Rapport final : Projet de Fin d'Études

Prévision des flux téléphoniques

Réalisé par :

Mahaut COLLIAUT

Aloïs DE OLIVEIRA

Koffi Guillaume MESSANH

Théo VELLETAZ

Sous la direction de :

David Flouriot

Emmanuelle Guyot

Groupama Loire-Bretagne

Année universitaire 2020 - 2021

Sommaire

Liste des Figures ii

Liste des Tableaux iii

Remercicements iv

Abstract v

Introduction 1

1 Les données de flux téléphoniques des clients de Groupama 1
1.1 Description littéraire des données, de leur qualité, et des retraitements effectués 1
1.2 Analyse de l'évolution des flux téléphoniques au cours du temps 3

2 Retour à la modélisation standard de séries temporelles : le modèle SARIMAX 4

3 Fonctionnement des algorithmes 5
3.1 Multi-Layer Perceptron (MLP) 5
3.2 Réseaux de Neurones Récurrents (RNN) 6
3.3 Long Short-Term Memory (LSTM) 8
3.4 Gated Recurrent Unit (GRU) 10
3.5 Support Vector Regression (SVR) 11

4 Évaluation de la performance des modèles 12

5 Mise en application 13
5.1 MLP 14
5.2 RNN simple 14
5.3 LSTM 14
5.4 GRU 15
5.5 SVR 15

6 Comparaison des résultats 16
6.1 Les performances des modèles obtenues sur l'échantillon test 16
6.2 Exemple d'application : prévision des 14 premiers jours de février 19

Conclusion 20

Bibliographie I

Annexes II
Présentation des résultats de MLP II
Présentation des résultats de RNN simple III
Présentation des résultats de GRU V
Présentation des résultats de SVR VII

Table des figures

1 Évolution du nombre d'appels reçus par jour en 2019 et en 2020 . . . 3
2 Représentation schématique d'un perceptron multi-couches . . . 6
3 Représentation schématique d'un Réseau de Neurones Récurrent . . . 7
4 Deux formes de représentation d'un neurone j de la couche récurrente d'un réseau RNN . . . 8
5 Représentation schématique d'une cellule d'un réseau LSTM . . . 9
6 Représentation schématique d'une cellule d'un réseau GRU . . . 10
7 Représentation de ϵ et ζ dans le modèle SVR . . . 11
8 MSE du LSTM sur les données au jour en fonction du nombre d'*epochs* . . . 17
9 MSE du LSTM sur les données à l'heure en fonction du nombre d'*epochs* . . . 17
10 Prévisions du nombre d'appels journalier faite par le LSTM face aux données réelles . . . 18
11 Prévisions du nombre d'appels journalier faite par le LSTM sur les données horaires face aux données réelles . . . 19
12 Prévisions obtenues pour les 2 premières semaines de février 2021 . . . 20
13 Évolution de la fonction de coût (MSE) sur les échantillons d'apprentissage et de validation au cours des *epochs* du meilleur modèle MLP pour la **prévision au jour** . . . II
14 Prévisions obtenues de ce modèle sur l'échantillon test . . . II
15 Évolution de la fonction de coût (MSE) sur les échantillons d'apprentissage et de validation au cours des *epochs* du meilleur modèle MLP pour la **prévision à l'heure** . . . II
16 Prévisions obtenues de ce modèle sur l'échantillon test . . . II
17 Prévisions au jour obtenues en sommant les prévisions à l'heure . . . III
18 Prévisions au jour obtenues pour 15 jours de suite . . . III
19 Évolution de la fonction de coût (MSE) sur les échantillons d'apprentissage et de validation au cours des *epochs* du meilleur modèle RNN pour la **prévision au jour** . . . III
20 Prévisions obtenues de ce modèle sur l'échantillon test . . . III
21 Évolution de la fonction de coût (MSE) sur les échantillons d'apprentissage et de validation au cours des *epochs* du meilleur modèle RNN pour la **prévision à l'heure** . . . IV
22 Prévisions obtenues de ce modèle sur l'échantillon test . . . IV
23 Prévisions au jour obtenues en sommant les prévisions à l'heure . . . IV
24 Prévisions au jour obtenues pour 14 jours de suite . . . IV
25 Évolution de la fonction de coût (MSE) sur les échantillons d'apprentissage et de validation au cours des *epochs* du meilleur modèle GRU pour la **prévision au jour** . . . V
26 Prévisions obtenues de ce modèle sur l'échantillon test . . . V
27 Évolution de la fonction de coût (MSE) sur les échantillons d'apprentissage et de validation au cours des *epochs* du meilleur modèle GRU pour la **prévision à l'heure** . . . V
28 Prévisions obtenues de ce modèle sur l'échantillon test . . . V
29 Prévisions au jour obtenues en sommant celles à l'heure . . . VI
30 Évolution de la fonction de coût (MSE) sur les échantillons d'apprentissage et de validation au cours des *epochs* du meilleur modèle GRU pour la **prévision au jour des 15 jours suivants** VI
31 Prévisions obtenues **au jour** du meilleur modèle SVR sur l'échantillon test . . . VII
32 Prévisions obtenues **à l'heure** du meilleur modèle SVR sur l'échantillon test . . . VII
33 Prévisions obtenues du meilleur modèle SVR sur la prévision **de 14 jours d'affilée** de l'échantillon test pour le 1^{er} jour à prédire . . . VII
34 Prévisions obtenues du meilleur modèle SVR sur la prévision **de 14 jours d'affilée** de l'échantillon test pour le **$14^{\text{ème}}$** jour à prédire . . . VII
35 Prévisions obtenues **au jour** en **sommant celles à l'heure**, sur l'échantillon test . . . VIII

Liste des tableaux

1 Illustration de la base de données sinistres auto 2
2 Valeurs prises par la variable confinement 2
3 Évolution du nombre moyen d'appels reçus par jour au cours des années 3
4 Variables utilisées pour la prédiction du nombre d'appels reçus à l'instant t 5
5 Scores MAE et RMSE des 5 modèles sur l'échantillon test 16
6 Scores MAE et RMSE des 5 modèles pour les 2 premières semaines de février 19

Remerciements

Nous remercions chaleureusement David Flouriot et Emmanuelle Guyot ainsi que *Groupama Loire-Bretagne* pour le sujet très intéressant proposé ainsi que pour leur accompagnement et leur disponibilité tout au long de ce projet.

Nous remercions également Arthur Katossky ainsi que l'ENSAI pour l'organisation de ce projet de fin d'études.

Abstract

Companies such as Groupama have found that the Covid-19 crisis and the lockdown have had a significant impact on the behavior of their customers. Indeed, the number of phone calls received by Groupama and the number predicted by the models they were using did not coincide anymore. These models failed to adapt to the new data. In this report we try to implement other models that are more flexible to changes. Five different models have been implemented. A support vector regression and 4 neural networks : MLP, RNN, LSTM and GRU. One of the major roles of this project is to be able to predict the number of phone calls over 15 days so that the company will be able to respond better to their customers. First, the performance of the models is compared on the period from October 2020 to January 2021 (which is our test sample), a period that has experienced various restrictions such as the November lockdown and the current curfew. Then each model will be tested on the first 15 days of February 2021, a period with uncommon observations to compare the adaptability of models to new data.

Introduction

La satisfaction des clients constitue un enjeu majeur pour toute entreprise. La mise en place de stratégies et de politiques pour maintenir et accroître la satisfaction clients permet de réduire les coûts directs et indirects liés à l'insatisfaction clients, de valoriser l'image d'une entreprise, et de faire des propositions d'offres plus adaptées.

Dans ce contexte, Groupama a développé des modèles de prévision des flux téléphoniques provenant de ses clients lui permettant de mieux planifier ses équipes et donc d'améliorer la satisfaction de ses clients en améliorant le délai de traitement auprès de ses clients. À date, trois modèles ont été mis en place : un modèle Seasonal Autoregressive Integrated Moving Average (SARIMA), un modèle de réseau de neurones auto-régressif à une couche cachée et le modèle Prophet de Facebook (FBProphet). Selon les résultats obtenus par l'équipe de Groupama, ces deux derniers modèles donnent des résultats plutôt satisfaisants. Cependant, elle a constaté que les modèles ne fonctionnement plus bien lors du confinement, celui-ci ayant changé le comportement des clients.

Face au problème susmentionné, Groupama souhaite développer d'autres modèles de prévision en tenant compte du contexte (période de confinement ou non) et en réduisant au maximum l'erreur de prévision. Ainsi, le but de ce projet est de répondre à cette problématique. Pour cela, nous testons des modèles de réseau de neurones récurrents (RNN, LSTM, GRU) qui sont une classe de réseaux de neurones puissants pour modéliser des données de séquence telles que les séries temporelles, les enregistrements audio, les données textuelles, etc. Nous mettons également en oeuvre un modèle SVR (Support Vector Regression) et un modèle MLP (Multi-Layer Perceptron).

La suite de ce rapport est divisée comme suit. Dans un premier temps, nous décrivons les données mises à notre disposition. Ensuite, nous faisons un bref retour à la modélisation classique des séries temporelles, puis nous présentons le fonctionnement des modèles mis en oeuvre. Quatrièmement, les indicateurs utilisés pour comparer les modèles sont présentés. Cette partie est suivie d'une partie dédiée à l'explication de la mise en application des modèles. Finalement, avant de conclure, nous comparons les résultats des modèles obtenus sur la base de test et nous utilisons nos modèles sur un cas concret d'application.

1 Les données de flux téléphoniques des clients de Groupama

1.1 Description littéraire des données, de leur qualité, et des retraitements effectués

Nous disposons d'une base de données sur la ligne d'appels téléphoniques "sinistres auto", contenant le nombre d'appels reçus par créneaux horaires du 1er janvier 2016 au 13 février 2021. La table 1 nous montre des exemples de données présentes dans cette base de données. La variable `LB_METIER` correspond à la ligne téléphonique concernée, ici la ligne sinistres auto. Les variables `TM_TRANCHE_HEURE`, `DT_DATE`, `LB_JOUR`, `LB_MOIS`, `LB_TRIMESTRE` et `NO_ANNEE` nous indiquent respectivement l'horaire, la date, le jour de la semaine, le mois, le trimestre et l'année. La variable `ferie` vaut 1 si le jour est férié, et 0 sinon, de même pour la variable `vacances`, et enfin la variable `NB_APPEL_RECU` nous indique le nombre d'appels reçus.

Nous avons effectué quelques traitements sur cette base de données avant de l'exploiter. Tout d'abord, nous avons remplacé les valeurs manquantes (NA) par 0, car les lignes téléphoniques correspondantes ne sont pas ouvertes ces jours. De plus, certains jours, plusieurs lignes téléphoniques peuvent être ouvertes en même temps et dans ce cas plusieurs lignes sont présentes pour le même horaire dans la base de données. Nous avons donc sommé les nombres d'appels pour les observations correspondant au même horaire. Par ailleurs, pour tenir compte du contexte dans la modélisation, nous avons créé une variable `confinement` selon les modalités de la table 2.

De plus, dans ce projet nous réalisons des prévisions de la variable `NB_APPEL_RECU` à la fois par heure et par jour. Ainsi, nous avons également créé une nouvelle table à partir de celle-ci en sommant les nombres d'appels reçus par jour, sans garder les informations sur les créneaux horaires.

LB_METIER	TM_TRANCHE_HEURE	DT_DATE	LB_JOUR	LB_MOIS
SIN_AUTO	08 :00 :00	2016-01-01	Vendredi	Janvier
SIN_AUTO	09 :00 :00	2016-01-01	Vendredi	Janvier
SIN_AUTO	10 :00 :00	2016-01-01	Vendredi	Janvier
SIN_AUTO	11 :00 :00	2016-01-01	Vendredi	Janvier
...	...	...	...	...
SIN_AUTO	15 :00 :00	2020-12-15	Mardi	Décembre
SIN_AUTO	16 :00 :00	2020-12-15	Mardi	Décembre
SIN_AUTO	17 :00 :00	2020-12-15	Mardi	Décembre
SIN_AUTO	18 :00 :00	2020-12-15	Mardi	Décembre

LB_TRIMESTRE	NO_ANNEE	ferie	vacances	NB_APPEL_RECU
1er trimestre	2016	1	1	NA
1er trimestre	2016	1	1	NA
1er trimestre	2016	1	1	NA
1er trimestre	2016	1	1	NA
...	...	...	...	...
4eme trimestre	2020	0	0	136
4eme trimestre	2020	0	0	82
4eme trimestre	2020	0	0	58
4eme trimestre	2020	0	0	NA

TABLE 1 – Illustration de la base de données sinistres auto

Note de lecture : le 15/12/2020 à 15h 00 le nombre total d'appels reçu sur une ligne téléphonique est de 136 alors que cette valeur est manquante (NA) le 01/01/2016 à 8h 00 qui est un jour de vacances férié.

Période	*Dates*	*Valeur*
1er confinement	du 17/03/2020 au 11/05/2020	1
2ème confinement	du 30/10/2020 au 15/12/2020	0.66
Couvre-feu	du 16/12/2020 au 14/02/2021	0.33
Ni confinement ni couvre-feu	du 01/01/2016 au 16/03/2020 et du 12/05/2020 au 29/10/2020	0

TABLE 2 – Valeurs prises par la variable confinement

Maintenant que les données sont présentées, nous présentons dans la section suivante quelques résumés issus de l'analyse de ce flux afin de visualiser l'effet du confinement sur le flux téléphonique des clients de Groupama.

1.2 Analyse de l'évolution des flux téléphoniques au cours du temps

Cette section expose une analyse du nombre d'appels reçus par jour, noté `NB_APPEL_RECU_JOUR`, en fonction de l'année. Le tableau 3 présente la moyenne et l'écart type de cette variable au cours du temps. D'une part, on peut voir qu'en moyenne, le nombre d'appels reçus par jour est de 406 en 2020. Ce nombre est bien plus élevé pour les autres années, le maximum étant obtenu en 2018 (495 appels reçus par jour). D'autre part, on note une grande variabilité dans les données. En effet, le minimum de l'écart type du `NB_APPEL_RECU_JOUR` est de 278 obtenu en 2020. Ces résultats montrent la particularité de l'année 2020 due à la crise sanitaire de Covid-19 et aux confinements.

NO_ANNEE	MOY_NB_APPEL_RECU_JOUR	SD_NB_APPEL_RECU_JOUR
2016	491	313
2017	492	311
2018	495	316
2019	468	295
2020	406	278
2021	453	299

TABLE 3 – Évolution du nombre moyen d'appels reçus par jour au cours des années

Note : la 2ème colonne contient le nombre moyen d'appels reçus par jour alors que l'écart type du nombre d'appels reçus par jour se trouve dans la 3ème colonne. Les données utilisées pour l'année 2021 sont celles du 01/01/2021 au 14/02/2021.

Pour mieux visualiser l'effet des confinements sur le comportement des clients, nous pouvons analyser la figure 1 ci-dessous. Elle nous montre l'évolution du nombre d'appels reçus par jour en 2019 et en 2020. On y voit que par rapport à la "tendance habituelle", en 2020 la série du `NB_APPEL_RECU_JOUR` est très basse entre fin mars et début juin. Cette baisse est également constatée en fin de l'année 2020. Ces baisses s'expliquent par les confinements connus en ces périodes.

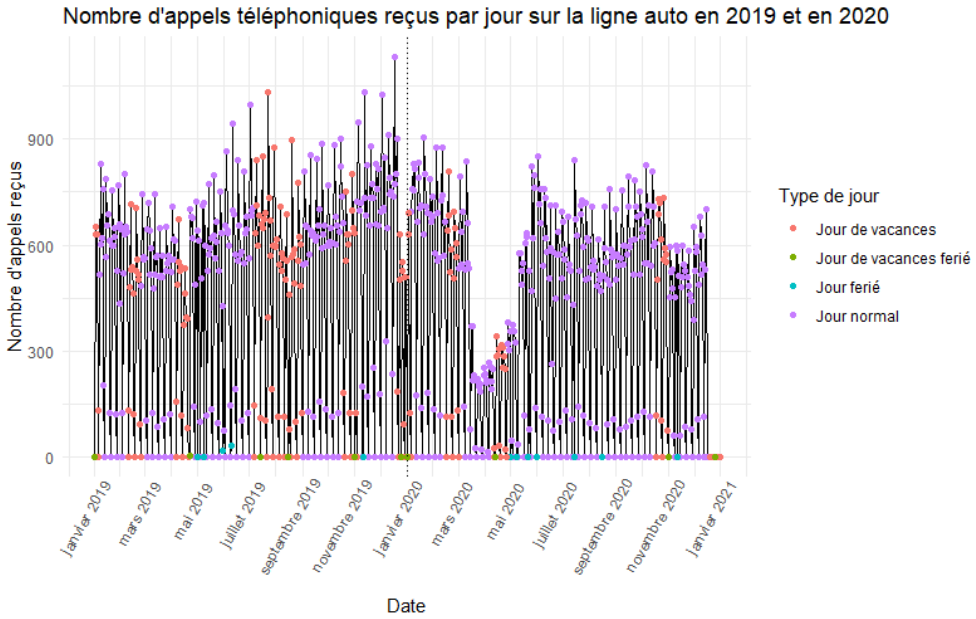

FIGURE 1 – Évolution du nombre d'appels reçus par jour en 2019 et en 2020

À partir des analyses précédentes, il est clair que le confinement a un impact significatif sur le nombre d'appels reçus par jour. Il serait maintenant intéressant de construire des modèles de prévision de cette série qui prennent en compte le contexte (période de confinement ou non) afin de mieux réduire les erreurs de prévision. Avant de présenter le fonctionnement des modèles que nous avons utilisés à cet effet, nous présentons dans la partie qui suit un résumé de la modélisation classique de séries temporelles. On s'inspire ensuite de cette modélisation standard pour construire les modèles mis en oeuvre.

2 Retour à la modélisation standard de séries temporelles : le modèle SARIMAX

La problématique que nous traitons dans ce rapport (prévision de séries temporelles tout en tenant compte du contexte) peut être traitée avec une extension des modèles standards SARIMA. Ces modèles sont connus sous le nom de *Seasonal Autoregressive Integrated Moving Average with Exogenous Variables* (SARIMAX). L'idée est qu'en plus d'utiliser les valeurs passées de la série temporelle y_t pour prédire ses valeurs futures, on tiendra également compte d'un ensemble de variables exogènes $\mathbf{X}_t$ (le contexte) qui pourraient avoir un impact sur la série y_t. Ainsi l'équation générale d'un modèle SARIMAX$(p,d,q)(P,D,Q)_S$ s'écrit ([1], pages 1-5) :

$$\phi(B)_p\Phi(B^S)_P(1-B)^d(1-B^S)^D y_t = c + \phi(B)_p\Phi(B^S)_P(1-B)^d(1-B^S)^D\mathbf{X}_t + \theta(B)_q\Theta(B^S)_Q\epsilon_t \qquad (1)$$

où B est un opérateur de retard tel que $B^k z_t = z_{t-k}$, c est une constante, d et D sont des ordres d'intégration de la série y_t pour la rendre stationnaire, $\phi(B)_p$ et $\theta(B)_q$ sont respectivement des opérateurs auto-régressif d'ordre p et moyenne mobile d'ordre q tels que par exemple $\phi(B)_p = I - \sum_{k=1}^{p}\phi_k B^k$, S est la période de la saisonnalité de la série y_t et ϵ_t est un bruit blanc. L'équation (1) régresse (ou prédit) ainsi y_t sur ses valeurs passées et sur des valeurs passées de $\mathbf{X}_t$ tout en tenant compte de la saisonnalité et d'une éventuelle non stationnarité de la série y_t.

Dans ce projet la série temporelle y_t à prédire est `NB_APPEL_RECU`, et elle sera prédite à partir de ses observations passées sur p jours $(y_{t-1}, y_{t-2}, \ldots, y_{t-p})$, des variables `ferie`, `vacances` et `confinement` notées $\mathbf{X}_t$, et de leurs valeurs passées sur p jours $(\mathbf{X}_{t-1}, \mathbf{X}_{t-2}, \ldots, \mathbf{X}_{t-p})$. $\mathbf{X}_t$ précise donc si le jour de prévision de la série y_t est un jour férié et/ou de vacances et/ou du confinement. La prise en compte de ces informations dans la modélisation est importante car nous avons remarqué que la série y_t vaut 0 les jours fériés, et lors des périodes du confinement elle prend de faibles valeurs par rapport aux autres jours. D'une manière détaillée, y_t sera prédite à partir des informations contenues dans la table 4. Ces informations forment une *séquence* de p éléments que nous notons par :

$$s_p = \begin{bmatrix} y_{t-1} & \mathbf{X}_t & \mathbf{X}_{t-1} \\ y_{t-2} & \mathbf{X}_t & \mathbf{X}_{t-2} \\ . & . & . \\ . & . & . \\ . & . & . \\ y_{t-p} & \mathbf{X}_t & \mathbf{X}_{t-p} \end{bmatrix} \qquad (2)$$

C'est cette séquence qui est attendue comme données d'entrée aux réseaux de neurones récurrents (RNN, LSTM, GRU) pour prédire la valeur future y_t. Pour les autres modèles un redimensionnement de cette séquence est nécessaire. La partie suivante expose comment chaque modèle utilise la séquence s_p.

Valeurs passées de la variable endogène	Variables exogènes					
$\text{NB_APPEL_RECU}_{t-1}$	ferie_t	vacances_t	confinement_t	ferie_{t-1}	vacances_{t-1}	confinement_{t-1}
$\text{NB_APPEL_RECU}_{t-2}$	ferie_t	vacances_t	confinement_t	ferie_{t-2}	vacances_{t-2}	confinement_{t-2}
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
$\text{NB_APPEL_RECU}_{t-p}$	ferie_t	vacances_t	confinement_t	ferie_{t-p}	vacances_{t-p}	confinement_{t-p}

TABLE 4 – Variables utilisées pour la prédiction du nombre d'appels reçus à l'instant t

3 Fonctionnement des algorithmes

Dans cette partie, nous expliquons le fonctionnement des modèles que nous avons mis en oeuvre. Nous commençons par le modèle *Multi-Layer Perceptron*.

3.1 Multi-Layer Perceptron (MLP)

Le perceptron multi-couches, ou *Multi-Layer Perceptron* en anglais (MLP), est un réseau de neurones composé de plusieurs couches organisées de la façon suivante : une couche d'entrée, une couche de sortie, et entre ces 2 couches se trouvent une à plusieurs couches cachées. Les informations vont circuler entre les neurones de la couche d'entrée vers la couche de sortie, on appelle ça un réseau de neurones à propagation avant (*feedforward*). Suivant sa couche d'appartenance, un neurone a un rôle particulier. Les neurones de la couche d'entrée n'ont aucune réelle fonction mis à part le transfert des données d'entrée vers les neurones de la $1^{\text{ère}}$ couche cachée. Les neurones des couches cachées et de la couche de sortie ont deux fonctions distinctes. La première est de produire un résumé des informations reçues en entrée. Pour cela, chaque neurone accorde un poids à chacune des données reçues et le résumé est la somme du produit des données reçues en entrée par leur poids respectif à laquelle on ajoute un biais. La seconde fonction d'un neurone est de calculer une valeur retour qui servira de donnée d'entrée à la couche suivante. Pour ce faire, il utilise une *fonction d'activation*. Les fonctions les plus utilisées sont : la fonction sigmoïde logistique $-sigmoid-$, la fonction tangente hyperbolique $-tanh-$ et la fonction Rectified Linear Unit $-ReLU$. Dans le cas d'une régression, il n'y a qu'un neurone sur la couche de sortie et aucune fonction d'activation n'est utilisée. En conclusion, un MLP reçoit en entrée un vecteur de données, transforme ces données à travers des couches cachées (on dit que le réseau apprend une représentation des données) et produit une prédiction $\hat{y}_t$ à partir de ces transformations.

La figure 2 ci-dessous est une représentation schématique d'un perceptron multi-couches composé d'une couche d'entrée de $7p$ neurones, pour des données en entrée de dimension $7p$, d'une couche cachée de 4 neurones et d'une couche de sortie à un neurone, qui donne la valeur prédite.

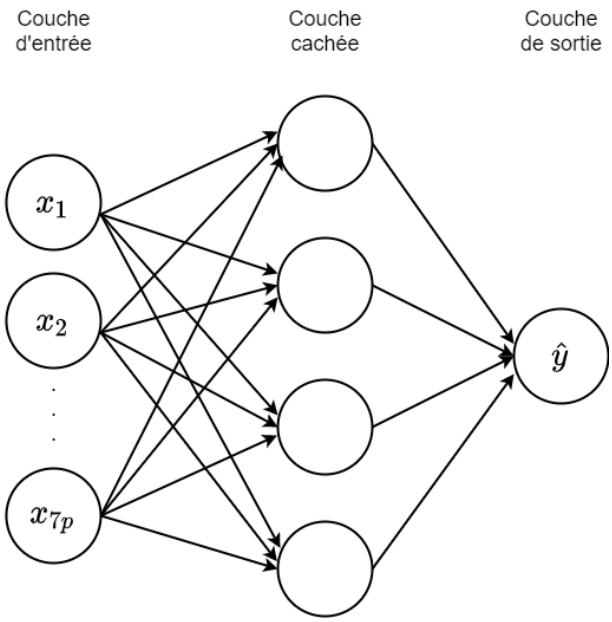

FIGURE 2 – Représentation schématique d'un perceptron multi-couches

En pratique, le nombre de couches cachées varie entre 1 et 2, et nous trouvons le plus souvent des MLP à une couche cachée dans la littérature.

De la description du fonctionnement des MLP, on voit qu'ils ne peuvent pas prendre directement en entrée la séquence s_p (équation (2)) pour la prédiction de la valeur d'une série temporelle à l'instant t. Nous devons donc transformer cette séquence en un vecteur $\mathbf{x} = (y_{t-1}, \mathbf{X}_t, \mathbf{X}_{t-1}, y_{t-2}, \mathbf{X}_t, \mathbf{X}_{t-2}, \ldots, y_{t-p}, \mathbf{X}_t, \mathbf{X}_{t-p})$ de dimension $7p$. Nous passons dans la section suivante aux réseaux de neurones récurrents qui prennent en entrée des séquences de données.

3.2 Réseaux de Neurones Récurrents (RNN)

Les Réseaux de Neurones Récurrents, ou *Recurrent Neural Networks* en anglais (RNN), sont des réseaux de neurones composés d'au moins une couche récurrente, c'est-à-dire que les neurones de cette couche communiquent entre eux. Nous ne sommes plus dans le cas d'un réseau de neurones à propagation avant car les informations ne circulent pas uniquement vers l'avant, elles circulent aussi au sein d'une couche. Les réseaux RNN sont plutôt adaptés pour modéliser des données de séquence telles que des séries temporelles ou le langage naturel. La figure 3 ci-dessous est la représentation schématique d'un réseau de neurones récurrent dont la couche intermédiaire est totalement récurrente, c'est-à-dire que chaque neurone communique avec tous les autres neurones de la couche.

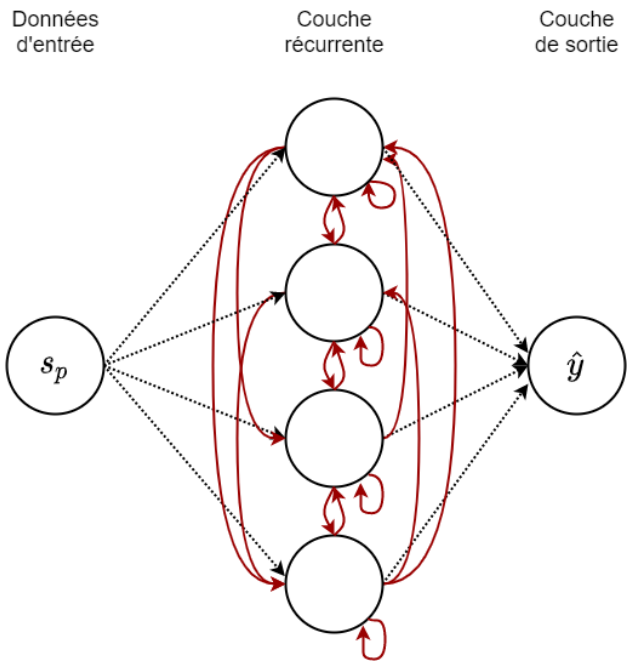

FIGURE 3 – Représentation schématique d'un Réseau de Neurones Récurrent

Regardons plus en détail le traitement de la séquence s_p par le réseau RNN pour la prédiction de la valeur y_t de la série temporelle. En effet, une connexion récurrente permet de réinjecter la sortie d'un neurone en entrée de celui-ci. Ceci est illustré par la figure 4 où un neurone j de la couche récurrente prend en entrée la séquence s_p et retourne une valeur h_{t-1}^j appelée l'*état caché* de la cellule (du neurone). Pour obtenir cette valeur à l'instant $t-1$, le neurone prend de manière séquentielle chaque élément de la séquence s_p dans l'ordre temporel et créé à chaque instant son état caché. En effet, on peut voir sur la figure 4 qu'un neurone j prend à l'instant $t=1$ l'état caché initial $h_0^j = 0$ et le vecteur $(y_{t-p}, \mathbf{X}_t, \mathbf{X}_{t-p})$ de la séquence s_p et fabrique son nouveau état h_1^j. Ce dernier est réinjecté à l'entrée du neurone que l'on combine ensuite avec l'élément suivant de la séquence, le vecteur $(y_{t-(p-1)}, \mathbf{X}_t, \mathbf{X}_{t-(p-1)})$, afin de mettre à jour l'état caché du neurone qui devient h_2^j. Ce processus est ainsi répété jusqu'à l'obtention du dernier état de la cellule h_{t-1}^j obtenu à partir du dernier élément de la séquence $-(y_{t-1}, \mathbf{X}_t, \mathbf{X}_{t-1})-$ et de h_{t-2}^j. La formule de mise à jour de l'état caché est :

$$h_k^j = \phi\left(\mathbf{W}\mathbf{x}_k + \mathbf{U}\mathbf{h}_{k-1}\right)_j, \quad \forall \quad k = 1, \ldots, t-1 \tag{3}$$

où ϕ est une fonction d'activation (*tanh*, *relu*, etc.), $\mathbf{W}$ et $\mathbf{U}$ sont des matrices de poids, $\mathbf{x}_k$ est l'élément de la séquence s_p à l'instant k, $\mathbf{h}_{k-1}$ est le vecteur des h_{k-1}^j pour l'ensemble des neurones de la couche récurrente (l'état caché précédent du réseau RNN), et $(.)_j$ dénote le $j^{\text{ème}}$ élément d'un vecteur.

L'équation (3) est adaptée à partir de celles écrites par ([2], page 2). Le vecteur $\mathbf{h}_k$ retient l'information passée pertinente contenue dans la séquence s_p jusqu'à l'instant k en découvrant les corrélations temporelles contenues dans les données. Ainsi, $\mathbf{h}_{t-1}$, le résumé de l'information passée contenue dans toute la séquence, est utilisé pour prédire la valeur y_t de la série temporelle à l'instant t. La formule de cette prévision est :

$$\hat{y}_t = \mathbf{W}_y \mathbf{h}_{t-1} \tag{4}$$

où $\mathbf{W}_y$ est une matrice de poids.

Dans le cas où le réseau RNN est composé de plusieurs couches récurrentes, l'ensemble des états cachés $(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{t-1})$ forme une nouvelle séquence qui sert de données d'entrée à la couche suivante.

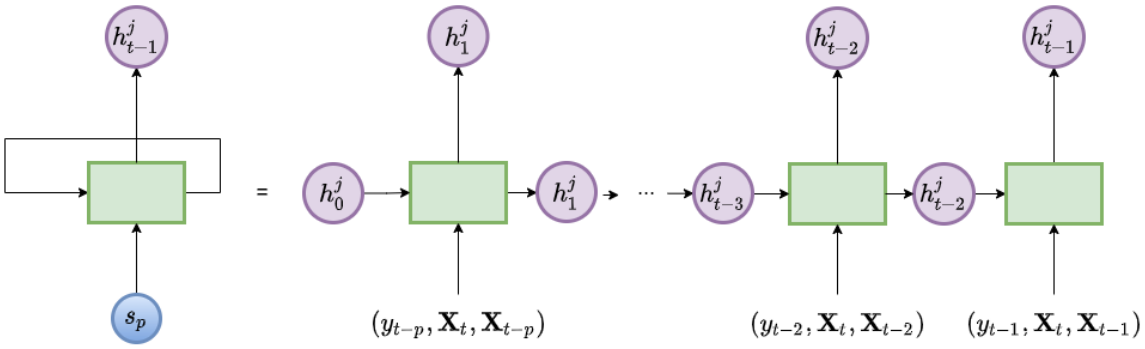

FIGURE 4 – Deux formes de représentation d'un neurone j de la couche récurrente d'un réseau RNN

Note de lecture : à gauche de l'égalité se trouve la version simplifiée du neurone tandis qu'à droite, se trouve la version dépliée dans le temps.

Les poids que l'on trouve dans les modèles de réseaux de neurones sont les paramètres du modèle que le réseau doit "régler" en prenant en entrée plusieurs exemples de la séquence s_p et de y_t. Les "bons" poids sont trouvés en minimisant une fonction de coût $\mathcal{L}(.)$. Celle-ci mesure la performance du réseau quant à la prédiction des valeurs y_t de la série temporelle. Pour minimiser cette fonction, les réseaux RNN, à l'instar des réseaux MLP, utilisent la méthode de la descente du gradient. Cette dernière nécessite le calcul du gradient par rapport aux poids, et ce calcul se fait de droite à gauche connu sous le nom de *Backpropagation Through Time* (BPTT). Si la séquence traitée est longue ($p \to \infty$), dans certains cas, à mesure que l'on avance vers la gauche, la norme du gradient devient très petit (proche de 0) et les poids des premières couches de neurones ne sont quasiment pas modifiés. Par conséquent, les réseaux RNN peuvent facilement oublier des données un petit peu anciennes de la séquence s_p pour la prédiction de y_t. Ce problème est connu dans la littérature sous le nom de *vanishing gradient* (voir [3], pages 1-5 pour plus de détail). Pour pallier à ce problème, des architectures RNN spéciales ont été créées que sont : les Long Short-Term Memory (LSTMs) et les Gated Recurrent Units (GRUs). Les sections suivantes présentent le fonctionnement de ces architectures.

3.3 Long Short-Term Memory (LSTM)

Les réseaux LSTM sont un type spécial de RNN, il s'agit toujours d'une structure récurrente : les neurones au sein d'une même couche communiquent entre eux. Comme susmentionné, la structure des LSTM permet de pallier au problème de *vanishing gradient* que rencontrent parfois les RNN. Cette structure permet aux réseaux récurrents de devenir très bons pour la détection des dépendances à long terme. En effet, pour permettre cela, chaque cellule j va être équipée d'une mémoire interne appelée *Cell state*, notée C^j_t. Cette dernière stockera les informations du passé et sera mise à jour en fonction des nouvelles données qui lui seront présentées. Nous allons décrire plus précisément ce fonctionnement : un réseau LSTM est un RNN auquel on a rajouté des portes pour gérer l'oubli/la prise en compte des valeurs précédentes. Généralement, un réseau LSTM est composé de trois portes. Une porte est en fait une fonction sigmoïde renvoyant des valeurs dans l'intervalle $[0, 1]$. Pour des valeurs proches de 0, la porte sera considérée comme fermée et peu d'informations seront accessibles, et respectivement pour des valeurs proches de 1, la porte sera considérée comme ouverte.

La première porte f^j_t s'appelle *forget gate* (la porte de l'oubli). En fonction de l'état caché précédent $\mathbf{h}_{k-1}$ et du nouvel input $\mathbf{x}_k$, elle calcule et sélectionne les parties de la mémoire qu'il faut garder, les autres parties sont oubliées. Toutes les équations que nous allons voir recensent les changements au sein d'une cellule j d'une couche LSTM. Le calcul fait au niveau de la première porte est donné par :

$$f^j_k = \sigma\left(\mathbf{W}_f \mathbf{x}_k + \mathbf{U}_f \mathbf{h}_{k-1} + \mathbf{b}_f\right)_j, \quad \forall \quad k = 1, \ldots, t-1 \tag{5}$$

où σ est la fonction sigmoïde logistique, $\mathbf{W}_f$ et $\mathbf{U}_f$ sont des matrices de poids, $\mathbf{b}_f$ est le vecteur des biais associés aux neurones, et $(.)_j$ dénote le $j^{\text{ème}}$ élément d'un vecteur.

La deuxième porte i_t s'appelle *input gate* (la porte de l'entrée/la contribution). Elle est composée de deux parties : toujours en fonction de l'état caché précédent $\mathbf{h}_{k-1}$ et du nouvel input $\mathbf{x}_k$, d'un côté elle calcule la partie des nouvelles informations qu'on va mettre à jour i_k^j, et de l'autre côté elle créé les nouvelles valeurs qu'on aurait eu si on avait choisi de toutes les mettre à jour (proposition de nouvelle mémoire interne $\tilde{C}_k^j$). Leur formule est :

$$i_k^j = \sigma\left(\mathbf{W}_i \mathbf{x}_k + \mathbf{U}_i \mathbf{h}_{k-1} + \mathbf{b}_i\right)_j, \quad \forall \quad k = 1, \ldots, t-1 \tag{6}$$

$$\tilde{C}_k^j = \tanh\left(\mathbf{W}_c \mathbf{x}_k + \mathbf{U}_c \mathbf{h}_{k-1} + \mathbf{b}_c\right)_j, \quad \forall \quad k = 1, \ldots, t-1 \tag{7}$$

Et en multipliant les valeurs issues des deux parties de cette porte, on obtient le résultat final de cette porte :

$$\hat{C}_k^j = i_k^j \times \tilde{C}_k^j, \quad \forall \quad k = 1, \ldots, t-1 \tag{8}$$

Les résultats à l'issue des portes f_t et i_t sont ensuite agrégés et représentent les modifications apportées à la mémoire interne du neurone j :

$$C_k^j = f_k^j \times C_{k-1}^j + \hat{C}_k^j, \quad \forall \quad k = 1, \ldots, t-1 \tag{9}$$

La troisième porte se nomme *output gate* (la porte de sortie). Cette porte ne modifie plus la mémoire interne de la cellule. Elle comporte aussi deux parties : d'un coté elle normalise le résultat des deux premières portes en appliquant la fonction *tanh* à C_k^j, et en parallèle une fonction sigmoïde est appliquée sur le nouvel input et l'état caché. Là encore, la multiplication des deux donne le résultat final de cette porte (l'état caché du neurone à l'instant k $-h_k^j$) :

$$o_k^j = \sigma\left(\mathbf{W}_o \mathbf{x}_k + \mathbf{U}_o \mathbf{h}_{k-1} + \mathbf{b}_o\right)_j, \quad \forall \quad k = 1, \ldots, t-1 \tag{10}$$

$$h_k^j = o_k^j \times \tanh(C_k^j), \quad \forall \quad k = 1, \ldots, t-1 \tag{11}$$

De l'équation (9), on voit que la mémoire interne du neurone est très peu modifiée, ce qui permet aux LTSM de se souvenir des informations à long terme. La figure 5 illustre la représentation schématique d'une cellule LSTM. On peut y voir les opérations effectuées.

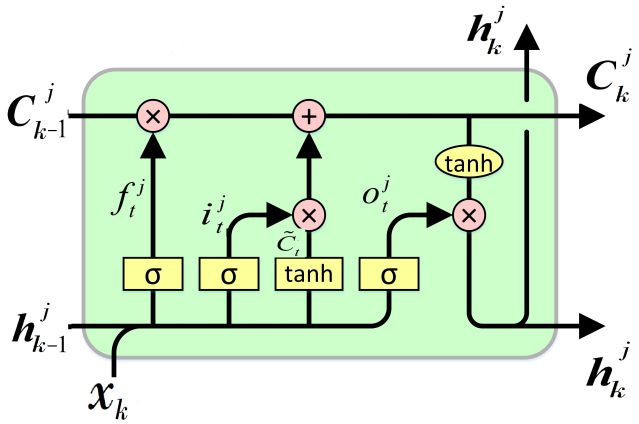

FIGURE 5 – Représentation schématique d'une cellule d'un réseau LSTM

Note : les signes mathématiques montrent les opérations effectuées à l'intérieur de la cellule.

3.4 Gated Recurrent Unit (GRU)

Un réseau *Gated Recurrent Unit* (GRU) est une variante des LSTM introduite en 2014 par [4]. Contrairement aux LSTM, les réseaux GRU pour leur part disposent seulement de deux portes et un état caché. Ces portes sont : *reset gate* et *update gate*.

La porte *reset gate* permet de contrôler la quantité d'informations passées que le réseau doit oublier. Elle regarde en effet l'état caché précédent $\mathbf{h}_{k-1}$, la nouvelle valeur de la séquence $\mathbf{x}_k$, et décide de garder ou pas l'ancien état de la cellule. L'équation de cette opération réalisée par un neurone j du réseau GRU est donnée par :

$$r_k^j = \sigma\left(\mathbf{W}_r \mathbf{x}_k + \mathbf{U}_r \mathbf{h}_{k-1}\right)_j, \quad \forall \quad k = 1, \ldots, t-1 \tag{12}$$

où σ est la fonction sigmoïde logistique, $\mathbf{W}_r$ et $\mathbf{U}_r$ sont des matrices de poids, et $(.)_j$ dénote le $j^{\text{ème}}$ élément d'un vecteur. Ainsi la valeur prise par r_p^j est entre 0 et 1, 0 signifiant "on oublie l'état caché précédent" et 1 signifiant "on laisse l'état caché précédent intact".

La porte *update gate* contrôle la quantité d'informations de l'état caché précédent $\mathbf{h}_{k-1}$ qui seront transférées à l'état caché actuel $\mathbf{h}_k$. Elle agit de manière similaire que la cellule mémoire $\mathbf{C}_k$ du réseau LSTM et aide le réseau GRU à se souvenir des informations à long terme. Cette opération est calculée comme suit :

$$z_k^j = \sigma\left(\mathbf{W}_z \mathbf{x}_k + \mathbf{U}_z \mathbf{h}_{k-1}\right)_j, \quad \forall \quad k = 1, \ldots, t-1 \tag{13}$$

La sortie d'une cellule j d'un réseau GRU, son actuel état caché h_k^j, est définie comme une interpolation linéaire entre l'état caché précédent h_{k-1}^j et un état caché candidat $\tilde{h}_k^j$. Elle est donnée par :

$$h_k^j = (1 - z_k^j) h_{k-1}^j + z_k^j \tilde{h}_k^j, \quad \forall \quad k = 1, \ldots, t-1 \tag{14}$$

où

$$\tilde{h}_k^j = \phi\left(\mathbf{W}\mathbf{x}_k + \mathbf{U}(\mathbf{r}_k \odot \mathbf{h}_{k-1})\right)_j, \quad \forall \quad k = 1, \ldots, t-1 \tag{15}$$

avec $\mathbf{r}_k$ le vecteur des r_k^j, $\odot$ le produit matriciel de Hadamard (produit élément par élément), et ϕ est une fonction d'activation (*tanh*, *relu*, etc.).

Les équations (12) à (15) sont adaptées à partir de celles écrites par ([4], page 3). La figure 6 illustre la représentation schématique d'une cellule GRU.

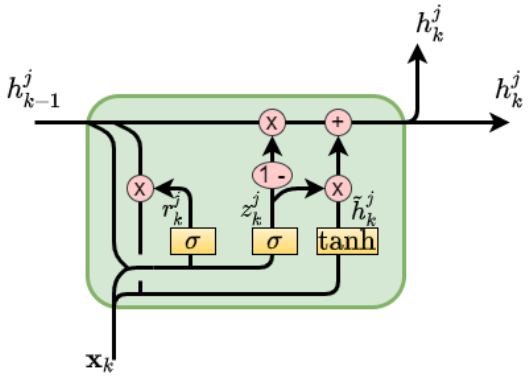

FIGURE 6 – Représentation schématique d'une cellule d'un réseau GRU

Note : les signes mathématiques montrent les opérations effectuées à l'intérieur de la cellule.

3.5 Support Vector Regression (SVR)

Si les modèles de Support Vector Machines (SVM) sont connus pour résoudre des problèmes de classification (à l'aide de Support Vector Classification, SVC), ils sont moins connus lors de problèmes de régression. Peu documentés, les Support Vector Regression (SVR ou encore Régression par séparateur à vaste marge) ont cependant souvent été mis en avant lors de travaux de prévision à court terme de séries temporelles.

L'intérêt de ce modèle vis-à-vis des régressions plus habituelles est de ne pas tenir compte de *petites erreurs* qui se situent dans une certaine marge de tolérance, cela permet d'améliorer la capacité de généralisation du modèle. Il est donc nécessaire d'utiliser une fonction de perte insensible à l'utilisation de ce paramètre de tolérance.

L'explication mathématique réalisée par Y. Yangl et al. [5] ressemble beaucoup à la formulation primale des SVM.

Soit un ensemble de n observations et m features (régresseurs), en partant de la fonction linéaire définie par :

$$Y = f(X) = Xw + b \tag{16}$$

avec Y un vecteur colonne de taille n correspondant aux valeurs à prédire, $X \in R^{n \times m+1}$ correspondant aux prédicteurs associés et w un vecteur colonne de taille $m+1$ les coefficients associés aux prédicteurs.

L'approche des moindres carrés prescrit de choisir les paramètres w et b de sorte à minimiser la somme des écarts au carré. Le problème peut être formulé comme le problème de minimisation suivant :

$$\begin{aligned} &\min \frac{1}{2} w^T w + C \sum_{i=1}^{n} |\zeta_i| \\ &s.t : Xw + b - Y \leq \epsilon + \zeta \end{aligned} \tag{17}$$

avec $C \in R_+$ le poids de pénalité pour les erreurs, $\epsilon \in R_+$ l'erreur maximale définissant la marge d'erreur et $\zeta \in R^n$ l'écart par rapport à la marge (voir la figure 7 suivante).

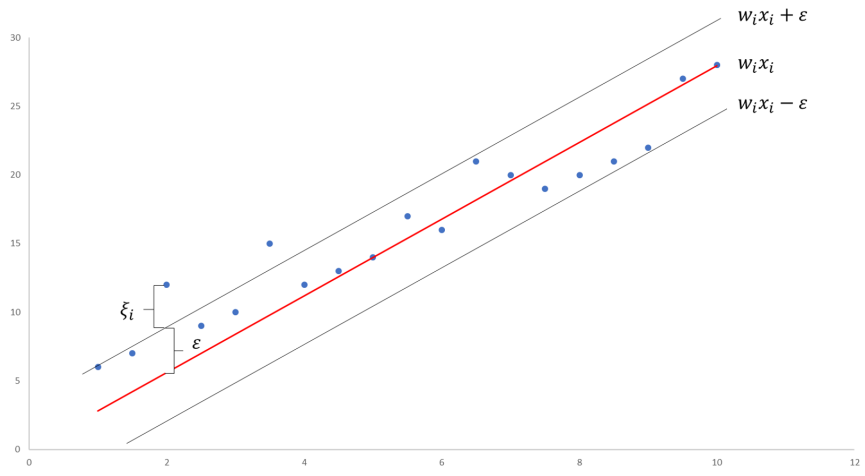

FIGURE 7 – Représentation de ϵ et ζ dans le modèle SVR

La formule (17) visant à minimiser la fonction de perte insensible à ϵ est un programme quadratique convexe et peut être résolu à l'aide de la méthode du Lagrangien :

$$L(w, b, \zeta_i, \alpha_i) = \frac{1}{2} w^T w + C \sum_{i=1}^{n} |\zeta_i| - \sum_{i=1}^{n} \alpha_i [y_i (w^T X_i + b) - 1 + \epsilon_i] \tag{18}$$

avec α_i les coefficients du Lagrangien.

En utilisant le théorème de Kuhn-Tucher, nous savons que la solution de l'optimisation est le point de selle de la fonction de Lagrange. Les coefficients non nuls du Lagrangien peuvent être déterminés de la manière suivante :

$$\begin{aligned} &\max_{\alpha} \Big[-\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j (X_i^T X_j) \alpha_i \alpha_j + \sum_{i=1}^{n} \alpha_i \Big] \\ &s.t : \sum_{i=1}^{n} \alpha_i y_i = 0, 0 \leq \alpha_i \leq C \\ &w = \sum_{i=1}^{n} y_i \alpha_i X_i \end{aligned} \tag{19}$$

Tout les formules précédentes correspondent au cas linéaire, or le cas non-linéaire nous intéresse davantage. Pour passer d'un résultat linéaire à non-linéaire, il est possible d'utiliser la technique du noyau (*kernel trick*) : en représentant w par $X^T u$ et en remplaçant XX^T par une matrice de noyaux $K(X, X^T)$, ainsi l'équation (17) devient :

$$\begin{aligned} \min \frac{1}{2} (u^T u + b^2) + C \sum_{i=1}^{n} |K(X, X^T) u + b - y_i| \\ s.t : wX + b - y \leq \epsilon + \zeta \end{aligned} \tag{20}$$

avec $K(X, X^T)$ un *kernelmap*.

Note : si la partie théorique utilise X et non s_p cela est dû à la dimension de ce dernier. En effet, pour le bon fonctionnement du SVR, il a été nécessaire de redimensionner les données en entrée. Ainsi au lieu d'avoir une dimension (n_{obs}, p, $n_{features}$), nous aurons (n_{obs}, $p \times n_{features}$).

Avant de passer à la mise en application des modèles présentés sur nos données, tout d'abord nous présentons les indicateurs de performances que nous avons utilisés pour les comparer.

4 Évaluation de la performance des modèles

L'objectif des modèles construits dans ce projet est de permettre à Groupama de prévoir, d'une manière assez précise, le nombre d'appels qu'elle est susceptible de recevoir dans les jours futurs en se basant sur des observations du passé (la séquence s_p). Par conséquent, nous devons évaluer la capacité prédictive de nos modèles. A cet effet, nous divisons la base de données en trois parties car l'évaluation d'un modèle sur la base de données ayant servi à sa construction engendrerait un biais positif. Ces échantillons sont :

- L'échantillon d'apprentissage servira à la construction des modèles et contient les données du 01/01/2016 au 31/05/2020.
- L'échantillon de validation permettra de contrôler le surapprentissage des réseaux de neurones au cours de l'apprentissage. Il servira ainsi au choix des meilleurs hyperparamètres des modèles. On lui a dédié les données du 01/06/2020 au 30/09/2020, soit des données sur 4 mois.
- L'échantillon test servira à l'évaluation de la performance prédictive des modèles et à la comparaison des différents modèles. Il contient les données du 01/10/2020 au 31/01/2020, soit également des données sur 4 mois.

Note : pour le SVR, cela diffère légèrement étant donné que l'échantillon de validation est intégré dans l'échantillon d'apprentissage.

Par ailleurs, la capacité prédictive des modèles se mesure au travers des indicateurs de performance. En effet, si les modèles fonctionnent bien, on doit s'attendre à ce que la valeur prédite $\hat{y_t}$ soit très proche de la vraie valeur y_t. Nous devons ainsi calculer des indicateurs de performance qui reflètent cette idée.

Dans ce projet, nous avons calculé les indicateurs suivants :

- Erreur Absolue Moyenne ou *Mean Absolute Error* (MAE) en anglais :

$$MAE = \frac{1}{T}\sum_{t=1}^{T}|y_t - \hat{y}_t| \tag{21}$$

- Racine Carrée de l'Erreur Quadratique Moyenne ou *Root Mean Squared Error* (RMSE) en anglais :

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2} \tag{22}$$

où T est le nombre total de y_t contenu dans l'échantillon test.

On pouvait également calculer d'autres indicateurs comme par exemple le *Mean Absolute Percentage Error* (MAPE) qui donne l'erreur moyenne commise pour la prévision de 100 appels reçus calculé comme :

$$MAPE = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \times 100\% \tag{22}$$

Cependant, il n'a pas été calculé car certaines valeurs de y_t valent 0.

5 Mise en application

Cette partie présente la mise en oeuvre des modèles. Avant de passer au détail, nous expliquons les hyperparamètres utilisés dans ces modèles puisque les modèles de réseaux de neurones sont des modèles paramétriques. Comme hyperparamètres nous distinguons :

— le nombre de couches cachées à faire intervenir dans la construction du réseau.
— le nombre de neurones à mettre sur chacune des couches cachées.
— le *rate* de la couche *dropout*. Le *dropout* est une méthode de régularisation utilisée pour réduire le surapprentissage. Cette méthode consiste à ignorer certains neurones des couches cachées pendant la phase d'apprentissage. Les neurones à désactiver sont sélectionnés aléatoirement selon une probabilité donnée qu'est le paramètre *rate*. L'idée ici est de permettre au réseau d'apprendre même en l'absence de certaines informations. Il devient donc moins sensible au poids $\mathbf{W}$ des neurones et plus stable face à la suppression de données. Ainsi, le réseau ne colle pas trop aux données et la généralisation devient plus facile (voir [6] pour plus de détail).
— la méthode de la descente du gradient à utiliser pour la minimisation de la fonction de coût $\mathcal{L}(.)$. Parmi ces méthodes, on en distingue : *Adam* (adaptive moment estimation), *RMSprop*, *Adagrad* etc.
— le pas de la méthode de la descente du gradient. Il correspond au *learning_rate* de la méthode.
— *n_epochs* : le nombre d'*epochs* que le modèle sera entraîné, une *epoch* correspondant à une itération complète (un apprentissage complet) sur l'échantillon d'apprentissage.
— le paramètre *patience* de la méthode *EarlyStopping*. Il correspond au nombre d'*epochs* sans amélioration de la fonction de coût de l'échantillon de validation après lesquelles l'entraînement sera arrêté. Tout comme la méthode de *dropout*, cette technique fait également partie des méthodes de régularisation des réseaux de neurones.

— *batch_size* : c'est le nombre d'exemples de (s_p, y_t) que le réseau utilise à chaque étape de mise à jour des poids $\mathbf{W}$ par la méthode de la descente du gradient.

N.B : Pour le SVR, les paramètres à optimiser sont présentés dans la sous-partie associée.

Ces hyperparamètres sont à fixer afin d'entraîner les réseaux de neurones. Étant donné qu'il n'existe pas de formules pour le choix de leurs valeurs optimales, celles-ci ont été déterminées expérimentalement en testant plusieurs combinaisons. Par ailleurs, dans ce projet, la fonction de coût $\mathcal{L}(.)$ optimisée lors de la phase d'entraînement est le *Root Mean Squared Error* (MSE). Les sections suivantes exposent les résultats obtenus.

Nous devons également fixer la valeur du paramètre p, qui correspond au nombre de lignes dans le tableau s_p présenté par l'équation (2), c'est-à-dire le nombre de jours/heures en arrière à utiliser pour prédire le nombre d'appels reçus suivant.

5.1 MLP

Dans le cadre du Multi-Layer Perceptron, nous choisissons une valeur de $p = 31$, ainsi nous mettons en entrée du modèle les données des 31 jours précédents pour prédire le nombre d'appels du jour suivant. Nous avons testé différentes valeurs de ce paramètre, et avons remarqué que plus ce nombre était grand plus les résultats étaient bons. Pour le modèle de prédiction à l'heure, nous avons gardé cette valeur de $p = 31$, qui correspond donc à un peu moins de 3 jours car nous disposons de 11 créneaux horaires par jour. Nous avons remarqué que tant que nous restons dans l'ordre des quelques jours, la valeur précise de ce paramètre a une très faible influence sur les résultats, qui n'est pas significative.

Pour le modèle de prédiction au jour, nous avons retenu le modèle avec 20 neurones dans la couche cachée, un maximum de 100 *epochs* avec la méthode *EarlyStopping* paramétrée avec une valeur de *patience* = 20, et nous avons fixé la valeur du *batch_size* à 10. De plus, la couche cachée utilise la fonction d'activation *ReLu*, nous utilisons également la fonction de coût MSE, et la méthode de descente de gradient *Adam* avec un $learning_rate = 0,001$. Pour le modèle de prédiction à l'heure, nous avons gardé les mêmes paramètres, qui nous donnent également de bons résultats pour ce modèle.

Pour prédire plusieurs jours ou plusieurs heures à la fois, nous rappelons successivement le modèle. Ainsi, nous prédisons la valeur du nombre d'appels du jour suivant, puis nous modifions notre tableau s_p en ajoutant la valeur prédite sur la première ligne et en supprimant la dernière ligne, puis nous prédisons une nouvelle fois le nombre d'appels pour le jour suivant avec ce nouveau tableau s_p, et ainsi de suite.

5.2 RNN simple

Pour le RNN simple, nous gardons le paramètre $p = 31$ pour le modèle de prédiction au jour, et 20 neurones dans la couche cachée. Nous entraînons notre modèle sur un maximum de 200 *epochs* en utilisant la méthode *EarlyStopping* avec un paramètre de 20 *epochs*, et une valeur de *batch_size* de 7. La couche cachée utilise la fonction d'activation *ReLu*, nous utilisons également la fonction de coût MSE et la méthode de descente de gradient *RMSprop* avec un *learning_rate* de 0,003.

Pour le modèle de prédiction à l'heure, nous avons uniquement changé la valeur du paramètre p à 11, soit un jour. Les valeurs des autres paramètres sont inchangées.

5.3 LSTM

Pour notre LSTM, nous avons retenu un historique de $p = 7$ jours pour les prévisions au jour et $p = 11$ heures pour les prévisions à l'heure (11 créneaux horaires forment une journée). Des historiques plus longs avec $p = 30$ *jours* ou $p = 7 \times 11$ *heures* ont d'abord été testés mais les résultats obtenus n'étaient pas bons et les temps de calculs devenaient très longs.

En ce qui concerne le réseau en lui même, le modèle qui s'est montré le plus performant sur les prévisions au jour comporte deux couches cachées de 70 neurones, et une couche *dropout* avec un $rate = 0,1$. Pour les prédictions à l'heure on utilise aussi deux couches cachées, la première avec 80 neurones et la seconde avec 50 neurones. Pour les fonctions d'activation on utilise des *ReLu*. On entraîne le modèle au maximum sur 500 *epochs*, et on utilise la méthode du *EarlyStopping* avec le paramètre $patience = 50$, c'est à dire que si il n'y a plus d'amélioration de la fonction de coût sur l'échantillon de validation après 50 *epochs*, l'entraînement du modèle s'arrête. La fonction de coût utilisée est le MSE, et on la minimise avec la méthode *Adam* avec un pas de $learning_rate = 0,0005$.

La dernière couche (la couche de sortie) du modèle est une couche $Dense(1)$. Ceci signifie que le modèle ne prédit qu'une valeur, et donc si l'on veut pouvoir prédire le nombre d'appels téléphoniques sur une séquence de 2 jours, il faut que le modèle prédise le nombre d'appel du premier jour, puis réutilise cette valeur pour prédire le second jour. Si l'on veut prédire des séquences allant jusqu'à 15 jours le modèle se comporte relativement bien, cependant vu que l'historique du modèle n'est que de $p = 7$ jours, un biais accumulé important peut apparaître lorsqu'on veut prédire des séquences plus longues.

5.4 GRU

Concernant le modèle GRU, l'historique utilisé est de 15 jours ($p = 15$ dans la séquence s_p) pour les prévisions au jour tout comme à l'heure. L'entraînement des modèles en utilisant un historique de 30 jours a été mis en oeuvre. Cependant, il prenait énormément de temps et donnait des résultats similaires à ceux des entraînements avec $p = 15$. Par ailleurs, l'hyperparamètre *patience* a été fixé à 50.

Après des expérimentations et tests de plusieurs modèles GRU, nous avons retenu un modèle GRU à deux couches cachées. Pour les prévisions à l'heure, les hyperparamètres donnant une bonne capacité de généralisation en matière de MAE et de RMSE (ceux donnant de meilleurs résultats sur l'échantillon test) ont été obtenus en mettant 200 neurones sur chaque couche cachée, une couche *dropout* avec un $rate = 0,2$ après chaque couche cachée, et en fixant $n_epochs = 150$ et $batch_size = 100$. La méthode *RMSprop* avec un pas de $learning_rate = 0,00001$ a été retenue pour la minimisation de la fonction de coût.

De même, pour les prévisions au jour, un modèle GRU à deux couches cachées est retenu. Cette fois-ci, la méthode *Adam* avec un pas de $learning_rate = 0,001$ était la meilleure pour la minimisation de la fonction de coût. De plus, les meilleurs résultats sont obtenus en mettant 100 neurones sur chaque couche cachée, une couche *dropout* avec un $rate = 0,1$ après chaque couche cachée, et en fixant $n_epochs = 600$ et $batch_size = 32$.

Une prévision directe des 15 jours suivants a été réalisée en faisant des prévisions à l'heure et au jour. Elle consiste à mettre 15 neurones sur la couche de sortie pour les prévisions au jour et $11 \times 15 = 165$ pour celles à l'heure. Les prévisions au jour présentent de meilleurs résultats et sont obtenues avec les mêmes paramètres que ceux des prévisions au jour suivant précédemment présentées. Ceci est probablement dû au fait qu'on prédit plus d'observations dans le cadre des prévisions à l'heure qu'au jour ($11 \times 15 = 165$ observations pour les prévisions à l'heure contre 15 au jour). De plus, la fonction d'activation *ReLu* a été utilisée pour tous les modèles GRU testés.

5.5 SVR

Les paramètres à optimiser diffèrent pour le SVR. Les paramètres à optimiser ont été :

— n_step : l'historique (correspond à p de s_p).
— *kernel* : le noyau à utiliser.
— C : le paramètre de régularisation (poids de pénalité pour les erreurs).
— *gamma* : le coefficient du noyau.

Afin d'optimiser ces paramètres, il avait été envisagé de réaliser un *GridSearchCV* mais n'ayant pas la capacité matérielle de le réaliser, cela a été fait manuellement.

Après différents tests, pour l'historique, il a été retenu 7 jours pour le modèle au jour et 11 heures (soit une journée) pour le modèle à l'heure.
Les noyaux "rbf", "poly" et "linear" (pour respectivement un noyau Radial Basis Function, Polynomial et Linéaire) ont été testés. Mais seul le noyau "rbf" a été retenu, étant donné qu'il était le plus rapide a entraîné et le plus performant.
De même pour *gamma*, la valeur "scale" (correspondant à $1/(n_features \times X.var())$) a été retenue. La valeur "auto" (correspondant à $1/n_features$) et les valeurs choisies arbitrairement ne permettaient pas d'obtenir un modèle plus performant que "scale".
Le coefficient C a été déterminé par tâtonnement. Les valeurs $1,625 \times 10^4$ et 1×10^4 ont été retenues respectivement pour le modèle au jour et le modèle à l'heure.

A présent que les meilleurs modèles retenus dans chaque cas sont présentés, la partie suivante présente leurs performances en matière de MAE et de RMSE. Une comparaison des modèles est faite.

6 Comparaison des résultats

6.1 Les performances des modèles obtenues sur l'échantillon test

Cette section résume les performances, en matière de MAE et de RMSE, obtenues avec les meilleurs modèles trouvés dans chaque cas sur la période de Octobre 2020 à Janvier 2021 . Ces performances sont données par le tableau 5.

	Jour suivant		**Heure suivante**		**Jour en sommant les heures**		**14 jours suivants**	
	MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
MLP	57,99 ≈ 58	77,16 ≈ 77	7,63 ≈ 8	11,06 ≈ 11	25,97 ≈ 26	33,76 ≈ 34	69,25 ≈ 69	89,24 ≈ 89
RNN	41,17 ≈ 41	65,09 ≈ 65	8,11 ≈ 8	12,02 ≈ 12	34,85 ≈ 35	42,45 ≈ 42	51,17 ≈ 51	77,48 ≈ 77
LSTM	40,75 ≈ 41	60,92 ≈ 61	7,11 ≈ 7	11,15 ≈ 11	29,53 ≈ 30	38,87 ≈ 39	51,55 ≈ 52	76,64 ≈ 77
GRU	39,07 ≈ 39	56,99 ≈ 57	6,63 ≈ 7	10,65 ≈ 11	28,65 ≈ 29	39,49 ≈ 39	78,50 ≈ 79	119,90 ≈ 120
SVR	46,62 ≈ 47	84,92 ≈ 85	6,98 ≈ 7	11,53 ≈ 12	29,42 ≈ 29	40,64 ≈ 41	63,80 ≈ 64	115,88 ≈ 116

TABLE 5 – Scores MAE et RMSE des 5 modèles sur l'échantillon test

Dans le tableau 5 nous recensons les différents scores obtenus par nos modèles pour 4 prévisions différentes :
— la prévision *Jour suivant* consiste à prédire uniquement le nombre d'appels du lendemain.
— la prévision *Heure suivante* consiste à prédire uniquement le nombre d'appels du prochain créneau horaire.
— la prévision *Jour en sommant les heures* consiste à prédire le nombre d'appels pour chaque créneau horaire d'une journée puis de sommer ces prévisions pour avoir une prévision au jour.
— la prévision 14 *jours suivants* consiste à prédire le nombre d'appels des 14 prochains jours.

Nous pouvons voir sur le tableau 5 qu'en termes de score MAE et RMSE, que tous les réseaux de neurones récurrents (RNN, LSTM et GRU) obtiennent des résultats similaires lorsqu'il s'agit de prédire qu'une seule valeur. C'est le modèle GRU qui se révèle être le plus performant pour ce type de prédiction. Cependant pour prédire sur des plus longues périodes, on observe que ce sont le RNN et le LSTM qui se comportent mieux. En effet, avec un MAE moyen sur 14 jours d'environ 51 pour le RNN, cela signifie qu'en moyenne, sur une période de 14 jours le modèle se trompera de 51 appels par jour. Dans un point de vu d'application pratique, cette prédiction est la plus importante puisque l'entreprise doit être capable d'établir un planning longtemps à l'avance et non pas seulement le jour précédent. C'est pourquoi le RNN ou le LSTM seraient sûrement les modèles retenus.

Vu que de manière générale c'est le LSTM qui fournit les meilleurs scores, nous allons présenter les graphiques de la fonction de coût en fonction du nombre d'*epochs* pour les deux modèles LSTM, celui fonctionnant sur les prévisions au jour (figure 8) et celui fonctionnant sur les données à l'heure (figure 9). Les graphiques obtenus pour les autres modèles sont présentés en annexes 6.2. On peut observer grâce à ces deux graphes que les modèles apprennent correctement et qu'ils ne sont pas sujet au sur-apprentissage. En effet, la fonction de coût (MSE) de l'échantillon de validation est en-dessous de celle de l'échantillon d'apprentissage au cours des *epochs*.

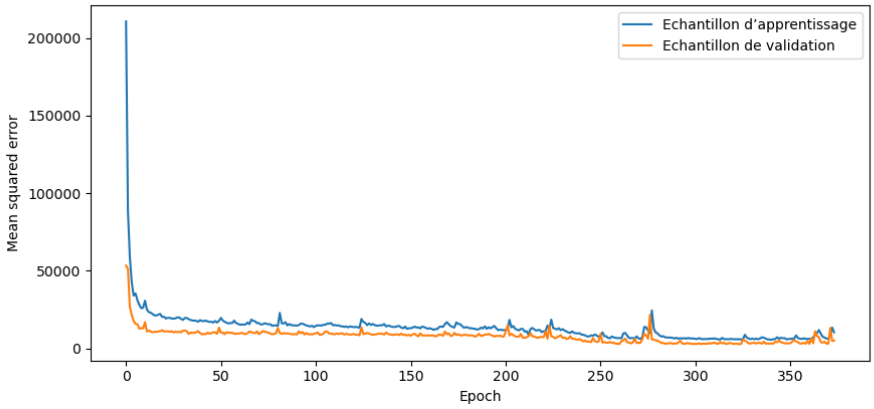

FIGURE 8 – MSE du LSTM sur les données au jour en fonction du nombre d'*epochs*

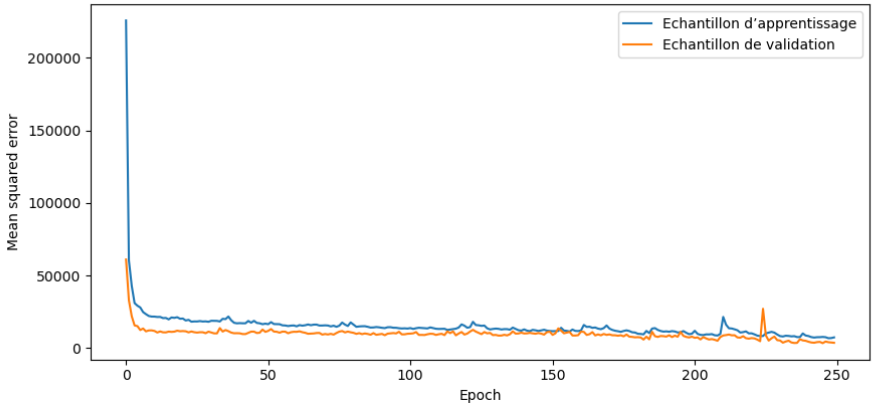

FIGURE 9 – MSE du LSTM sur les données à l'heure en fonction du nombre d'*epochs*

Nous allons à présent regarder les prévisions faites par ces deux modèles sur les échantillons test. D'abord on peut observer sur la figure 10 le nombre d'appels journaliers prévu par le LSTM entraîné sur les données au jour. Et sur la figure 11, il s'agit aussi du nombre d'appels journaliers mais cette fois ci, prévu par le LSTM fonctionnant sur les données horaires. Pour construire le nombre d'appels journaliers avec le LSTM horaire, on effectue 11 prévisions correspondant aux 11 créneaux d'une journée de service chez Groupama, que l'on somme ensuite pour avoir le nombre d'appel sur un jour entier. Cependant ces deux résultats ne sont pas complètement comparables. Ceci est dû au fait qu'avec le LSTM horaire, on se sert de plus d'informations qu'avec le LSTM journalier. En effet, pour établir nos prédictions avec le LSTM horaire on utilise les valeurs observées de la journée que nous sommes en train de prédire. Autrement dit, pour prédire le nombre d'appels au jour d, ce modèle utilise les observations faites à chacun des 10 premiers créneaux de ce jour. Alors

qu'avec le LSTM journalier on prédit le jour d seulement en utilisant les observations des jours précédents. En procédant comme suit, le modèle va s'adapter tout au long de la journée en fonction des valeurs observées. Pour pouvoir réellement comparer ces deux valeurs, il faudrait prédire les 11 heures de la journée d'un coup.

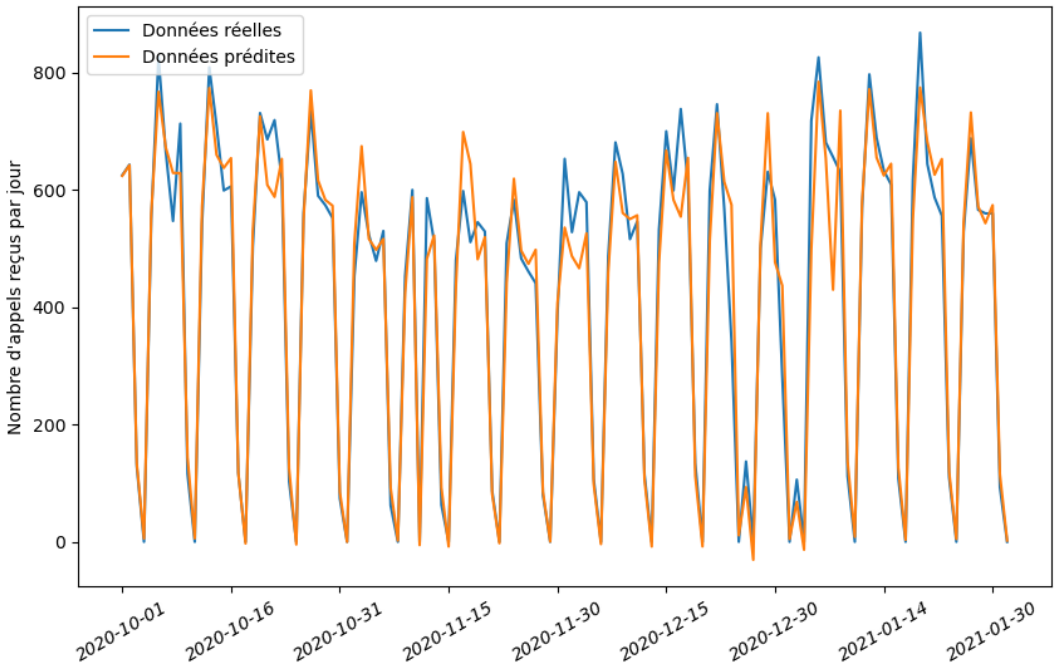

FIGURE 10 – Prévisions du nombre d'appels journalier faite par le LSTM face aux données réelles

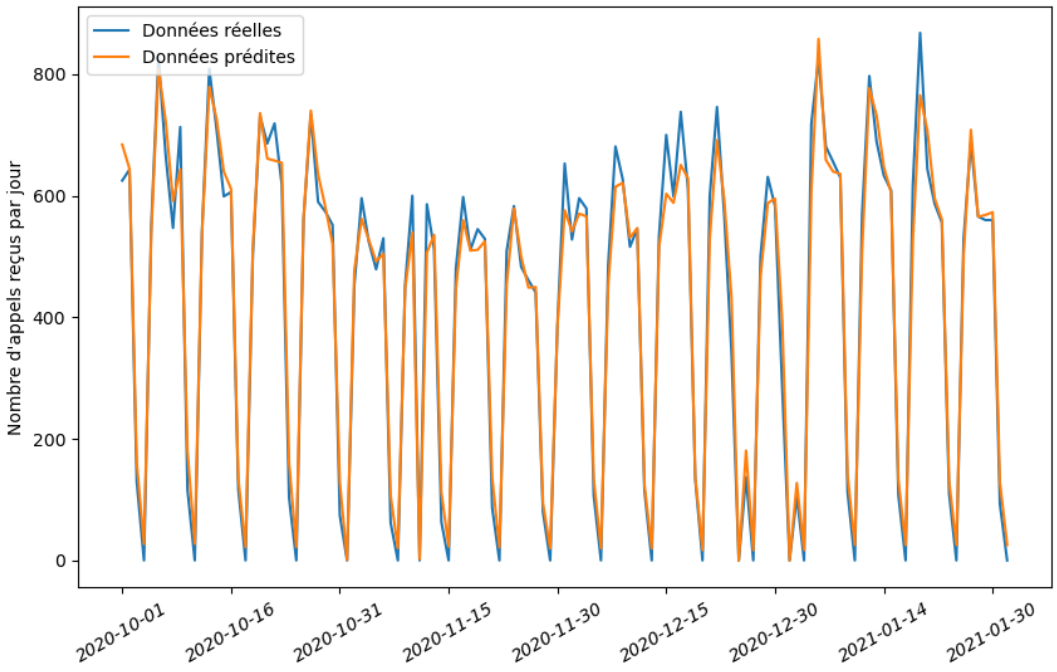

FIGURE 11 – Prévisions du nombre d'appels journalier faite par le LSTM sur les données horaires face aux données réelles

6.2 Exemple d'application : prévision des 14 premiers jours de février

Nous mettons maintenant en application nos modèles dans un cas concret. Nous nous plaçons le dimanche 31 janvier au soir, et nous souhaitons prédire les valeurs des nombres d'appels reçus pour les 2 semaines à venir. Nous pouvons rappeler que sur cette période nous sommes hors période de vacances, il n'y a pas de jours fériés et il y a un couvre-feu, donc la variable `confinement` prend la valeur 0,33. Pour cela, nous utilisons les 5 modèles présentés dans ce rapport. À l'issue de ces 2 semaines, nous avons pu disposer des valeurs observées, et ainsi comparer les valeurs prédites aux valeurs réelles. Nous représentons graphiquement les résultats dans la figure 12 ci-dessous. La courbe pleine rouge correspond aux valeurs observées des nombres d'appels reçus, et les courbes discontinues correspondent aux prévisions obtenues à l'aide de nos 5 modèles. Les valeurs des MAE et RMSE sont données pour chacun des modèles dans la table 6.

	MAE	**RMSE**
MLP	49,63 ≈ 50	67,52 ≈ 68
RNN	80,38 ≈ 80	111,04 ≈ 111
LSTM	85,34 ≈ 85	112,85≈ 113
GRU	73,51 ≈ 74	104,53 ≈ 105
SVR	60,69 ≈ 61	83,85 ≈ 84

TABLE 6 – Scores MAE et RMSE des 5 modèles pour les 2 premières semaines de février

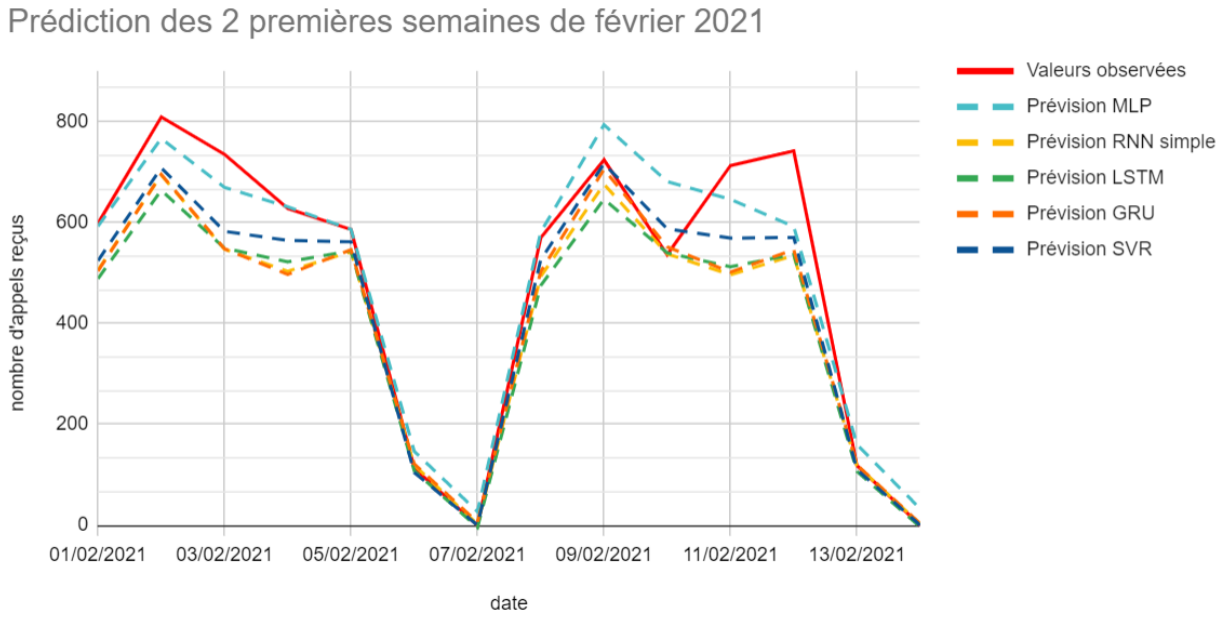

FIGURE 12 – Prévisions obtenues pour les 2 premières semaines de février 2021

Nous observons graphiquement que les résultats obtenus avec le MLP sont assez proches des nombres d'appels observés la première semaine. Nous pouvons remarquer que les modèles RNN, LSTM et GRU prédisent des résultats très semblables sur ces 2 semaines. Le SVR a tendance à prédire légèrement plus d'appels en milieu de semaine. Nous pouvons également constater une tendance inhabituelle sur les données observées la 2ème semaine de février. Il est important de rappeler que nous essayons de prédire le nombre d'appels reçus sur une ligne téléphonique de déclaration de sinistres automobiles pour une compagnie d'assurances, et que même si nous sommes sur des nombres relativement grands, il y a tout de même une part d'aléatoire que nous ne pouvons pas prédire.

Si nous regardons les scores de MAE et de RMSE, nous voyons que dans cette situation le MLP est le modèle qui est le plus proche des données observées, puis suivent le SVR, le GRU, et enfin le RNN et le LSTM qui ont des résultats très proches.

Conclusion

Nous avons réussi à créer différents modèles en utilisant 5 méthodes différentes de machine learning pour essayer de répondre à cette problématique, qui est de prédire le nombre d'appels reçus sur la ligne d'appels téléphoniques de déclaration de sinistres automobiles. Nous pouvons rappeler que ceci sert ensuite à planifier les équipes de *Groupama*, afin d'améliorer la satisfaction client.

Nous avons ainsi créé un modèle permettant de prédire le nombre d'appels du jour suivant et un autre pour l'heure suivante pour chacune des 5 méthodes de machine learning utilisées, qui sont les *Multi-Layer Perceptron* (MLP), *Recurrent Neural Network* (RNN), *Long Short-Term Memory* (LSTM), *Gated Recurrent Units* (GRU) et *Support Vector Regression* (SVR). Nous avons également fait des prévisions pour plusieurs jours suivants, soit en rappelant le modèle au jour suivant plusieurs fois en intégrant la prédiction aux données en entrée à chaque fois (MLP, RNN, LSTM, SVR), soit en créant un nouveau modèle en modifiant la dimension de la couche de sortie (GRU).

Nous avons défini les valeurs des paramètres pour chacun des modèles expérimentalement à l'aide des données de la base de test, et nous avons comparés leurs performances. Nous avons pu voir que sur la prédiction au jour suivant, le RNN, le LSTM et le GRU sont les meilleurs, et présentent des résultats similaires.

Pour la prédiction à l'heure suivante, le GRU a de meilleurs résultats, puis suivent le LSTM, le MLP et le SVR qui ont des résultats équivalents. La prédiction au jour en sommant les heures est la plus proche des données observées pour le MLP, nous pouvons rappeler que cette valeur prédite est plus juste que celle en utilisant le 1er modèle permettant de prédire la valeur du jour suivant, mais ceci s'explique par le fait que nous prédisons ici les valeurs pour chaque créneau horaire au fur et à mesure de la journée, en utilisant les données observées. Ainsi le modèle va s'adapter au cours de la journée, et la valeur de la somme obtenue à la fin de la journée n'est pas comparable à la valeur obtenue par la prédiction au jour suivant. Pour la prédiction sur plusieurs jours suivants, le RNN et le LSTM sont les meilleurs modèles, qui présentent des résultats très proches. Cela nous indique qu'ils sont assez stables lorsque nous les appelons plusieurs fois en utilisant les valeurs prédites en entrée du modèle. Nous remarquons que le GRU, pour lequel nous avons créé un nouveau modèle avec une nouvelle dimension de sortie, nous permettant ainsi d'obtenir d'un seul coup les valeurs prédites des 14 prochains jours, est le moins bon.

Nous avons ensuite appliqué ces modèles à un cas concret en février, nous avons cherché à prédire les nombres d'appels pour les 14 premiers jours de février, en se plaçant le 31 janvier 2021 et en n'ayant aucune information sur les valeurs réelles de février. Sur cet exemple précis nos prévisions ne sont pas toujours très proches des données observées, mais nous pouvons rappeler qu'une part d'aléatoire est présente dans ces nombres d'appels. Certes les nombres observés sont relativement grands, de l'ordre de plusieurs centaines d'appels par jours, mais il est possible d'observer des nombres inattendus sans pouvoir les expliquer.

Un travail scientifique ne pouvant prétendre à la perfection, quelques points d'amélioration sont à noter au niveau de la présente étude. Pour cela, il serait intéressant de tester d'autres valeurs pour les hyperparamètres des modèles, d'autres architectures des modèles mis en oeuvre (ajout de couches cachées supplémentaires). On peut également entraîner les autres modèles (MLP, RNN et LSTM) pour la prévision d'un seul coup de plusieurs jours afin de voir leurs performances et d'en déduire le plus performant pour cette tâche.

Références

[1] Nari Sivanandam Arunraj, Diane Ahrens, and Michael Fernandes (2016), « *Application of SARIMAX Model to Forecast Daily Sales in Food Retail Industry* », International Journal of Operations Research and Information Systems, 10.4018/IJORIS.2016040101.

[2] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio (2014), « *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling* », arXiv :1412.3555v1.

[3] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio (2013), « *On the difficulty of training Recurrent Neural Networks* », arXiv :1211.5063v2.

[4] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014), « *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation* », arXiv : 1406.1078.

[5] Y. Yangl, R. Fulil, C. Huiyou and X. Zhijiaol, « *SVR mathematical model and methods for sale prediction* », in Journal of Systems Engineering and Electronics, vol. 18, no. 4, pp. 769-773, Dec. 2007, doi : 10.1016/S1004-4132(08)60018-3.

[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014), « *Dropout : A Simple Way to Prevent Neural Networks from Overfitting* », Journal of Machine Learning Research 15 1929-1958.

Annexes

Présentation des résultats de MLP

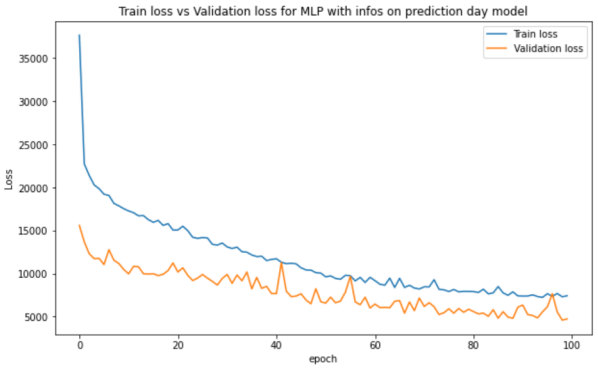

FIGURE 13 – Évolution de la fonction de coût (MSE) sur les échantillons d'apprentissage et de validation au cours des *epochs* du meilleur modèle MLP pour la **prévision au jour**

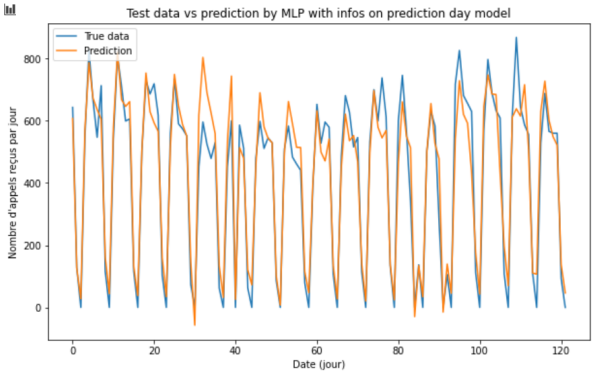

FIGURE 14 – Prévisions obtenues de ce modèle sur l'échantillon test

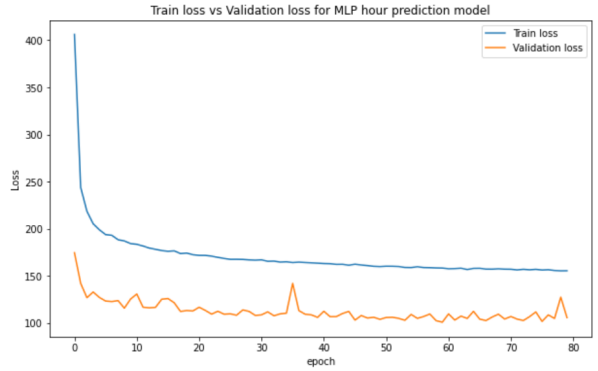

FIGURE 15 – Évolution de la fonction de coût (MSE) sur les échantillons d'apprentissage et de validation au cours des *epochs* du meilleur modèle MLP pour la **prévision à l'heure**

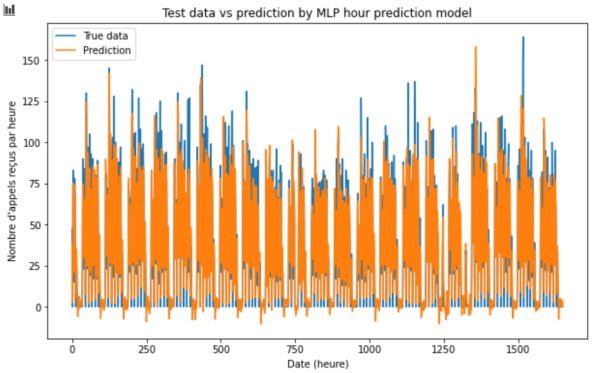

FIGURE 16 – Prévisions obtenues de ce modèle sur l'échantillon test

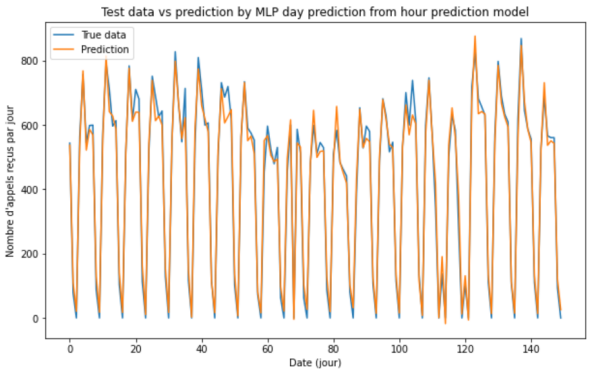

FIGURE 17 – Prévisions au jour obtenues en sommant les prévisions à l'heure

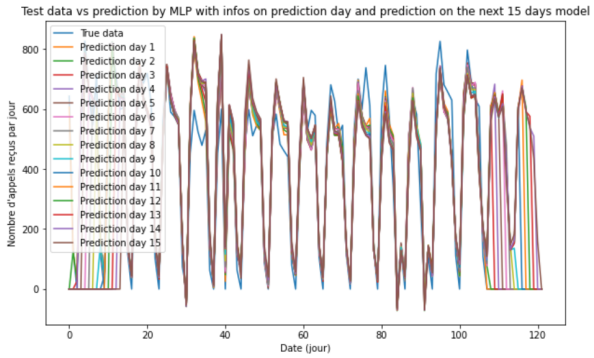

FIGURE 18 – Prévisions au jour obtenues pour 15 jours de suite

Présentation des résultats de RNN simple

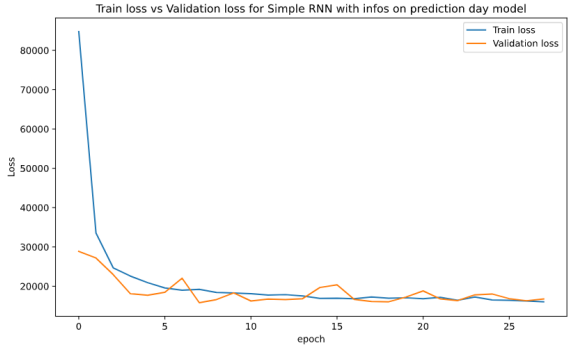

FIGURE 19 – Évolution de la fonction de coût (MSE) sur les échantillons d'apprentissage et de validation au cours des *epochs* du meilleur modèle RNN pour la **prévision au jour**

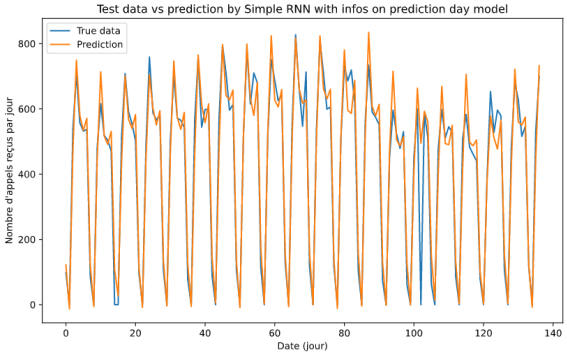

FIGURE 20 – Prévisions obtenues de ce modèle sur l'échantillon test

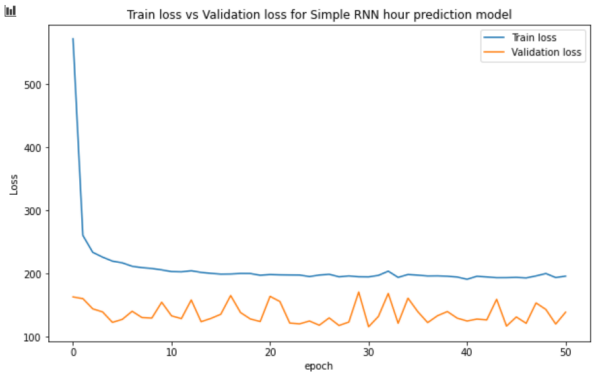

FIGURE 21 – Évolution de la fonction de coût (MSE) sur les échantillons d'apprentissage et de validation au cours des *epochs* du meilleur modèle RNN pour la **prévision à l'heure**

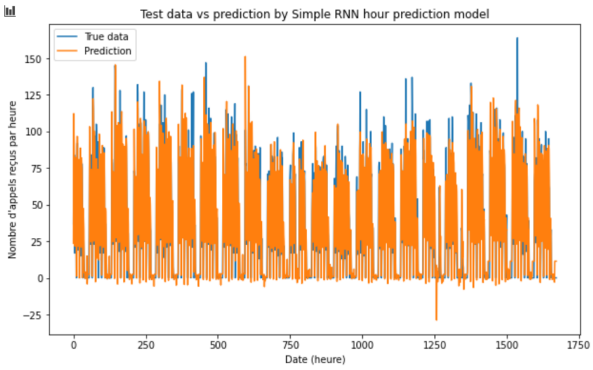

FIGURE 22 – Prévisions obtenues de ce modèle sur l'échantillon test

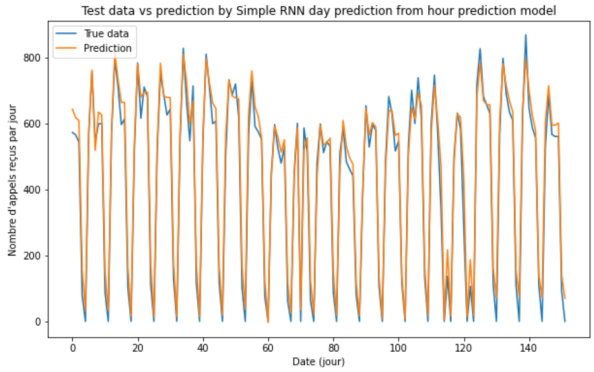

FIGURE 23 – Prévisions au jour obtenues en sommant les prévisions à l'heure

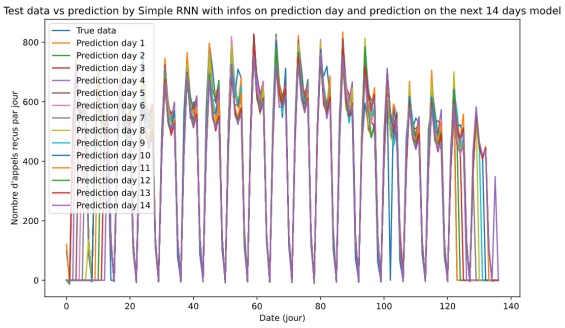

FIGURE 24 – Prévisions au jour obtenues pour 14 jours de suite

Présentation des résultats de GRU

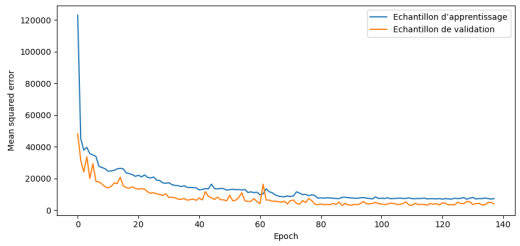

FIGURE 25 – Évolution de la fonction de coût (MSE) sur les échantillons d'apprentissage et de validation au cours des *epochs* du meilleur modèle GRU pour la **prévision au jour**

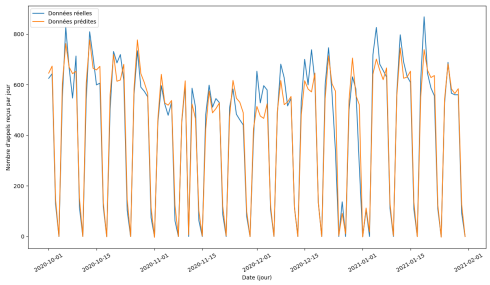

FIGURE 26 – Prévisions obtenues de ce modèle sur l'échantillon test

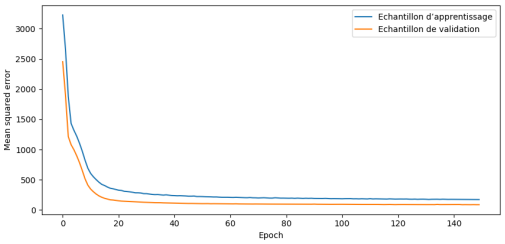

FIGURE 27 – Évolution de la fonction de coût (MSE) sur les échantillons d'apprentissage et de validation au cours des *epochs* du meilleur modèle GRU pour la **prévision à l'heure**

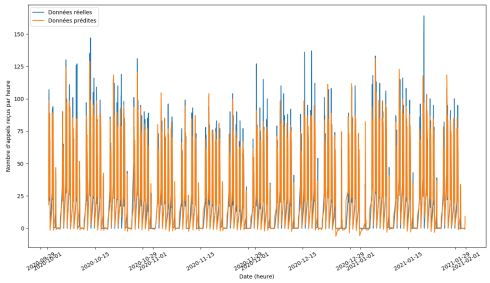

FIGURE 28 – Prévisions obtenues de ce modèle sur l'échantillon test

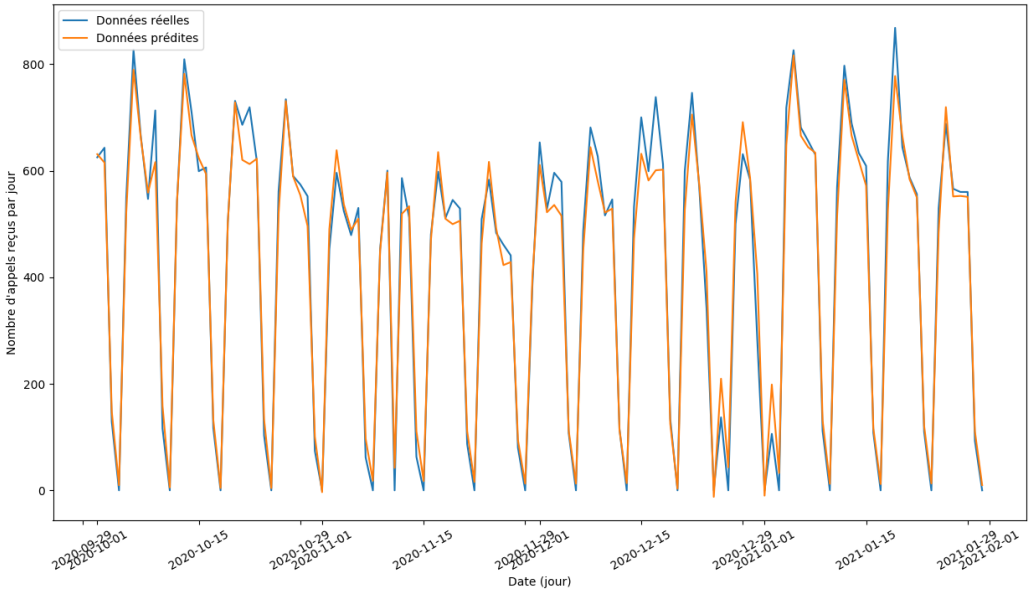

FIGURE 29 – Prévisions au jour obtenues en sommant celles à l'heure

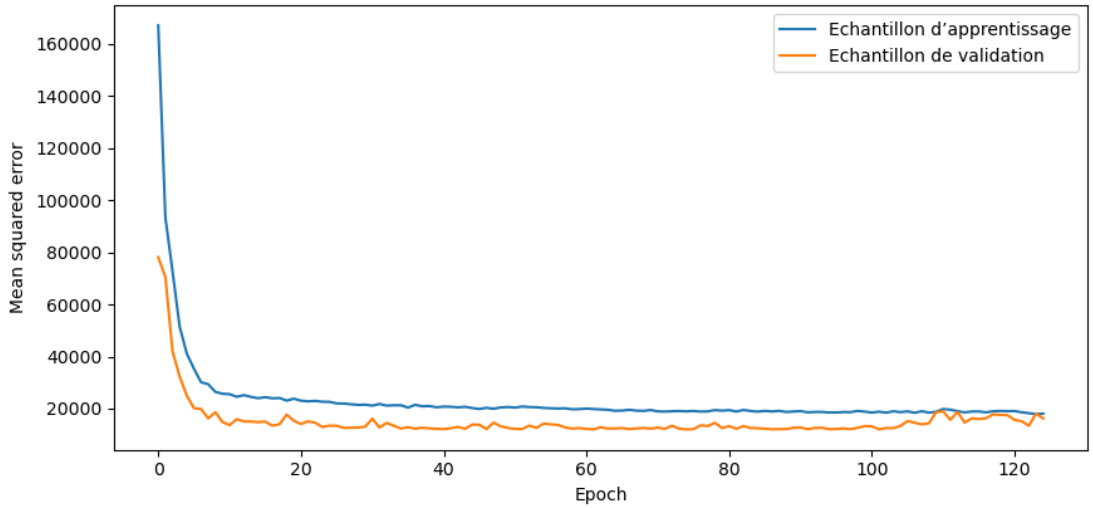

FIGURE 30 – Évolution de la fonction de coût (MSE) sur les échantillons d'apprentissage et de validation au cours des *epochs* du meilleur modèle GRU pour la **prévision au jour des 15 jours suivants**

Présentation des résultats de SVR

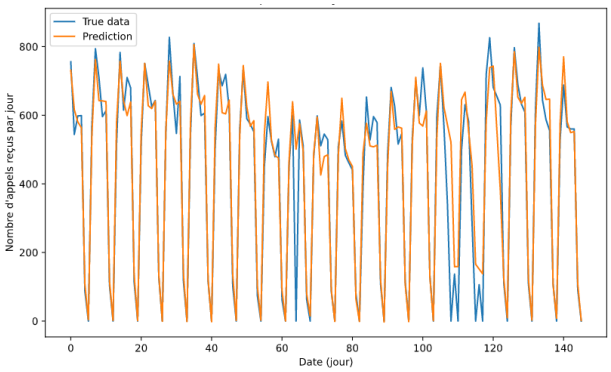

FIGURE 31 – Prévisions obtenues **au jour** du meilleur modèle SVR sur l'échantillon test

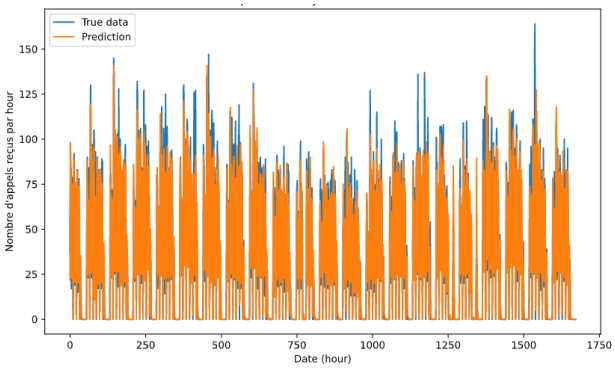

FIGURE 32 – Prévisions obtenues **à l'heure** du meilleur modèle SVR sur l'échantillon test

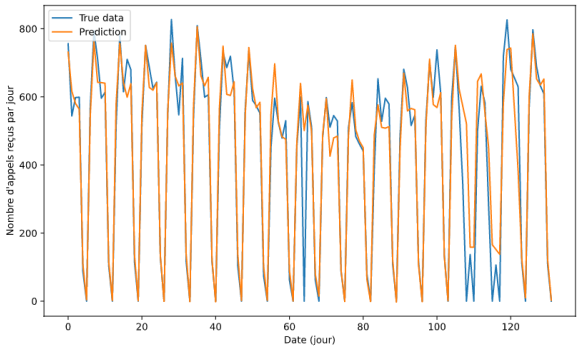

FIGURE 33 – Prévisions obtenues du meilleur modèle SVR sur la prévision **de 14 jours d'affilée** de l'échantillon test pour le 1^{er} jour à prédire

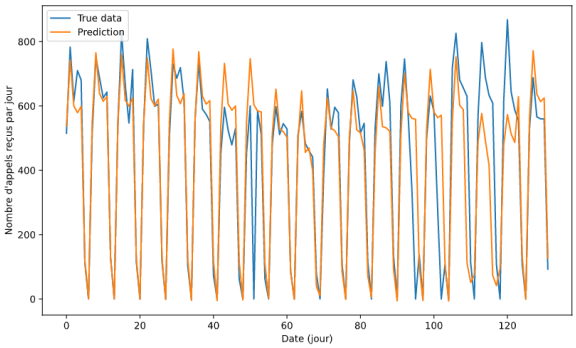

FIGURE 34 – Prévisions obtenues du meilleur modèle SVR sur la prévision **de 14 jours d'affilée** de l'échantillon test pour le **14ème** jour à prédire

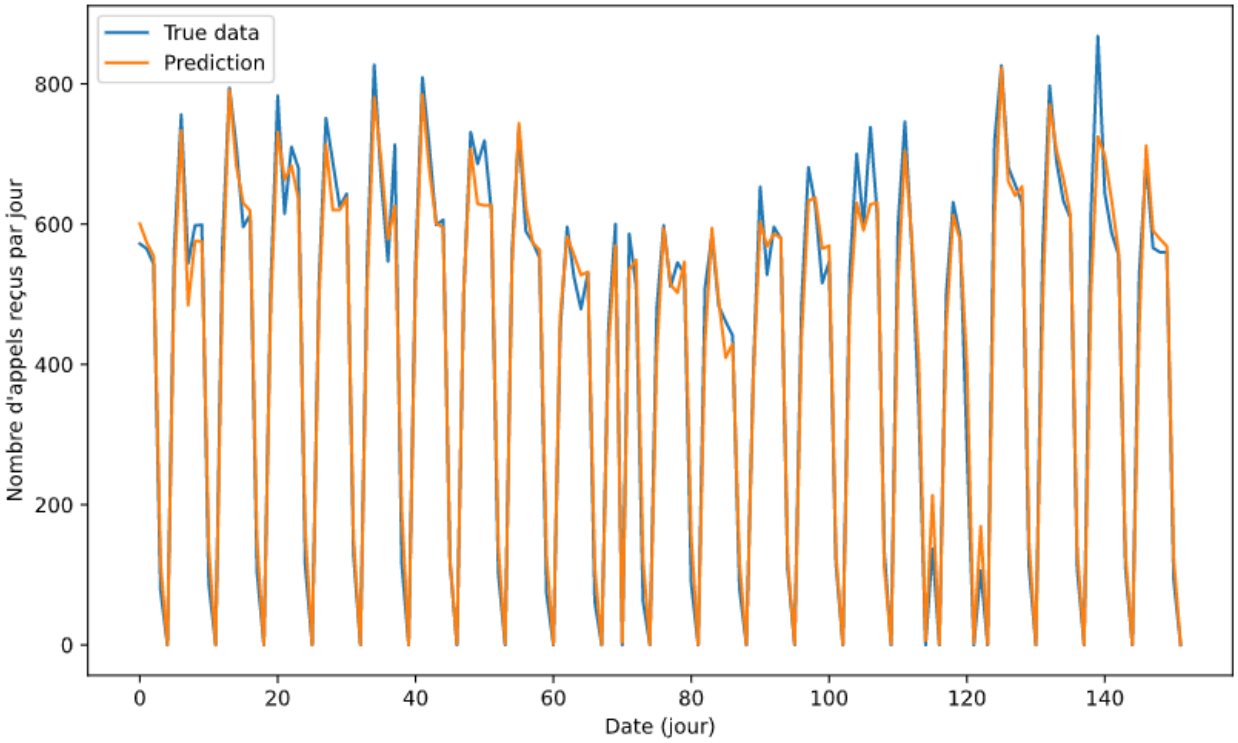

FIGURE 35 – Prévisions obtenues **au jour** en **sommant celles à l'heure**, sur l'échantillon test