

Rapport Final de Stage - (2021-2022)

MODELISATION DE LA DEMANDE DE PRODUIT DANS LES MAGASINS BOXY : ML-Series Chronologiques

7 juin 2022 – 09 septembre 2022



Élève Ingénieur Polytechnicien :

Emmanuel GNABEYEU MBIADA
X2020, Stg Data Science

Tuteur :

Pierre Hulot
X2013, Senior DataScientist Boxy



REMERCIEMENTS

Je tiens à remercier tout particulièrement mon Tuteur de stage, Pierre Hulot, et mon Manager, Yann Merchier, pour m'avoir guidé et soutenu tout au long de mon stage, et pour m'avoir aidé à effectuer rigoureusement mon travail.

Je remercie également la cellule " data " de Boxy, pour m'avoir fourni rapidement les données et pour avoir suivi le sujet avec intérêt.

Je remercie également mon référent de stage en la personne de Marie Claire Wastieux pour le suivi du stage.

Je remercie également Boxy et tous ses employés qui m'ont accompagné tout au long de ce travail et m'ont aidé à résoudre mes problèmes, leur suggestion et critiques m'ont fait prosperer.

Enfin, je remercie l'administration de l'Ecole Polytechnique et mes enseignants pour leur incitation au travail méthodique et rigoureux dans une perspective aux frontières de la connaissance des grands enjeux pluridisciplinaire dans l'optique de relèver les grands défis scientifiques et technologiques.

RESUMÉ OPÉRATIONNEL

Boxy est une startup tricolore spécialisé dans le retail qui reinvente le commerce de proximité avec un concept de supérettes connectées et ouvertes. Elle offre des services d'hyper-proximité de commerce locale en France en alimentant depuis son entrepot environ 35 magasins autonomes de produits aux catégories diverses. L'ambition est de prévoir pour chaque produit, la quantité optimale à utiliser pour l'alimentation de chaque magasin afin d'éviter des ruptures de stocks et des pertes d'une part et d'autres part oeuvrer au service à la clientèle et à la fidelisation des clients.

Des modèles ont été mis en place pour estimer la demande des clients en utilisant l'historique des ventes(resp. des demandes antérieures) à l'instar d'une moyenne en fenêtre roulante d'horizon 3 jours. Avec une colette de données de ventes assez importantes, il est judicieux d'en ressortir des comportements généraux et intemporels à l'instar des tendances, des saisonnalités. C'est dans ce cadre que s'inscrit sous l'hypothèse de performance au test, notre travail consistant à mettre en place un modèle de prédiction de la demande des produits dans les magasins boxy.

La Méthode que nous avons définie se contente de prédire l'espérance de la demande de chaque produit par magasin en utilisant l'historique des ventes. Notre Méthode présente une architecture en deux étapes : **Le processus d'entraînement** qui d'abord réduit le problème à un problème plus simple, minimisant la perte d'information, puis un algorithme prédictif est appris sur le problème réduit. **Le processus de prédiction** de la demande (utilisation du modèle) qui inverse ce mécanisme en quittant de la prédiction de algorithme prédictif dans l'espace réduite pour reconstruire la solution générale dans l'espace réelle.

L'entraînement se fait sur les produits les plus populaires (les actifs qui à priori font le plus grand historique de ventes) et sur les nouveaux(présents en magasin depuis moins de 14 jours) car peut-être ya t'il des choses intéressantes à apprendre sur ces produits récents ou moins populaires. Cependant les composantes apprise sur ces nouveaux sont mises à jour grâce à l'expérience acquise par les produits avec un long historique via un lissage exponentielle.

Plusieurs méthodes de simplification(PCA,AutoEncoder,UMAP,TCA) et de prédiction(Régressions ,Bagging,Boosting,Réseaux de Neurones) sont testées et comparées en termes de précision(MSE, MAE,MSLE) et de temps de calcul(cas de la décomposition tensorielle sur plusieurs Librairies).

Les résultats montrent que le meilleur algorithme est de la famille des modèles linéaires(Regression linéaire et svr à noyau). En ce qui concerne le nombre de magasins qui évolue avec le temps, notre méthode propose l'utilisation de la demande des magasins voisins(du même cluster par exemple) pour exprimer l'espérance de la demande de chacun des produits des nouveaux magasins comme une fonction linéaire des comportements principaux extraits de ces autres Magasins.

Les conclusions de ce travail sont validées sur les ventes réellement observées. Enfin, une infrastructure pour ce modèle (Interface Python) a été mise en place et déployer pour servir les applications ayant besoin des prédictions de demandes des produits.

ABSTRACT

Avec une historique des ventes des produits dans les Magasins Boxy, dans l'optique d'apprendre des comportements généraux et intemporels à l'instar des tendances et des saisonnalités, nous avons mis en place un modèle de prédiction de la demande espérée des produits dans ces magasins.

Notre Méthode présente une architecture en deux étapes : **Le processus d'entraînement** qui d'abord réduit le problème à un problème plus simple, minimisant la perte d'information, puis un algorithme prédictif est appris sur le problème réduit. **Le processus de prédiction** de la demande (utilisation du modèle) qui inverse ce mécanisme en quittant de la prédiction de algorithme prédictif dans l'espace réduite pour reconstruire la solution générale dans l'espace réelle.

Plusieurs méthodes de simplification et de prédiction sont testées et comparées en termes de performances(minimisation de l'erreur sur la prédiction et l'observation) et de temps de reconstruction de l'espace réelle.

Les résultats montrent que le meilleur algorithme est de la famille des modèles linéaires(Regression linéaire et svr à noyau) et l'approche du modèle implique d'exprimer l'espérance de la demande de chacun des produits des nouveaux magasins(évolutifs)comme une fonction linéaire des comportements principaux extraits des autres Magasins.

Les conclusions de ce travail sont validées sur les ventes réellement observées. Enfin, une infrastructure pour ce modèle (Interface Python) a été mise en place et déployer pour servir les applications ayant besoin des prédictions de demandes des produits.

Table des matières

1	Introduction	
1.1	Définition des enjeux	1
1.2	Objectif de la prédition	2
2	Typologie des méthodes de prédition	
2.1	Cadre Mathématique	2
2.2	Typologie des méthodes ou approches.....	3
3	Selection et Métriques	
3.1	Selection des variables ou caractères.....	4
3.2	Métriques d'évaluation	4
4	Methodes de Reduction	
4.1	Identité	5
4.1.1	<i>Réduction de dimensions :Réduction du nombre de variables de sortie</i>	5
4.1.2	<i>Autoencodeurs</i>	6
4.1.3	<i>Uniform Manifold Approximation and Projection for Dimension Reduction (Umap)</i> 6	
4.1.4	<i>TCA ou Tensor Decomposition or Factorization</i>	7
5	Prétraitemet et Représentation de l'information	
5.1	Données d'entrée (Input Data)	8
5.2	Matrix Representation	9
5.3	Tensor Representation pour TCA	9
6	Méthodes d'entraînement	
6.1	Transformation en données d'apprentissage supervisé	11
6.2	Les Modèles d'apprentissage Automatique mis en place	11
6.3	Les Modèles d'apprentissage Profond ou de type ANN.....	20
7	Résultats et illustrations	
7.1	Modèle temporel produit par Magasin	23
7.2	Modèle atemporel aggrégé	25
7.2.1	<i>Stratégie Tactico-technique</i>	25

7.2.2	<i>Analyse en composante Principale(PCA)</i>	26
7.2.3	<i>Autoencodeur</i> :	31
7.2.4	<i>UMAP</i> :	31
7.2.5	<i>Tensor decomposition (TCA)</i> :	32
7.3	feature engineering et Ajout de Covariables	33
7.3.1	<i>Historique et données calendaires</i>	34
7.3.2	<i>Historique et données Metéo</i>	35
7.3.3	<i>Historique, données calendaires et données Météorologiques</i>	36
7.3.4	<i>Conclusion de l'étude de l'intégration des données calendriers et météorologiques.</i>	37
8	Interface Prediction	
8.1	Sélection du Modèle.....	37
8.2	Auto-correction en temps réel	37
8.2.1	<i>Cas des Nouveaux Produits par Magasin</i>	38
8.2.2	<i>Cas des Produits en "to come" et en "projects"</i>	38
8.3	Ajout de Nouveaux Magasins :.....	39
8.3.1	<i>Comparatif des performances de algorithmes</i>	39
9	Déroulement du Stage	
9.1	L'entité d'accueil	41
9.1.1	<i>Secteur d'activité</i>	41
9.1.2	<i>Situation économique, compétitive et stratégie</i>	41
9.1.3	<i>Organisation, fonctionnement et questions éthiques</i>	41
9.2	La mission	41
9.2.1	<i>Rôle</i>	41
9.2.2	<i>Réalisations et intégration dans l'organisation</i>	42
9.3	Les enseignements	42
9.3.1	<i>Analyse de l'adéquation entre les constats réalisés sur l'organisation et son secteur, et les aspirations personnelles/professionnelles.</i>	42
9.3.2	<i>Enseignements personnels (softskills) : qualités et axes de progrès</i>	42
9.3.3	<i>Impact sur les projets scolaires et professionnels.</i>	42

10 Conclusion et Perspectives

11 Références et Bibliographie

Annexes

A Code

B Nouvelles Métriques après Filtration des données

B.1	Modèle sans covariables	44
B.2	Historique et données calendaires	45
B.3	Historique et données météorologiques	46
B.4	Historique, données calendaires et données météo	46
B.5	Tensor decomposition (TCA) :.....	47

1 Introduction

La gestion des stocks et l'approvisionnement des magasins est souvent sensible notamment en cas de rupture de produit voire de cannibalisation ; Car alors,d'une part le client risque de se tourner vers la concurrence pour ledit produit, mais au pire pour le chariot tout entier et de l'autre coté,entrainer l'augmentation des ventes d'un autre produit(un substitut) au détriment de la diminution des ventes du précédent. Une maîtrise incertaine des stocks peut donc s'avérer particulièrement coûteuse pour l'entreprise : qu'il s'agisse de ne plus disposer du produit demandé (et donc de perdre le client) ou de trop stocker en ayant « peur de manquer » (et donc de dégager des surcoûts de stockage). Il convient donc de **mieux optimiser ses stocks**.

La solution réside dans **l'anticipation du besoin client, et par conséquent dans l'anticipation de la demande.**

La modélisation prédictive que nous nous proposons d'orchestrer dans une approche bottom-up(prévision de la demande par produit par magasin pour déduire la demande totale par produit) permet d'intégrer une multitude de paramètres essentiels à une gestion optimisée, afin d'anticiper les tendances de consommation et de fournir des prévisions de demande : Nous prendrons en compte les données temporelles mais aussi des contraintes locales relatives à chaque magasin.

- effets de saisonnalité :
- modes de consommation spécifiques
- événements culturels locaux
- événements climatiques

De la sorte, notre modèle integrera des données temporelles récurrentes et prendra en compte des contraintes locales afin de faire émerger des comportements de consommation pour ainsi dire « géolocalisés ». Notre algorithme devra offrir à la direction Marketing une vision juste et précise des comportements d'achat à venir. Et lui permet alors d'ajuster, à la hausse ou à la baisse, ses commandes en fonction de cette demande prévisionnelle.

Une ouverture vers une auto-correction en temps réel : C'est-à-dire que le modèle pourra réévaluer automatiquement ses projections/prédictions/prévisions après les avoir comparer à la réalité pour intégrer les écarts éventuellement constatés. Nous ferons juste une mise à jour de la matrice des composantes contenant l'essentiel de l'information réduites sur l'historique des produits par magasin(Ancien, nouveau,à venir)

Les différentes méthodes de prévision

Parmi les méthodes les plus connues nécessitant de disposer d'un certain nombre d'information, on peut citer :

- Les méthodes subjectives : elles se basent sur les opinions des personnes.
- Les méthodes analogiques : elles s'appuient sur l'analyse d'un marché similaire au marché de l'entreprise étudiée.
- Les méthodes des tests : elles se fondent sur l'estimation de la demande à partir de sondages.
- Les méthodes des marchés témoins : les entreprises sélectionnent des petites zones géographiques qui présentent les mêmes caractéristiques que leur marché et y appliquent leur politique commerciale.
- **Les méthodes statistiques** : elles extrapolent dans l'avenir les tendances observées dans le passé. C'est cette famille de méthode qui constituera l'ossature de notre travail inspiré de l'article de recherche [4] **Towards Station-Level Demand Prediction for Effective Rebalancing in Bike-Sharing Systems.**

1.1 Définition des enjeux

La prévision de la demande consiste à utiliser les données historiques des ventes afin de prévoir la demande future des clients. De façon générale les prévisions permettent d'anticiper de potentiels

problèmes en offrant un vaste éventail d'avantages allant de l'amélioration de l'efficacité de la chaîne d'approvisionnement à l'optimisation de la planification financière.

- Au niveau Logistiques : Les logisticiens doivent prendre des marges de sécurité afin d'éviter les ruptures de stock face aux variations de la demande d'une part et les pertes d'autre part.(gestion du stock, transport,...)
- Au niveau Marketing : (Négociation fournisseur, volume de commande,...) La prédition de la demande peut être utile quant à la négociation fournisseur, les campagnes promotionnelles, À court terme, elle permet de planifier sa production/les livraisons(cas de Boxy) et donc, d'optimiser la gestion des stocks, À plus long terme, il permet d'estimer le chiffre d'affaires annuel et d'élaborer le budget pour l'année à venir. Ainsi, elle fournit une mine d'informations qui peuvent être utilisées pour augmenter les marges bénéficiaires et le succès global. Ainsi, de la **prise de décision à l'évolution de la stratégie d'entreprise**, une prévision précise de la demande est un outil fiable et précieux
- Au niveau éthique : (Disponibilité et satisfaction client) -On note des effets importants à court et à long terme sur la longévité et le succès de l'entreprise au travers de l'évaluations des performances. Si la demande des clients n'est pas estimée avec précision, les processus métier, depuis la gestion des stocks jusqu'à la planification financière, peuvent être gravement affectés. Une prévision précise de la demande peut fournir un avantage concurrentiel et un succès financier notable. Elle peut être utilisée à l'avantage de l'entreprise pour influencer positivement et pratiquement tous ses aspects. -Le maintien de niveaux de stock adéquats est essentiel au service à la clientèle et à la fidélisation des clients qui peuvent rapidement devenir mécontents lorsque le produit qu'ils ont visité un magasin pour acheter n'est pas disponible.
- Projections commerciales : (estimation du CA annuel, élaboration du Budget, planification financière)

Nous notons qu'il ne s'agit pas de prédire les ventes qui peuvent être impactées ou non par l'état du stock(notamment en cas de rupture) n'affecte pas la demande qui n'est jamais complètement observée et provient d'un modèle statistique.

1.2 Objectif de la prédition

Le but de ce travail est de faire des prévisions de la quantité de produit à différentes échelles.

- La demande par produit : Il s'agit de la quantité totale de chaque produit.
- La demande par magasin/produit : Il s'agit de la quantité de chaque produit au niveau de chaque magasin.

Pour notre travail, nous adopterons une méthode Bottom-Up, qui consiste à partir de la prédition de la demande de chaque produit au niveau de chaque magasin pour remonter à la demande totale par produit.

2 Cadre Mathématique et Typologie des méthodes de prédition

2.1 Cadre Mathématique

On suppose les produits numérotés de 1 à d, appelé catalogue. Pour $i \in [1, d]$, $(x_{i,t})$ dénote la quantité de produit i à la date t.

on observe les ventes $(x_{i,t}) \in \mathbb{R}$ que l'on suppose connue jusqu'à temps t_0 . Soit h l'horizon de prédition (c'est-à-dire l'algorithme prédira aux temps $t+1$ jusqu'à $t+h$) ou le temps nécessaire au réapprovisionnement : Le but est d'estimer \hat{x}_{i,t_0+h} que nous modélisons comme :

$$\hat{x}_{i,t_0+h} = \underset{x}{\operatorname{argmin}} \mathbb{E}[L(x, \sum_{t=t_0+1}^{t_0+h} x_{i,t}) | \forall j, t < t_0, x_{j,t}] \quad (1)$$

où L est une fonction perte dont nous pouvons matérialiser comme une perte de poisson si l'on suppose que les dates d'arrivée d'un client suivent un processus de poisson dont le paramètre change chaque jour/semaine.

2.2 Typologie des méthodes ou approches

- Approche temporelle : Elle consiste à construire une fonction f_h tel que $\hat{x}_{i,t_0+h} = f_h(x_{i,t}, \theta_{i,t})$ où $\theta_{i,t}$ sont des variables externes concernant un produit i à la date t pouvant être apprise grâce à un algorithme d'apprentissage automatique. Ainsi, soit I l'ensemble des produits divisé en K différents catégories noté I_k avec $I = \cup_k I_k$. Supposons avoir N séries $(y_{i,t})$ où $y_{i,t}$ est le nombre de ventes du produit $i \in I$ au cours de la période(ici le jour) t . On a des observations pendant T jours. Si $Y_{i,T} = (y_{i,0}, \dots, y_{i,T})$ alors $f(Y_{i,T}, \theta)$ est un estimateur de $y_{i,t+h}$ pour un ensemble de paramètre θ pouvant être appris.

Dans cette catégorie, on distingue plusieurs modèles ou algorithmes d'apprentissage :

- Les méthodes auto-regressives qui sont une classe de modèles qui tentent de prédire les valeurs futures d'une variable sur la base de ses observations passées comme la moyenne Mobile(et ses variantes ARMA, ARIMA, SARIMA, SARIMAX) et les Méthodes de lissage exponentiel
- Algorithme de type boosting comme XgBoost
- Algorithme d'apprentissage profond comme les réseaux de neurones récurrents à l'instar du LSTM.

Le problème que soulève une telle approche est l'horizon de prédition h . En effet, la question est celle de savoir si une telle approche peut faire des prévisions pertinentes à n'importe quelle période futurs, la réponse est la négative en générale.

- Approche atemporelle : Cette approche va permettre de fixer la difficulté de la précédence en ce qui concerne l'horizon de prédition en permettant de faire des prévisions de la demande future à tout temps qu'importe la période. Il s'agit d'un point de vue généralisé. Les modèles ou algorithmes d'apprentissages sont des regressions ou des Machines à support de vecteurs, mais aussi des modèles à arbres comme type Bagging comme Random Forest et type Boosting comme XgBoost.
- Approche Héritique :
- Approche Bayésienne : Dans cette catégorie, on prédit la demande comme une distribution de valeurs plausibles, ce qui peut aider à la création de stock de sécurité dans la mesure où on a une idée de la dispersion(DeepAR d'Amazon s'inspire de cela).
- Approche Héritique Bayésien :

Nous nous concentrerons dans la première partie aux méthodes temporelles et atemporelles.

3 Selection des variables et métriques d'évaluation

3.1 Selection des variables ou caractères

Il s'agit des différentes caractéristiques qui décrivent chaque instance d'apprentissage. On distingera :

- Des variables temporelles, qui sont des covariables dépendant du temps t uniquement matérialisant la saisonnalité, à savoir entre autre la saison, jour de la semaine, jour dans le mois (impact de la paye qui tombe en fin de mois), vacances scolaires, soldes, événements, les événements spéciaux(Noël,Black Friday, Vacances, Jour férié...) etc.
- Les Variables catégorielles qui sont des covariables dépendant du produit i , des magasins, et toute autre variable constante. On peut dénombrer à titre d'illustration le type de produit, la marque(importante lorsqu'il ya des substituants), la famille ou catégorie, game, prix(prix de vente, réductions, évolution historique du prix, mécanisme promotionnel, etc.),le packaging, avis client etc.
- Des variables mixtes ou combinaison des deux précédentes :Force de vente (rémunération, qualification, nombre, etc.) Point de vente (localisation, taille, assortiment, événements locaux, etc.) etc.
- Des facteurs extérieurs qui influencent la valeur future de la variable étudiée : Concurrence (nature, densité, chevauchement des offres, etc.) ; Canal (physique, web, drive, livraison, relais colis) ; Promotions (budget pub, merchandising, réseaux sociaux, catalogues, etc.) ; Météo(tempete,Beau temps,orage,canicule) ; Indicateurs macroéconomiques (taux de change, salaire moyen, taux d'inflation, cours de bourse, etc.)

3.2 Métriques d'évaluation

Elle servent de grandeurs principales pour la comparaison de la performance des modèles et le/la choix/selection du modèle optimale. Nous utiliserons principalement :

- le MAE ou erreur absolue moyenne, qui pénalise les erreurs de façon uniforme.
- le MSE ou erreur quadratique Moyenne, qui pénalise les fortes erreurs de façon importante.

$$MSE = \frac{1}{n} \sum_{i=1}^n (x_i - y_i)^2 \quad (2)$$

- le MSLE ou erreur quadratique logarithmique moyenne, qui prend en compte l'écart entre les valeurs.
- le Score R^2 Le coefficient R^2 ou coefficient de détermination est une mesure de la déviation de la moyenne expliquée par le modèle.

4 Methodes de Reduction

Le problème original de prédiction consiste à prédire 260*30 valeurs par pas de temps sur l'horizon.

Cette partie vise à réduire ce nombre à un nombre plus petit. Le grand nombre d'objectifs pose plusieurs problèmes. Tout d'abord, il nécessite une grande capacité de calcul (temps et puissance) et de mémoire (RAM et ROM). Réduire le temps de calcul est utile pour construire des modèles plus complexes,mais un compromis doit être trouvé entre la précision (conserver la plupart des informations) et la complexité du problème (réduire le nombre de colonnes).

4.1 Identité

Cette méthode est une base de référence, aucune réduction n'est effectuée et aucune perte d'information, les données sont conservées telles quelles, mais conduit à un modèle de prédiction plus complexe en raison du nombre de dimensions.

4.1.1 Réduction de dimensions :Réduction du nombre de variables de sortie

Intéressons nous aux variables de base X_1, \dots, X_p . Les techniques de réduction de dimensions consistent à transformer les variables en utilisant des combinaisons linéaires des X_i (Familles de Méthodes Linéaires). Par exemple, pour se ramener à $m < p$ variables, on va construire Z_1, \dots, Z_m des combinaisons linéaires des variables, i.e. $Z_k = \sum_{i \in N} a_{ik} X_i$

Il faut donc bien choisir les coefficients. Une méthode utilisée est la Principal Component Analysis (PCA). Elle consiste à définir les Z_k comme les directions où les données varient le plus.

Pour définir les a_{ij} , on commence par choisir les a_{i1} afin que la variance de Z_1 soit maximale. Ensuite, on choisit les a_{i2} afin que la variance de Z_2 soit maximale et décorrélée de Z_1 , et ainsi de suite jusqu'à obtenir m prédicteurs indépendants les uns des autres.

la méthode PCA est une décomposition en valeurs singulières (SVD) qui centre et normalise les données avant de calculer la transformation. La décomposition en valeurs singulières (SVD) étant une technique de réduction très courante. Elle utilise le théorème suivant pour réduire la dimensionnalité d'une matrice :

Theorem 1. Étant donné une Matrice D de taille $n \times m$ dans $\mathbb{R}^{n \times m}$ de rang r alors elle peut être factorisée comme suit : $D = U \times \Sigma \times V^T$ Où U et V sont des matrices orthogonales unitaires de forme $n \times r$ et $r \times m$ et Σ est la matrice diagonale $r \times r$ des valeurs singulières, ordonnées de la plus grande à la plus petite.

La réduction SVD consiste à sélectionner les dimensions les plus significatives (en termes de valeurs singulières), et à mettre à zéro toutes les autres. Notons $V' = [V_1, \dots, V_k]$, les k premières colonnes de V , pour une valeur prédefinie de k . Ensuite, la méthode de réduction calcule avec la transformation inverse :

$$\text{red}(D) = D \times V' \text{ et } \text{inv_red}(D') = D' \times V'^T \quad (3)$$

L'orthogonalité de V justifie cette pseudo-inverse.

Nous savons tous que la réduction de la dimensionnalité du problème peut entraîner une diminution de la variance d'un modèle statistique, ce qui peut compenser l'augmentation du biais (compromis biais-variance). Cependant, l'un des inconvénients de la réduction de la dimensionnalité est que nous risquons de perdre des informations importantes, puisque la variance de la covariable semble être une bonne mesure de l'information. Plus la variance est grande, plus la quantité d'informations que contient la variable est importante.

L'information totale que contient les composantes principales est la même que pour les composantes originelles(En d'autres termes, montrons qu'en appliquant l'ACP, nous n'avons pas perdu d'information (nous l'avons simplement répartie différemment)) Soit X la matrice prédicteur d'origine et Y la matrice prédicteur transformée, toutes deux de dimensions $n \times p$. Afin de transformer X , nous effectuons un changement de variable, ce qui implique de multiplier X par une autre matrice inconnue P de dimensions $p \times p$ afin d'obtenir Y .

La matrice de Covariance de X notons S est symétrique dont diagonalisable comme suit $S = PDP^{-1}$ avec P orthogonale(ce qui permet de maintenir inchangée la variance totale des données car la mul-

tiplication par une matrice orthogonale ne modifie pas les longueurs des vecteurs ni leurs angles). Et on a donc $P^T S P = D$. De plus

$$\text{Cov}(Y) = \text{Cov}(X \times P) \text{ Changement de variable} \quad (4)$$

$$= P^T \times \text{Cov}(X) \times P \text{ Propriété de la Covariance} \quad (5)$$

$$= P^T \times S \times P \quad (6)$$

Donc $D = \text{Cov}(Y)$ et donc $\text{Cov}(Y)$ est diagonal, donc les composantes sont incorrélatées.

4.1.2 Autoencodeurs

Les autoencodeurs[2] sont des réseaux de neurones artificiels qui tentent d'apprendre une représentation d'un ensemble de données, généralement pour réduire la dimensionnalité. Il s'agit d'un algorithme d'apprentissage non supervisé (Hinton et Salakhutdinov, 2006). Le réseau neuronal de l'autoencodeur tente de prédire x' étant donné x en transmettant les informations à un espace de dimension inférieure. Il apprend donc une représentation de x dans un espace plus petit.

Cette couche plus petite coupe le réseau en deux parties, la partie encodage ou encodeur F de l'espace original vers un espace latent et la partie décodage décodeur G pour revenir à l'espace d'origine.

Si l'espace latent est de dimension inférieure, l'auto-encodeur doit capturer une "bonne" représentation des données.

$$\text{red}(D) = f_n(f_{n1}(\dots f_1(DW_1 + b_1)\dots)W_n + b_n) \text{ et } \text{inv_red}(D) = f_{2n}(f_{2n1}(\dots f_{n+1}(DW_{n+1} + b_{n+1})\dots)W_{2n} + b_{2n}) \quad (7)$$

Où les f_i sont des fonctions d'activation, les W_i des matrices de poids et les b_i des vecteurs constants pour $i \in 1, \dots, 2n$.

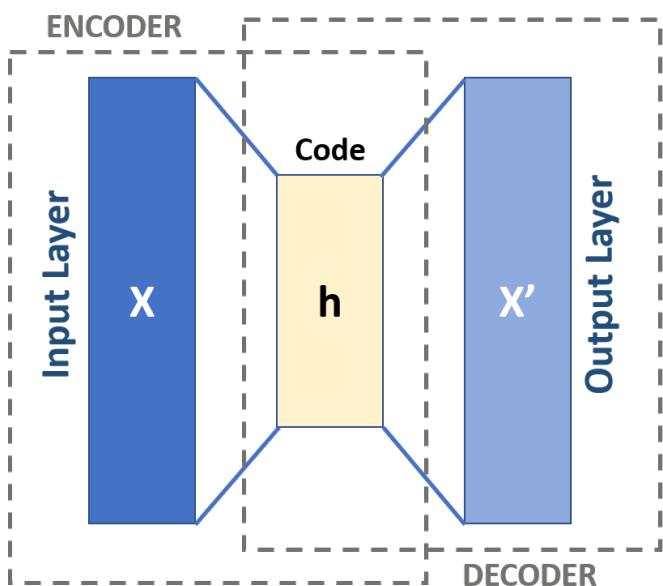


FIGURE 1 – Matérialisation de l'autoencodeur

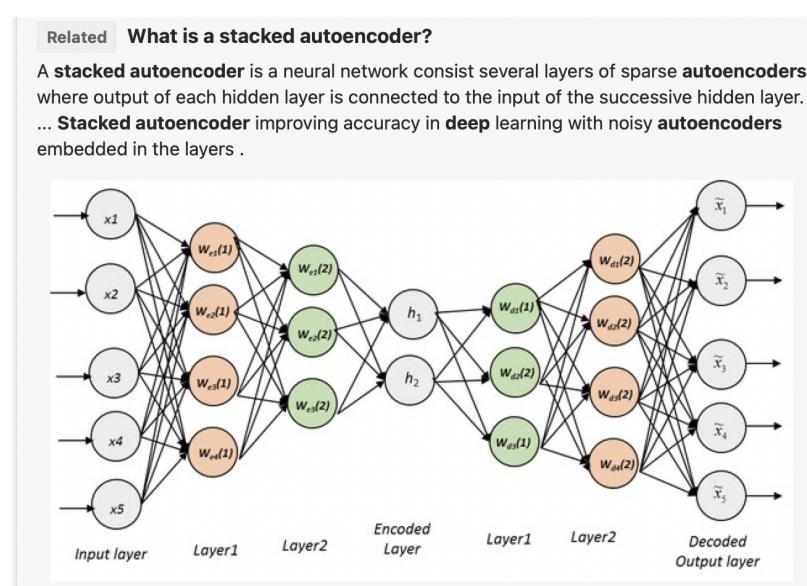


FIGURE 2 – stack encoder

4.1.3 Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP)

L'UMAP[6] est une technique de réduction de dimension utilisant l'analyse topologique des données qui peut être utilisée pour la visualisation de manière similaire à t-SNE, mais aussi pour la réduction de dimension non linéaire générale. On détermine l'intégration en cherchant une projection en basse

dimension des données dont la structure topologique se rapproche le plus (ou est similaire à) de la structure topologique floue (de départ) construite.

Les bases mathématiques solides de Umap assurent que l'algorithme est robuste et interprétable. Trouver une bonne représentation à "basse dimension", dépend de notre capacité à mesurer la "proximité" d'une correspondance que nous avons trouvée en termes de structures topologiques floues, pour ainsi transformer le problème en une question d'optimisation mesuré par l'entropie croisée. L'idée est de recouvrir les points par des boules et construire un complexe simplicial divisé en plusieurs composants connectés (c'est la phase de construction de la structures topologiques floues via le Nearest-Neighbor-Descent algorithm of Dong et al) et puis un algorithme d'optimisation (descente de gradient stochastique) permet de trouver la bonne représentation topologique de dimension plus petite.

4.1.4 TCA ou Tensor Decomposition or Factorization

Tenseur : Un tenseur est un tableau multidimensionnel. Il est également connu sous le nom de tableau d-way, dans lequel "d" signifie "dimensions". Par conséquent, la quasi-totalité des structures de données géométriques avec lesquelles nous travaillons sont des tenseurs. Jusqu'à d=2, ces tenseurs ont des noms spécifiques : tenseur zéro : scalaire tenseur à sens unique : vecteur tenseur à deux voies : matrice

Décomposition : La décomposition est un processus de découpage en éléments constitutifs. En analyse mathématique, elle signifie la factorisation d'un tenseur à plusieurs voies (d-way tensor). En science des systèmes, elle consiste à trouver une partition optimale d'un système en termes de ses sous-systèmes.

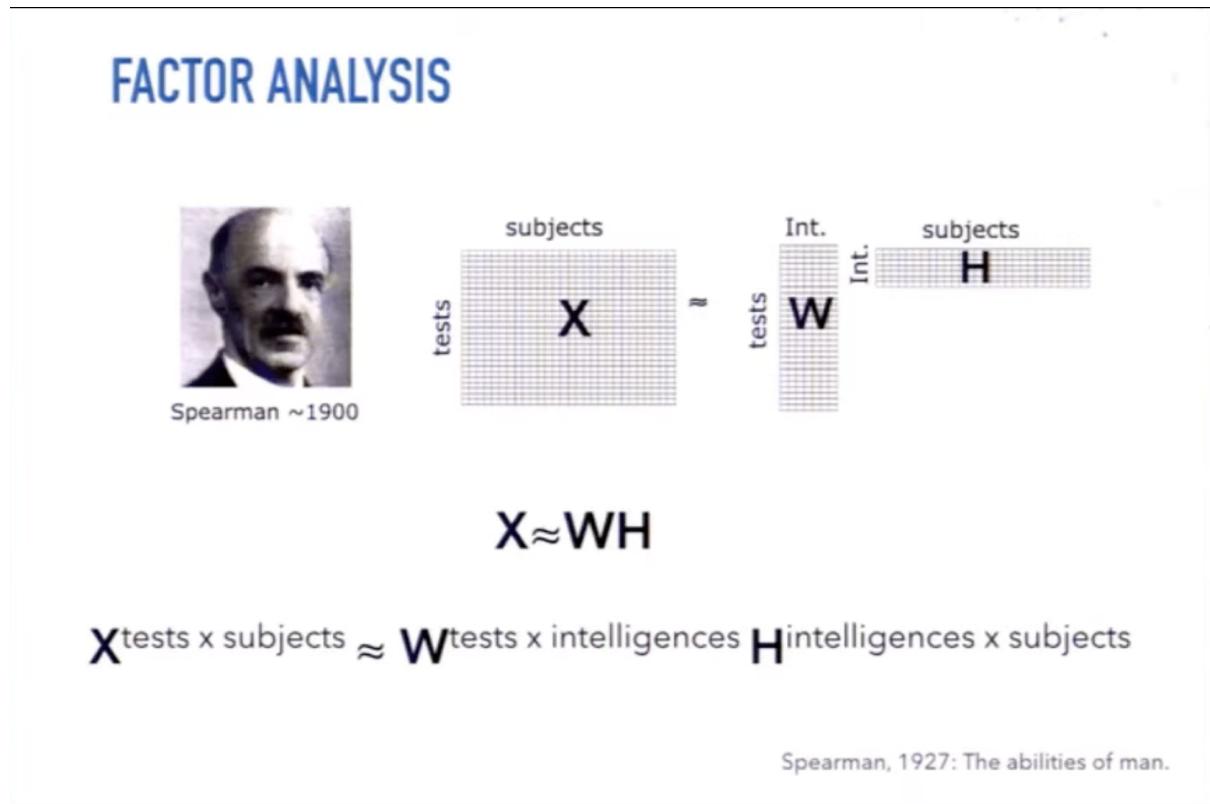


FIGURE 3 – Spearman Decomposition

En général, les décompositions sont motivées par la nécessité d'obtenir un ensemble d'éléments constitutifs beaucoup plus simple, capable de représenter au mieux un système donné.

Pour l'ACP(ou PCA en anglais), une telle décomposition est connue sous le nom d'analyse factorielle. La formulation $X = A \times B$ souffre d'un problème appelé **le problème de rotation**. C'est-à-dire que nous pouvons insérer n'importe quelle matrice de rotation non singulière, Z, dans la formulation ci-dessus, et obtenir la même approximation de X (étant donné que les colonnes de Z ont une amplitude de 1). Par conséquent, si la formulation ci-dessus n'est pas contrainte, elle donne lieu à une infinité de combinaisons de A et B.

Décomposition tensorielle à trois voies(Three-way Tensor Decomposition) : La décomposition à trois voies est simplement l'extension de la décomposition à deux voies. Cependant, bien que dans le cas de la décomposition à deux voies, des contraintes explicites doivent être imposées au problème pour obtenir une solution unique, la haute dimensionnalité du format tensoriel présente des avantages, notamment **la possibilité d'obtenir des représentations compactes, l'unicité des décompositions, la flexibilité dans le choix des contraintes et la généralité des composants qui peuvent être identifiés**.

Dans le cas d'une décomposition à trois voies, nous avons un tenseur à trois voies et nous aimeraions avoir un modèle M qui approxime $X \in \mathbb{R}^{I \times J \times K}$, via $a \in \mathbb{R}^I$, $b \in \mathbb{R}^J$, et $c \in \mathbb{R}^K$ tel que :

$$X \approx M = \sum_{r=1}^R a_r \otimes b_r \otimes c_r = \mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C} \quad (8)$$

Où $X \in \mathbb{R}^{I \times J \times K}$, $A \in \mathbb{R}^{I \times R}$, $B \in \mathbb{R}^{J \times R}$, et $C \in \mathbb{R}^{K \times R}$ et R est la nouvelle dimension (réduite) de nos données, souvent appelée rang et correspond au nombre de termes dans la somme. À la suite de cette décomposition, nous aurons trois matrices $A \in \mathbb{R}^{I \times R}$, $B \in \mathbb{R}^{J \times R}$ et $C \in \mathbb{R}^{K \times R}$. Cette opération est simplement la somme du produit externe de chaque colonne de A,B et C où l'indice de colonne est spécifié par r comme illustré ci-dessous :

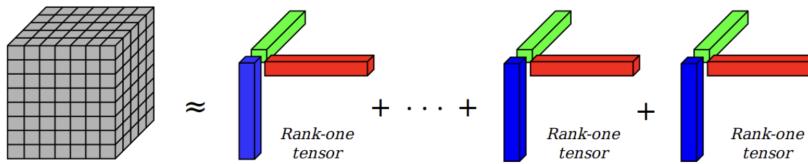


FIGURE 4 – Tensor Decomposition 3-Mode

Remarques :Toutes les méthodes de décomposition(décomposition de Tucker,Non-negative CP de-composition etc) sont basées sur l'optimisation suivante :

$$\min_{a_r, b_r, c_r, r=1,..,R} \left\| X - \sum_{r=1}^R a_r \otimes b_r \otimes c_r \right\|^2 \quad (9)$$

Perspectives : Décomposition en R composantes(choix de R sur la base de l'erreur de reconstruction) et apprentissage sur l'axe temporels.

5 Prétraitemennt et Représentation de l'information

5.1 Données d'entrée (Input Data)

Contrairement à certains algorithmes utilisant intégralement ou en partie una approche autorégrressive sur l'historique passée,nous utilisons un encodage pour matérialiser la saisonalité de façon

périodique : avec entre autre la saison, le jour de la semaine, le jour dans le mois le mois de l'année, vacances scolaires, événements, les événements spéciaux Il s'agit notamment :

- one hot encoder(similaire au get_dummies de pandas) qui transforment les données en variables fictives ou indicatrices :
- La décomposition en serie Fourrier qui représente la saisonalité comme une somme de cosinus et sinus périodique et de période le nombre de jour de la semaine ou du mois, ou le nombre de semaine ou de mois de l'année etc. C'est ce qu'utilise l'algorithme Prophet de Facebook, en plus d'avoir la capacité d'ajouter en entrée les jours fériés, l'information métier (les événements spéciaux) et des covariables(tout comme DeepAR d'Amazon)

5.2 Matrix Representation

La PCA et l'Umap utilisent des données sous format tableau (matrice ou DataFrame pandas) dont l'axe 0 des lignes corresponds aux pas de temps futurs et l'axe 1 des colonnes, les produits par magasins. schéma

Nous créons une matrice d'une ligne par pas de temps(donc le nombre de lignes de la matrice est égale au nombre de pas de temps des données d'entraînement) et une colonne par produit/- magasin (soit environ 230*30 colonnes

5.3 Tensor Representation pour TCA

Tensor decomposition avec Python: Apprentissage de structures à partir de données multidimensionnelles.

Interet de l'approche TCA : Renforcer l'indépendance des produits de chaque magasin, via une représentation en dimension 3 où chacun des composants des données(product_id, store_id et time_step) a sa base. Ici les produits sont découplés du temps et des Magasins, contrairement à l'approche PCA en dimension 2 où ce sont les produits/magassins sont découplés du temps.

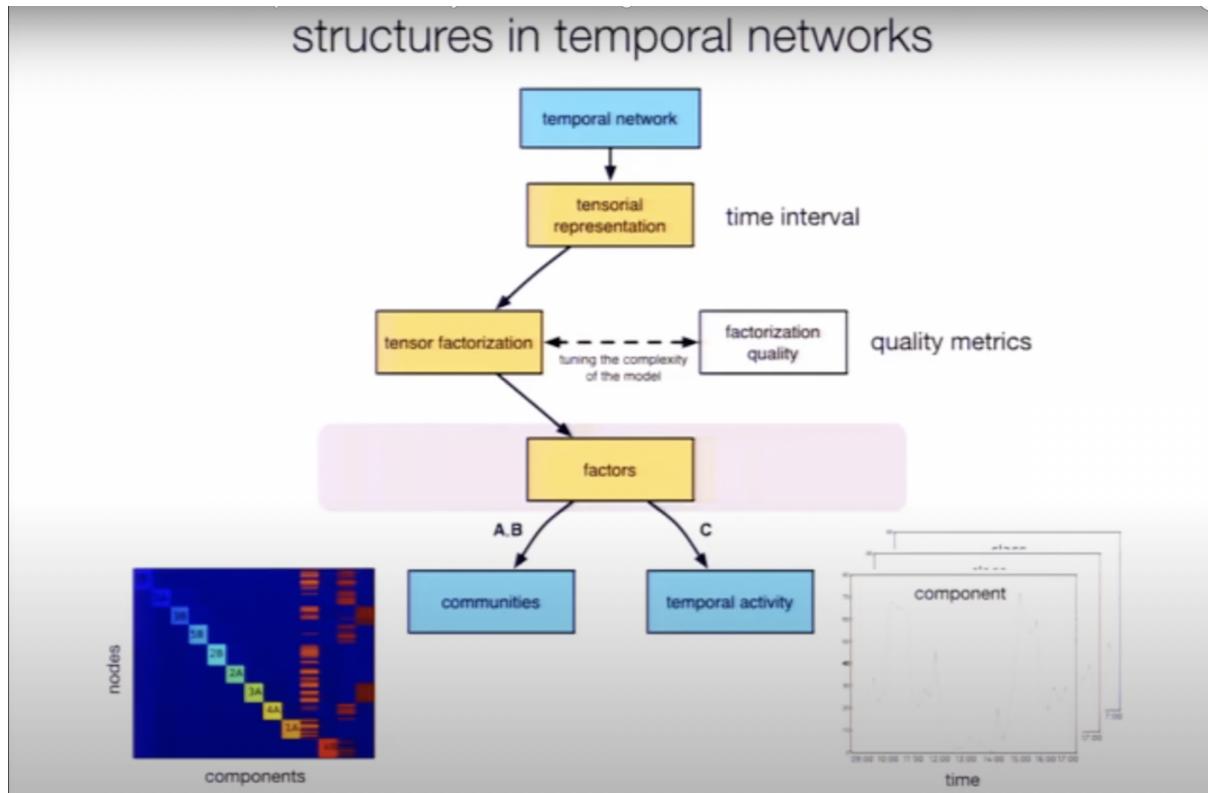


FIGURE 5 – Achitecture Tensor Decomposition 3-Mode

Comment trouver A, B et C ?

Dans cette section, nous allons nous concentrer sur l'implémentation de la décomposition tensorielle à trois voies en utilisant deux bibliothèques Python : TensorLy et tensortools. De plus, nous implémenterons également un décomposeur tensoriel à trois voies très simple en utilisant Numpy et un algorithme d'optimisation alternatif.

Python offre trois bibliothèques pour faire la décomposition ou factorisation tensorielle. tensortools, tensorly et numpy. La comparaison entre ces bibliothèques via l'**erreur et le temps de Reconstruction** nous a permis de sélectionner Tensorly pour la suite du travail.

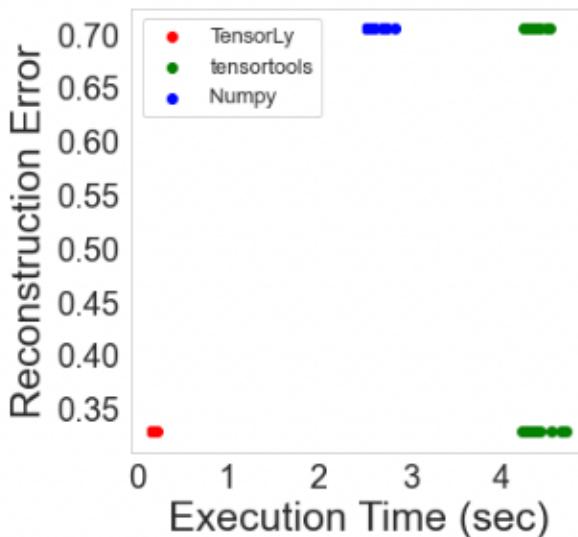


FIGURE 6 – Performance sur la reconstruction des Méthodes

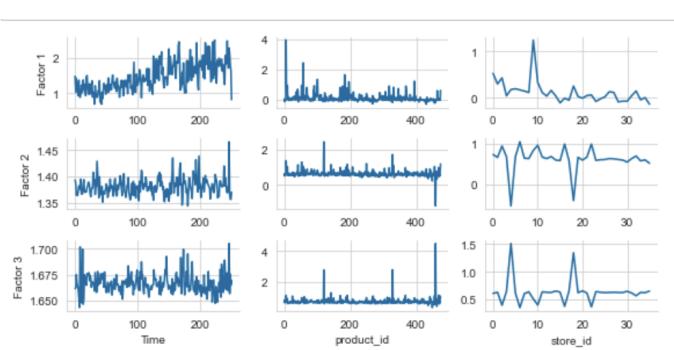


FIGURE 7 – Factors computed with Tensorly

Remarques : D'autres Méthodes de décomposition peuvent être utilisées comme la décomposition de Tucker, Non-negative CP decomposition etc tous basées sur l'optimisation suivante :

$$\min_{a_r, b_r, c_r, r=1, \dots, R} \left\| X - \sum_{r=1}^R a_r \otimes b_r \otimes c_r \right\|^2 \quad (10)$$

Perspectives : Décomposition en R composantes (choix de R sur la base de l'erreur de reconstruction) et apprentissage sur l'axe temporels.

6 Méthodes d'entraînement

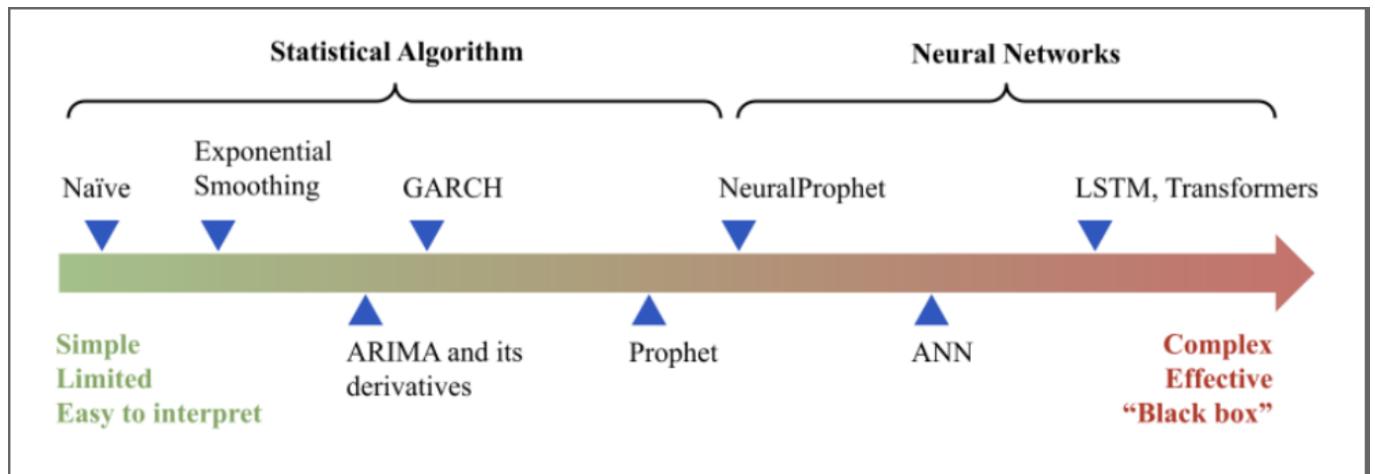


FIGURE 8 – Méthodes d'analyses de séries chronologiques

6.1 Transformation en données d'apprentissage supervisé

Étant donné une séquence de chiffres pour un ensemble de données de séries temporelles ou chronologiques, nous pouvons restructurer les données pour qu'elles ressemblent à un problème d'apprentissage supervisé. La série peut être transformée en échantillons avec des composantes d'entrée et de sortie qui peuvent être utilisées comme partie d'un ensemble d'apprentissage pour former un modèle d'apprentissage supervisé. Cette transformation est appelée "fenêtre glissante" (sliding window transformation) car elle revient à faire glisser une fenêtre sur les observations antérieures qui sont utilisées comme entrées du modèle afin de prédire la(s) prochaine(s) valeur(s) de la série.

6.2 Les Modèles d'apprentissage Automatique mis en place

— La Moyenne mobile :

La moyenne mobile est une méthode simple permettant d'extraire les composantes basses fréquences d'une série temporelle autrement dit sa tendance. Elle est également connue comme une méthode de lissage car elle agit comme un filtre passe bas et donc élimine le bruit.

Le calcul de la moyenne mobile dépend d'un paramètre l appelé la largeur de fenêtre. Ce paramètre correspond au nombre d'observations incluses dans le calcul de la moyenne glissante effectuée. Plus l est grand plus le lissage est important (jusqu'à atteindre la fonction constante

égale à la moyenne).

La moyenne mobile se calcule ainsi :

$$\hat{y}_t = \frac{1}{2l+1} \sum_{t'=t-l}^{t+l} y_{t'} \quad (11)$$

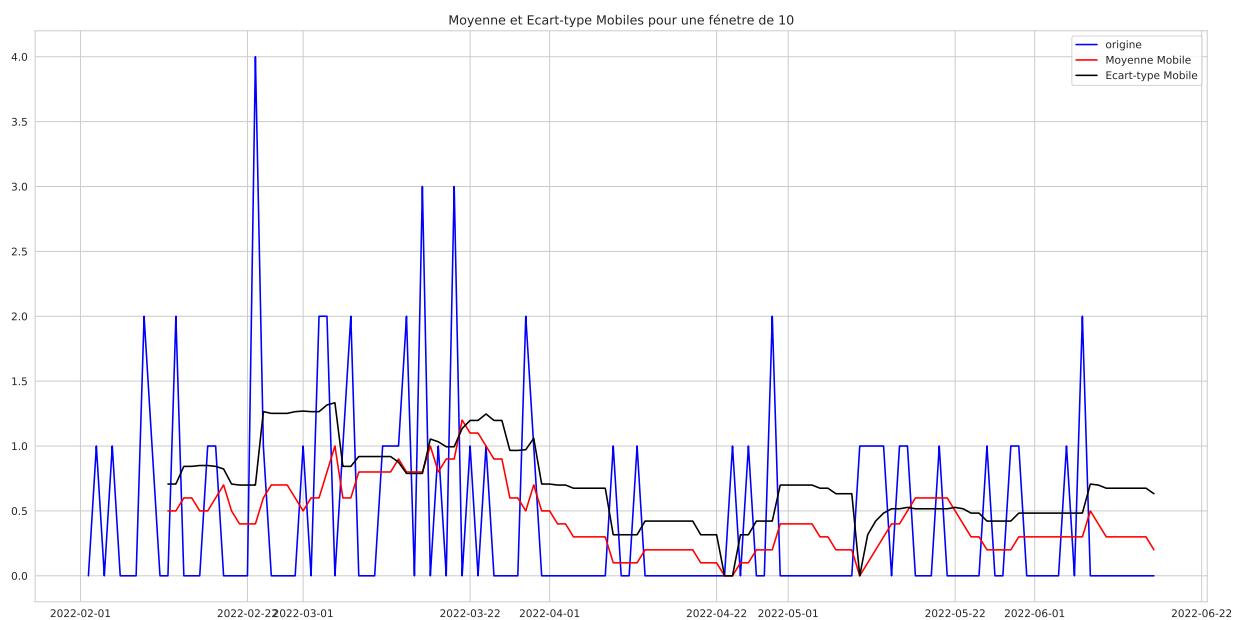


FIGURE 9 – Moyenne Mobile

- La Moyenne Mobile Autoregressive Intégrée (ARIMA) :
ARIMA

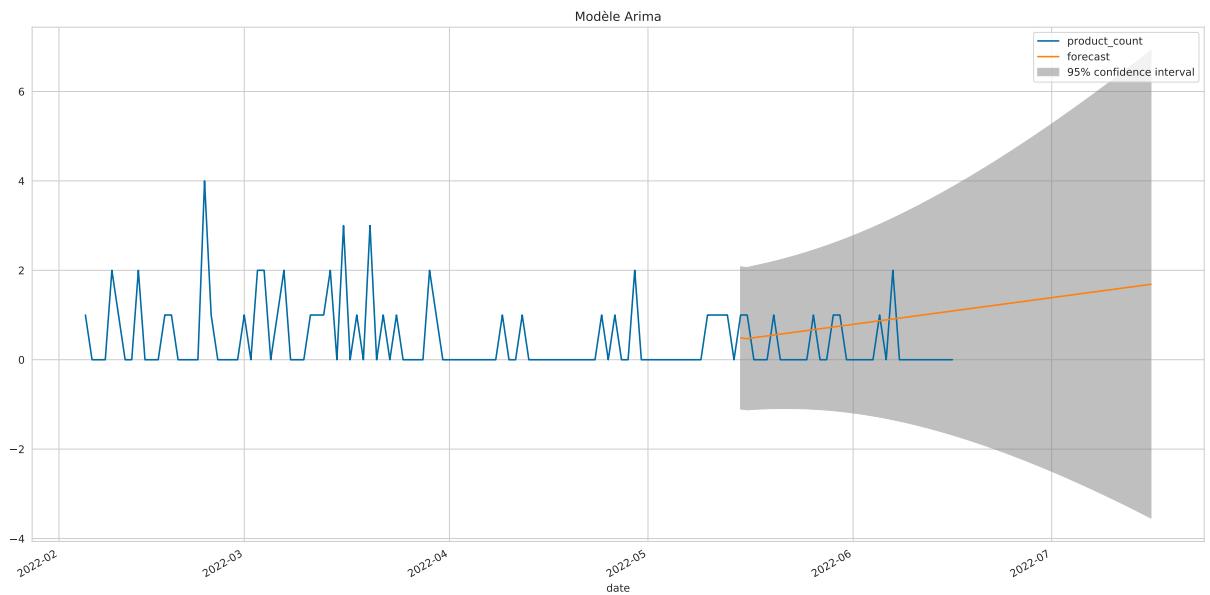


FIGURE 10 – Moyenne Mobile Autoregressive Intégrée

- **La Moyenne Mobile Autoregressive Intégrée saisonnière (SARIMA or Seasonal ARIMA) :** SARIMA ajoute un terme retardé à ARIMA qui tient compte de la saisonnalité des données. C'est-à-dire il intègre des décalages multiples de la saisonnalité.

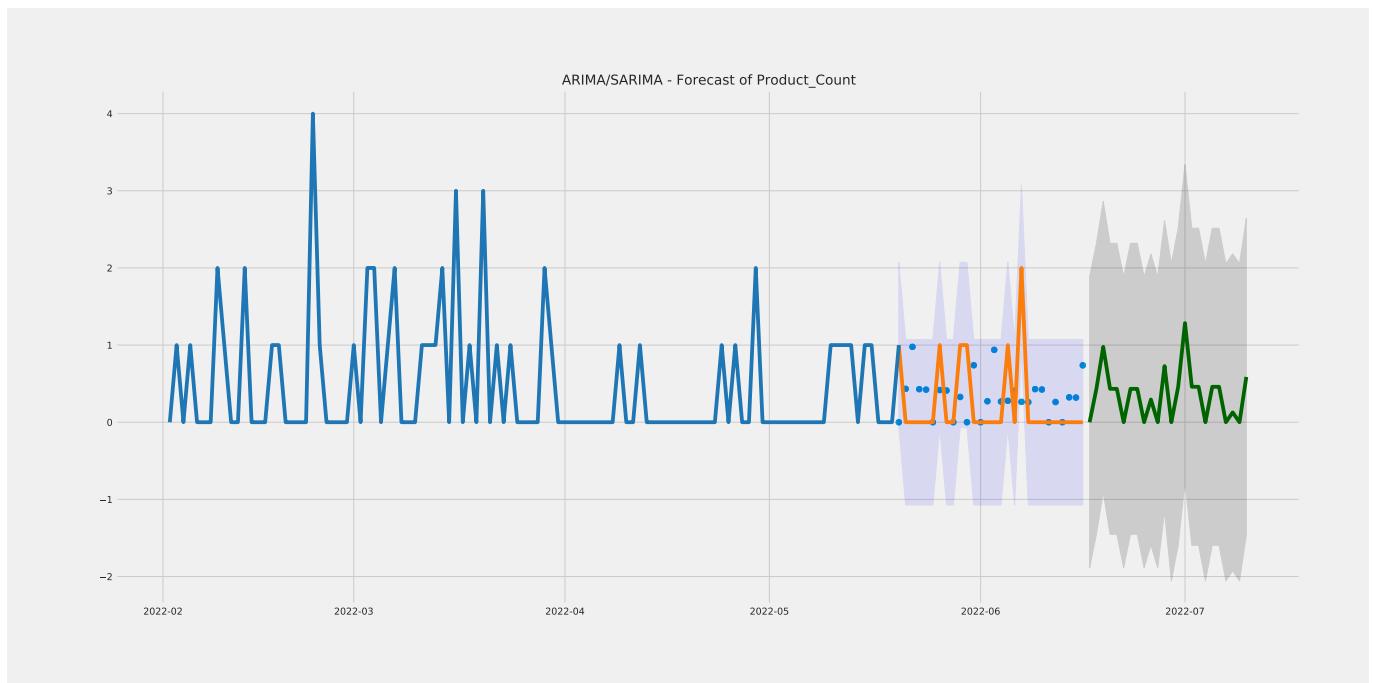


FIGURE 11 – Moyenne Mobile Autoregressive Intégrée

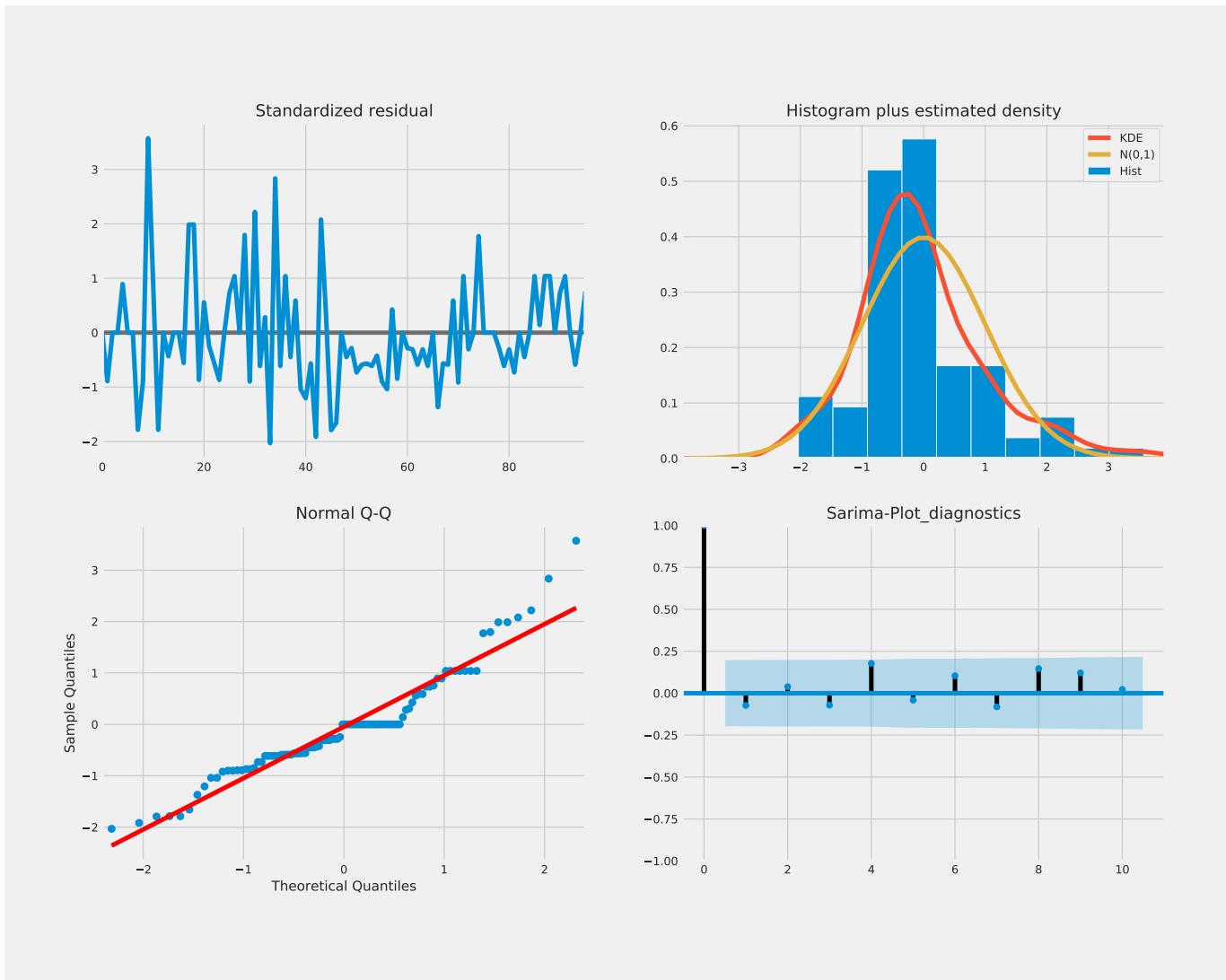


FIGURE 12 – Sarima_plot_diagnostics

— Le Lissage exponentiel simple :

Les prévisions produites à l'aide de méthodes de lissage exponentielles sont des moyennes pondérées d'observations antérieures, les poids décroissant de manière exponentielle à mesure que les observations vieillissent. En d'autres termes, plus l'observation est récente, plus le poids associé est élevé.

Le lissage exponentiel simple convient pour la prévision de données sans tendance claire ni de tendance saisonnière et pour un horizon de prévision restreint.

Pour une série temporelle y_t on appelle lissage exponentiel simple de paramètre $\alpha \in [0, 1]$ de cette série le processus \hat{y}_t défini ainsi : $\hat{y}_{t+1/t} = \alpha y_t + (1 - \alpha)\hat{y}_{t/t-1}$ on a donc :

$$\hat{y}_{t+1/t} = \sum_{i=0}^{t-1} \alpha(1 - \alpha)^i y_{t-i} \quad (12)$$

la prévision de l'instant $t+1$ est donc une somme pondérée des valeurs passées de la série, les poids décroissant exponentiellement dans le passé. La mémoire de la prévision dépend de α . Plus α est proche de 1 plus les observations récentes influent sur la prévision, à l'inverse un α proche de 0 conduit à une prévision très stable prenant en compte un passé lointain.

Une autre façon d'écrire le lissage exponentiel (forme d'erreur de correction) : $\hat{y}_{t+1/t} = \hat{y}_{t/t-1} + \alpha(y_t - \hat{y}_{t/t-1})$. $\hat{y}_{t+1/t}$ est une prévision à horizon 1. Il est parfois nécessaire d'effectuer une

prévision à un horizon h quelconque. On notera par la suite, $\hat{y}_{t+h/t}$ la prévision de y_{t+h} conditionnellement à (y_1, \dots, y_t) . Pour le lissage exponentiel simple cette prévision est tout simplement : $\hat{y}_{t+h/t} = \hat{y}_{t+1}$ car on approxime le futur de la série à une constante.

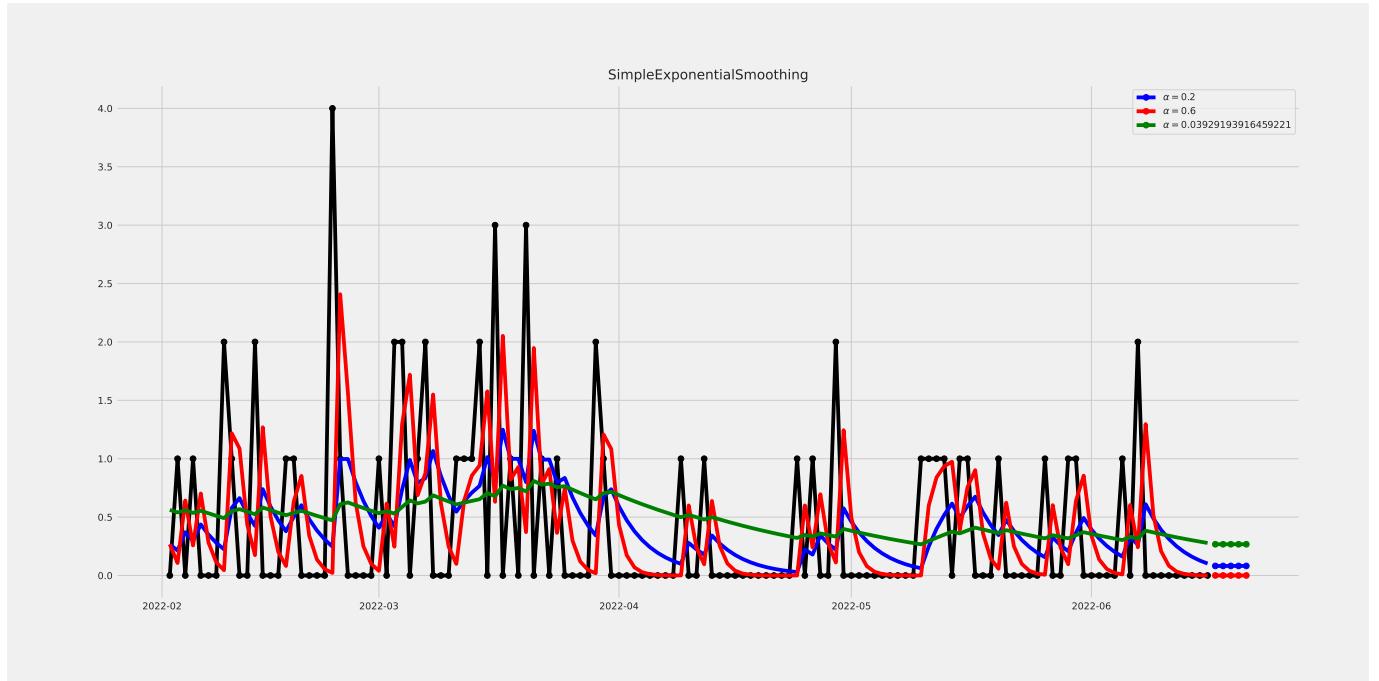


FIGURE 13 – Lissage Exponentiel Simple

— Le Lissage exponentiel double (ou de Holt) :

Holt (1957) a étendu le lissage exponentiel simple au cas du lissage exponentielle linéaire. L'idée est d'ajuster une droite au lieu d'une constante dans l'approximation locale de la série.

Définition : Soit une série temporelle y_t . On appelle lissage exponentiel double (ou de Holt) de paramètre $\alpha \in [0, 1]$ de cette série le processus \hat{y}_t défini ainsi : $\hat{y}_{t+h/t} = l_t + b_t h$:Prévision

$$\text{avec : } \begin{cases} l_t &= l_{t-1} + b_{t-1} + (1 - (1 - \alpha)^2)(y_t - \hat{y}_{t-1}) : \text{Niveau} \\ b_t &= b_{t-1} + \alpha^2(y_t - \hat{y}_{t-1}) : \text{Tendance} \end{cases}$$

l_t est une estimation du niveau de la série, b_t une estimation de la tendance (pente) de la série à l'instant t , α est le paramètre de lissage du niveau et de la tendance. l_t et b_t minimisent à chaque instant :

$$\underset{l, b}{\operatorname{argmin}} \sum_{i=0} (1 - \alpha)^i (y_{t-i} - (l + bi))^2 \quad (13)$$

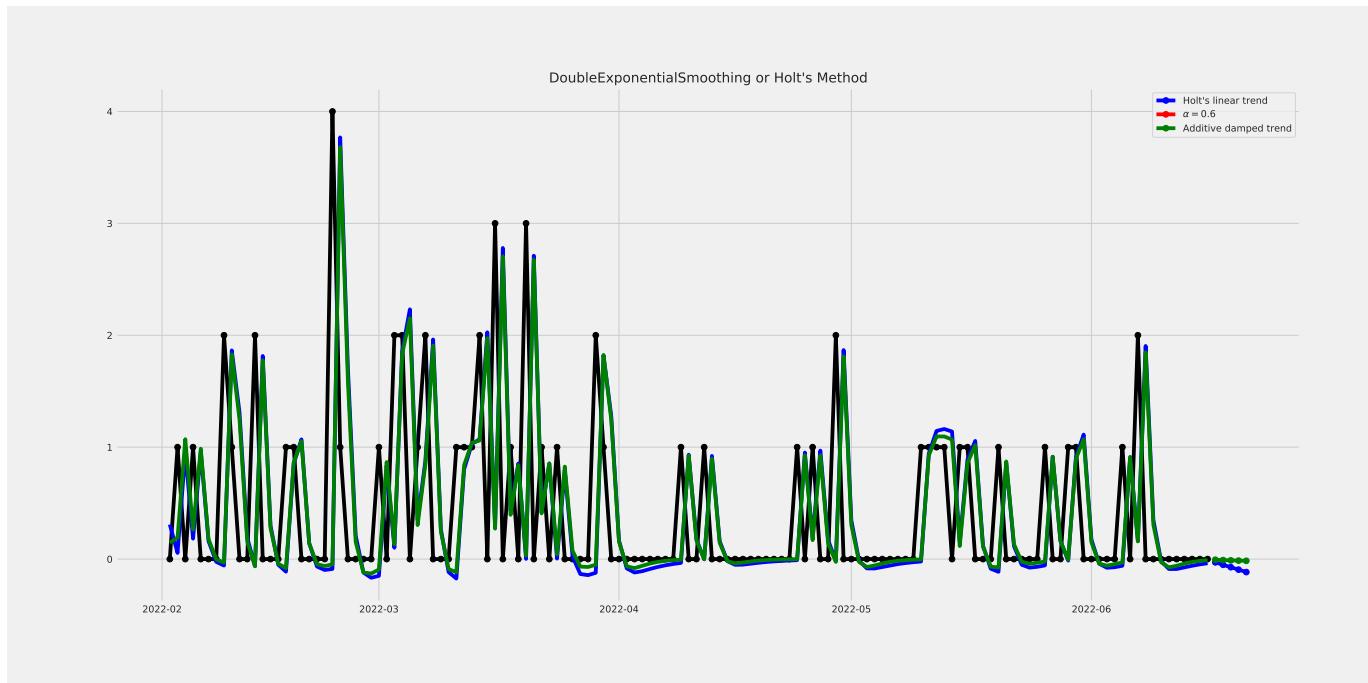


FIGURE 14 – Lissage Exponentiel Double

— La regression Linéaire :

L'algorithme de régression linéaire (un seul prédicteur ou variable prédictrice) et de régression linéaire multiple (plus d'un prédicteur) est basé sur l'équation d'une ligne ou d'un hyperplan ($y = m * X + C$) et permet de trouver les meilleures valeurs des coefficients/matrices "m" et "C" pour des données données en utilisant une fonction de coût. L'une des fonctions de coût est connue sous le nom de somme des erreurs quadratiques (SSE) ou la racine de l'erreur quadratique moyenne (RMSE). Il en existe beaucoup d'autres comme la moyenne des erreurs quadratiques (MAE), la perte logarithmique ou la perte d'entropie croisée, etc.

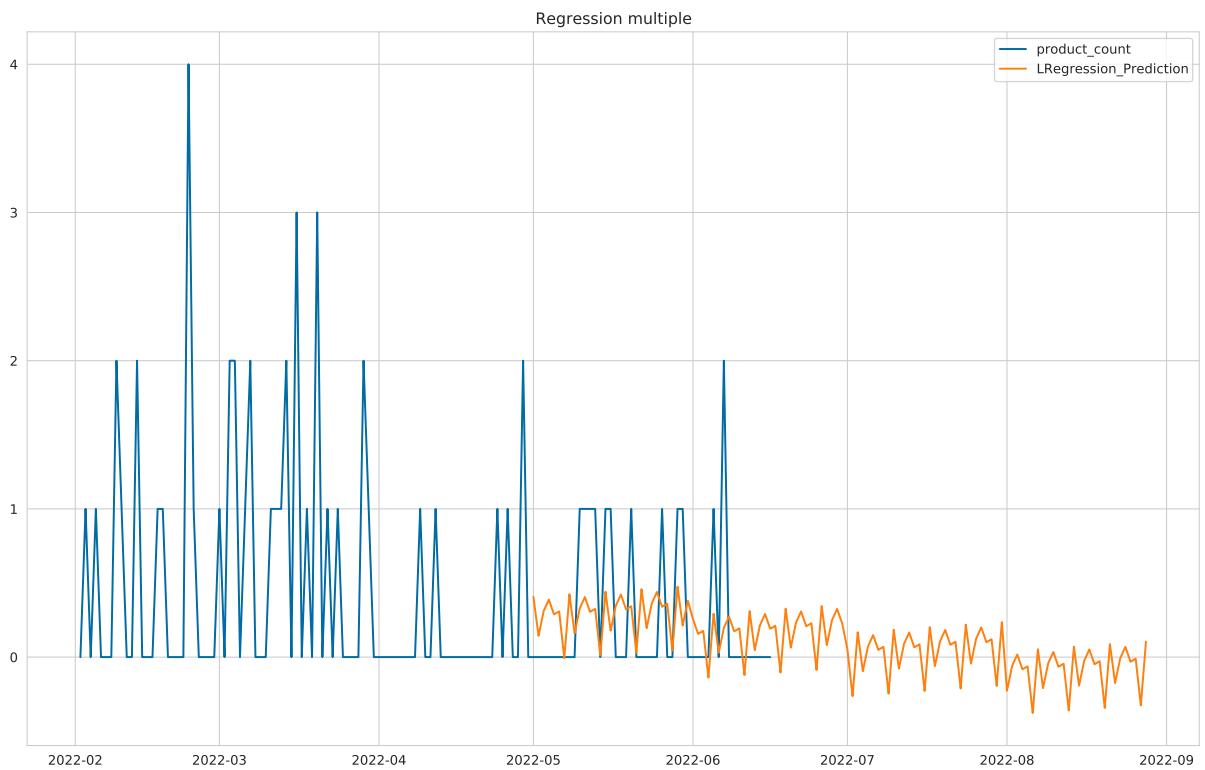


FIGURE 15 – Regression Linéaire Multiple

Cette méthode statistique est très importante en prévision parce qu'elle permet d'introduire les facteurs extérieurs qui influencent la valeur future de la variable étudiée. Pour les prévisions de ventes, ce sont les prix et les promotions de l'entreprise qui viennent d'abord en tête comme variables explicatives. Idéalement, les prix et les promotions des principaux concurrents devraient être considérés. S'ils sont probablement disponibles pour le passé, ils ne le sont sûrement pas pour le futur.

— Algorithme de type Bagging : Les forêts aléatoires ou RandomForest :

Le Bagging, le Boosting et Stacking sont des techniques d'Ensemble Learning qui permettent de développer les modèles de machine learning les plus puissants au monde. Tout repose sur le concept : "The Wisdom of the Crowd." Les modèles de ML réunis ensemble sont plus forts qu'un modèle de ML tout seul. Pour ça, il faut que les modèles soient un minimum compétents et suffisamment diversifiés.

Le Bagging (et la random forest) permet d'obtenir des ensemble de modèles diversifiés en entraînant chaque modèle sur une portion aléatoire des données (en échantillonnant le dataset avec le Bootstrapping)

Les forêts aléatoires (création de plusieurs arbres de décision indépendants, connue sous le nom de forêt aléatoire) sont essentiellement des arbres de décision multiples mis ensemble. Cette méthode est également connue sous le nom de "bagging". La réponse finale s'obtient en combinant les prédictions de chacun des arbres. Dans le cas de la régression, la réponse finale est la moyenne des prédictions faites par tous les arbres et dans le cas de la classification, la réponse finale est le mode (vote majoritaire) des prédictions faites par tous les arbres.

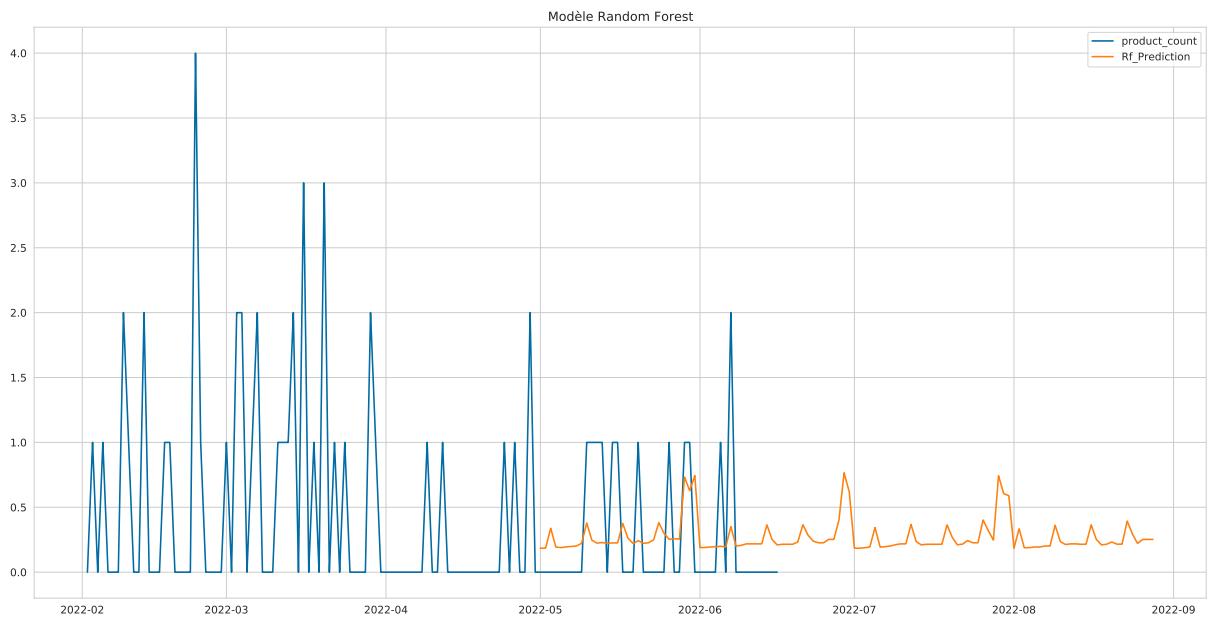


FIGURE 16 – Forets Aléatoires

— Algorithme de type Boosting : XGboost :

XGboost est l'abréviation de **Xtreme Gradient Boosting**. Le Boosting permet quant à lui de construire des modèles les uns après les autres, en demandant à chaque modèle de corriger les erreurs de son prédecesseur. (Adaboost et GradientBoosting sont des exemples d'algorithmes tout comme XgBoost)

Quel que soit le type de tâche de prédiction à accomplir, régression ou classification. XGBoost est bien connu pour fournir de meilleures solutions que les autres algorithmes d'apprentissage automatique. En fait, depuis sa création, il est devenu l'algorithme d'apprentissage automatique "de pointe" pour traiter les données structurées.

La logique de base derrière XGboost est le Boosting. Sa particularité est qu'il corrige exactement les erreurs du modèle précédent. Ceci consiste à créer un ensemble de modèle prédictif successif dont les prochains corrigent les erreurs des modèles précédents. Ce processus est répété jusqu'à ce que le modèle soit précis à 100 % ou que le nombre maximum d'itérations autorisées (nombre d'arbres) soit atteint.

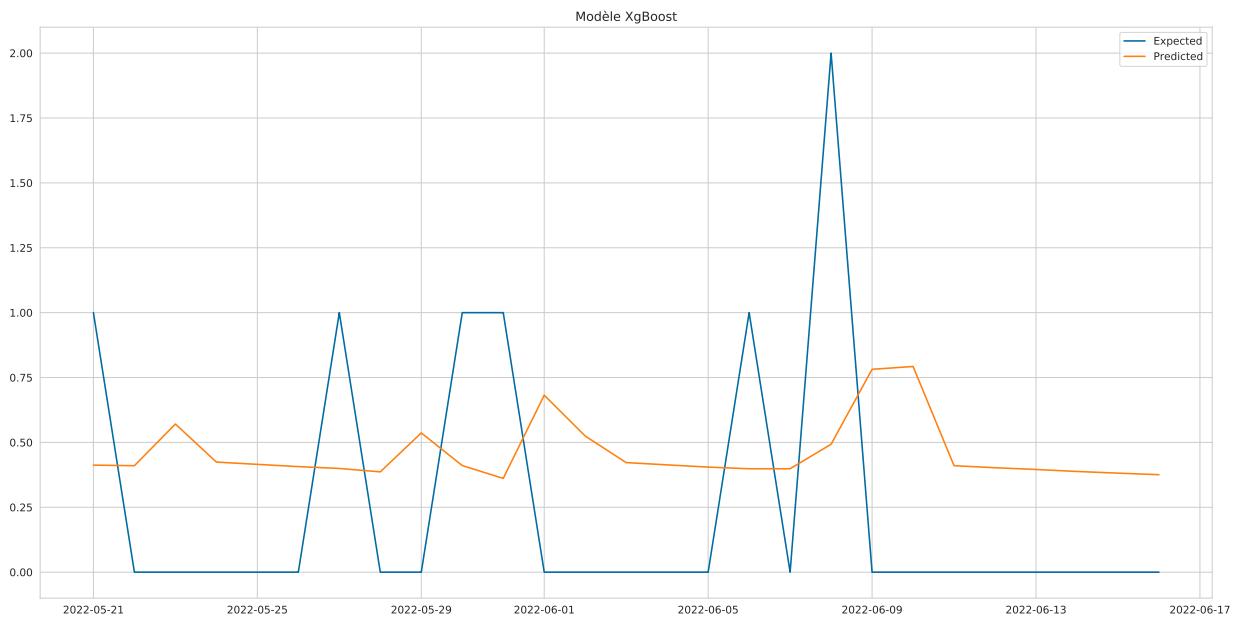


FIGURE 17 – Arbres Boostés auto-regressif

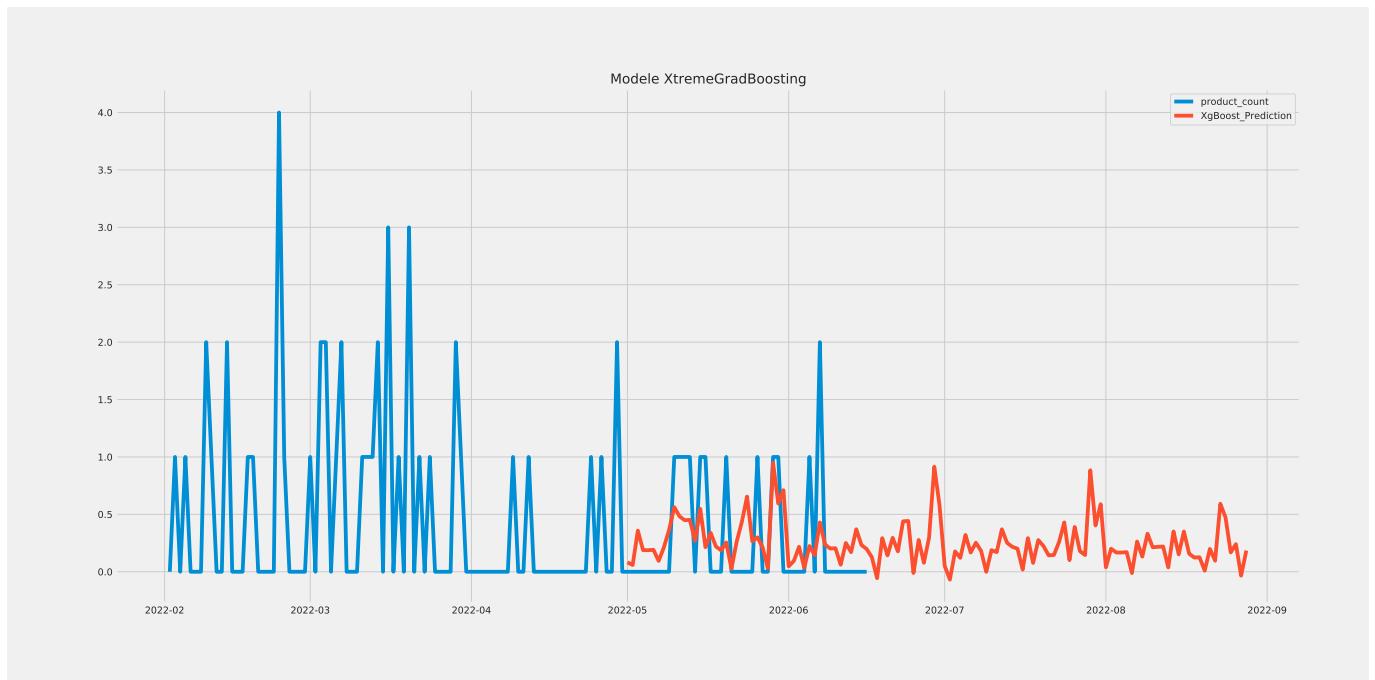


FIGURE 18 – Arbres Boostés sur features temporelles

— Prophet [3]

L'équipe Core Data Science de Facebook a récemment publié une nouvelle procédure de prévision des données de séries chronologiques appelée Prophet. Elle est basée sur un modèle additif où les tendances non linéaires sont ajustées avec la saisonnalité annuelle et hebdomadaire, plus les vacances. Elle permet d'effectuer des prévisions automatisées qui sont déjà mises en œuvre dans R à l'échelle dans Python 3.

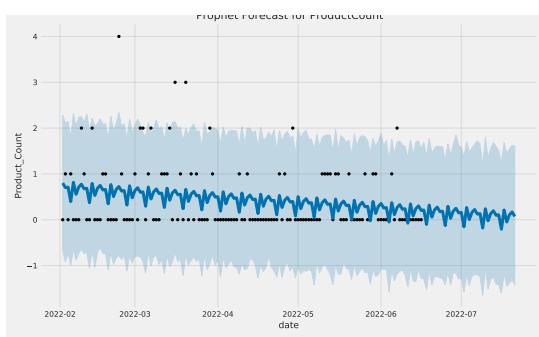


FIGURE 19 – Prophet_plot

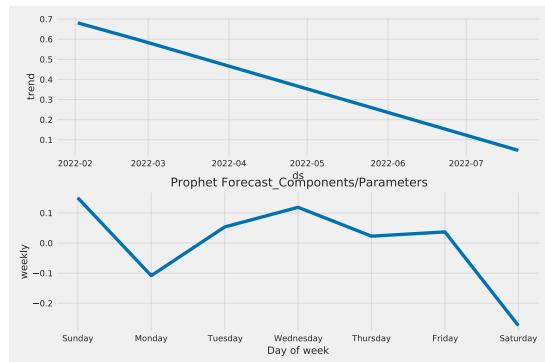


FIGURE 20 – Prophet_Components-Parameters

Prophet [10] est comme une extension des modèles AR de base. Au lieu de se contenter d'utiliser les valeurs décalées de la variable cible, le modèle offre une forme supplémentaire d'ingénierie des caractéristiques - il applique des séries de Fourier à la variable d'entrée. En outre il s'agit d'un modèle facilement interprétable, stable dans les résultats et ouverts à l'ajout de l'information métier.

6.3 Les Modèles d'apprentissage Profond ou de type ANN

Les modèles de type Artificial Neural Network (ANN) mis en place sont principalement le MLP[5] et le LSTM[1]

- **MultiLayerPerceptron(MLP)** :

Le modèle MLP va apprendre une fonction qui fait correspondre une séquence d'observations passées en entrée à une ou plusieurs observation en sortie(horizon de prédition). En tant que telle, la séquence d'observations doit être transformée en plusieurs exemples à partir desquels le modèle peut apprendre : c'est la transformation en données d'apprentissage supervisé.

Nous avons utilisé comme modèle une **séquence de deux couches(100 neurones pour le premier et n_{steps} pour le second)** d'activations de type Relu avec une optimisation d'adams sur la perte quadratique.

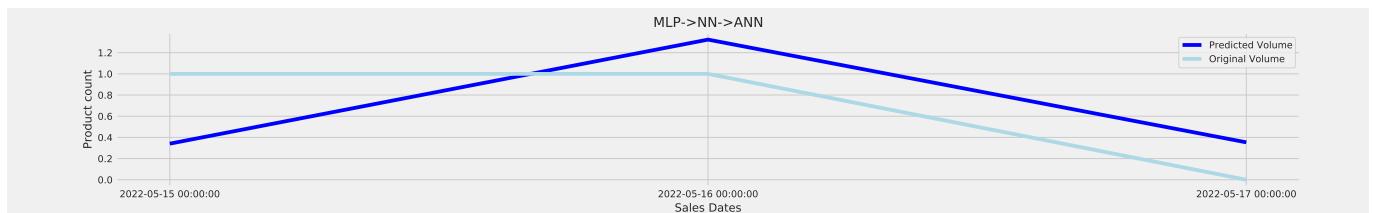


FIGURE 21 – Perceptron Multi Couches

- **Long Short Term Memory (LSTM)** Il s'agit d'un modèle de réseau de Neuron récurrent ou RNN.

Principe du RNN Le RNN essaie de regarder en arrière (les pas de temps précédent) dans le temps les événements qui se sont produits et apprend la relation entre la sortie et l'entrée. Cependant, admet certains problèmes tels que le gradient évanescence(vanishing gradient) et à l'opposé, gradient explosif(Exploding gradient). Ces problèmes deviennent plus prononcés dans les RNNs à cause de la rétropropagation dans le temps. Cela conduit à l'incapacité des RNNs à se souvenir des informations à long terme. C'est là que les LSTM entrent en jeu !

Les LSTM résolvent ce problème en introduisant le concept de "cellules mémoire"(memory cells). L'idée est de maintenir un journal de ce qui est important et de ce qui ne l'est pas à travers le temps.

L'idée derrière les cellules de mémoire LSTM est de maintenir un journal(a log) de ce qui est important et de ce qui ne l'est pas à travers le temps. Ces cellules de mémoire stockent les informations importantes et rejettent les informations non importantes.

Par conséquent, les LSTM sont actuellement les algorithmes les plus utilisés pour de nombreuses applications modernes qui impliquent la compréhension des dépendances à long terme des données. Cette propriété des LSTM en fait un merveilleux algorithme pour apprendre des séquences qui sont interdépendantes et peut aider à construire des solutions comme la traduction de langues,les prévisions du cours des actions, les séries chronologiques de ventes/demande, les chatbots, les autocorrections, les suggestions de mots suivants, etc.

Nous avons utilisés comme constituants du reseau :

Un reseau de 3 LSTM ou 3 couches ; La sortie de chaque étape temporelle doit être partagée avec la couche cachée suivante.

Notez que le nombre de neurones dans la couche dense(la plus interne) est le nombre de pas de temps futurs sur lesquels nous voulons prédire la demande avec des activations de type Relu(Rectify Linear Unit).

TimeSteps=10 : La prédiction des quantités des prochains jours est basée sur les quantités des derniers jours. FutureTimeSteps=3 : Nombre de jours dans le futur pour prédire les quantités.

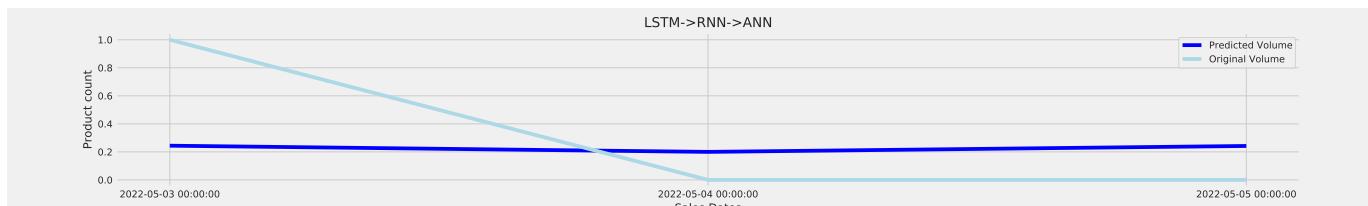


FIGURE 22 – Mémoire à long et court terme

— NeuralProphet

NeuralProphet [9] est une bibliothèque python pour la modélisation de données de séries temporelles basée sur des réseaux neuronaux. Elle est construite au-dessus de PyTorch et s'inspire fortement des bibliothèques Prophet et AR-Net(Auto-Regressive Network)[8] de Facebook.(Pour résumer l'AR-Net en une phrase, il s'agit d'un réseau à une seule couche qui est formé pour imiter le processus AR dans un signal de série temporelle, mais à une échelle beaucoup plus grande que les modèles traditionnels.)

NeuralProphet utilise la descente de gradient de PyTorch pour l'optimisation, ce qui rend la modélisation beaucoup plus rapide. L'autocorrélation des séries temporelles est modélisée à l'aide du réseau autorégressif. Les régresseurs décalés sont modélisés à l'aide d'un réseau neuronal Feed-Forward distinct.

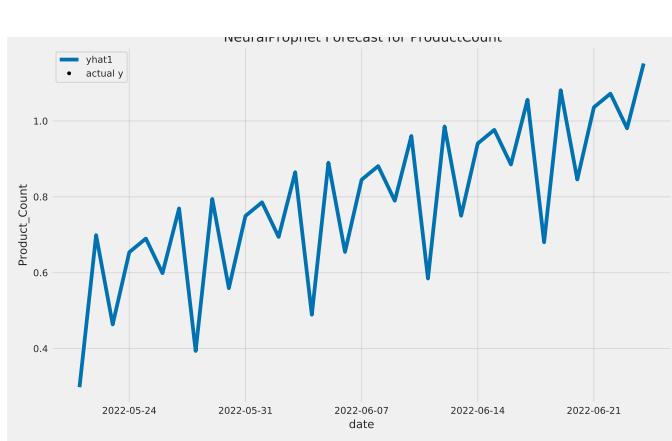


FIGURE 23 – NeuralProphet_plot

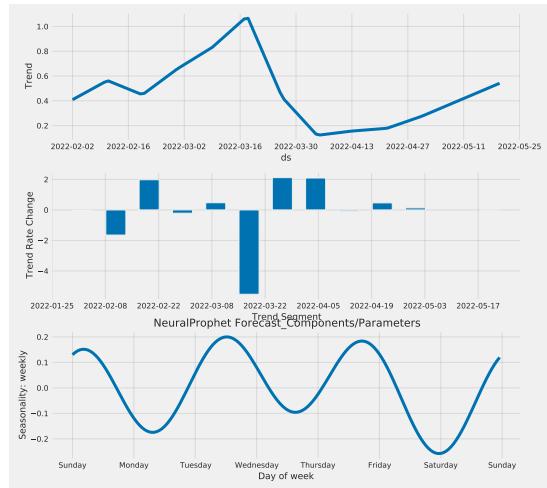


FIGURE 24 – NeuralProphet_Components-Parameters

— DeepAR :

Probabilistic Forecasting with Autoregressive Recurrent Networks [9] L'algorithme DeepAR a quant à lui été développé par Amazon pour répondre à leur besoin de prédition de vente de l'ensemble des produits en entraînant un unique modèle. En effet, les produits ont certes des dynamiques différentes mais l'expérience acquise par des produits avec un long historique peut bénéficier à la prédition des ventes des produits plus récents. Le modèle de DeepAR repose sur un réseau de neurones récurrents (recurrent neural networks ou RNN) par dessus duquel est ajoutée une couche d'apprentissage d'une distribution de probabilité. Le modèle calcule en effet chaque prédition en fonction des valeurs précédentes, qui peuvent elles-mêmes être des prédictions. Il apprend ainsi, au fur et à mesure, les meilleurs paramètres permettant de générer les paramètres de la distribution de probabilité à partir de l'output du RNN, noté h sur le graphique ci-dessous :

Schéma du modèle de DeepAR

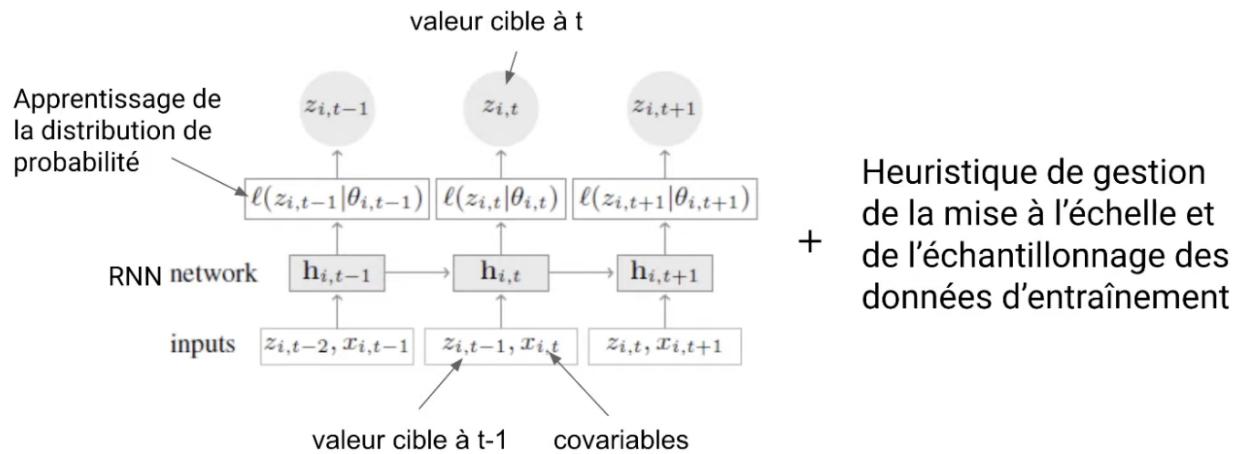


FIGURE 25 – Probabilistic Forecasting with Autoregressive Recurrent Networks[11]

7 Résultats et illustrations

7.1 Modèle temporel produit par Magasin : cas $product_id = a$ et $store_id = b$

a="5cebf8d0b259700045dc8aa" b="5f1198de8746ae00042869ba"

Nous essayons dans cette phase pour un produit d'un magasin donné, de tester plusieurs modèles de prédiction de sa quantité dont les performances sont comparées à celles de la moyenne simple. Notons qu'il est important de faire un prétraitement général consistant à compléter les données manquantes.

Le tableau des performances des différents algorithmes est le suivant :

Model	MSE	MAE	MSLE
XtremeGradBoosting	0.185534	0.326580	0.079107
Rand_Forest	0.224834	0.377911	0.097590
LinearRgression	0.231455	0.373122	0.099485
SimpleExpSmoothing	0.253642	0.447524	0.118435
Prophet	0.309907	0.349487	0.104881
S.Average	0.311858	0.522596	0.156173
Arima	0.314126	0.525374	0.157654
LSTM-ANN	0.319436	0.454230	0.129314
Xgboost_roll (day by day)	0.337735	0.535137	0.161505
Sarima	0.392443	0.491587	0.176947
DoubleExpSmoothing/Holt	0.403778	0.384031	0.161938
NeuralProphet	0.521565	0.676701	0.260783
MLP-ANN	0.522334	0.564940	0.193966
Rolling or MA	0.582741	0.568148	0.194343

TABLE 1 – Metriques pour différents modèles

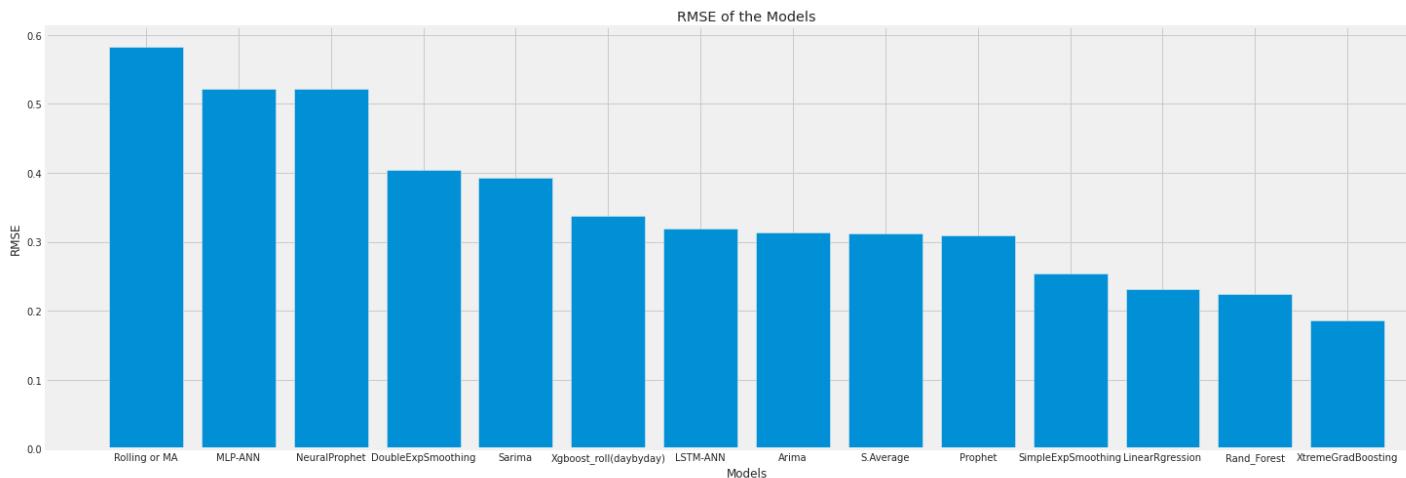


FIGURE 26 – Graph Final pour le RMSE de chaque modèle

La difficulté avec cet approche est le nombre très élevée de produits par magasins. Ainsi, à supposer 260 produits actifs par magasin, nous aurons 260 prédictions par pas de temps futur par magasin et soit 260×30 prédictions par pas de temps futur pour les 30 magasins.

Ce qui représente un nombre élevé et pourrait être algorithmiquement chronophage. Il se pose donc la question à savoir comment minimiser le nombre de prédictions ?

Pour y apporter une réponse motivée, il semble judicieux d'effectuer une analyse en composante principale sur les produits/magasins dans l'optique de réduire le nombre ou la dimension de prévision en sélectionnant uniquement les composantes ou prédictions pertinentes.

En outre, l'ambition ou l'idée de départ est d'entrainer un unique modèle pour prédire les demandes de l'ensemble des produits.

À quoi ça sert d'entraîner un unique modèle pour un ensemble de séries temporelles liées les unes aux autres ? On imagine bien qu'on pourrait faire un modèle pour chaque produit, mais déjà

cela voudrait dire avoir des milliers de modèles différents. Un modèle pour chaque produit, où chaque catégorie et en plus tous les produits n'ont pas forcément le même historique. Un produit qui vient de sortir depuis 2 semaines, ça voudrait dire qu'on ne serait pas trop capable de savoir comment ses ventes/demandes vont évoluer. Alors qu'en fait intuitivement, on peut se dire que l'expérience des autres produits, par exemple des produits de la même catégorie, pourrait informer la prédition de notre nouveau produit. Et c'est là dessus que nous posons les jalons de la stratégie tactico-technique adoptée.

7.2 Modèle atemporel aggrégé

L'objectif dans cette approche atemporelle est de prédire la demande à n'importe quel temps futur, soit avoir comme horizon de prédition l'infini.

Elle permet aussi de rendre les caractéristiques indépendantes. En effet, comme certaines caractéristiques sont fortement corrélées, l'idée est de les décorreler en utilisant la PCA et ne représenter que leurs composantes orthogonales. L'architecture du modèle reste la même, mais les caractéristiques sont remplacées par leurs versions décorrélées.

7.2.1 Stratégie Tactico-technique

La démarche adopter est la suivante :

- Nous créons une matrice d'une ligne par pas de temps(donc le nombre de lignes de la matrice est égale au nombre de pas de temps des données d'entraînement) et une colonne par produit/-magasin (soit environ 230*30 colonnes).
- Effectuer une normalisation ou une standardisation sur cette Matrice, ce qui permet de définir un 1 er espace vectoriel pour les variables de sorties
- Effectuer une Analyse en composante principale sur cette matrice et plus exactement une réduction de dimension, dans l'obtique de réduire le nombre de variables de sorties du modèle(soit le nombre de prédition). Ceci permet de définir un second espace vectoriel sur les variables de sorties.
- Analyser les composantes principales et sélectionner un nombre k de composantes pertinentes pour la prédition.
- Mise en place d'un modèle de prédition g sur les composantes sélectionnées. $g(X_{train}, y_{train})$ où X_{train} représente les variables temporelles + meteo et autres.

Les produits n'ont pas la même popularité, les mêmes ordres de grandeur de ventes/demandes, et on a pas forcément envie que le modèle s'entraîne juste sur les produits qui font le plus de volume de ventes, peut-être ya t'il aussi des choses intéressantes à apprendre sur les produits les moins populaires. Notre Modèle doit donc apprendre de l'ensemble des produits(mise à l'échelle des différents produits entre eux) même ceux qui sont moins populaires(les nouveaux et ceux à venir)

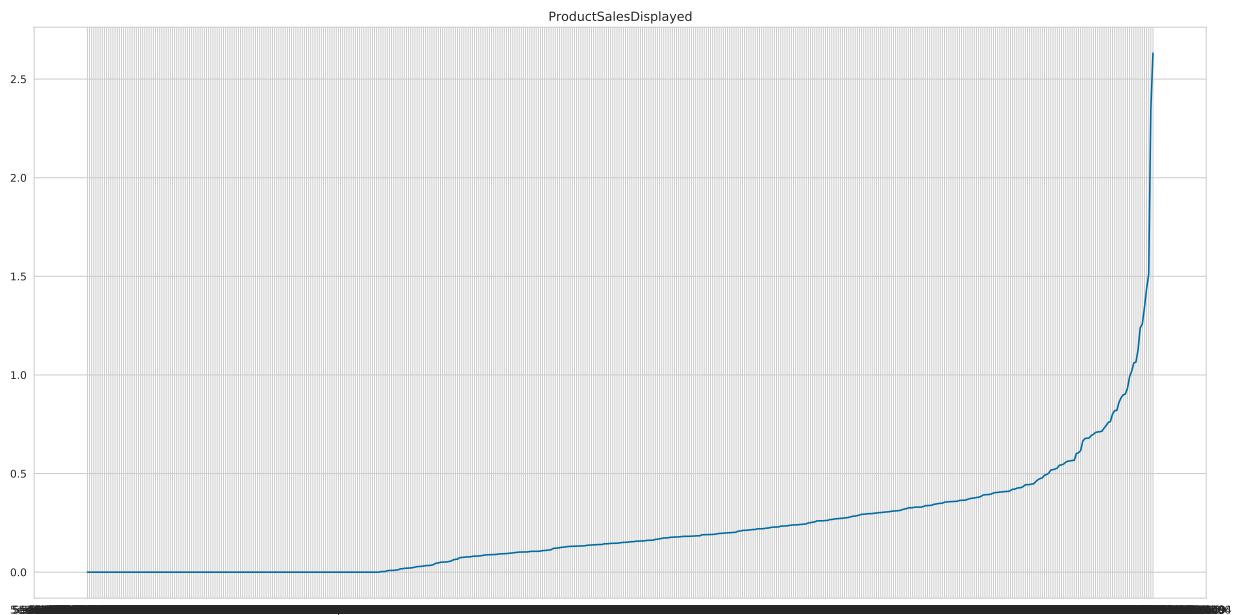


FIGURE 27 – Transformation et Visualisation des ventes des Produits par Magasin

Remarque : La figure ci-dessus permet de sélectionner les produits/magasin qui se vendent peu ou pas, et donc peuvent être supprimé de la gamme des produits proposés dans ces magasins. Enfin, pour la prédition, on procède en deux étapes :

- Prédiction des composantes principales
- Inversion de la PCA , puis transformation Inverse de la Normalisation : Pour ainsi retourner dans l'espace réelle de départ.

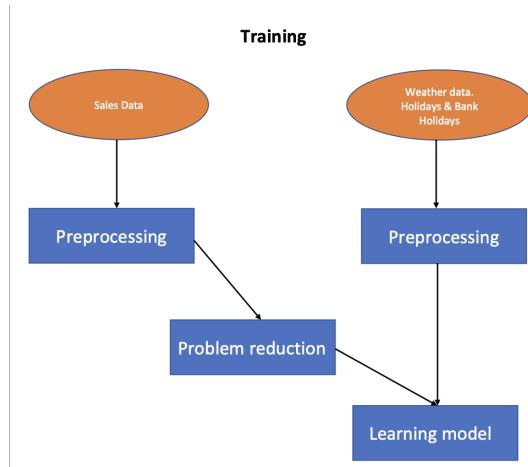


FIGURE 28 – Architecture pour l'entraînement

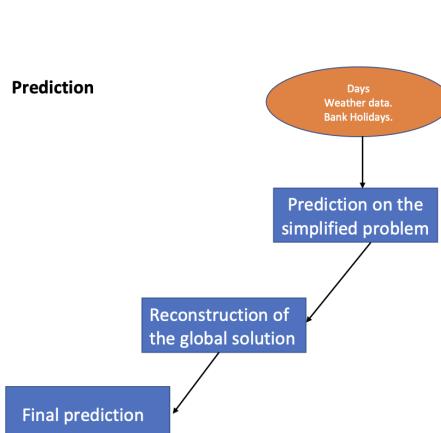


FIGURE 29 – Architecture pour la prédition

7.2.2 Analyse en composante Principale(PCA)

un outil extrêmement puissant de synthèse de l'information, très utile lorsque l'on est en présence d'une somme importante de données quantitatives à traiter et interpréter

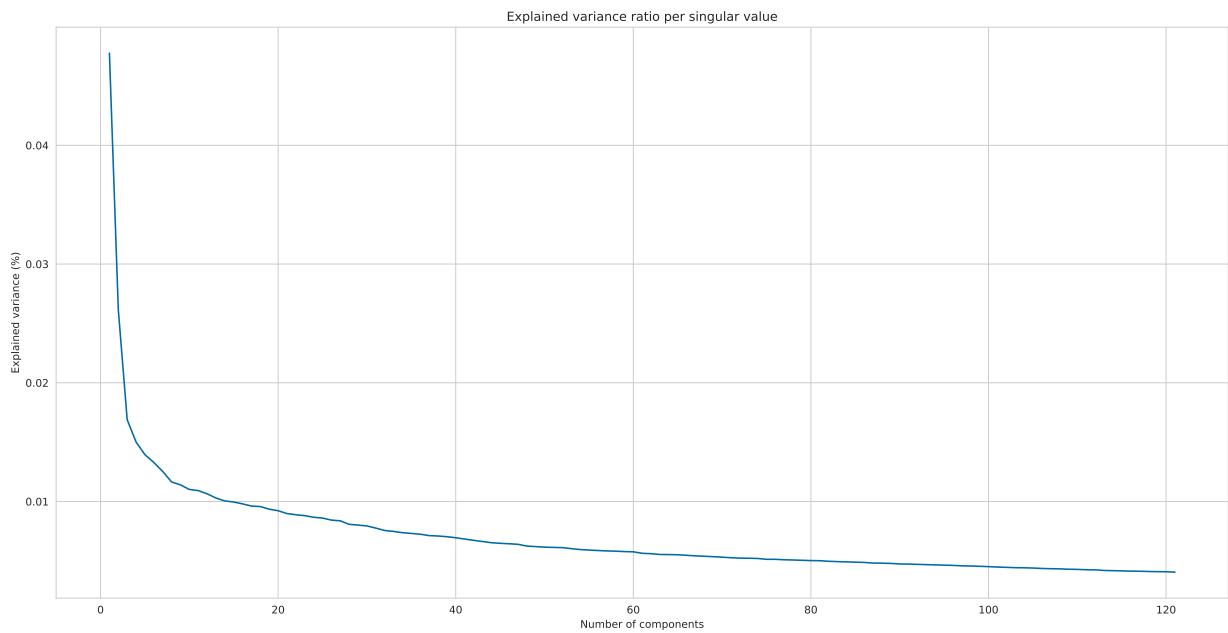


FIGURE 30 – Rapport de variance expliquée par valeur singulière :RMSE par dimension pour PCA

Nous avons présélectionné la dimension de l'espace réduit sur la base de la perte d'information des méthodes de réduction même si cette perte ne représente pas notre objectif final, qui est de prédire les données futures.

Remarque :Stabilité des valeurs propres et non des composantes principales(vecteurs propres)
Résultats sur l'ensemble de données et Interprétation :

Nous avons utilisé et entrainé plusieurs modèles g, dont les métriques calculées à chaque fois sur l'ensemble test sont repertoriées dans le tableau ci-après :

L'approche PCA ou plus généralement SVD(Décomposition en valeur singulière ou réduction de dimension) obtient les meilleurs résultats en utilisant seulement 5 dimensions de prédiction, ce qui réduit la taille du problème de 99,97 %. Par conséquent, la méthode SVD est capable de conserver la plupart des informations pertinentes. Comme la méthode SVD est plus performante que la réduction identité(pas de réduction), elle est capable d'effacer une partie du bruit des données.

Sélection du nombre de dimensions, optimisation des méthodes de réduction :

Dans cette section, ayant réduit le problème,nous proposons une valeur de dimension pour les espaces réduits($k=5$) et comparons les métriques/scores afin d'obtenir le meilleur algorithme, tout en conservant un espace de sortie restreint.

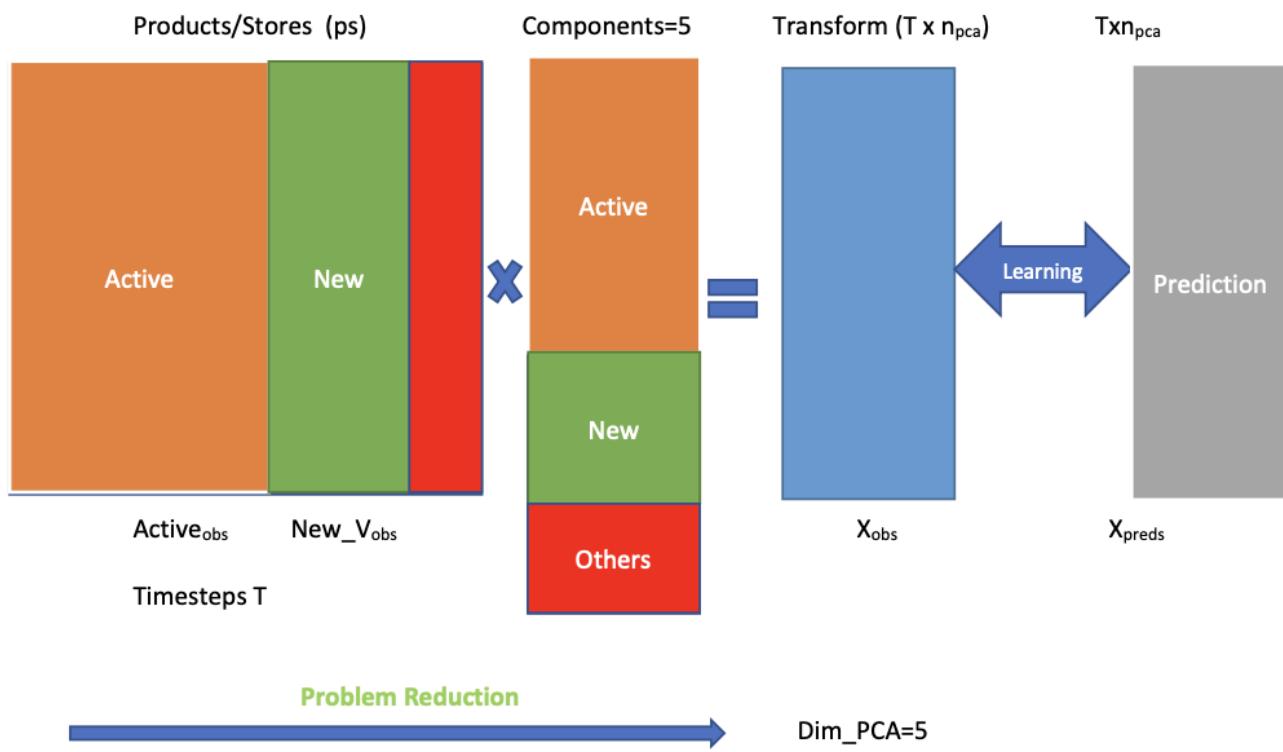


FIGURE 31 – PCA

Dans la suite, nous entraînerons deux Modèles Temporels (Sarima et LSTM) sur ces 5 composantes principales, et les transformations inverses(sur ces composantes et sur les prédictions de ces modèles) permettront de revenir dans l'espace réel des données.

Model	n _{pca}	MSE	MAE	MSLE
ID-BruteForceRegressor	ID-BruteForceRegressor	3.283227	0.313150	0.090376
perform_pca	5	3.292158	0.307720	0.091061
	10	3.303245	0.311837	0.093646
	20	3.302638	0.312213	0.093729
	50	3.301614	0.312784	0.093730
	100	3.299751	0.313827	0.093640
Rolling or MA	0	1.074999	0.297313	0.091118
PCA-Lin_Reg	5	3.279865	0.307688	0.088697
	10	3.279939	0.308068	0.088887
	20	3.280446	0.308548	0.089087
	50	3.281353	0.310704	0.089605
	100	3.282391	0.312030	0.089974
PCA-Multi_svm	5	3.290382	0.314740	0.091430
	10	3.291870	0.315708	0.091840
	20	3.291181	0.314978	0.091562
	50	3.293195	0.316461	0.092144
	100	3.294142	0.317067	0.092324
PCA-Rand_Foret	5	3.325596	0.308017	0.100648
	10	3.345128	0.314295	0.105260
	20	3.333645	0.311179	0.102520
	50	3.294962	0.310487	0.092101
	100	3.292907	0.309726	0.091788
PCA-XGB	5	3.384669	0.334448	0.113299
	10	3.447832	0.353705	0.122277
	20	3.498592	0.388211	0.129602
	50	3.607245	0.437698	0.144278
	100	3.663767	0.462224	0.151706
Prophet	5	3.319467	0.321958	0.098824
	10	3.321169	0.322717	0.099137
SimExpSmoothing	5	3.184207	0.284111	0.080148
	10	3.203358	0.287643	0.082658
DeepAR	5	3.286413	0.316945	0.089050
	10	3.286413	0.316945	0.089050
Simple_Average	Simple_Average	0.561258	0.430636	0.127516

TABLE 2 – les scores de chaque modèle sur l'ensemble de test

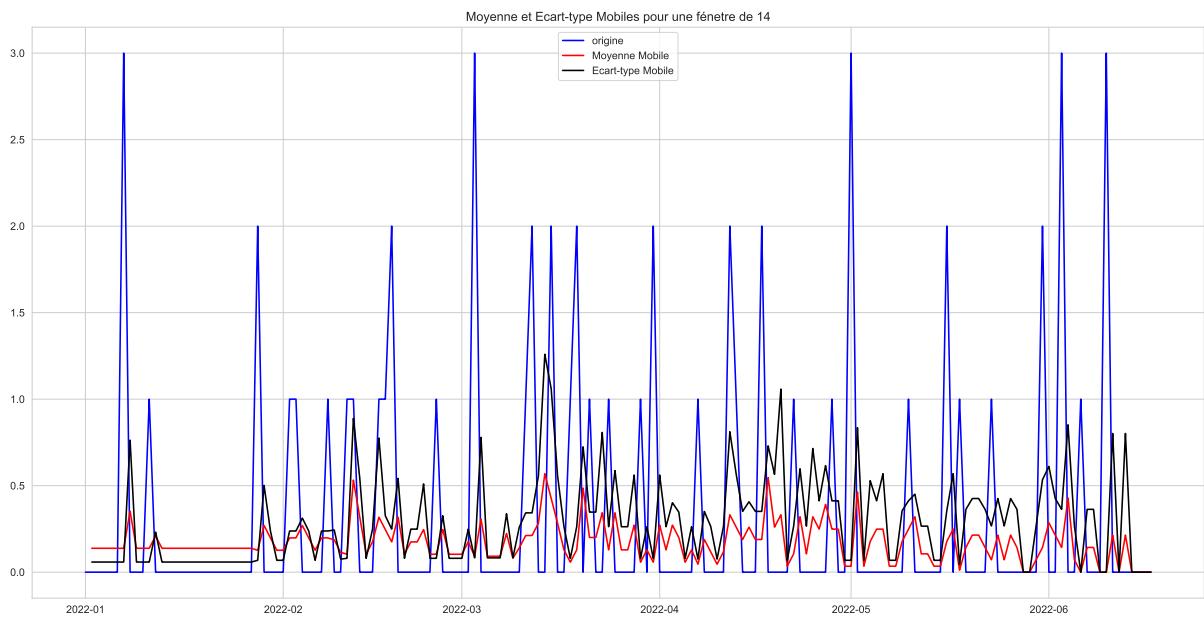


FIGURE 32 – Moyenne Mobile Autoregressive Intégrée

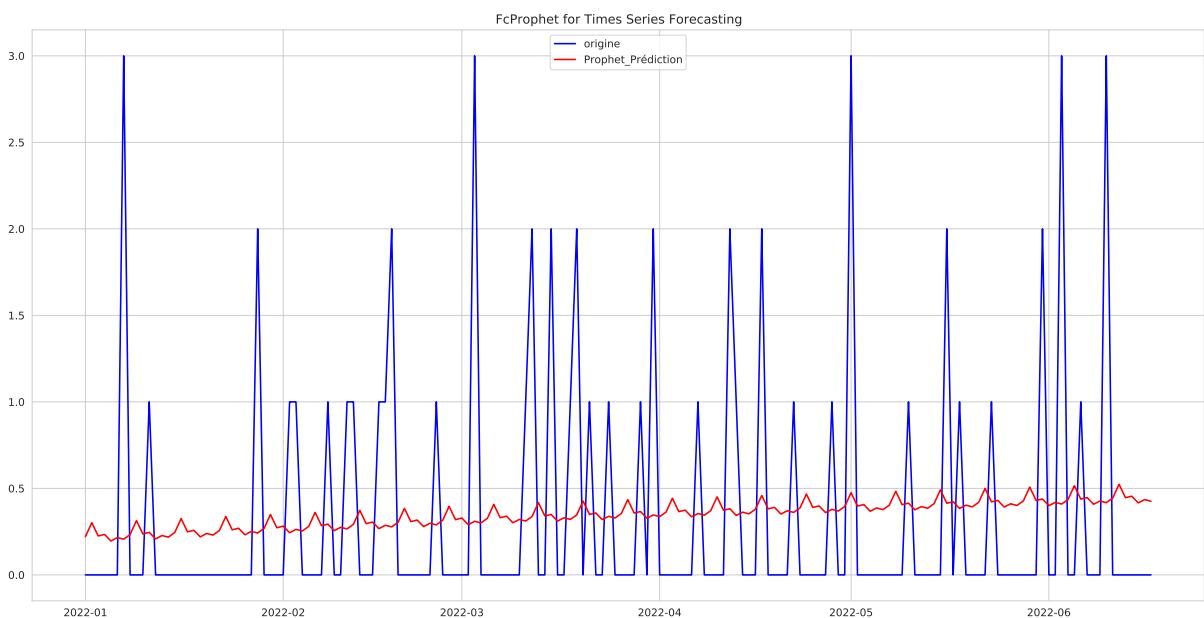


FIGURE 33 – Prophet Après PCA

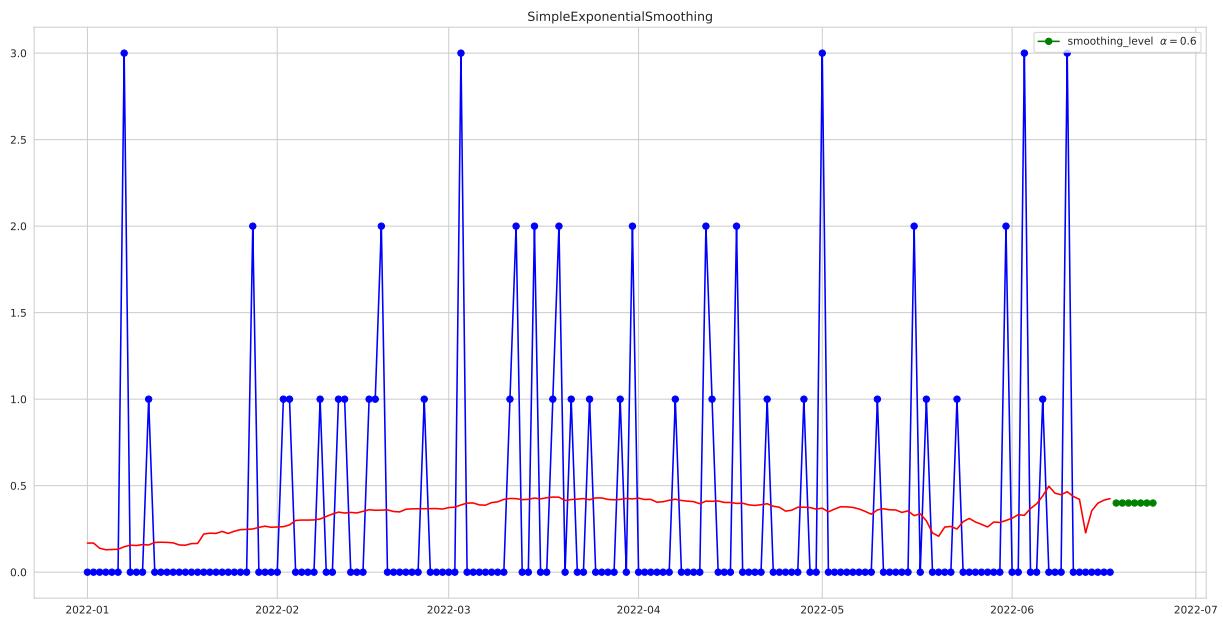


FIGURE 34 – Lissage Exponentielle apres PCA

7.2.3 Autoencodeur :

encoding_dim	MSE	MAE	MSLE	Model
5	3.289457	0.328655	0.091289	rebuild_error
5	3.686014	0.441231	0.139562	RandonForest
5	3.711829	0.446020	0.141516	SVM
5	3.713980	0.446438	0.141705	LinearReg
5	3.290382	0.314740	0.091430	PCA-Multi_svr(ref)

TABLE 3 – les scores de chaque modèle sur l'ensemble de tests pour l'autoencodage

7.2.4 UMAP :

n_components	MSE	MAE	MSLE	Model
5	0.148485	0.170231	0.014829	Rebuild_Error
5	3.569334	0.204718	0.019507	Multi_svm
5	3.575483	0.198051	0.019313	RFRegressor
5	3.577750	0.199393	0.019555	LinRegressor
5	3.290382	0.314740	0.091430	PCA-Multi_svr(ref)

TABLE 4 – les scores de chaque modèle sur l'ensemble de tests pour Umap

7.2.5 Tensor decomposition (TCA) :

Dans cette approche, nous présélectionnons le nombre de facteur dans la décomposition sur la base de la perte d'information des méthodes de réduction, en minisant l'erreur et le temps de reconstruction.

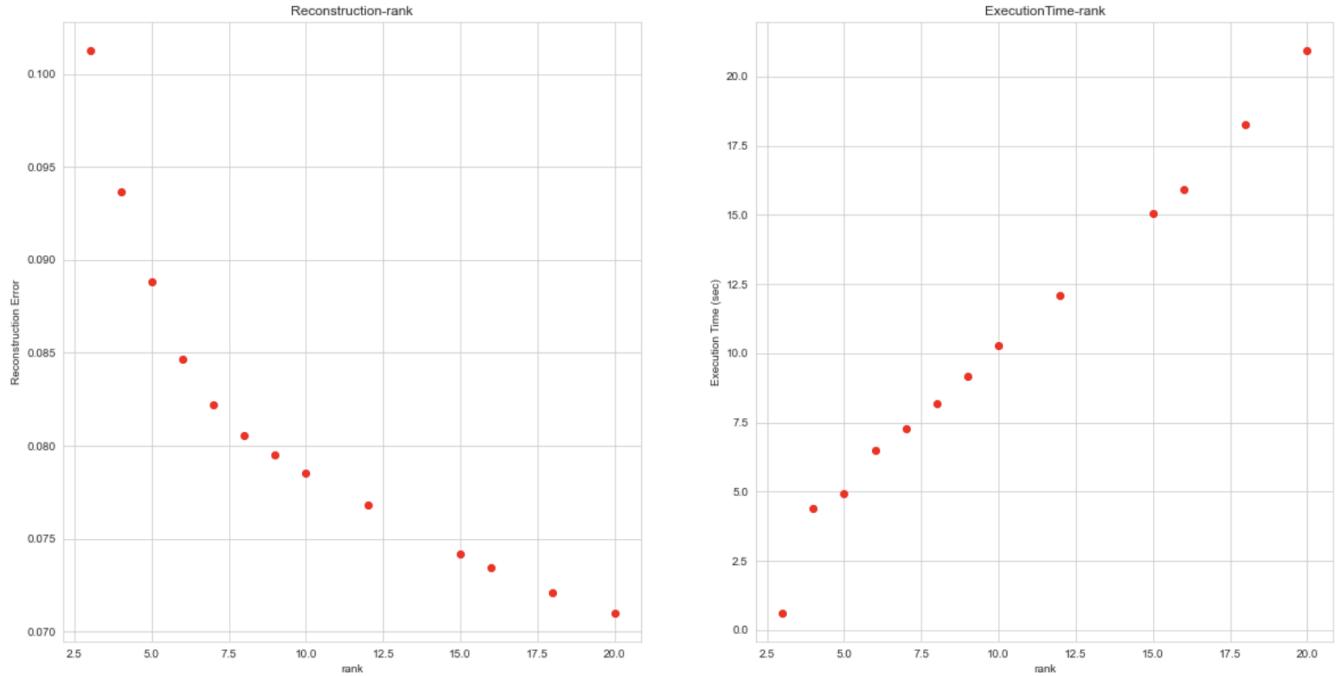


FIGURE 35 – Selection du rang(Rank) de la décomposition

Une fois la décomposition faite, en entrainant plusieurs modèles sur la partie temporelle de la décomposition et en faisant la transformation inverse sur la partie invariante (base des produits et des magasins) pour revenir dans l'espace réelle et calculer les métriques, on obtient le tableau suivant :

n_components or Rank	MSE	MAE	MSLE	R2	Model
20	3.514137	0.119037	0.013091	0.039758	LinRegressor
10	3.514186	0.118359	0.013087	0.140317	RFRegressor
10	3.514243	0.117970	0.013083	0.059182	LinRegressor
20	3.514371	0.121759	0.013201	0.147351	RFRegressor
10	3.515471	0.121298	0.013232	0.157209	Multi_svm
5	3.517385	0.114312	0.013060	0.100096	RFRegressor
5	3.517465	0.114460	0.013063	0.020033	LinRegressor
5	3.517702	0.116028	0.013108	0.043319	Multi_svm
10	3.517926	0.123477	0.013491	0.996290	XgBoost
20	3.518364	0.131510	0.013782	0.165483	Multi_svm
5	3.518946	0.116552	0.013196	0.994357	XgBoost
20	3.522040	0.132536	0.014096	0.996878	XgBoost

TABLE 5 – les scores de chaque modèle sur l'ensemble de test pour l'approche TCA

7.3 feature engineering et Ajout de Covariables

Time features	Weather features
Month (int $\in [1,..,12]$)	Temperature (float)
Day of month (int $\in [1,..,31]$)	Visibility (float)
Day of week (int $\in [1,..,7]$)	Humidity (float)
Week of year(int $\in [1,..,52]$)	Pressure (float)
Holidays (one hot encoder)	Wind (float)
Bank Holidays (one hot encoder)	Rain (float)
	solar energy (float)

TABLE 6 – Feature Engeneering for the Model

À partir des dates calendaires, nous avons pu extraire de nombreuses caractéristiques, qui ont permis d'étudier l'amélioration de la qualité des prédictions pour chaque jour. Cette technique de « feature engineering » a créé les variables suivantes :

- Saisonnalité : Le mois, Le jour du mois (impact de la paye qui tombe en fin de mois), La semaine, Le jour de la semaine, Les jours fériés, Le type de vacances scolaires (hiver, printemps, été, automne, fin d'année), soldes, événements.

Les conditions météorologiques ont une influence sur le commerce. La récupération de ces informations peut se faire via l'interrogation d'API météo(**Visualcrossing Weather Data API Global Forecast History Data**) qui nous retournent :

Un gros volume historique météo depuis janvier 2022 à aujourd'hui pour l'entraînement des modèles. Des prédictions météorologiques de qualité sur une période de 15 jours pour réaliser des prédictions en temps réel.

- Les variables climatiques que nous avons sélectionnées sont :

La température moyenne, la radiation solaire(solarradiation similar to solarenergy) et l'humidité. Les précipitations et la neige La vitesse du vent etc ne sont pas très corrélés à la vente agrégée.

De La pertinence d'ajouter des données météorologiques ?

La météo a une influence sur notre vie de tous les jours : ventes, visites, consommation, etc. Les saisons dictent nos comportements d'achat. Mais, est-ce qu'il s'agit d'une simple saisonnalité ou d'une météo-sensibilité ? Pour savoir si l'intégration de la météo au sein de nos modèles était pertinente, nous avons fait un comparatif.

Le comparatif des modèles s'est fait dans des conditions d'expériences égales

Nos métriques d'évaluation des modèles sont un score global RMSE et le MAE.

Étapes [7] :

- Apprentissage sur l'historique + données calendaires(Jours Fériés,vacances)
- Apprentissage sur l'historique + données météo
- Apprentissage sur l'historique + données calendaires + données météo

Resultats, comparaisons des métriques et commentaire sur la pertinence des coviables

7.3.1 Historique et données calendaires

Bank Holidays	Holidays
Jour de l'an	Vacances de printemps
Fête du Travail	Vacances de Noël
Armistice 1945	
Fête nationale	Vacances d'hiver
Armistice 1918	Vacances d'été
Lundi de Pâques	.
Lundi de Pentecôte	Vacances de la Toussaint
Ascension	
Assomption	
Toussaint	
Noël	

TABLE 7 – One HotEncoder for Holidays

L'ajout de ces variables en One HotEncoder dans le modèle a permis d'obtenir les métriques d'évaluations suivantes où (ref) désigne le meilleur modèle sélectionné sous la contrainte de performance(erreurs) lors de l'entraînement sur les seuls attributs temporelles codées en termes de jour de la semaine, semaine de l'année et mois de l'année :

Dim_PCA	MSE	MAE	MSLE	R2	Model
5	3.290382	0.314740	0.091430	-	PCA-Multi_svr(ref)
10	3.514623	0.118773	0.013078	0.0693	LinRegressor
20	3.515015	0.120870	0.013137	0.05618	LinRegressor
10	3.515095	0.129128	0.013502	-	FbProphet
20	3.515142	0.130691	0.013554	-	FbProphet
5	3.515677	0.122796	0.013288	-	FbProphet
10	3.515882	0.127817	0.013438	0.141973	Multi_svm
5	3.516496	0.116049	0.013084	0.082647	LinRegressor
5	3.516890	0.122269	0.013323	0.134293	Multi_svm
10	3.517372	0.113714	0.013065	-	DeepAR
5	3.517372	0.113714	0.013065	-	DeepAR
20	3.517372	0.113714	0.013065	-	DeepAR
20	3.521144	0.136193	0.013977	0.137594	Multi_svm
5	3.531248	0.138041	0.014910	0.82385	RFRegressor
10	3.532754	0.143093	0.015247	0.823495	RFRegressor
20	3.533382	0.145817	0.015291	0.825725	RFRegressor
5	3.567225	0.154116	0.018514	0.999191	XgBoost
10	3.582604	0.171219	0.020475	0.998386	XgBoost
20	3.587888	0.182791	0.021528	0.996071	XgBoost

TABLE 8 – Apprentissage sur l'historique + données calendaires

7.3.2 Historique et données Metéo

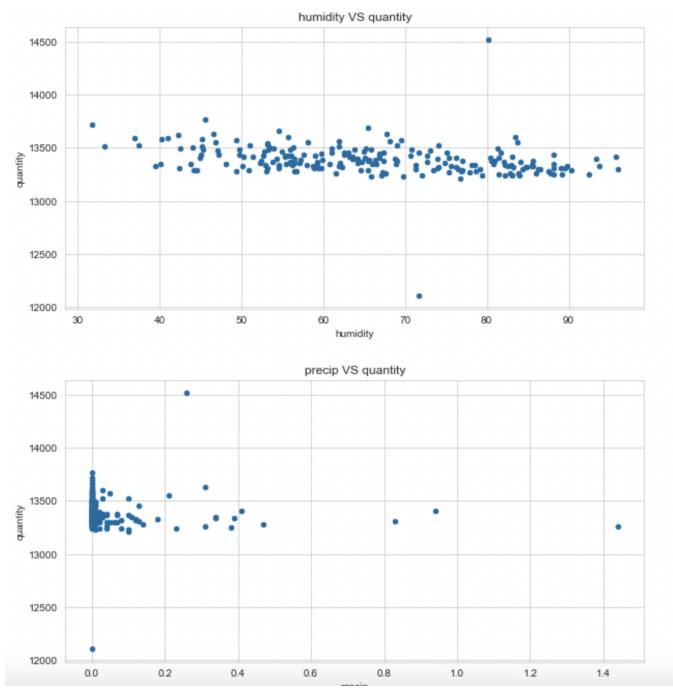


FIGURE 36 – scatter chart 1

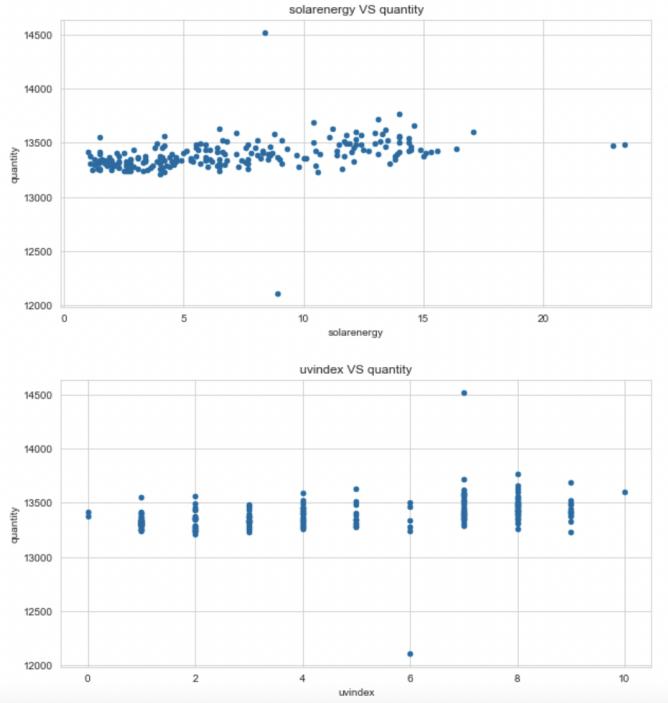


FIGURE 37 – scatter chart2

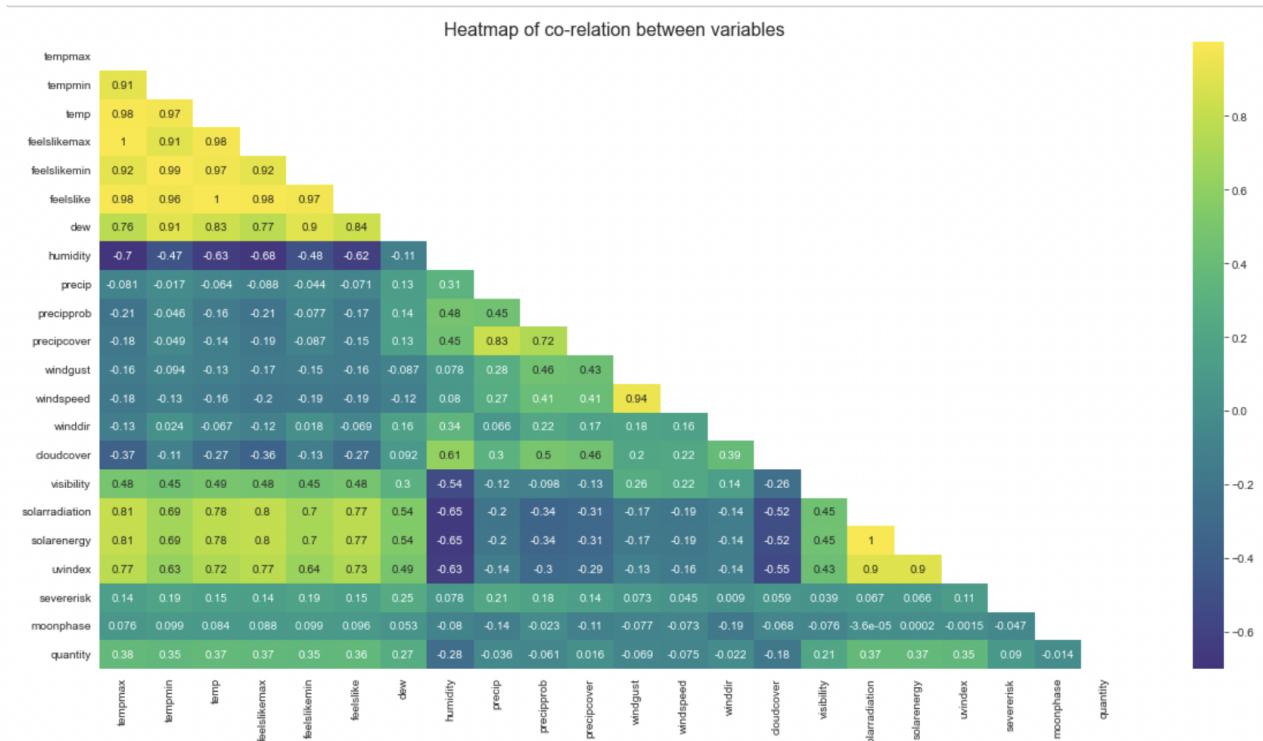


FIGURE 38 – Matrice triangulaire de corrélation

Les nuages de points et la matrice de correlation permettent de selectionner comme covariables météo pour l'entraînement : **La température moyenne, l'humidité et la radiation solaire**

Dim_PCA	MSE	MAE	MSLE	R2	Model
5	3.290382	0.314740	0.091430	-	PCA-Multi_svr(ref)
10	3.513641	0.121377	0.013131	0.078059	LinRegressor
20	3.513720	0.123404	0.013186	0.063333	LinRegressor
10	3.514873	0.128693	0.013459	-	FbProphet
20	3.515081	0.130497	0.013504	-	FbProphet
5	3.515345	0.117921	0.013098	0.079295	LinRegressor
5	3.515409	0.121692	0.013230	-	FbProphet
10	3.516180	0.128908	0.013539	0.169535	Multi_svm
5	3.517071	0.122878	0.013339	0.154586	Multi_svm
20	3.517372	0.113714	0.013065	-	DeepAR
5	3.517372	0.113714	0.013065	-	DeepAR
10	3.517372	0.113714	0.013065	-	DeepAR
20	3.523275	0.142737	0.014464	0.183365	Multi_svm
20	3.528142	0.144323	0.014987	0.842615	RFRegressor
10	3.528417	0.142055	0.014927	0.845611	RFRegressor
5	3.528431	0.138421	0.014730	0.855985	RFRegressor
5	3.563739	0.158639	0.018503	0.999998	XgBoost
10	3.588477	0.175396	0.020941	0.999996	XgBoost
20	3.597364	0.187706	0.022132	0.999994	XgBoost

TABLE 9 – Apprentissage sur l'historique + données météo

7.3.3 Historique, données calendaires et données Météorologiques

Dim_PCA	MSE	MAE	MSLE	R2	Model
5	3.290382	0.314740	0.091430	-	PCA-Multi_svr(ref)
10	3.513920	0.121082	0.013113	0.109254	LinRegressor
10	3.514292	0.127149	0.013355	-	FbProphet
20	3.514427	0.122327	0.013151	0.074138	LinRegressor
20	3.514752	0.129266	0.013428	-	FbProphet
5	3.515601	0.121079	0.013203	-	FbProphet
5	3.515692	0.121092	0.013211	0.192679	Multi_svm
5	3.516091	0.117120	0.013093	0.114473	LinRegressor
10	3.516844	0.130035	0.013584	0.178328	Multi_svm
10	3.517372	0.113714	0.013065	-	DeepAR
20	3.517372	0.113714	0.013065	-	DeepAR
5	3.517372	0.113714	0.013065	-	DeepAR
20	3.521401	0.139360	0.014197	0.193208	Multi_svm
5	3.526220	0.136105	0.014449	0.840102	RFRegressor
10	3.528084	0.143299	0.014924	0.862275	RFRegressor
20	3.528657	0.143600	0.015033	0.833374	RFRegressor
5	3.548969	0.153167	0.017000	0.999999	XgBoost
10	3.579986	0.170982	0.020363	0.999996	XgBoost
20	3.592487	0.185754	0.022095	0.999991	XgBoost

TABLE 10 – Apprentissage sur l'historique + données calendaires + données météo

7.3.4 Conclusion de l'étude de l'intégration des données calendriers et météorologiques.

L'ajout de covariables calendaires et météo semble détériorer les métriques obtenues avec les seules attributs temporels. Est ce à dire que l'ajout de ces covariables n'est pas pertinente ? La réponse est la négative dans le cas général, car en effet, on peut évoquer ici la :

- **Nécessité d'une historique de vente plus large** (1 et 2 ans de données) pour mieux apprécier l'influence des données calendaires(jours Fériées, vacances scolaires) et des covariables Météorologique(Température, humidité,Energie solaire).
- **Nécessité de faire une analyse plus fine** car peut-être faut il aller par produit (évolution météo non capturée par la PCA)

8 Selection et construction de l'interface de l'algorithme de Prediction

8.1 Sélection du Modèle

Dans le tableau ci-dessous, nous regroupons pour un choix de dimension de réduction égale à 5, les performances en termes de métriques d'évaluation de différents algorithmes d'apprentissages.

Model	Métriques d'évaluation Dimension_PCA	MSE	MAE	MSLE
ID-BruteForceRegressor		3.283227	0.313150	0.090376
perform_pca	5	3.292158	0.307720	0.091061
PCA-Lin_Reg	5	3.279865	0.307688	0.088697
	10	3.279939	0.308068	0.088887
PCA-Multi_svm	5	3.290382	0.314740	0.091430
	10	3.291870	0.315708	0.091840
PCA-Rand_Foret	5	3.325596	0.308017	0.100648
	10	3.345128	0.314295	0.105260
PCA-XGB	5	3.384669	0.334448	0.113299
	10	3.447832	0.353705	0.122277
Prophet(Facebook)	5	3.319467	0.321958	0.098824
	10	3.321169	0.322717	0.099137
SimExpSmoothing	5	3.184207	0.284111	0.080148
	10	3.203358	0.287643	0.082658
DeepAR(Amazon)	5	3.286413	0.316945	0.089050
	10	3.286413	0.316945	0.089050

TABLE 11 – les scores de chaque modèle sur l'ensemble de test

Détails sur le choix du SVM Ajout éventuel des features calendaires et Meteo

8.2 Auto-correction en temps réel : Mise à jour pour les produits moins populaires.

Une ouverture vers une auto-correction en temps réel, C'est-à-dire que le modèle pourra ré-évaluer automatiquement ses projections/prédiction/prévisions après les avoir comparer à la réalité pour intégrer les écarts éventuellement constatés.

Il s'agit d'un **Apprentissage continu** qui répond à la question : comment mettre à jour nos prédictions continuellement pour s'adapter à l'évolution de la demande ?

8.2.1 Cas des Nouveaux Produits par Magassim

Tous les produits n'ont pas les mêmes ordres de ventes/demandes, et on a pas forcément envie que le modèle s'entraîne juste sur les produits qui font le plus de volume de ventes, peut-être ya t'il aussi des choses intéressantes à apprendre sur les produits les moins populaires(les nouveaux). Toutefois les composantes de ces derniers sont mises à jour sur la base de celles ayant un plus grand historique, c'est à dire en utilisant l'information appris par le modèle des produits d'historique plus large. cette mise à jour consiste à trouver par une méthode linéaire(Pour la plupart des méthodes) le meilleur ensemble de paramètres pour exprimer l'espérance de la demande du produit comme une fonction linéaire des comportements ou informations extraits de l'historique des plus anciens(produits actifs).

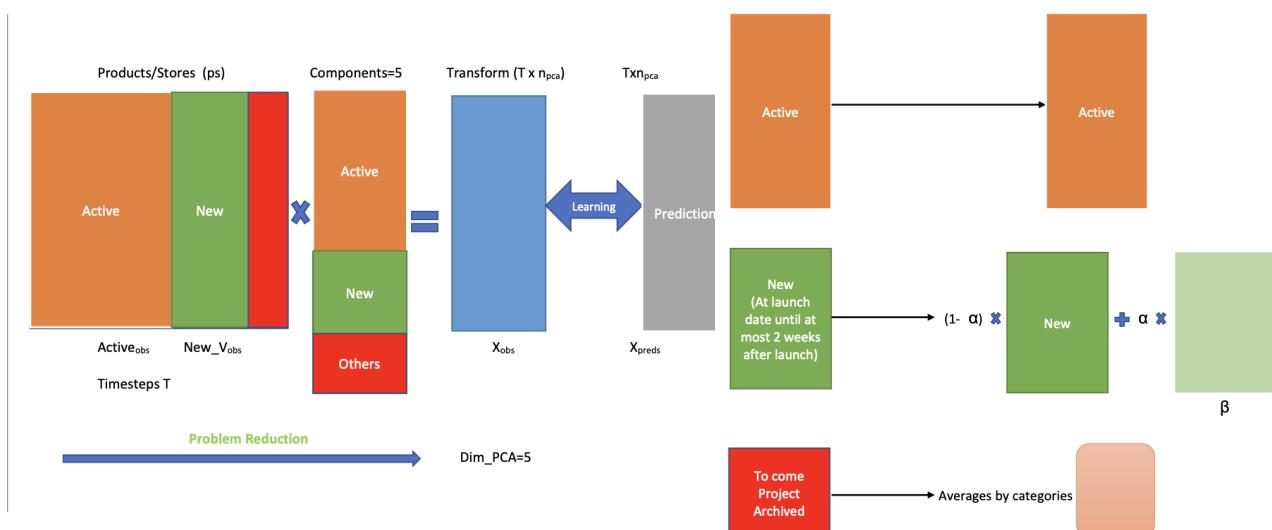


FIGURE 39 – Estimation Linéaire de l'Expérience de la demande

FIGURE 40 – Mise à jour sur les nouveaux produits

$$\hat{\beta} = \underset{\beta}{\operatorname{arg\,min}} \underset{X \in \{X_{obs}, X_{preds}\}}{\|X \times \beta - New_V_{obs}\|^2} \quad (14)$$

8.2.2 Cas des Produits en "to come" et en "projects"

Pour les produits à venir, puisque jusque là précisément, on ne possède aucune historique de vente, la méthode intuitive(mais non naïve) que nous avons adopté consiste à remplacer les composantes principales de ces produits à venir par la moyenne de ceux de la même catégorie. L'idée de base est la même et consiste à dire que l'information sur l'évolution de la demande de la catégorie pourrait servir pour estimer voire avoir une ordre de grandeur sur le comportement de ces produits en Projets.

Peut t'on faire mieux?

La réponse est la positive, mais ne sera détaillé ici qu'ultérieurement car fait l'objet d'un autre travail.

8.3 Ajout de Nouveaux Magasins :

Le nombre de Magasin évolue dans le temps, par l'ajout nouveaux magasins et voire même la suppression de certains.

Une telle démarche permet un Apprentissage continu : En effet, pour les nouveaux produits/magasin, il suffit d'apprendre des facteurs des composantes principales $g(X_{train2}, y_{train2})$, puis appliquer les transformations inverses.

La partie réduction du modèle rend l'ajout d'un nouveau Magasin assez facile et est capable de prédire son comportement avec une petite quantité d'informations. L'architecture du modèle modélise l'attente de ces principaux comportements, et la fonction de réduction inverse reconstruit à partir de ses comportements l'attente du Magasin. Cette fonction de réduction inverse est pour ce travail, déduite de l'inversion de la méthode de réduction.

Ajouter un nouveau Magasin équivaut donc à trouver pour la plupart des méthodes (linéaires) le meilleur ensemble de paramètres pour exprimer l'espérance de la demande de chacun des éventuels produits de ce Magasin comme une fonction linéaire des comportements extraits des autres Magasins(Du même cluster par exemple)

8.3.1 Comparatif des performances de algorithmes



FIGURE 41 – metriques par algorithme

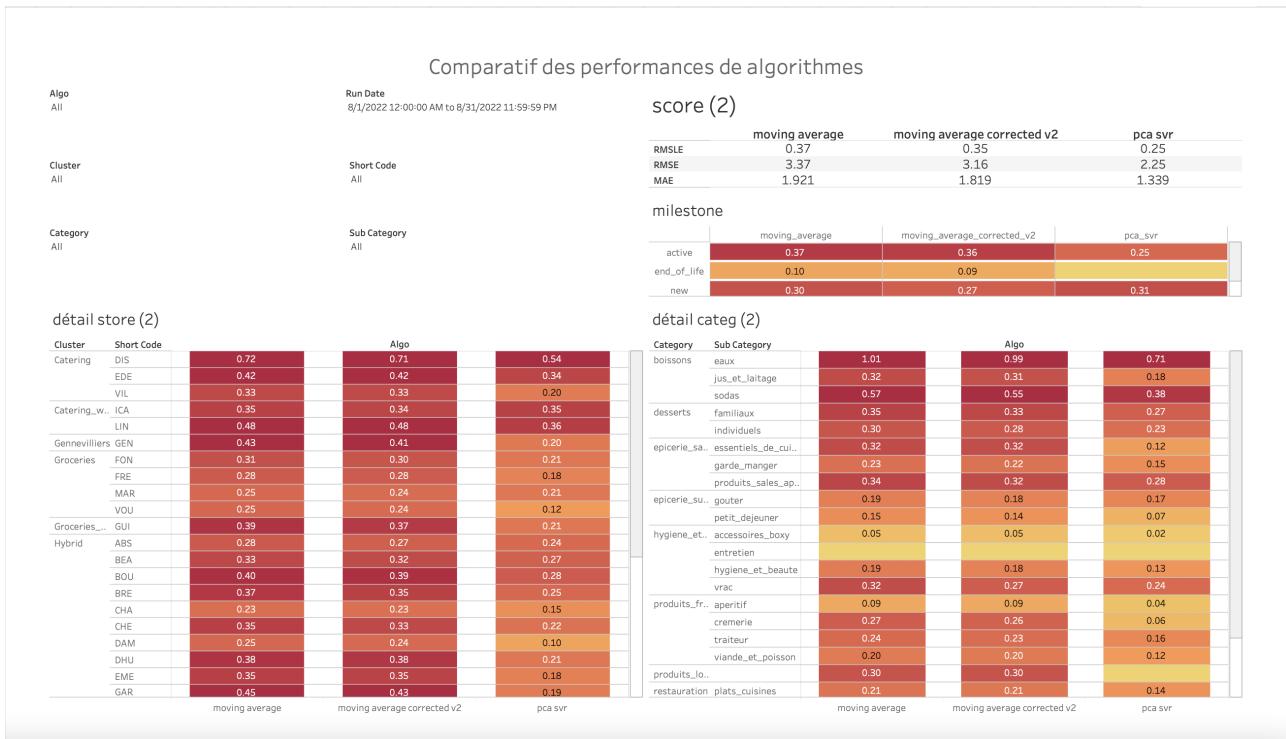


FIGURE 42 – Détail des performances des algos de prediciton de la demande

9 Déroulement du stage : (MIE)

9.1 L'entité d'accueil

Lancée en 2018, Par David Gabai et Tom Hayat, Boxy est une startup tricolore spécialisé dans le retail qui reinvente le commerce de proximité avec un concept de supérettes connectées et ouvertes.

9.1.1 Secteur d'activité

Boxy est une Industrie du Retail qui veut révolutionner le commerce de proximité avec un concept inédit et sans compromis : Elle offre des services d'hyper-proximité de commerce locale en France en alimentant depuis son entrepot environ 35 magasins autonomes de produits aux catégories diverses. Elle simplifie le quotidien de sa clientèle grâce à sa technologie innovante : 100% automatisé, sans caisse, proposant 300 produits essentiels à des prix accessibles, boxy en faisant gagner du temps, de l'argent, et libérer des contraintes habituelles des courses ! Une supérette ouverte 24/7 - sans passage en caisse pour ainsi mettre fin aux portes closes et à la queue à la caisse. un système connecté de capteurs et de caméras détecte automatiquement et en temps réel les produits pris et/ou que reposer sur les étagères par le client, sans avoir besoin de les scanner.

9.1.2 Situation économique, compétitive et stratégie

Boxy est née d'un constat : Dès que l'on sort des centres urbains, l'offre de proximité alimentaire devient rare et souvent très chère, voire carrément inexistante. Ainsi, pour Boxy, la technologie, loin d'être gadget, doit servir de solution à des problématiques réelles.

Boxy a fait une levée de fonds de 25M€ pour accélérer son développement en France avec un objectif de 60 boxy en Ile-de-France fin 2022 et avec pour stratégie une boxy toutes les deux semaines, avant de se lancer dans le reste de la France, puis en Europe.

9.1.3 Organisation, fonctionnement et questions éthiques

9.2 La mission

9.2.1 Rôle

le stagiaire devra d'une part mettre en place un modèle de prédiction de la demande des produits boxy et d'autre part mettre en place une infrastructure pour servir les applications ayant besoin de cette prédiction

9.2.2 Réalisations et intégration dans l'organisation

9.3 Les enseignements

9.3.1 Analyse de l'adéquation entre les constats réalisés sur l'organisation et son secteur, et les aspirations personnelles/professionnelles.

9.3.2 Enseignements personnels (softskills) : qualités et axes de progrès

9.3.3 Impact sur les projets scolaires et professionnels.

10 Conclusion et Perspectives

Dans ce travail, nous avons essayé de construire un modèle de prédiction de la demande espérée de chaque produits par Magasin. Au regard du nombre très élevé et croissant des produits par magasin, la taille du problème était à priori très grande. Le modèle a été conçu pour minimiser la complexité du problème en maximisant sa précision. Ainsi, le nombre de produits par magasin (environ dix mille en dimension) a été réduit à cinq composantes ou cinq dimensions sans perte de l'essentiel de l'information. Cette réduction a été utilisée pour construire un modèle prédictif, pour prévoir la demande espérée de chaque produits par magasin.

Mettre à jour le modèle pour les produits les moins populaires (nouveaux et en projets) et ajouter de nouveaux magasins sont faciles dans une telle architecture de façon à ce que le modèle puisse considérer les changements à court terme sur les données.

Dans l'optique d'améliorer la précision du service de demande, nous avons étudié la pertinence de l'intégration des données calendaires et météorologiques dans l'entraînement. Ceci se conclut par la nécessité d'une historique plus large ou d'une analyse plus fine au niveau de chaque magasin pour observer la pertinence de ces covariables sur la prédiction.

En outre, pour renforcer l'indépendance entre les produits et les magasins, il a été judicieux d'opter pour une nouvelle représentation et réduction des données via l'approche de décomposition tensorielle à trois modes (produits, magasins et les pas de temps) et de rang cinq (nombre de termes dans la décomposition) : Le comportement sur les métriques d'évaluation restent pertinent pour la construction ultérieure d'une interface de prédiction pour cette approche afin de servir les applications ayant besoin des prédictions de demandes des produits.

Aussi, en ce qui concerne l'algorithme prédictif dans l'espace réduit, nous nous donnons comme perspectives de mettre au point une architecture avec DeepAR d'Amazon et/ou Prophet de Facebook, qui sont deux algorithmes particulièrement intéressants en termes de performance (Table 2 page 29) et en même temps facilite l'intégration des événements spéciaux, des covariables météo et l'information métier.

11 Références et Bibliographie

Références

- [1] Lead Data Scientist FARUKH HASHMI. *Predicting stock prices using Deep Learning LSTM model in Python*. <https://thinkingneuron.com>. 2022.
- [2] Ian GOODFELLOW, Yoshua BENGIO et Aaron COURVILLE. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [3] Facebook DataScientists PROPHET DOCUMENTATION. *Forecasting at scale*. <https://facebook.github.io/prophet/>. 2018.
- [4] Pierre HULOT et al. “Towards Station-Level Demand Prediction for Effective Rebalancing in Bike-Sharing Systems (Masters thesis, École Polytechnique de Montréal)”. In : (oct. 2018). URL : <https://publications.polymtl.ca/3160/>.
- [5] PhD JASON BROWNLEE. *Deep Learning for Time Series Forecasting Predict the Future with MLPs, CNNs and LSTMs in Python*. <https://machinelearningmastery.com/start-here/#deeplearning>. Machine Learning Mastery, 2022.
- [6] L. MCINNES, J. HEALY et J. MELVILLE. “UMAP : Uniform Manifold Approximation and Projection for Dimension Reduction”. In : ArXiv e-prints (fév. 2018). arXiv : 1802.03426 [stat.ML]. URL : <https://arxiv.org/abs/1802.03426>.
- [7] Cédric SIBAUD, Bruno GOUTORBE et Bertrand CHABRIER. *Intelligence Artificielle La Grande Mutation du E-commerce*. https://www.openstudio.fr/app/uploads/2022/01/DGOpenStudio_LivreIAeCommerce_V7_PublicationNumerique.pdf. OpenStudio, decembre 2020.
- [8] Oskar TRIEBE, Nikolay LAPTEV et Ram RAJAGOPAL. “AR-Net : A simple Auto-Regressive Neural Network for time-series”. In : (2019). URL : <https://arxiv.org/abs/1911.12436>.
- [9] Oskar TRIEBE et al. *NeuralProphet : Explainable Forecasting at Scale*. 2021. arXiv : 2111.15397 [cs.LG].
- [10] Machine Learning TUTORIALS. *Step-By-Step Machine Learning Tutorials*. <https://grabngoinfo.com/tutorials/>. Go Info, 2021.
- [11] DataScientist VINCENT VILLET. *Time Series made easy*. <https://github.com/xebia-france/Xebicon19-time-series-made-easy>. 2020.

Annexes

A Code

B Nouvelles Métriques après Filtration des données

Dues aux problèmes d'incertitude et précisions des balances, certains produits ont eu théoriquement des ventes très élevées(plus de 20 par jour dans un magasin précis), ce qui est une aberration dispersant la distribution de nos historiques. Dans cette suite, nous avons filtrer les données de ventes en supprimant ceux de nombres journaliers supérieurs à 20. Il convient de prendre en compte le fait qu'au regard de ces nouvelles métriques, certaines conclusions ci-dessus peuvent être modifiées car s'avèreraient non correctes.

B.1 Modèle sans covariables

Model	index	MSE	MAE	MSLE
perform_pca	5	0.309096	0.282548	0.082090
	10	0.310338	0.283116	0.082552
	20	0.309715	0.283183	0.082605
	50	0.309490	0.283325	0.082624
	100	0.307868	0.283354	0.082276
ID-BruteForceRegressor	ID-BruteForceRegressor	0.312871	0.287143	0.083764
Rolling or MA	0	0.318329	0.278234	0.084590
PCA-Lin_Reg	5	0.310099	0.282536	0.082599
	10	0.310473	0.283191	0.082736
	20	0.310536	0.283742	0.082853
	50	0.310953	0.285117	0.083138
	100	0.311987	0.286040	0.083435
PCA-Multi_svm	5	0.310113	0.282533	0.082597
	10	0.310557	0.283113	0.082713
	20	0.310387	0.283600	0.082833
	50	0.311298	0.285062	0.083226
	100	0.312114	0.286198	0.083485
PCA-Rand_Foret	5	0.353864	0.287974	0.091448
	10	0.360620	0.285678	0.093655
	20	0.354470	0.288212	0.093732
	50	0.364710	0.287438	0.096252
	100	0.360259	0.285016	0.095397
DeepAR	5	0.311518	0.287551	0.081275
	10	0.311518	0.287551	0.081275
Prophet	5	0.377400	0.317105	0.097683
	10	0.374993	0.315030	0.097082
SimExpSmoothing	5	0.277753	0.254890	0.072571
	10	0.286380	0.255994	0.075070
Simple_Average	Simple_Average	0.608092	0.462233	0.145955

n_pca	MSE	MAE	MSLE	R2	Model
5	0.123981	0.092957	0.010695	0.056252	LinRegressor
5	0.125905	0.097961	0.010908	0.140574	Multi_svm
5	0.130120	0.101712	0.011411	0.857452	RFRegressor
5	0.152955	0.116220	0.013379	0.999994	XgBoost
5	0.126853	0.104856	0.011291	null	FbProphet
5	0.125388	0.089980	0.010666	null	DeepAR
10	0.123289	0.094373	0.010696	0.050377	LinRegressor
10	0.125458	0.105249	0.011138	0.150588	Multi_svm
10	0.128645	0.107844	0.011506	0.8355	RFRegressor
10	0.165131	0.133832	0.014841	0.999989	XgBoost
10	0.126116	0.107535	0.011332	null	FbProphet
10	0.125388	0.089980	0.010666	null	DeepAR
20	0.122846	0.096880	0.010726	0.050925	LinRegressor
20	0.128374	0.111017	0.011424	0.153535	Multi_svm
20	0.130346	0.113856	0.011778	0.845929	RFRegressor
20	0.172968	0.143180	0.015668	0.999977	XgBoost
20	0.127199	0.110804	0.011511	null	FbProphet
20	0.125388	0.089980	0.010666	null	DeepAR

B.2 Historique et données calendaires

n_pca	MSE	MAE	MSLE	R2	Model
5	0.124048	0.093052	0.010698	0.035841	LinRegressor
5	0.127099	0.101882	0.011117	0.127991	Multi_svm
5	0.134162	0.111937	0.012049	0.809426	RFRegressor
5	0.161580	0.123045	0.014128	0.998542	XgBoost
5	0.126735	0.102990	0.011173	null	FbProphet
5	0.125388	0.089980	0.010666	null	DeepAR
10	0.122530	0.095800	0.010706	0.040465	LinRegressor
10	0.126338	0.106008	0.011201	0.123305	Multi_svm
10	0.136095	0.116754	0.012324	0.819508	RFRegressor
10	0.176832	0.142225	0.016375	0.99577	XgBoost
10	0.127767	0.111261	0.011530	null	FbProphet
10	0.125388	0.089980	0.010666	null	DeepAR
20	0.122233	0.097085	0.010733	0.036161	LinRegressor
20	0.130937	0.114798	0.011711	0.137786	Multi_svm
20	0.141567	0.123307	0.012909	0.810977	RFRegressor
20	0.212283	0.170735	0.020346	0.991166	XgBoost
20	0.128789	0.113683	0.011651	null	FbProphet
20	0.125388	0.089980	0.010666	null	DeepAR

B.3 Historique et données météorologiques

n_pca	MSE	MAE	MSLE	R2	Model
5	0.124031	0.093053	0.010703	0.05978	LinRegressor
5	0.126291	0.099364	0.010965	0.139188	Multi_svm
5	0.129157	0.103113	0.011386	0.831406	RFRegressor
5	0.159190	0.118036	0.013835	0.999993	XgBoost
5	0.125585	0.102965	0.011139	null	FbProphet
5	0.125388	0.089980	0.010666	null	DeepAR
10	0.122842	0.095098	0.010703	0.058825	LinRegressor
10	0.125736	0.105186	0.011123	0.143946	Multi_svm
10	0.129628	0.108343	0.011570	0.840323	RFRegressor
10	0.165404	0.131852	0.014902	0.999989	XgBoost
10	0.124560	0.105703	0.011175	null	FbProphet
10	0.125388	0.089980	0.010666	null	DeepAR
20	0.122666	0.096952	0.010721	0.055722	LinRegressor
20	0.132039	0.116257	0.011794	0.160478	Multi_svm
20	0.130205	0.112522	0.011760	0.840613	RFRegressor
20	0.171335	0.142676	0.015484	0.999974	XgBoost
20	0.125666	0.109482	0.011366	null	FbProphet
20	0.125388	0.089980	0.010666	null	DeepAR

B.4 Historique, données calendaires et données météo

n_pca	MSE	MAE	MSLE	R2	Model
5	0.124190	0.093735	0.010720	0.090274	LinRegressor
5	0.125872	0.098207	0.010913	0.144071	Multi_svm
5	0.128796	0.101839	0.011315	0.827717	RFRegressor
5	0.156187	0.116272	0.013580	0.999993	XgBoost
5	0.126477	0.102719	0.011133	-	FbProphet
5	0.125388	0.089980	0.010666	-	DeepAR
10	0.122981	0.096287	0.010726	0.081351	LinRegressor
10	0.126416	0.105619	0.011160	0.160017	Multi_svm
10	0.128991	0.110536	0.011607	0.825422	RFRegressor
10	0.159916	0.129049	0.014350	0.999987	XgBoost
10	0.127167	0.111006	0.011469	-	FbProphet
10	0.125388	0.089980	0.010666	-	DeepAR
20	0.122851	0.098640	0.010778	0.068273	LinRegressor
20	0.127924	0.110670	0.011411	0.167685	Multi_svm
20	0.131759	0.113672	0.011899	0.829947	RFRegressor
20	0.165096	0.140773	0.015043	0.999975	XgBoost
20	0.129477	0.113729	0.011625	-	FbProphet
20	0.125388	0.089980	0.010666	-	DeepAR

B.5 Tensor decomposition (TCA) :

n_components or Rank	MSE	MAE	MSLE	R2	Model
5	0.125012	0.092016	0.010671	0.042140	LinRegressor
5	0.126314	0.097975	0.010866	0.218916	Multi_svm
5	0.128928	0.107750	0.011663	0.847737	RFRegressor
5	0.141930	0.112213	0.012493	0.995420	XgBoost
10	0.122612	0.097047	0.010798	0.068304	LinRegressor
10	0.128294	0.107459	0.011435	0.158727	Multi_svm
10	0.130275	0.114967	0.012149	0.275223	RFRegressor
10	0.150653	0.131840	0.014064	0.997114	XgBoost
20	0.122604	0.097343	0.010804	0.044389	LinRegressor
20	0.127486	0.108350	0.011441	0.118805	Multi_svm
20	0.126786	0.110497	0.011617	0.171156	RFRegressor
20	0.158343	0.137990	0.015258	0.996485	XgBoost