag-1 du graphe ACF. La figure 3.22 montre un pic négatif au lag-1 du graphe ACF, et nous ajoutons donc des termes **MA** au modèle. Sur l'ACF, on voit un pic significatif (positif ou négatif) en dehors de l'intervalle de confiance au lag 4, ce qui signifie que l'on devrait ajouter $q = 4$ termes **MA**. Cependant, il est déconseillé de trop mélanger les termes **MA** et **AR** car ils sont susceptibles de se compenser. Après plusieurs tests, nous décidons d'ajouter $q = 2$ termes **MA**.

Nous identifions à présent les trois termes saisonniers restants (P, D, Q).

- Choix de m : nous avions choisi $m = 16$ et ceci est confirmé par l'observation d'un pic significatif au lag 16, lié à la période de la saisonnalité.
- Choix de D : comme nous n'avons pas utilisé de différenciation saisonnière $y'_t = (1 - B^m)y_t$ pour rendre la série temporelle stationnaire, nous laissons $D = 0$.
- Le choix de P s'apparente au choix de p . Cependant, au lieu de compter le nombre de pics qui sortent de l'intervalle de confiance du PACF, on compte le nombre de pics *saisonniers* qui sortent de l'intervalle de confiance. Ainsi, à partir du lag $m = 16$, on observe 2 pics significatifs au lags 16 et 17 et on ajoute donc $P = 2$ termes **SAR**.
- Le choix de Q s'apparente au choix de q . Cependant, au lieu de compter le nombre de pics qui sortent de l'intervalle de confiance du ACF, on compte le nombre de pics *saisonniers*

qui sortent de l'intervalle de confiance. On voit uniquement 1 pic au lag $m = 16$ lié à la saisonnalité. On n'ajoute pas de termes **SMA** au modèle ($Q = 0$).

Ainsi, le modèle final s'écrit SARIMA(7, 0, 2)(0, 0, 2, 16).

Il existe une librairie python `pmdarima` permettant d'identifier automatiquement les paramètres du modèle SARIMA. Cependant, après plusieurs tests, les paramètres trouvés avec `pmdarima` ne donnent pas des résultats satisfaisants et nous avons décidé de poursuivre avec l'identification manuelle des paramètres.

Entraînement du modèle et prédictions

L'entraînement du modèle SARIMA(7, 0, 2)(0, 0, 2, 16) sur la série `TS_train` utilise la fonction `arima.model` de la librairie `statsmodel`. Les prédictions sont calculées sur la série `TS_test`.

Une fois les prédictions réalisées, nous appliquons la transformation Box-Cox inverse [3.2](#) sur les données prédites afin de pouvoir les comparer avec les véritables valeurs. Pour quantifier l'erreur entre les données réelles et les données prédites, nous utilisons la métrique *RMSE* qui évalue la performance de prédiction du modèle et pénalise fortement les valeurs aberrantes :

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2} \quad (3.5)$$

où y_i est la i -ième valeur réelle de la série `TS_test` et \hat{y}_i est la valeur prédite. Plus le RMSE est faible et meilleure est la prédiction. Nous calculons aussi la MAE entre les données réelles et les données prédites. La MAE est la moyenne des erreurs et permet de savoir si le modèle représente bien la tendance principale des données :

$$MAE = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i| \quad (3.6)$$

Plus la MAE est faible et meilleure est la prédiction. Nous présentons les résultats obtenus dans la section [3.5.1](#).

3.4.4 Modélisation par l'algorithme MiniRocket

Comme le modèle SARIMA testé dans la section précédente ne donne pas des résultats satisfaisants (voir section [3.5.1](#)), nous utilisons une méthode de deep learning plus complexe avec l'algorithme MiniRocket (2021) [\[11\]](#).

La plupart des méthodes donnant des résultats de pointe pour la prédiction des séries temporelles ont une complexité de calcul élevée. MiniRocket est basé sur l'algorithme Rocket (2019) [\[12\]](#), qui permet d'effectuer la classification et la régression de séries temporelles univariées ou multivariées. L'avantage de Rocket est qu'il atteint une précision presque identique aux méthodes de deep learning de pointe plus complexes en un temps de calcul beaucoup plus court. Naturellement parallèle, il peut être entraîné sur un seul cœur CPU ce qui est utile dans notre cas car la CEMP ne possède pas de GPUs. L'algorithme MiniRocket

est une reformulation de Rocket et a été choisi pour ce projet car il est plus rapide que Rocket tout en conservant la même précision.

Présentation théorique du modèle

Bien que très intéressant, il n'est pas nécessaire de connaître le fonctionnement des algorithmes Rocket et MiniRocket pour comprendre la suite de ce rapport. Une description brève de ces algorithmes est disponible en Annexe A3.

Pour l'implémentation de MiniRocket, nous nous baserons sur le tutoriel fourni par les auteurs de MiniRocket [13] qui utilise les librairies python `tsai` et `fastai`.

3.4.4.1 Approche multivariée

Dans un premier temps, nous testons l'approche multivariée en utilisant la table `credits_MTS` qui est une série temporelle multivariée, construite dans la section 3.4.1.

One-hot encoding

Tout d'abord, nous encodons les variables qualitatives de `credits_MTS` en variables numériques en utilisant le one-hot encoding. Il n'est pas nécessaire de normaliser la série temporelle d'entrée avec MiniRocket.

Application de fenêtres glissantes

Pour que la série temporelle multivariée ait le bon format pour MiniRocket, nous appliquons des fenêtres glissantes comme détaillé dans le tutoriel [14] en utilisant la fonction `SlidingWindow` de la librairie `tsai`. Ainsi, nous appliquons sur notre série temporelle $n_{windows}$ fenêtres glissantes de taille l_{window} chacune. Les fenêtres se déplacent d'une distance $stride$ à chaque pas, comme illustré dans la figure 3.23.

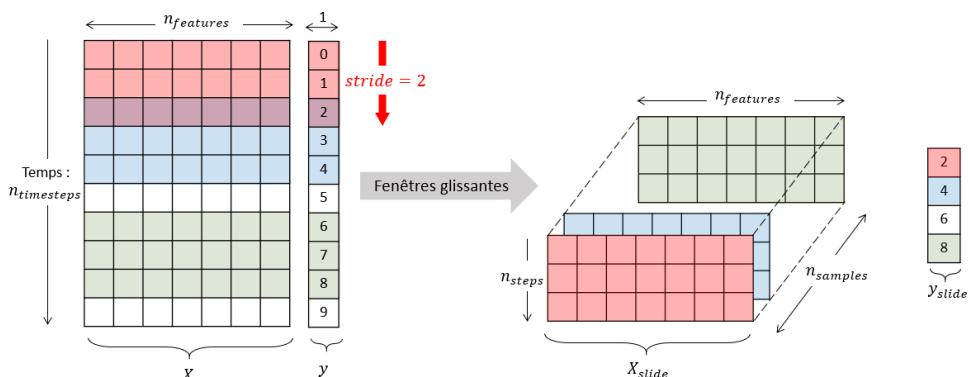


FIGURE 3.23 – Illustration des fenêtres glissantes pour $l_{window} = 3$ et $stride = 2$.

Pour les variables explicatives X (voir figure 3.14), les fenêtres glissantes permettent de passer d'un format $X = (n_{timesteps}, n_{features})$ à un format $X_{slide} = (n_{samples}, n_{features}, n_{steps})$ compréhensible par MiniRocket. La transformation est automatiquement calculée par la fonction `SlidingWindow` comme suit : $n_{steps} = l_{window}$ et $n_{samples} = n_{windows} = (n_{timesteps} - l_{window})//stride + 1_{[n_{timesteps} \text{ est pair}]}$ où $//$ est la division Euclidienne. De même, la réponse y passe du format $y = (n_{timesteps})$ au format $y_{slide} = (n_{samples})$. Pour la création de y_{slide} , `SlidingWindow` prend une valeur de y toutes les $stride$ valeurs. Par exemple, si $y = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]$ avec $l_{window} = 3$ et $stride = 2$ alors $y_{slide} = [2, 4, 6, 8, 10, 12]$.

Après plusieurs tests, nous adoptons une taille de fenêtre $l_{window} = 9$ et un $stride = 1$ ce qui permet de prédire un résultat tous les jours.

Jeu d'entraînement et jeu de test

Nous utilisons la fonction `get_splits` de `tsai` qui effectue automatiquement la séparation du jeu de données (X_{slide}, y_{slide}) en jeu d'entraînement (80% des premières valeurs) et en jeu de test (20% des dernières valeurs).

Entraînement du modèle et prédictions

L'apprentissage sur le jeu d'entraînement créé par `get_splits` se fait avec la fonction `MiniRocketRegressor` de la librairie `tsai`. La métrique employée dans `MiniRocketRegressor` pour l'optimisation du régresseur est le *RMSE* (3.5) calculé avec les fonctions `make_scoring` et `mean_squared_error` de la librairie `scikit-learn`. Les prédictions et *RMSE* sont calculés sur le jeu de test.

Enfin, comme nous avons appliqué des fenêtres glissantes avec $stride = 1$, les valeurs prédites sont décalées dans le temps de l_{window} . Nous redéfinissons les indices pour avoir les bonnes valeurs. Les résultats sont présentés dans la section 3.5.2.

3.4.4.2 Approche univariée

Nous testons à présent l'approche univariée et testons MiniRocket dans le cas des données regroupées au jour en utilisant le vecteur `TS_day` et dans le cas des données regroupées à la semaine avec le vecteur `TS_week`. Avec l'approche univariée, nous n'avons pas à faire l'étape de One-hot encoding puisque seule la variable réponse est utilisée dans l'algorithme.

Application de fenêtres glissantes :

De la même façon que pour le cas multivarié, nous appliquons des fenêtres glissantes sur les séries `TS_day` et `TS_week` pour qu'elles aient le bon format. Avec la fonction `SlidingWindow`, nous appliquons $n_{windows}$ fenêtres glissantes de taille l_{window} chacune se déplaçant d'une distance $stride$ à chaque pas.

Dans le cas univarié, les fenêtres glissantes permettent de passer d'un format $X = y = (n_{timesteps})$ (voir figure 3.15) à un format $X_{slide} = (n_{samples}, 1, n_{steps})$ et $y_{slide} = (n_{samples})$.

Les calculs sont similaires au cas multivarié. Après plusieurs tests, nous conservons une taille de fenêtre $l_{window} = 9$ et un $stride = 1$ pour avoir un résultat tous les jours si nous travaillons avec `TS_day` ou toutes les semaines si nous travaillons avec `TS_week`.

Jeu d'entraînement et jeu de test

De même que précédemment, la fonction `get_splits` de `tsai` effectue automatiquement la séparation du jeu de données (X_{slide}, y_{slide}) en jeu d'entraînement et en jeu de test.

Entraînement du modèle et prédictions

L'entraînement et le calcul des prédictions sont similaires au cas multivarié. Pour tenter d'améliorer les performances du modèle, nous débruitons au préalable le signal comme indiqué section 3.4.2, avant d'appliquer MiniRocket. Par la suite, nous testons également

d'appliquer la transformation Box-Cox (3.1), puis effectuons les prédictions sur les données transformées avant d'appliquer la transformation inverse (3.2) pour retrouver les vraies valeurs. Les résultats obtenus sont présentés section 3.5.2.

3.5 Résultats

3.5.1 SARIMA

Après entraînement du modèle SARIMA(7, 0, 2)(0, 0, 2, 16) sur `TS_train`, nous traçons figure 3.24 le graphe des résidus ε_t afin de vérifier qu'ils soient bien un bruit blanc gaussien. Sur le graphe 3.24, on observe que les résidus sont bien stationnaires et s'apparentent à une gaussienne, ce qui montre que les hypothèses du modèle SARIMA sont bien vérifiées.

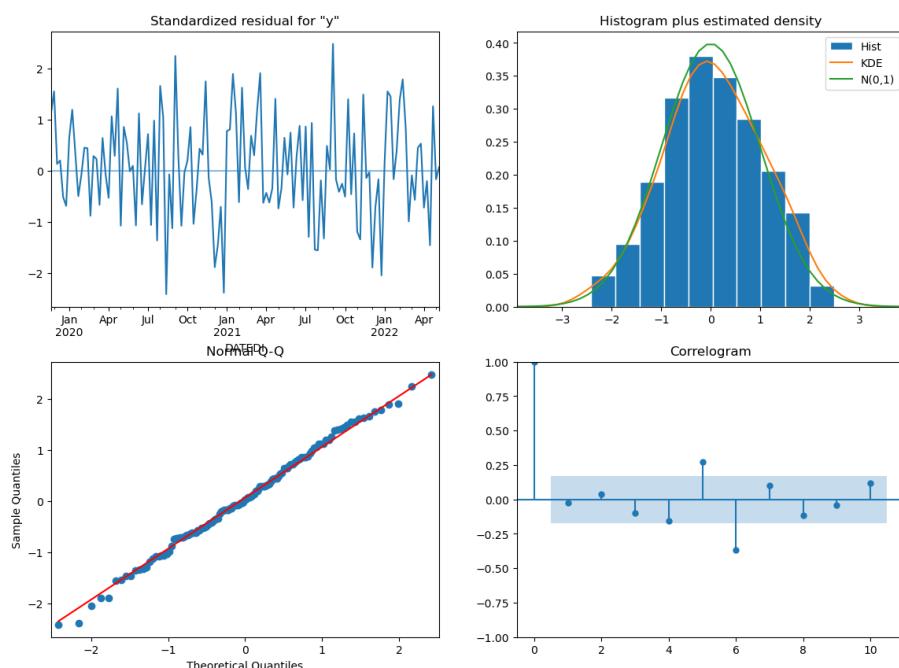


FIGURE 3.24 – Graphes des résidus du modèle SARIMA.

Le graphique figure 3.25 présente les prédictions obtenues avec le modèle SARIMA(7, 0, 2)(0, 0, 2, 16). Pour un nombre moyen de 220 dossiers par semaine sur la période sélectionnée (en ayant retiré l'année 2018), le MAE s'élève à 56.25 soit environ 56 dossiers d'erreur ce qui est trop important pour le besoin métier. La RMSE à une valeur de 69.10.

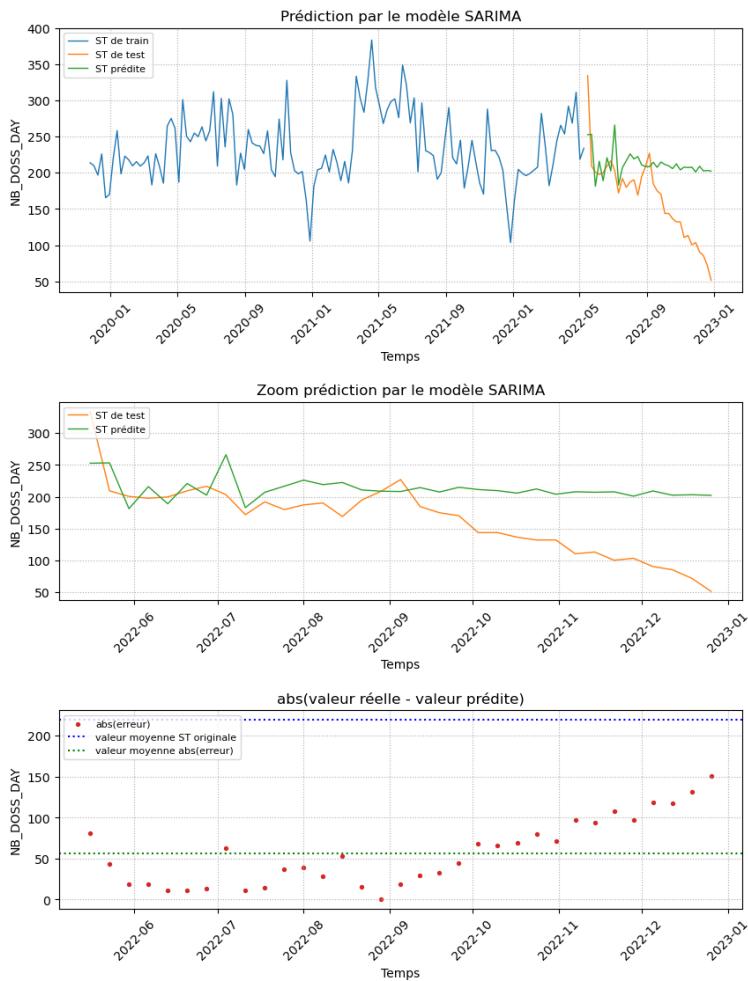


FIGURE 3.25 – Prédiction avec le modèle SARIMA(7, 0, 2)(0, 0, 2, 16) sur TS_week.

3.5.2 MiniRocket

Nous entraînons le modèle MiniRocket sur les données d'apprentissage pour différentes configurations :

- (a) sur les données multivariées à la journée credits_MTS,
- (b) sur les données univariées à la semaine TS_week,
- (c) sur les données univariées à la journée TS_day,
- (d) sur les données univariées à la journée TS_day au préalable débruitées,
- (e) sur les données univariées à la journée TS_day transformées par Box-Cox,
- (f) sur les données univariées à la semaine TS_week au préalable débruitées,
- (g) sur les données univariées à la semaine TS_week transformées par Box-Cox.

Pour chaque configuration, nous calculons le RMSE (3.5) ainsi que le MAE (3.6) sur les données de test.

Dans le cas univarié, nous observons que le débruitage et la transformation Box-Cox empêrent les prédictions par rapport aux données non transformées. De fait, nous présenterons figures 3.26 , 3.27 et 3.28 uniquement les graphes des configurations (a), (b) et (c).

Pour les données regroupées par semaine, le nombre moyen de dossiers est de 156 dossiers par semaine sur toute la période (début 2018 à fin 2022). Avec la configuration (b), nous obtenons un RMSE de 40.43 et un MAE de 30.37 soit une erreur moyenne de 30 dossiers par semaine. Pour les données regroupées par journées, le nombre moyen de dossiers est de 22 dossiers par jour sur toute la période (début 2018 à fin 2022). Avec la configuration (a), nous obtenons un RMSE de 13.91 et un MAE de 12.06, soit une moyenne de 12 dossiers d'erreur par jour donc environ $12 \times 7 = 84$ dossiers d'erreur par semaine. Avec la configuration (b), nous avons un *RMSE* de 11.78 et un MAE de 7.90, soit une moyenne de 8 dossiers d'erreur par jour donc environ $8 \times 7 = 56$ dossiers d'erreur par semaine.

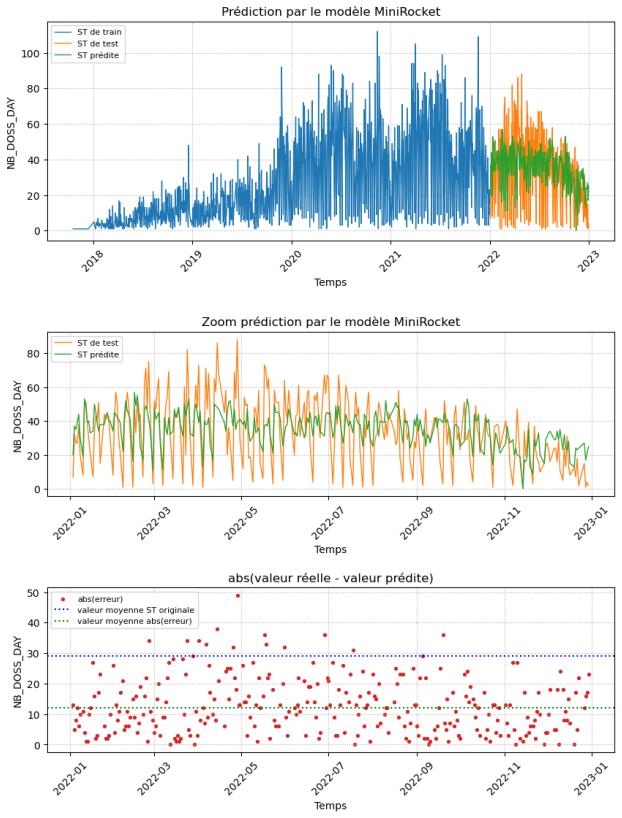


FIGURE 3.26 – Prédictions avec MiniRocket sur `credits_MTS`, configuration (a).

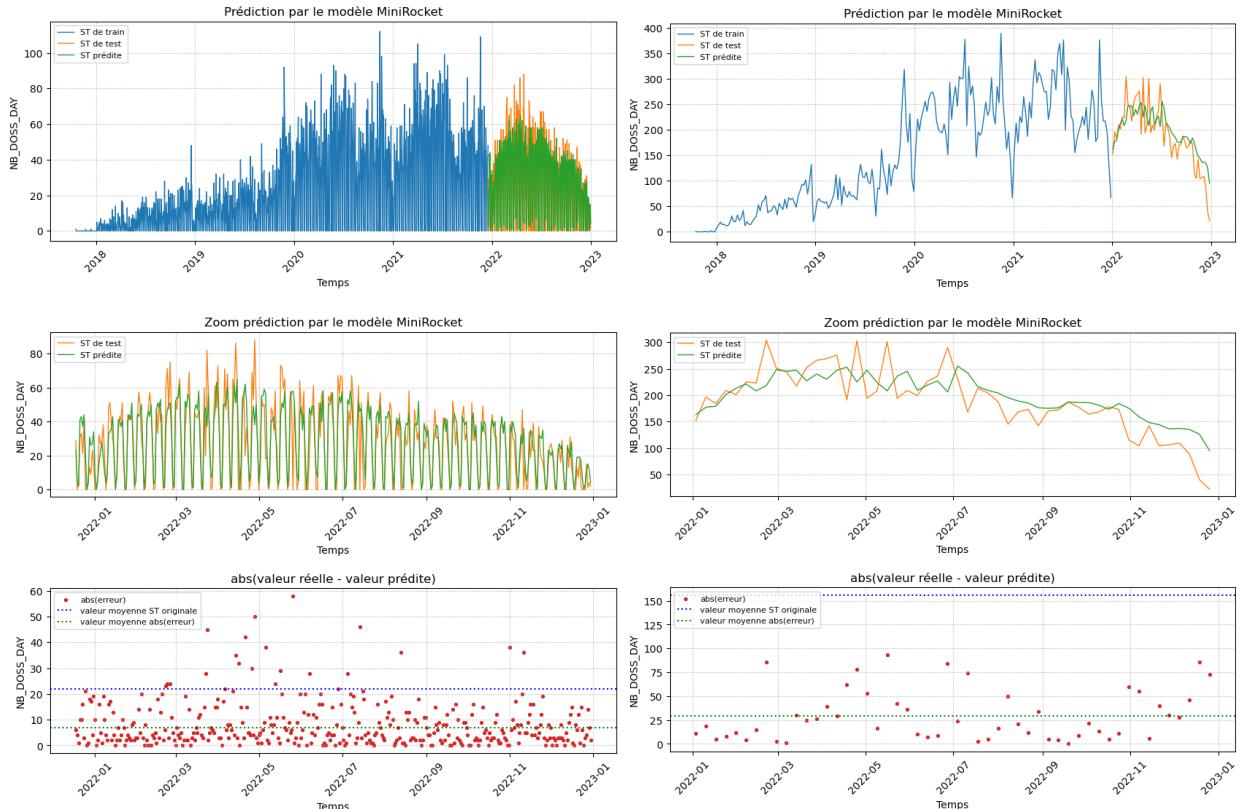


FIGURE 3.27 – Prédictions avec MiniRocket sur `TS_day`, configuration (b).

FIGURE 3.28 – Prédictions avec MiniRocket sur `TS_week`, configuration (c).

On constate donc que les meilleurs résultats sont obtenus dans le cas des données univariées regroupées à la semaine avec la configuration (c).

3.5.3 Tableau comparatif

Le tableau 3.2 présente un résumé de tous les résultats obtenus.

Méthode	Données	Valeur moyenne	Débruitage	Transformation Box-Cox	RMSE	MAE
MiniRocket	credits_MTS	22	✗	✗	14.91	12.06
	TS_day	22	✗	✗	11.78	7.90
			✓	✗	11.77	7.97
			✗	✓	12.10	8.05
	TS_week	156	✗	✗	40.43	30.37
			✓	✗	40.89	30.67
			✗	✓	44.66	32.90
SARIMA(7, 0, 2)(0, 0, 2, 16)	TS_week	156	✓	✓	69.10	56.25

TABLE 3.2 – Tableau récapitulatif des résultats obtenus pour l'ensemble des méthodes testées.

Comme indiqué précédemment, le RMSE (3.5) et le MAE (3.6) sont calculés sur les données de test. Nous pouvons donc conclure que l'algorithme MiniRocket sur les données `TS_week` non transformées donne les meilleurs résultats avec un RMSE de 40.43 et un MAE de 30.37 soit environ une erreur moyenne de 30 dossiers d'erreur par semaine.

3.6 Conclusion

Le but de ce projet était de prédire le nombre de crédits immobiliers qui arrivent au back-office par semaine. Nous avons décidé de conserver la structure temporelle des données en considérant des méthodes spécifiques aux séries temporelles.

Nous avons commencé par réaliser un premier pré-traitement des données crédit ainsi qu'une analyse exploratoire en section 3.3. Nous avons observé que les variables explicatives étaient peu corrélées avec la variable réponse. A l'issue de cette première partie, nous avons pu sélectionner les variables pertinentes en 3.3.6, imputer les données manquantes en 3.3.7 et mettre en forme les séries temporelles en 3.4.1. Nous avons obtenu deux séries temporelles univariées `TS_week` et `TS_day` permettant de compter le nombre de crédits par semaine et par jour respectivement, et une série temporelle multivariée `credits_MTS` permettant de compter le nombre de crédits par jour uniquement.

Nous avons ensuite implémenté le modèle SARIMA en 3.4.3 qui est une méthode statistique simple pour la prédiction de séries temporelles univariées (`TS_week` et `TS_day`). Nous avons commencé par débruiter le signal à l'aide de la décomposition en base d'Ondelettes en section 3.4.2. Puis, nous avons transformé le signal avec la transformation Box-Cox avant de réaliser une décomposition STL en section 3.4.3.1. La décomposition STL nous a permis de retirer la saisonnalité du signal. Nous avons tracé les graphes ACF et PACF nous permettant de déterminer les paramètres du modèle SARIMA en 3.4.3.2 et avons entraîné le modèle.

Par la suite, nous avons testé l'algorithme de deep-learning MiniRocket en [3.4.4](#). Après une brève présentation de son fonctionnement, nous nous sommes d'abord penchés sur la prédiction de la série temporelle multivariée `credits_MTS` en section [3.4.4.1](#). Pour ce faire, nous avons one-hot encodé les variables et appliqué des fenêtres glissantes sur les données avant d'entraîner le modèle. Par la suite, nous avons appliqué MiniRocket aux deux séries temporelles univariées `TS_week` et `TS_day` en testant différents scénarios (avec ou sans débruitage et transformation Box-Cox) en section [3.4.4.2](#).

Enfin, nous avons présenté les résultats de SARIMA et MiniRocket en [3.5.1](#) et [3.4.4](#) avant de conclure en [3.5.3](#) sur le meilleur modèle : MiniRocket sur `TS_week` sans débruitage ni transformation Box-Cox produisant un RMSE de 40.43 et un MAE de 30.37, soit une erreur moyenne de 30 dossiers par semaine.

4 Projet détection de signature

- De Mars à Juin 2023 -

4.1 Introduction

Chaque mois, la CEMP récupère des milliers de documents PDF fournis par ses clients. Un organisme de contrôle est chargé de vérifier si les documents fournis sont conformes et signés. Cependant, le nombre de documents est trop important pour effectuer un contrôle exhaustif. L'organisme de contrôle sélectionne alors chaque mois une poignée de documents au hasard pour vérification et ne vérifie pas l'ensemble des documents fournis. Bien qu'une partie seulement des documents soit contrôlée, cette procédure de vérification est très chronophage.

L'objectif de ce projet est d'implémenter une méthode permettant de vérifier de façon automatique la présence d'une signature sur plusieurs types de documents PDF.

Pour réaliser ce projet, nous commençons par implémenter un code en Python ce qui nous laisse la liberté d'explorer différentes solutions. Nous adaptons ensuite en Alteryx la meilleure solution trouvée en Python. Avec les pare-feux de l'entreprise, l'installation de packages Python sur l'ordinateur et sur le serveur sont compliqués. Nous développons alors une méthode Python utilisant autant que possible un minimum de librairies différentes. De plus, l'entreprise ne dispose pas de GPUs. La méthode implantée ne doit donc pas nécessiter de grosses capacités de calcul. Sans GPU, il n'est pas possible d'entraîner des réseaux de neurones, qui sont généralement les méthodes donnant les meilleurs résultats pour le traitement d'image.

Dans un premier temps, nous présenterons les données en section 4.2 puis détaillerons les méthodes utilisées en section 4.3. Pour ce faire, nous commencerons par expliquer brièvement en section 4.3.1 les premières approches testées. Puis nous détaillerons section 4.3.2 une première méthode implantée en Python pour la détection de signature. Une deuxième méthode implantée en Alteryx pour la détection de signature sera brièvement présentée en section 4.3.3. Enfin, les résultats des deux méthodes seront annoncés en section 4.4.

4.2 Présentation des données

Dans un premier temps, nous souhaitons vérifier la signature sur quatre types de documents différents : document LEA (synthèse du parcours épargne personnalisé), EAI (auto-certification destinée aux personnes physiques), BS (bulletin de souscription assurance-vie) et QCF-QR (questionnaire de profil investisseur). La méthode doit cependant pouvoir se généraliser à d'autres types de documents. La figure 4.1 illustre la page contenant la signature sur les quatre types de documents à analyser.

Un jeu de donnée comportant l'ensemble des quatre documents PDF depuis 2022 est fourni. Certains documents comme le LEA ou le QCF-QR, sont tous identiques. Cependant, il existe des documents présentant différentes versions comme le EAI (possédant deux versions différentes) ou le BS (possédant plus de quatre versions différentes). Par ailleurs, le jeu de donnée peut comporter des erreurs. Par exemple, un document indiqué comme étant un LEA peut être en réalité un autre document. La méthode implémentée doit être capable de traiter les différentes versions d'un même document et d'identifier le type de document que l'on analyse pour savoir si on est en présence du bon document. Par la suite, un document PDF peut contenir plusieurs pages, les documents n'ont pas tous le même nombre de pages et la signature ne se trouve pas toujours sur la même page du document. La solution imaginée doit être capable d'identifier la page supposée contenir la signature. Enfin, nous devons traiter les cas des documents mal scannés, rotationnés, contenant des pages blanches ou autres artefacts.

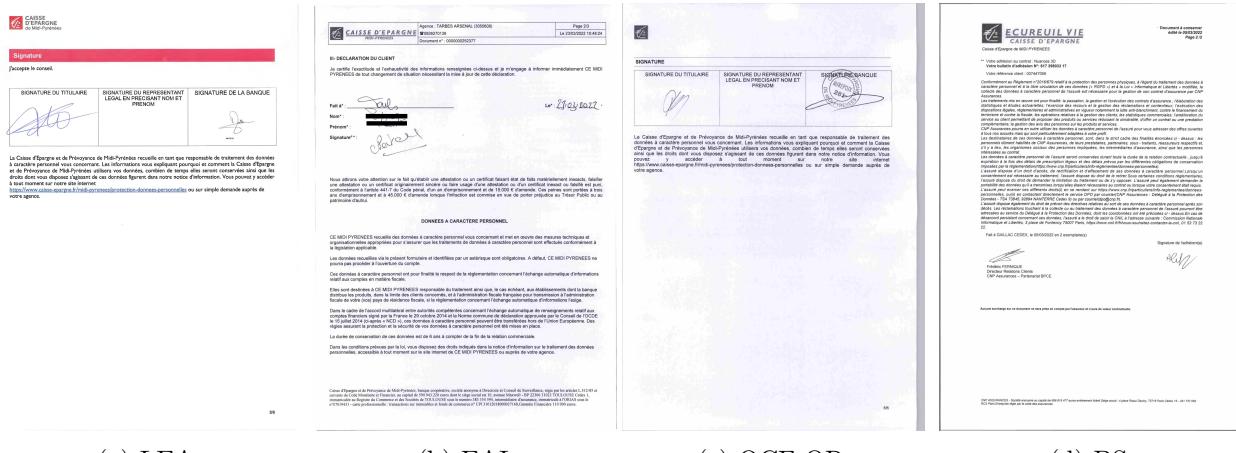


FIGURE 4.1 – Page contenant la signature pour les quatre documents à analyser.

4.3 Méthodes utilisées

4.3.1 Premières approches

Nous avons commencé par tester des méthodes déjà implémentées en commençant par le package `signature-detect` [15] qui permet de détecter la présence d'une signature sur un fichier PDF ou sur une image en utilisant la librairie Python OpenCV. Cependant, cette méthode ne fonctionne pas dans notre cas car elle ne permet pas de détecter des signatures trop petites ou de la même taille que le texte. Elle ne fonctionne pas non plus pour les signatures qui ne sont pas des dessins mais des écritures. Néanmoins, nous nous inspirerons de ce package pour implémenter notre solution (voir section 4.3.2.3).

Par la suite, nous avons trouvé les poids d'un modèle YOLO-v5 (réseau de neurones) entraîné à détecter des signatures sur des images [16]. Comme nous n'avons pas de GPUs, nous n'avons pas pu fine-tuner (réentraîner) le modèle sur notre jeu de données et avons dû utiliser le modèle déjà entraîné tel quel. Ceci n'a pas conduit à des résultats satisfaisants sur nos données en plus de nécessiter un temps de calcul important pour chaque document.

Après des recherches, nous n'avons pas trouvé de méthodes fonctionnant pour notre problème de détection de signatures et ne nécessitant pas de puissance de calcul. Nous avons dû créer notre propre solution, que nous détaillerons dans les sections suivantes.

Enfin, notons que nous n'avons pas disposé de la librairie Python **Tesseract** sur laquelle repose notre solution dès le début du projet. Une première méthode de détection de signature basée sur la détection du cadre puis le calcul du pourcentage de remplissage à l'intérieur du cadre a été implémentée. Cette méthode, que nous ne détaillerons pas dans ce rapport, ne fonctionne pas pour les documents ne possédant pas de cadre autour de la signature et a été abandonnée au profit de la méthode présentée ci-dessous.

4.3.2 Programme Python

Pour détecter la présence de signature, une première méthode a été implémentée en Python en utilisant essentiellement la librairie **OpenCV** pour le traitement d'image et l'OCR (Reconnaissance Optique de Caractères) **Tesseract** [17] avec l'extension Python **pytesseract** pour la reconnaissance du texte dans les images.

Tesseract est un logiciel open source développé par Google permettant de convertir une image en texte [18]. Cet algorithme est capable de reconnaître plus de 100 langues différentes ainsi que des polices de caractères variées, y compris les polices de caractères cursives. Pour fonctionner, **Tesseract** utilise tout d'abord des techniques de pré-traitement d'image pour nettoyer l'image et améliorer la reconnaissance des caractères . Ensuite, il segmente l'image en caractères individuels et utilise des modèles pour reconnaître la police de caractère. Une fois la police identifiée, l'algorithme identifie chaque caractère. Enfin, **Tesseract** réassemble les caractères individuels en mots et en phrases [19].

Pour un type de PDF donné (LEA, EAI...), la méthode Python implémentée demande à l'utilisateur (personne chargée de contrôler la signature des documents) de sélectionner un PDF servant de modèle et de tracer sur ce modèle le ou les rectangles supposés contenir la signature. L'utilisateur saisit également une liste de mots clés présents sur la page du PDF modèle contenant la signature. A l'aide de **Tesseract** et **OpenCV**, ces informations sont utilisées pour calculer la position des rectangles supposés contenir la signature pour tous les autres PDF à analyser du même type que le PDF modèle. Enfin, l'algorithme détermine si les rectangles extraits contiennent ou non une signature. Le fonctionnement de cette méthode est détaillé dans les sections ci-dessous.

4.3.2.1 Pré-traitement de l'image

Les librairies **pdf2image** et **Pillow** sont utilisées pour charger les PDF et convertir chaque page en une image qui est un tableau **numpy** lisible par **OpenCV** et **Tesseract**.

Avant de lire le texte d'une image avec **Tesseract**, il est recommandé d'appliquer un pré-traitement de l'image afin d'optimiser les performances de lecture [20].

Un module python **preprocessing.py** est implémenté afin de regrouper toutes les opérations de pré-traitement.

Rotation de l'image

Certains PDF mal scannés peuvent être rotationnés ce qui empêche l'OCR de détecter le texte et causera des imprécisions par la suite lors du calcul de la position des rectangles supposés contenir la signature. Afin de remettre l'image dans le bon sens, on calcule tout d'abord l'angle de rotation de l'image. Une image correctement scannée doit présenter un angle de 0° . Pour cela, la librairie OpenCV est utilisée avec la fonction `Canny` pour extraire les contours de l'image et la fonction `HoughLinesP` pour extraire les lignes principales de l'image à partir des contours. On itère ensuite sur chaque ligne et on calcule l'angle entre la ligne et l'horizontale. On obtient une liste d'angles et l'angle médian est choisi comme angle de l'image. Par la suite, une fonction permettant de rotationner l'image de l'angle calculé précédemment est implémentée. La rotation se fait avec OpenCV et ne doit pas ronger les contours de l'image.

Cependant, cette méthode de calcul d'angle ne fait pas la différence entre une image rotationnée à 0° et une image à 180° . Ainsi, une deuxième méthode de calcul d'angle, utilisée après la première méthode permet de résoudre ce problème. La deuxième méthode utilise la fonction `image_to_osd` de `pytesseract` pour calculer à nouveau l'angle de l'image et déterminer si cet angle est de 0° ou de 180° . Dans le cas d'un angle à 180° , l'image est rotationnée pour atteindre le bon angle. Cette deuxième méthode ne peut pas de substituer à la première car la fonction `image_to_osd` identifie uniquement des angles de rotation de 0° , 90° , 180° ou 270° tandis que les images possèdent souvent des angles de rotation quelconques. La figure 4.2 montre la rotation de l'image.

Seuillage et débruitage

Pour améliorer la lecture des images, on utilise OpenCV pour convertir l'image en niveaux de gris puis la débruiter. Le débruitage a pour conséquence de flouter l'image. Afin de conserver les contours de l'image et donc la lisibilité du texte, on utilise un filtre bilatéral avec la fonction `bilateralFilter`. Contrairement aux autres filtres populaires de débruitage basés sur la position des pixels sur l'image (filtre médian ou filtre gaussien), le filtre bilatéral prend également en compte la plage de valeurs des pixels ce qui permet de mieux conserver les contours du texte [21].

On applique enfin un seuillage adaptatif avec la fonction `adaptiveThreshold` pour améliorer la qualité de l'image. Si la valeur d'un pixel de l'image est supérieure à la valeur du seuil, la valeur du pixel est fixée à la valeur maximale (blanc). Le seuillage adaptatif est plus précis qu'un seuillage binaire et doit être utilisé si les conditions d'éclairage d'une image varient d'une zone à l'autre. En effet, il détermine le seuil de chaque pixel à partir d'une petite région autour de celui-ci et permet d'obtenir différents seuils pour différentes régions d'une même image. A l'issue du seuillage, nous obtenons une image binaire (en noir et blanc), illustrée figure 4.2.

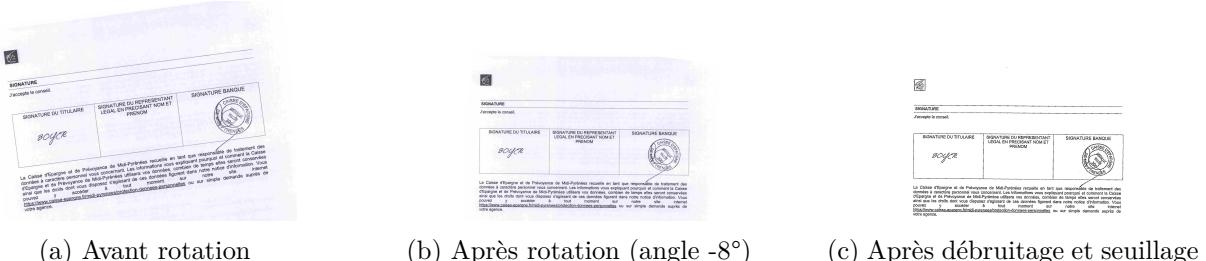


FIGURE 4.2 – Pré-traitements sur la page du document LEA contenant la signature.

4.3.2.2 Cration du modle

Nous implemons  present le module `template.py` permettant de definir le PDF mode pour la detection de signature. Le mode n'est  definir qu'une seule fois pour chaque type de PDF (LEA, EAI...).

tape 1 - Selection de la page du PDF mode, extraction de la taille et du texte

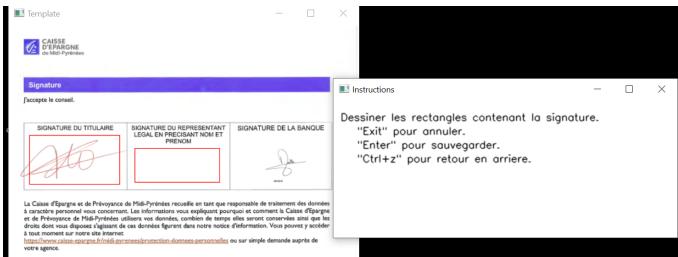
Tout d'abord, ce module demande  l'utilisateur de choisir un PDF servant de mode et de definir le numero de page du PDF contenant la signature. On deruite ensuite la page du PDF mode avec le module `preprocessing.py` et on utilise la fonction `image_to_data` de `pytesseract` pour lire le texte de la page. La taille `taille_mode` et le texte `texte_mode` de la page du PDF mode sont sauvegardes pour une utilisation ulterieure.

tape 2 - Dessin des rectangles supposes contenir la signature et calcul de leur position

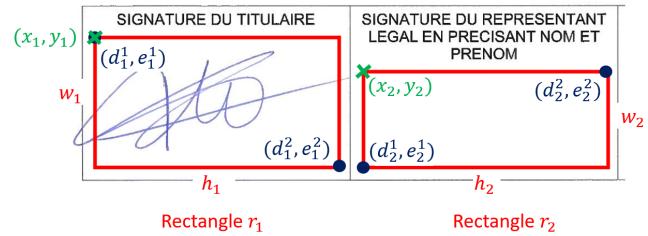
Dans un deuxime temps, l'utilisateur doit dessiner sur la page du PDF mode, les rectangles supposes contenir la signature. Pour ce faire, on utilise la librairie `OpenCV` pour faire apparatre  l'cran une fentre sur laquelle l'utilisateur peut dessiner les rectangles, comme illustre figure 4.3a. `OpenCV` permet de calculer la taille de l'cran d'ordinateur de l'utilisateur afin que l'image affiche s'adapte  son cran. L'utilisateur peut choisir d'annuler la selection avec la touche `echap` ou de la valider avec la touche `enter`. La touche `ctrl+z` permet un retour en arrire. Par exemple, sur le PDF mode du document LEA presente figure 4.3b, l'utilisateur a dessin deux rectangles r_1 et r_2 car la signature peut se trouver soit sous l'intitul "signature du titulaire", soit sous l'intitul "signature du representant lgal". Les coordonnees $[(d_i^1, e_i^1), (d_i^2, e_i^2)]$ definissant chaque rectangle dessin r_i sont utilises pour determiner le point (x_i, y_i) en haut  gauche du rectangle ainsi que la hauteur h_i et la largeur w_i du rectangle r_i . La liste (x_i, y_i, h_i, w_i) est sauvegarde pour chaque rectangle r_i dessin.

tape 3 - Definition d'une liste de mots ou expressions cles

L'utilisateur doit par la suite saisir une liste de mots ou expressions cles `keyWords_template` = $[w_1, w_2, \dots, w_j, \dots]$ presents sur la page du PDF mode contenant la signature, par exemple `keyWords_template` = ['prénom', 'signature', 'declaration du client'] = $[w_1, w_2, w_3]$ comme illustre figure 4.4a. Ces expressions cles serviront par la suite  calculer la position des rectangles supposes contenir la signature sur l'ensemble des PDF  analyser. L'utilisateur doit suivre plusieurs regles lors de la selection des expressions cles de `keyWords_template` :



(a) Fenêtre apparaissant à l'écran de l'utilisateur pour choisir le modèle.



(b) Dessin des rectangles supposés contenir la signature.

FIGURE 4.3 – Choix du modèle et dessin des rectangles contenant la signature pour le document LEA.

- Tous les éléments de la liste `keyWords_template` doivent être présents et lisibles par l'OCR sur la page du PDF modèle. La page du PDF modèle est la seule page qui requiert d'être "parfaite" c'est-à-dire correctement scannée, lisible et centrée, ne pas présenter d'angle de rotation ou autre artefact.
- Chaque élément de `keyWords_template` peut être un seul mot ('signature') ou une expression ('déclaration du client') pour davantage de robustesse dans la recherche des mots clés.
- Chaque élément de `keyWords_template` ne doit être présent qu'une seule fois dans la page.
- Il est recommandé de choisir des mots ou expressions facilement lisibles par l'OCR, proches du centre de l'image (au cas où l'image serait rognée) et proches de la zone supposée contenir la signature (au cas où les PDF différeraient légèrement entre eux, par exemple s'il existe différentes versions d'un même document, et que les rectangles supposés contenir la signature ne seraient pas toujours situés à la même distance de tous les mots clés). Il est recommandé d'éviter les éléments avec des apostrophes qui peuvent être mal interprétés par l'OCR. Cependant, la recherche des mots est insensible aux apostrophes et aux accents donc leur présence ne causera pas d'erreur. Enfin, la recherche est sensible à la casse.



(a) Recherche des mots clés `keyWords_template` sur la page signée du EAI.

(b) Recherche des mots clés `keyWords_template` sur la page signée du LEA.

FIGURE 4.4 – Recherche des mots clés `keyWords_template` sur la page signée des documents EAI et LEA. La liste `keyWords_template` est différente selon le document que l'on souhaite analyser.

Étape 4 - Calcul de la position des mots ou expressions clés

Une fois la liste des expressions clés $\text{keyWords_template} = [w_1, w_2, \dots, w_j, \dots]$ correctement définie, on calcule la position $p_{\text{modèle}} = [(a_1, b_1), (a_2, b_2), \dots, (a_j, b_j), \dots]$ de chaque élément

w_j sur la page du PDF modèle. Pour ce faire, on utilise la fonction `image_to_data` de `pytesseract` pour extraire les informations de la page du PDF modèle. Cette fonction renvoie notamment le texte contenu sur la page sous forme d'une liste de mots $\mathbf{T} = [m_1, m_2, \dots, m_k, \dots]$ ainsi que les coordonnées $[(\tilde{a}_1, \tilde{b}_1), (\tilde{a}_2, \tilde{b}_2), \dots, (\tilde{a}_k, \tilde{b}_k), \dots]$ définissant la position de chaque mot m_k sur la page. Par exemple, $\mathbf{T} = ['Voici', 'la', 'déclaration', 'du', 'client', 'son', 'prénom', 'et', 'sa', 'signature :']$

Ainsi, pour chaque élément w_j de `keyWords_template`, on recherche le mot m_k correspondant dans la liste de texte \mathbf{T} . Cette recherche permet de trouver l'indice de w_j dans \mathbf{T} et par conséquent la position $(a_j, b_j) = (\tilde{a}_k, \tilde{b}_k)$ du mot w_j . La recherche est insensible aux accents et aux apostrophes et se fait en utilisant la librairie Python `RegEx` afin que les mots w_j comprenant des espaces ou des caractères au début et à la fin soient également trouvés. Par exemple, pour w_j valant "déclaration" les mots m_k "Déclaration", "déclaration", " _déclaration :4" ou "declaration" peuvent être identifiés. Notons cependant que l'utilisateur peut choisir pour w_j des expressions composées de plusieurs mots. Par conséquent, un élément $w_j = 'déclaration du client'$ correspond à plusieurs mots $m_k = ['déclaration', 'du', 'client']$ dans la liste \mathbf{T} et la recherche de l'indice de w_j n'est plus immédiate. Pour résoudre ce problème, l'ensemble des mots m_k de \mathbf{T} sont concaténés et séparés par un caractère spécial, par exemple : 'Voici#la#déclaration#du#client,#son#prénom#et#sa#signature :'. On compte alors le nombre de caractères spéciaux '#' avant le premier mot de w_j (ici 'déclaration'). La position (a_j, b_j) du premier mot de w_j est ainsi retournée.

Étape 5 - Calcul de la position relative des rectangles par rapport aux mots clés

Une fois la position (a_j, b_j) de chaque élément w_j trouvée, on calcule la position (x_i, y_i) de chaque rectangle r_i par rapport à chaque coordonnée (a_j, b_j) . On obtient donc une liste `list_relative_point` contenant les coordonnées relatives de chaque point (x_i, y_i) par rapport à la nouvelle origine (a_j, b_j) . Ainsi, $\text{list_relative_point}_{i,j} = (x_i - a_j, y_i - b_j, h_i, w_i)$ pour chaque élément w_j et chaque rectangle r_i .

Résumé des étapes précédentes

Pour résumer les étapes précédentes, nous obtenons `list_relative_point` contenant la position, sur la page du PDF modèle, des rectangles r_i supposés contenir la signature par rapport à la position $p_{\text{modèle}}$ des différents éléments clés w_j de `keyWords_template`.

De fait, pour tout autre PDF à analyser du même type que le PDF modèle, nous calculerons la position p_{analyser} des éléments w_j de `keyWords_template`, sur la page du PDF à analyser. Puis nous utiliserons `list_relative_point` pour placer les rectangles r_i en fonction de la position p_{analyser} des éléments w_j .

La position des rectangles s'adaptera donc à chaque PDF à analyser, notamment si celui-ci a été scanné de manière décalée. Nous sommes ainsi certains que les rectangles supposés contenir la signature sont bien placés et ne risquent pas de contenir autre chose qu'une signature. Nous détaillons cette méthode dans la section suivante.

4.3.2.3 Détection de la signature

Nous implémentons le module `detectSignature.py` permettant de vérifier si un PDF est signé. Au préalable, l'utilisateur a dû définir un modèle avec le module `template.py` pré-

senté dans la section [4.3.2.2](#) précédente et nous avons calculé les variables `taille_modèle`, `texte_modèle` ainsi que `list_relative_point` qui seront utilisées dans cette partie.

Étape 1 - Détection d'une signature numérique

Les documents à analyser peuvent être papier ou numériques. Dans le cas d'un document signé numériquement, l'information de la signature est présente dans les métadonnées du PDF et nous n'avons pas besoin d'analyser en détail le document. Nous utilisons la fonction `get_sigflags` [22] du module `fitz` implémenté dans la bibliothèque PyMuPDF pour détecter la présence d'une signature numérique sur un document PDF. Si aucune signature numérique n'est détectée, le PDF est probablement un document papier scanné et nous passons à l'étape suivante ci-dessous.

Étape 2 - Extraction de la page contenant la signature

La signature se trouve souvent sur la dernière page d'un document PDF mais il arrive régulièrement que certains PDF soient scannés dans le désordre, que d'autres documents soient accolés à la fin, que le document soit différent du document type pour lequel le modèle a été calculé ou bien que le document soit uniquement composé de pages blanches. Avant de déterminer si le PDF est signé, il est donc nécessaire d'identifier si le document à analyser est bien un document similaire au PDF modèle, si la page contenant la signature existe et dans ce cas, identifier quelle page du document contient la signature.

Pour ce faire, nous commençons par itérer sur chaque page du document PDF à analyser en débutant par la dernière page du document car c'est là où se situe généralement la signature. Si la page est identifiée comme contenant la signature, la recherche s'arrête et la page est renvoyée, sinon, on passe à la page suivante. Pour chaque page sur laquelle nous itérons, nous redimensionnons tout d'abord la page à la même taille que celle du PDF modèle en utilisant les dimensions données par `taille_modèle`. Puis, nous utilisons le module `preprocessing.py` pour rotationner correctement la page et la débruiter afin d'optimiser la lecture du texte par l'OCR. La fonction `image_to_data` de `pytesseract` est ensuite utilisée pour extraire le texte de la page à analyser. Nous calculons alors la similarité entre le texte de la page à analyser et le texte du PDF modèle `texte_modèle` avec la distance de Jaccard qui se définit comme suit :

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \in [0, 1] \quad (4.1)$$

où A et B sont deux ensembles de mots et $|A|$ est la cardinalité de l'ensemble A . Une valeur de 0 indique qu'il n'y a pas d'éléments communs entre les deux ensembles, et 1 indique que les ensembles sont identiques. Il existe d'autres méthodes populaires pour le calcul de distance entre deux textes comme la distance de Levenshtein ou la méthode LCS (longest common subsequence) tenant compte de la position des mots dans le texte. Cependant dans notre cas, l'indice de Jaccard est pertinent car l'OCR ne lit pas toujours les mots dans le même ordre selon les documents et nous souhaitons seulement calculer le taux de mots communs entre les deux textes.

Ainsi, si le texte de la page sur laquelle nous itérons présente une similarité supérieure à un certain seuil avec le texte de la page du PDF modèle au sens de la distance de Jaccard,

nous considérons que la page en question est la page contenant la signature. Si à la fin de toutes les pages, nous n'avons pas trouvé de page suffisamment similaire à celle du PDF modèle, nous levons une exception et ne poursuivons pas l'analyse de la signature : le document analysé ne contient pas la page signée et/ou il n'est pas du même type que le document modèle et nous sommes en présence d'une erreur de document.

Étape 3 - Calcul des coordonnées d'un élément de keyWords_template

Si la page supposée contenir la signature a été identifiée sur le document à analyser, nous calculons la position du premier mot ou expression de `keyWords_template` que nous trouvons sur la page, à l'aide de la fonction `image_to_data` de `pytesseract`. Nous notons ce mot ou expression \hat{w} et (\hat{a}, \hat{b}) ses coordonnées. Si aucun mot de `keyWords_template` n'a pu être trouvé, une exception est levée et nous ne poursuivons pas l'analyse de la signature.

Étape 4 - Extraction des rectangles supposés contenir la signature

Le but est de placer au "bon endroit" sur la page du PDF à analyser, chaque rectangle r_i dessiné sur le PDF modèle. Pour trouver le "bon endroit", nous utilisons les coordonnées relatives `list_relative_points` calculées avec le module `template.py` pour placer, sur la page à analyser, les rectangles r_i par rapport à la position (\hat{a}, \hat{b}) de \hat{w} .

Pour ce faire, nous identifions les éléments de `list_relative_point` correspondant au mot ou expression clé \hat{w} . Par exemple, \hat{w} est le mot d'indice l dans `list_relative_point` : $list_{relative_point}_{i,l} = (x_i - a_l, y_i - b_l, h_i, w_i)$ où x_i, y_i, h_i, w_i sont la position, la hauteur et la largeur de chaque rectangle r_i dessiné sur le modèle. La position du rectangle r_i à placer sur la page à analyser est donc calculée comme suit : $[A_i] : \hat{a} + x_i - a_l$, $[B_i] : \hat{a} + x_i - a_l + w_i$, $[C_i] : \hat{b} + y_i - b_l$ et $[D_i] : \hat{b} + y_i - b_l + h_i$ où $[A_i], [B_i], [C_i]$ et $[D_i]$ sont les 4 sommets du rectangle r_i .

Étape 5 - Vérification de la signature

Une fois les rectangles r_i correctement placés sur la page à analyser, il ne reste plus qu'à vérifier si au moins un des rectangles contient une signature. Pour cela, nous nous inspirons des fonctions `Cropper` et `Judger` du package `signature-detect` [15]. En utilisant `OpenCV`, le `Cropper` permet de rogner légèrement la bordure d'une image binaire puis de la découper en régions de pixels connectés. En conservant uniquement la région de pixels connectés la plus grande dans chaque rectangle, on s'assure que l'on retire des rectangles les éventuels artefacts, traits noirs et bordures qu'ils pourraient contenir. Le `Judger` lit la région de pixels connectés la plus grande de chaque rectangle et calcule le taux de remplissage du rectangle en faisant le ratio entre le nombre de pixels noirs et le nombre de pixels blancs. Ainsi, si le ratio d'un rectangle est supérieur à un certain seuil, on considère que ce rectangle est signé. Les figures 4.5 et 4.6 illustrent les rectangles r_i supposés contenir la signature avant et après le `Cropper`. Certaines modifications ont été apportées aux fonctions `Cropper` et `Judger` de `signature-detect` pour qu'elles s'adaptent à notre problème. Par exemple, le calcul de la bordure dans le `Cropper` a été modifié et nous avons supprimé dans le `Judger` le calcul du rapport de taille qui était un test supplémentaire pour déterminer si l'image est signée.

Nous avons donc terminé la détection de signature avec Python. Nous utilisons la librairie

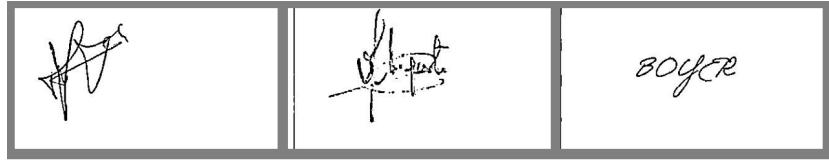


FIGURE 4.5 – Rectangles r_i contenant les signatures extraites, avant le Cropper.

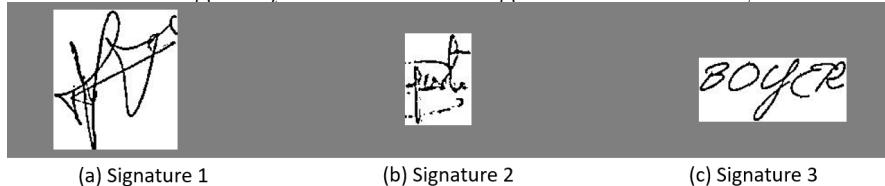


FIGURE 4.6 – Rectangles r_i contenant les signatures extraites, après le Cropper.

`argparse` pour pouvoir lancer notre programme depuis le terminal. Étant donné le volume important de documents que nous devons analyser, nous implémentons notre méthode de sorte que l'on puisse arrêter le code en cours de route et reprendre l'analyse à l'endroit où elle s'était interrompue. Les résultats seront présentés dans la section 4.4.

4.3.2.4 Limites de la méthode

Si le texte du PDF à analyser est différent de celui du modèle, une exception est généralement levée lors du calcul de la page supposée contenir la signature. Le résultat de cette exception est "page contenant la signature non détectée". Cependant, il pourrait exister des cas où le document à analyser contient le même texte que le modèle, mais une structure différente. Dans ce cas, aucune exception n'est levée mais le placement des rectangles n'est pas correct. Nous n'avons pas réussi à adresser ce problème, mais nous n'avons pas ce genre de situation dans notre jeu de données.

4.3.3 Workflow Alteryx

Après avoir implémenté une première solution en Python, nous cherchons maintenant à reproduire autant que possible cette solution sur le logiciel Alteryx. En effet, bien que les possibilités avec Alteryx soient beaucoup plus restreintes qu'en Python, ce logiciel a pour avantage d'être facile à maintenir et à mettre en production pour l'équipe Data de la CEMP. Pour pouvoir traiter les documents PDF, nous utilisons les outils de la palette Computer Vision du module Alteryx Intelligence Suite. Dans ce rapport, par manque de place, nous n'expliquerons pas en détail la solution Alteryx mais nous en présenterons les grandes lignes en Annexe A4. La principale différence entre le code Python précédemment implémenté et le Workflow Alteryx est qu'avec Alteryx, nous n'avons pas accès à la position des rectangles dessinés ni à la position du texte sur la page. Par conséquent, les rectangles contenant la signature sont fixes et ne s'adaptent pas à chaque PDF, à la différence de la méthode Python. Les résultats sont présentés à la section 4.4.

4.4 Résultats

La méthode Python et la méthode Alteryx ont été testées sur les quatre types de documents (LEA, EAI, BS, QCF-QR). Pour constituer les quatre jeux de données de test, nous avons sélectionné au hasard 100 PDF pour chaque type de document et les avons labellisés à la main. Nous avons ensuite exécuté le code Python et le code Alteryx sur ces jeux de test.

Les quatre jeux de test présentent en proportions différentes des documents signés papier, des documents signés numériquement, des documents non signés et des documents présentant une anomalie (illisibles, mal scannés ou des documents différents du PDF modèle). Idéalement, pour pouvoir correctement analyser les performances, il aurait fallu que les proportions entre les documents signés papier, numériquement ou présentant une anomalie soient identiques entre les quatre jeux de test. Cependant, ceci aurait impliqué de vérifier énormément de documents à la main pour constituer les jeux de test et pour des raisons de temps, une sélection au hasard a été préférée.

Tous les documents LEA et QCF-QR ont une structure identique. Une seule analyse avec un seul modèle est donc nécessaires pour détecter la signature sur ces types de documents. Par la suite, le document EAI possède deux versions différentes. Une approche en deux étapes est alors utilisée : une première analyse de l'ensemble des PDF est réalisée avec un premier PDF modèle, puis les documents pour lesquels la page signée n'a pas pu être identifiée sont récupérés et soumis à une deuxième analyse avec un deuxième modèle. Enfin, le document BS possède une dizaine de versions différentes. Le texte de la page contenant la signature est quasiment identique pour chaque version et seule la position de la signature diffère. De plus, de nombreux documents (un quart du jeu de test) sont notés comme étant des BS alors qu'il s'agit en réalité d'autres types de documents. Avec un seul PDF modèle et en choisissant judicieusement les mots clés `keywords_template` proches de la zone signée, la méthode Python parvient à adapter la position du cadre à chaque version. En revanche, la méthode Alteryx qui utilise un cadre fixe pour détecter la signature ne fonctionne pas pour le document BS.

Selon le type de document, la méthode Python est deux à quatre fois plus rapide que la méthode Alteryx. Lorsqu'elles fonctionnent, les deux méthodes donnent des résultats plutôt satisfaisants. Alteryx obtient de meilleures performances que Python pour les documents LEA et EAI tandis que Python obtient de meilleures performances qu'Alteryx sur les documents QCF-QR et BS, Alteryx ne fonctionnant pas sur le document BS. Concernant les erreurs commises, la méthode Python a surtout tendance à dire que les documents ne sont pas signés alors qu'ils le sont. La méthode Alteryx a des difficultés à lire certains documents, notamment lorsque ceux-ci ne sont pas très bien scannés ou lorsqu'ils sont un peu décalés. Ceci s'explique par le fait qu'avec Alteryx, le cadre est fixe et ne s'adapte pas à chaque document. L'analyse détaillée des résultats pour chaque document est présentée en Annexe A5. Cependant, compte tenu de la constitution inhomogène des jeux de test, il est important de nuancer les résultats présentés ci-dessus. De plus, nous n'avons pas eu le temps d'approfondir le réglage des paramètres pour chaque type de document, notamment avec la méthode Python. Ainsi, les mêmes paramètres pour le `Cropper` et le `Judger` ont été utilisés

pour les quatre types de documents.

4.5 Conclusion

L'objectif de ce projet était de détecter la présence d'une signature sur quatre types de documents PDF (LEA, EAI, BS, QCF-QR). Cette méthode devait également pouvoir se généraliser à d'autres types de documents.

Dans un premier temps, nous avons recherché en section 4.2 des méthodes déjà implémentées pour détecter la présence d'une signature, dont le package `signature-detect` ou le réseau de neurones `YOLO-v5`. Comme ces méthodes n'ont pas fonctionné dans notre cas, nous avons implémenté notre propre algorithme en Python, détaillé en section 4.3.2.

Nous avons commencé par effectuer un débruitage de l'image avec le module `preprocessing.py`. Puis nous avons implémenté le module `template.py` afin de définir un PDF modèle pour chaque type de document donné. Ce modèle permet de trouver la page contenant la signature et la position des cadres supposés contenir la signature sur l'ensemble des PDF du même type que le PDF modèle. Par la suite, le module `detectSignature.py` implémenté permet de vérifier la présence d'une signature dans les cadres extraits.

Une deuxième méthode pour la détection de signature implémentée sous Alteryx est brièvement décrite en section 4.3.3. La principale différence entre les méthodes Python et Alteryx est que la méthode Python permet d'adapter les cadres supposés contenir la signature à chaque document tandis que les cadres sont fixes avec Alteryx.

Pour évaluer les solutions trouvées, quatre jeux de test de 100 documents de chaque type ont été créés. Les résultats présentés en section 4.4 montrent que la méthode Python est deux à quatre fois plus rapide que la méthode Alteryx. Alteryx obtient de meilleures performances que Python sur les documents LEA et EAI tandis que Python obtient les meilleures performances pour les documents QCF-QR et BS (Alteryx ne fonctionnant pas pour le document BS). Cependant, ces résultats sont à nuancer car les quatre jeux de tests ont été sélectionnés au hasard et ne comportent pas tous les mêmes proportions de documents signés, non signés et avec anomalie.

5 Projet détection de mots interdits

- De Juin à Juillet 2023 -

5.1 Introduction

Les incidents de risque opérationnel (pertes suite à un dysfonctionnement, erreurs humaines, fraudes, problèmes liés à la gestion du personnel, accidents de travail etc) de la CEMP sont enregistrés dans l'outil Osirisk géré par le groupe BPCE. Osirisk est une plateforme permettant un regroupement des données liées aux risques opérationnels et une gestion prospective de l'exposition aux risques. Certains incidents saisis sur Osirisk comportent des mots interdits par le RGPD. L'objectif de ce projet est d'identifier les mots interdits dans le texte de la base de données Osirisk.

Par manque de place dans ce rapport, nous décrirons brièvement ce projet sans entrer dans le détail.

5.2 Présentation des données

La table Osirisk à analyser comporte 50 008 lignes et 6 colonnes. Par ailleurs, une liste de 1039 mots interdits est également fournie.

5.3 Méthodes utilisées

Comme pour le projet précédent, nous commençons par implémenter une solution en Python pour pouvoir facilement explorer le jeu de données et tester diverses solutions. Puis nous adaptons le code python en un Workflow Alteryx.

5.3.1 Programme Python

La première méthode utilisée est la méthode de base qui consiste en une simple recherche des mots interdits dans la base Osirisk. Nous effectuons d'abord un pré-traitement du texte de la base Osirisk et des mots interdits (retrait des accents, de la ponctuation, des chiffres, des valeurs manquantes, des majuscules). Nous recherchons ensuite les mots interdits qui sont dans le texte de la base Osirisk. Cependant, il arrive que certains prénoms comme *Nègre* *Véronique* soient considérés à tort comme des mots interdits. En effet, la recherche identifie *Nègre* comme étant un mot interdit alors qu'il s'agit ici d'un nom de famille. Pour pallier ce problème, nous téléchargeons la liste de tous les prénoms de 1900 à 2021 fournie par l'Insee [23] et soumettons cette liste au même pré-traitement du texte que pour la base Osirisk et les mots interdits. Nous retirons aussi les prénoms composés de moins de deux lettres. Ainsi, lorsque nous recherchons les mots interdits qui sont dans Osirisk, nous considérons qu'un mot n'est pas un mot interdit si celui-ci est précédé ou suivi d'un mot qui se trouve dans

la liste des prénoms. Notons que nous aurions pu utiliser les NER (Reconnaissance d’Entité Nommée) pour déterminer si un mot est un prénom. Cependant, de nombreux prénoms ne sont pas reconnus avec le NER comme par exemple le prénom *Urbain* qui est classé comme un adjectif ou un nom.

La liste de 1039 mots interdits contient plusieurs formes possibles d’un même mot (agressif, agressifs, agressive, agressives) mais n’est pas exhaustive (contient chrétien mais pas chrétienne). Nous implémentons une deuxième méthode qui s’apparente à la première méthode mais inclut avant le pré-traitement de texte une étape de lemmatisation du texte de la base Osirisk, des mots interdits et des prénoms. La lemmatisation consiste à prendre la forme neutre canonique du mot. Par exemple, la phrase « *L’oiseau est posé sur une branche.* » devient « *le oiseau être poser sur un branche* ». Cette étape supplémentaire permet de capturer davantage de mots interdits. La lemmatisation est effectuée avant le pré-traitement afin de ne pas fausser la compréhension du texte par le lemmatiseur.

Les deux méthodes ci-dessus sont des solutions simples ne permettant pas de traiter les cas nécessitant une compréhension du texte. Par exemple, dans la phrase « *La voiture a été percutée par un blaireau* » le mot « *blaireau* » est considéré à tort comme un mot interdit. Il serait alors intéressant d’implémenter une troisième méthode utilisant des algorithmes de NLP plus complexes comme des embeddings ou un réseau de neurones afin d’avoir une meilleure compréhension du texte. Comme il n’existe pas de réseau de neurones spécifiques aux mots interdits de la CEMP, une idée serait de récupérer un réseau de neurones pré-entraîné à détecter des insultes en français. Puis utiliser ensuite un générateur de texte (comme ChatGPT) pour générer des phrases comportant les mots interdits de la CEMP. Enfin réentraîner (fine-tuning) le réseau de neurones sur nos phrases générées afin de détecter spécifiquement les mots interdits de la CEMP. Malheureusement, par volonté de libérer du temps pour d’autres projets, par manque de ressources (pas de GPUs) et car les deux premières méthodes répondent déjà au besoin métier, ces pistes n’ont pas pu être approfondies.

5.3.2 Workflow Alteryx

Nous adaptons les deux méthodes Python en Workflows Alteryx. La solution Alteryx reprend exactement le même principe que la solution Python. Néanmoins, elle a été plus longue à mettre en place en raison du manque de flexibilité d’Alteryx pour la manipulation des chaînes de caractères, de la difficulté d’implémenter des boucles « *for* » ou « *while* » et de problèmes de mémoire. Il en résulte quatre Workflows Alteryx imbriqués les uns dans les autres.

5.4 Résultats

La solution Alteryx est conservée pour sa facilité de mise en production, par comparaison à la solution Python. Avec la première méthode sur Alteryx, on trouve 402 mots interdits. La deuxième méthode Alteryx avec la lemmatisation permet d’identifier 455 mots interdits.

6 Projet avis d'imposition

- De Juillet à Septembre 2023 -

Le dernier projet réalisé porte à nouveau sur de l'extraction d'information à partir de documents PDF. L'objectif de ce projet est de récupérer un maximum d'informations présent sur les avis d'imposition des clients de la CEMP (ayant donné leur consentement). Le but est d'identifier des clients intéressants afin de leur proposer divers services.

Le projet s'apparente au projet de détection de signature vu au chapitre 4, pour ce qui est des outils utilisés pour lire le texte des documents PDF. Le projet a été uniquement réalisé en Alteryx.

Comme précédemment, par manque de place dans ce rapport, ce projet ne sera pas détaillé.

7 Conclusions et perspectives

Dans le cadre de mon stage de fin d'études au département Génie Mathématiques et Modélisation de l'INSA de Toulouse, j'ai effectué un contrat de professionnalisation (alternance) au sein de la Caisse d'Epargne Midi-Pyrénées (CEMP). En intégrant le pôle Data & Décisionnel, j'ai eu l'occasion de réaliser plusieurs projets variés.

J'ai débuté mon alternance par l'installation et la mise en place du poste de travail, avant de traiter mon premier projet décrit au chapitre 3, portant sur de l'analyse exploratoire de données et la prédiction de séries temporelles. La meilleure solution utilise le réseau de neurones MiniRocket mais n'a pas donné des résultats suffisamment précis pour pouvoir être utilisés par le métier¹. J'ai poursuivi avec mon deuxième projet expliqué au chapitre 4 portant sur le traitement d'image et l'extraction des informations des images avec l'OCR Tesseract. Comme aucune méthode existante ne s'appliquait à mon cas, j'ai du implémenter ma propre solution. Les résultats en Python et en Alteryx sont satisfaisants et pourront être utilisés par le métier. Mon troisième projet, présenté au chapitre 5 impliquait du traitement de texte et des méthodes basiques de NLP. Les résultats sur Alteryx satisfaisant le besoin métier et par manque de ressources, je n'ai pas pu approfondir le sujet comme je l'aurais souhaité. Enfin, mon dernier projet évoqué au chapitre 6 portait sur de l'extraction d'information à partir d'avis d'imposition au format PDF. Par manque de place dans ce rapport, je n'ai pas détaillé la méthode utilisée.

1. Différentes équipes représentant les collaborateurs dans des activités diverses.

Durant cette alternance, j'ai beaucoup apprécié la variété des projets que j'ai eu l'occasion de traiter. Ceci m'a permis d'appliquer de nombreuses notions vues à l'INSA mais également d'apprendre énormément. Ayant réalisé l'ensemble de mes projets en quasi-autonomie, cette alternance m'a permis de prendre confiance en mes capacités à rechercher l'information par moi-même et à implémenter des solutions qui fonctionnent.

Par la suite, comme évoqué en section 2.3, la CEMP souhaitait que j'effectue un maximum de projets sur l'ETL Alteryx. De mon point de vue, je ne trouvais pas de grand intérêt à développer en Alteryx, en raison du grand manque de flexibilité de ce logiciel qui ne me semble pas adapté à la complexité de la plupart des projets de Data Science. Je souhaitais également continuer à apprendre et progresser en Python, qui est actuellement un langage phare dans ce domaine, davantage utilisé par rapport à Alteryx. J'ai alors beaucoup apprécié que la CEMP me laisse coder en Python avant de devoir transformer la solution en Alteryx.

Cette expérience m'a permis de me rendre compte que je souhaitais travailler dans une entreprise avec davantage de ressources dédiées au développement de la Data Science, en R&D ou bien en recherche. En effet, j'aime prendre le temps de comprendre et d'approfondir les notions ainsi que de développer de nouvelles solutions, ce qui n'est pas toujours possible dans une entreprise qui ne fait pas de la recherche. Bien que le pôle Data ait des objectifs à remplir, j'ai apprécié que l'on me laisse du temps pour me documenter et tester diverses solutions. Ensuite, je pense que travailler dans une équipe avec d'autres Data Scientists aurait été très instructif mais le travail en autonomie ne m'a pas déplu.

Enfin, cette alternance m'a permis d'avoir une expérience professionnelle très enrichissante dans une grande entreprise, tout en travaillant dans un petit service. Je suis très satisfaite de mon alternance qui s'est très bien déroulée, notamment grâce à la bienveillance et l'aide de mes tuteurs et de toute l'équipe Data.

Annexes

Annexe 1 : Présentation de l'entreprise

0.1 Le groupe Banque Populaire Caisse d'Epargne (BPCE)

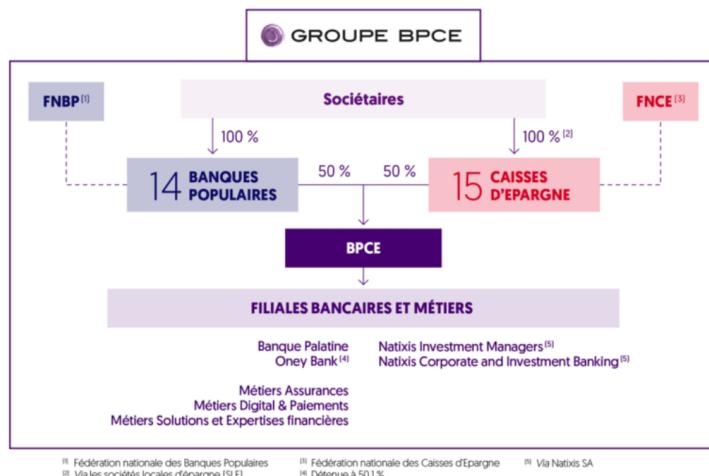


FIGURE 7.1 – Organisation du groupe BPCE

Au terme d'un rapprochement entamé en 2006, les réseaux Banque Populaire et Caisse d'Epargne fusionnent le 31 juillet 2009. Le groupe BPCE est constitué de 15 caisses d'épargne régionales, 14 banques populaires régionales et de multiples filiales. Au total, le groupe BPCE compte 35 millions de clients dont 9 millions de sociétaires au 31 décembre 2022 et contribue à hauteur de 20% au financement de l'économie française. BPCE est présent dans plus de 40 pays et emploie également près de 100 000 personnes. BPCE est organisé de la manière suivante d'après la figure 7.1 :

- **Les sociétaires** : ils sont les propriétaires de 100% du capital des Banques Populaires et des Caisses d'Epargne au travers de parts sociales. Leurs représentants constituent les conseils d'administration des Banques Populaires et les conseils d'orientation et de surveillance des Caisses d'Epargne.
- **Les Banques Populaires et les Caisses d'Epargne** : elles détiennent à parité 100% du capital de BPCE.
- **BPCE** : organe central du groupe.
- **Les filiales de BPCE**, dont Natixis, la Banque Palatine, Oney et les filiales regroupées dans le pôle Solutions et Expertises financières.

0.2 La Caisse d'Epargne de Midi Pyrénées (CEMP)

La Caisse d'Épargne joue un rôle primordial dans la fonctionnalité du groupe BPCE en assurant un rôle de banque de proximité. Elle est divisée sur le territoire en caisses régionales, puis en direction, en secteur et enfin en agence. Ces banques de proximité assurent des services bancaires et financiers, mais se développent en proposant des conseils et financements spécialisés ainsi que des assurances. Les chiffres clés de la CEMP sont résumés dans la figure 7.2 ci-dessous :

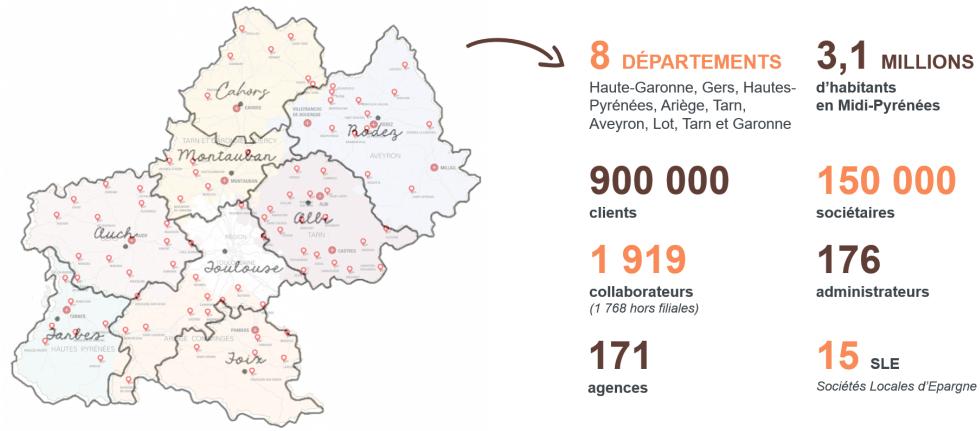


FIGURE 7.2 – Les chiffres clés de la CEMP

L'objectif de la CEMP est d'accompagner tous les clients (particuliers, professionnels, entreprises, collectivités locales, associations) en leur proposant des produits et services bancaires différents (crédit immobilier, assurance, épargne, etc) sur les 8 départements de l'ancienne région Midi-Pyrénées.

La CEMP est composé de 5 pôles :

- Ressources humaines
- Présidence du directoire
- Banque de développement régional
- Finances et moyens généraux
- Banque de détail

Le département dans lequel j'ai effectué mon alternance se situe dans le pôle des finances et moyens généraux, celui-ci est lui-même divisé en plusieurs départements :

- Juridique et contentieux
- Comptabilité fiscalité
- Contrôle de gestion
- Immobilier et services généraux
- Gestion financière
- Data et décisionnel

J'ai effectué mon stage dans le département Data & Décisionnel.

Annexe 2 : Tables des données pour le projet Back Office crédit

Libellé	Définition	Variable retenues pour l'analyse exploratoire	Variable retenues après sélection	quantitative(1)/qualitative(0)/date(d)
COETB	Code Etablissement			0
LIETB	Libellé Etablissement	✓		0
CODOSB	Identifiant dossier bancaire	✓		0
COCO	Identifiant contrat	✓		0
COPRO	Code produit local			0
LIPRLG	Libellé code produit local			0
COPROG	Identifiant produit groupe	✓	✓	0
LIPRO	Libellé produit groupe			0
COOBJ	Code objet prêt	✓	✓	0
LOBJ	Libellé du code objet prêt			0
MEDOS	Montant nominal du dossier	✓	✓	1
TAINT	Taux intérêt	✓	✓	1
TITEGI	TEG Initial prêt	✓	✓	1
QLDDOS	Durée du prêt	✓	✓	1
NBASSGPE	Nbre Assurance Groupe	✓	✓	1
NBASSEXT	Nbre Assurance Externe	✓	✓	1
NBGAR	Nbre de garantie(s)	✓	✓	1
TOPPSC	Top prescription	✓	✓	0
COFAAP	Code famille apporteur affaire			0
LIBLGG	Libellé long du CBO prescripteur			0
COPOST	Code postal du bien financé	✓		0
CONSCE	Nbre de scénarios (Accord Banque à traiter)	✓	✓	1
DDDSPS	Date instruction à traiter	✓		d
DTINS	Date instruction traitée	✓		d
COEMINS	Identifiant employé instruction	✓		0
DATDEC	Date accord banque à traiter	✓		d
DTDEC	Date accord banque traitée	✓		d
COEMDEC	Identifiant employé décision	✓		0
DATEDI	Date édition à traiter	✓	✓	d
DTEDI	Date édition traitée	✓		d
COEMEDI	Identifiant employé édition	✓		0
DELINS	Nbre jrs Instruction	✓	✓	1
DELDEC	Nbre jrs Décision	✓	✓	1
DELEDI	Nbre Jrs Edition	✓	✓	1
TOP_EDITION_BACK_OFFICE	Transfert BackOffice	✓		0

TABLE 7.1 – Tableau récapitulatif de l'ensemble des variables crédits disponibles.

Annexe 3 : Fonctionnement de l'algorithme Rocket

Rocket est basé sur un réseau de neurones convolutif (CNN) qui utilise des noyaux de convolution pour apprendre les caractéristiques des séries temporelles d'entrée. Les paramètres d'un noyau sont la taille, les poids, le biais, la dilation et le padding. Dans le cas des séries temporelles, les noyaux sont des vecteurs de poids $[w_1, \dots, w_{l_{kernel}}]$ où l_{kernel} est la longueur du noyau. Cependant, contrairement aux CNN classiques pour lesquels les paramètres des noyaux sont appris durant l'entraînement, Rocket utilise un grand nombre de noyaux de convolution **aléatoires** en générant des noyaux avec longueur, dilation, padding, poids et biais aléatoires. Pris individuellement, un seul noyau aléatoire n'est pas précis. Mais c'est la combinaison d'une grande variété de noyaux qui permet de capturer des caractéristiques pertinentes pour la prédiction des séries temporelles. Comme les paramètres des noyaux ne sont pas appris et que la génération de noyaux aléatoires n'est pas coûteuse, le coût de calcul des convolutions est faible, ce qui permet à Rocket d'être rapide.

Finalement, Rocket est constitué d'une seule couche de convolution avec un grand nombre de noyaux de convolution. À la fin de cette couche de convolution, la série temporelle d'entrée est transformée en une *feature map*. Par la suite, deux paramètres sont extraits de cette *feature map* et sont utilisés en entrée d'un régresseur linéaire Ridge. Les deux paramètres extraits sont la valeur maximale (similaire au max pooling) et la proportion de valeurs positives *ppv*. Aucune fonction d'activation telle que ReLU n'est appliquée. La figure 7.3 illustre l'architecture de Rocket.

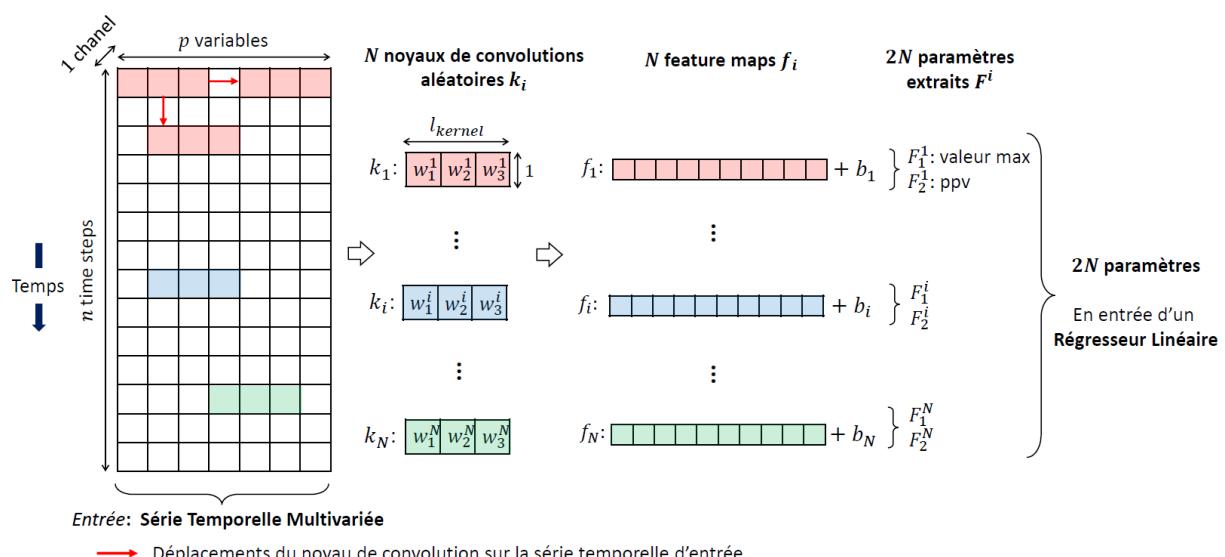


FIGURE 7.3 – Schéma de l'architecture de Rocket dans le cas de séries temporelles multivariées.

L'algorithme MiniRocket (2021) [11] est une reformulation de Rocket et a été choisi pour

ce projet car il est plus rapide que Rocket tout en conservant la même précision. Comme Rocket, MiniRocket transforme les séries temporelles d'entrée en utilisant des noyaux de convolution, et calcule un unique paramètre (ppv) qui servira d'entrée à un régresseur linéaire. Cependant, contrairement à Rocket, MiniRocket utilise un petit ensemble fixe de noyaux, et est presque entièrement déterministe.

Annexe 4 : Workflow Alteryx pour la détection de signature

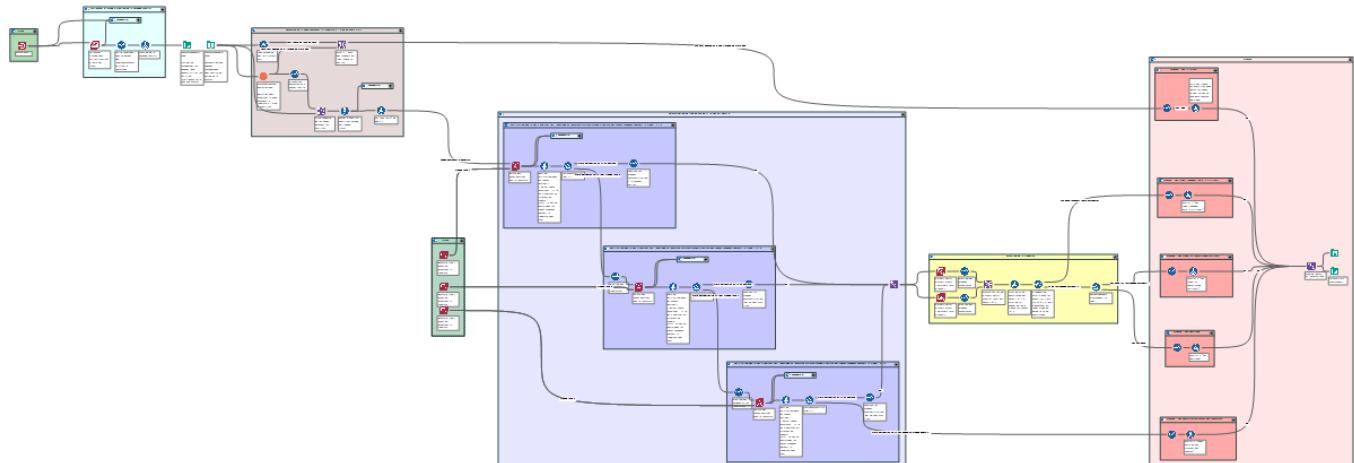


FIGURE 7.4 – Workflow Alteryx pour la détection de signature pour le document EAI.

La figure 7.4 ci-dessus montre le Workflow Alteryx implémenté dans le cas de la détection de signature pour le document EAI. Nous avons un workflow différent par type de PDF (LEA, EAI, BS...). Ce Workflow fait appel à un autre Workflow (non représenté ici) pour détecter la page contenant la signature.

La première étape du Workflow présenté figure 7.4 consiste à charger les PDF puis à effectuer un traitement de l'image pour faciliter la lecture du texte par l'OCR.

On souhaite ensuite identifier la page contenant la signature. Pour cela, on extrait le texte des PDF et on recherche dans le texte la présence de certains mots clés, spécifiques au document, par exemple "Signature", "Déclaration du client", "Données à caractère personnel", etc, pour le document EAI. Si la page que l'on analyse possède suffisamment de mots clés alors il s'agit de la page contenant la signature et on poursuit l'analyse. Sinon, si après avoir analysé toutes les pages, nous n'avons toujours pas trouvé suffisamment de mots clés, nous considérons que le PDF est illisible ou que la page supposée contenir la signature n'existe pas et donc qu'il s'agit peut-être du mauvais document.

Par la suite, nous définissons les PDF modèles et créons autant de modèles que de versions différentes d'un document. Par exemple, comme le EAI possède deux versions différentes, nous créons deux modèles. Nous dessinons sur chaque modèle les rectangles supposés contenir la signature. A la différence de Python, nous n'avons pas accès à la position des rectangles dessinés ni à la position du texte sur la page. Les rectangles contenant la signature sont donc fixes et ne s'adaptent pas à chaque PDF. Pour être sûrs que le PDF n'est pas décalé et donc que l'information contenue dans les rectangles est bien une signature, nous traçons des zones de vérification sur chaque modèle. Si les mots détectés dans les zones de vérification du premier modèle sont corrects, la page que l'on analyse est bien identique

au premier modèle et on poursuit l'analyse. Sinon, nous sommes peut-être en présence du deuxième modèle. Si les mots détectés dans les zones de vérification du deuxième modèle sont corrects, on poursuit l'analyse. Sinon, cela veut dire que le PDF est décalé ou alors le PDF est différent des deux modèles définis.

Enfin, une fois certains que le PDF n'est pas décalé, nous lisons le texte situé dans les rectangles supposés contenir la signature. Si ce texte est suffisamment long, le document est considéré comme signé. Sinon on calcule le pourcentage de remplissage des rectangles. Si ce remplissage est suffisant, le PDF est signé, sinon il n'est pas signé.

Annexe 5 : Résultats détaillés pour le projet Détection de Signature

Les résultats de la section 4.4 sont détaillés ci-dessous. Avec la méthode Python, les résultats possibles obtenus sont :

- PDF signé
- PDF signé numériquement
- PDF non signé
- Page contenant la signature non détectée, noté (P1)

En Alteryx, les résultats possibles obtenus sont :

- PDF signé
- PDF non signé
- Cadre décalé ou PDF différent des modèles, noté (A1)
- PDF illisible ou page supposée contenir la signature non détectée, noté (A2).

Les tables 7.2, 7.3, 7.4 et 7.5 présentent les résultats pour chaque jeu de test sur les quatre types de documents (LEA, EAI, BS, QCFQR). Les résultats écrits en rouge indiquent une erreur.

Label	Résultat Alteryx	Résultat Python	Nombre d'occurrences
Documents papier signés	signé	signé	37
	signé	non signé ou (P1)	9
	(A1) ou (A2)	signé	3
	(A1) ou (A2)	(P1) ou non signé	2
Documents numériques signés	signé	signé numériquement	11
Documents non signés	non signé ou (A2)	non signé ou (P1)	20
Documents mal scannés	(A1) ou (A2)	(P1) ou non signé	2
Documents différents du modèle	(A1) ou (A2)	(P1)	15
	(A2)	signé numériquement	1
Erreurs totales	5	11	
Temps de calcul	43 min	20 min	

TABLE 7.2 – Résultat de la détection de signature sur le jeu de test constitué de 100 LEA.

Tous les documents LEA ont une structure identique et un seul modèle est nécessaire pour détecter la signature. Avec la table 7.2, on observe que les documents non signés, les documents signés numériquement ainsi que les documents présentant une anomalie sont bien identifiés par la méthode Python et Alteryx. Cependant, pour les documents signés papier Python identifie 9 documents comme non signés et ne parvient pas à trouver la page signée pour 2 documents. Alteryx n'arrive pas à lire 5 documents.

Label	Résultat Alteryx	Résultat Python	Nombre d'occurrences
Documents papier signés	signé	signé	22
	signé	(P1)	1
	(A1)	signé	1
	(A1)	non signé	1
Documents numériques signés	signé	signé numériquement	39
	signé	non signé	14
Documents non signés	non signé	non signé	14
	signé	non signé	1
Documents mal scannés	(A1) ou (A2)	(P1)	2
Documents différents du modèle	(A2)	non signé	2
	(A2)	signé	1
	(A2)	signé numériquement	2
Erreurs totales	3	16	
Temps de calcul	38 min	9 min	

TABLE 7.3 – Résultat de la détection de signature sur le jeu de test constitué de 100 EAI.

Le document EAI possède deux versions différentes. Une première analyse de l'ensemble des PDF est alors réalisée avec un premier PDF modèle. Puis, les documents pour lesquels la page signée n'a pas pu être identifiée sont récupérés et soumis à une deuxième analyse avec un deuxième modèle. Avec cette approche en deux étapes, la table 7.3 montre que les documents papier signés ainsi que les documents anormaux sont bien identifiés par Python et Alteryx. En revanche, Python identifie 14 documents comme non signés alors qu'il sont signés numériquement et Alteryx commet une erreur "grave" (indiquée en gras et rouge dans la table) en disant qu'un document est signé alors qu'il ne l'est pas. Python parvient à trouver correctement tous les documents non signés.

Label	Résultat Alteryx	Résultat Python	Nombre d'occurrences
Documents papier signés	-	signé	70
	-	non signé	1
Documents numériques signés	-	signé numériquement	1
Documents non signés	-	non signé	1
Documents mal scannés	-	(P1) ou non signé	2
Documents différents du modèle	-	(P1)	25
Erreurs totales	-	1	
Temps de calcul	-	55 min	

TABLE 7.4 – Résultat de la détection de signature sur le jeu de test constitué de 100 BS.

Le document BS possède une dizaine de versions différentes. Le texte de la page contenant la signature est quasiment identique pour chaque version et seule la position de la signature

diffère. De plus, de nombreux documents (un quart du jeu de test) sont notés comme étant des BS alors qu'il s'agit en réalité d'autres types de document. Avec un seul modèle et en choisissant judicieusement les mots clés `keywords_template` proches de la zone signée, la méthode Python arrive à adapter la position du cadre à chaque version et commet seulement une erreur en disant que le document est non signé alors qu'il l'est, comme le montre la table 7.4. En revanche, comme le cadre contenant la signature est fixe dans la méthode Alteryx, cette méthode ne fonctionne pas pour le document BS.

Label	Résultat Alteryx	Résultat Python	Nombre d'occurrences
Documents papier signés	signé	signé	4
	signé	(P1)	1
	(A1) ou (A2)	signé	22
Documents numériques signés	signé	signé numériquement	69
	(A2)	(P1)	1
Documents non signés	non signé	(A2)	1
Documents mal scannés	(A2)	(P1)	2
Erreurs totales	23	2	
Temps de calcul	23 min	8 min	

TABLE 7.5 – Résultat de la détection de signature sur le jeu de test constitué de 100 QCF-QR.

Tous les documents QCF-QR ont une structure identique et un seul modèle est nécessaire pour détecter la signature. Avec la table 7.5 on voit que les documents signés numériquement ainsi que les documents non signés sont correctement identifiés par les méthodes Python et Alteryx. Cependant, pour les documents papier signés, Alteryx ne parvient pas à lire 22 documents tandis que Python obtient de très bonnes performances.

Bibliographie

- [1] S. Mesnildrey, “Big data : Statistiques et chiffres clés (2023).” <https://www.sales-hacking.com/post/statistiques-big-data>, 2023. En ligne, accès Juillet 2023. (page 1)
- [2] J. Howarth, “57+ amazing artificial intelligence statistics (2023).” <https://explodingtopics.com/blog/ai-statistics#ai-adoption>, February 2023. En ligne, accès Juillet 2023. (page 1)
- [3] M. Baak, R. Koopman, H. Snoek, and S. Klous, “A new correlation coefficient between categorical, ordinal and interval variables with pearson characteristics,” 2018. (page 9)
- [4] V. A. Pimpalkhute, R. Page, A. Kothari, K. M. Bhurchandi, and V. M. Kamble, “Digital image noise estimation using dwt coefficients,” *IEEE Transactions on Image Processing*, vol. 30, pp. 1962–1972, 2021. (page 14)
- [5] C. Dossal, “Cours image optim 5a gmm,” Septembre 2022. (page 14)
- [6] R. Hyndman and G. Athanasopoulos, *Forecasting : Principles and Practice*. Australia : OTexts, 2nd ed., 2018. (page 15, 17)
- [7] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, “Stl : A seasonal-trend decomposition procedure based on loess (with discussion),” *Journal of Official Statistics*, vol. 6, pp. 3–73, 1990. (page 15)
- [8] P. J. Bickel and K. A. Doksum, “An analysis of transformations revisited,” *Journal of the American Statistical Association*, vol. 76, no. 374, pp. 296–311, 1981. (page 16)
- [9] J. Mackinnon, “Critical values for cointegration tests,” *Long-Run Economic Relationships*, 02 1990. (page 19)
- [10] R. Nau, “Statistical forecasting : notes on regression and time series analysis,” August 2020. (page 19)
- [11] A. Dempster, D. F. Schmidt, and G. I. Webb, “MiniRocket,” in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ACM, aug 2021. (page 20, 48)
- [12] A. Dempster, F. Petitjean, and G. I. Webb, “ROCKET : exceptionally fast and accurate time series classification using random convolutional kernels,” *CoRR*, vol. abs/1910.13051, 2019. (page 20)
- [13] M. Malcom and O. Ignacio, “Github repository : timeserie-sai/tsai/10_time_series_classification_and_regression_with_minirocket.ipynb.” https://github.com/timeseriesAI/tsai/blob/main/tutorial_nbs/10_Time_Series_Classification_and_Regression_with_MiniRocket.ipynb, commit be3c787 on November 2022. En ligne, accès Février 2023. (page 21)

- [14] I. Oguiza, “Mise en forme des données avec tsai : Time _series_data_preparation.ipynb.” https://colab.research.google.com/github/timeseriesAI/tsai/blob/master/tutorial_nbs/00c_Time_Series_data_preparation.ipynb#scrollTo=5f1tQ1G2-GxY. En ligne, accès Février 2023. (page 21)
- [15] J. LIU and V. Singh, “Signature detection - a simple tool to detect if there is a signature in an image or a pdf file..” https://github.com/EnzoSeason/signature_detection, commit 7fdfb68 August 2021. En ligne, accès Mai 2023. (page 29, 36)
- [16] A. Joseph, “Deep learning based signature detection and verification.” https://github.com/amaljoseph/Signature-Verification_System_using_YOL0v5-and-CycleGAN, commit ca7f992 March 2022. En ligne, accès Mai 2023. (page 29)
- [17] Hewlett-Packard and R. Smith, “Github repository : Tesseract.” <https://github.com/tesseract-ocr/tesseract>, commit 58b75b3 on May 2023. En ligne, accès Mai 2023. (page 30)
- [18] “Comment fonctionne tesseract : l’ocr open source de google.” <https://commentouvrir.com/tech/comment-fonctionne-tesseract-locr-open-source-de-google/>, 2023. En ligne, accès Juin 2023. (page 30)
- [19] “How does optical character recognition work.” <https://www.baeldung.com/cs/ocr>, 2022. En ligne, accès Juin 2023. (page 30)
- [20] F. Zelic and A. Sable, “How to ocr with tesseract, opencv and python.” <https://nanonets.com/blog/ocr-with-tesseract/>, April 2023. En ligne, accès Mai 2023. (page 30)
- [21] P. Kang and A. K. Singh, “Theailearner - mastering artificial intelligence - bilateral filtering.” <https://theailearner.com/2019/05/07/bilateral-filtering/>, 2019. En ligne, accès Mai 2023. (page 31)
- [22] “Pymupdf 1.22.5 documentation - artifex software inc.-.” https://pymupdf.readthedocs.io/en/latest/document.html#Document.get_sigflags, 2023. En ligne, accès Juin 2023. (page 35)
- [23] “Fichier des prénoms - État civil, institut national de la statistique et des études économiques (insee).” <https://www.insee.fr/fr/statistiques/2540004>, 2021. En ligne, accès Juin 2023. (page 40)