Prédiction des ventes d'un produit Walmart

Statistique pour la prévision

Rapport de projet

Janvier 2023

Haythem Boucharif
Ferdinand Genans Boiteux

Table des matières

1 Introduction

La prévision des ventes constitue un élément clé de la gestion stratégique d'un magasin, permettant d'anticiper les besoins en termes de ressources et de planifier les activités de manière à optimiser les résultats financiers. En effet, en s'appuyant sur des prévisions précises concernant l'évolution des ventes à venir, le magasin peut s'assurer de disposer de suffisamment de stock pour satisfaire la demande de ses clients, ce qui contribue à minimiser les risques de rupture de stock et de perte de chiffre d'affaires. Par ailleurs, la prévision des ventes peut être utilisée pour établir un budget et fixer des objectifs.
Dans note projet, nous proposons une étude statistique pour la prévision des ventes quotidiennes d'un produit Walmart. Notre but est de prévoir la vente de ce produit sur une période de 28 jours (25-04 2016 au 22-05 2016) en minimisant l'erreur quadratique (pénalisant fortement les grosses erreurs).
Nous nous sommes appuyés sur différentes variables, notamment le prix de l'article, ses ventes passés et le calendrier pour essayer d'avoir une bonne prévision tout en gardant de l'interprétabilité.
Remarque : Tous les codes, notebooks et jeux de données sont disponibles sur Github [1], dont une version aggrégée avec tout le code d'un seul coup.

2 Ingénierie des données

2.1 Création d'un jeu de données

Pour notre projet de statistique, il a fallu chercher un dataset convenable pour une étude statistique pour la prévision, comportant des variables exogènes.
Notre recherche nous à amener vers la compétition m5 de Kaggle [2] qui comporte des données sur plusieurs magasins Wallmart et une centaine de produits différents sur une période de 5 ans. Après avoir analyser ce jeu de données, nous avons décidé de réduire de manière drastique nos données et le but de la prévision pour ce concentrer sur les ventes d'un unique produit d'un des magasins du Wisconsin. Ce choix était motivé par plusieurs critères, notamment la volonté de se différencier complètement de la compétition m5, passer plus de temps sur l'exploitation statistique des variables exogènes et de rajouter des variables météorologiques plus simplement.

Finalement, nous nous sommes concentrés sur le produit alimentaire n°90 du magasin "Wisconsin_3". L'absence d'informations quant à la nature de notre produit alimentaire ne facilite pas la tâche, n'ayant aucun à priori sur l'influence de la saison et des mois sur ce dernier. Le choix du produit s'est fait en regardant le nombre total de ventes de notre produit, ne voulant pas que celui-ci ne soit pas trop faible pour pouvoir l'étudier.

De plus, nous avons rajouté des variables météorologiques [3] sur les 5 années que nous avions (2011-2016). Intuitivement, nous pensons que la météo peut influencer les ventes d'un magasin, que ce soit de manière positive ou négative.

1. https://github.com/Ferdinand-Genans/Walmart_prevision
2. https://www.kaggle.com/c/m5-forecasting-accuracy
3. https://www.visualcrossing.com/weather/weather-data-services

2.2 Imputation des données manquantes

Bien que l'imputation de données pourrait être vu comme un sujet de prévison à part entière étant donnée la quantité non négligeable de données manquantes, nous avons utilisé la librairie MICE pour cette tache et nous concentrer plus rapidement sur notre sujet de prévision.
La méthode MICE (Multiple Imputation by Chained Equations) est une approche d'imputation permettant de traiter de manière efficace les jeux de données ayant des valeurs manquantes réparties de manière aléatoire et comportant plusieurs variables pouvant être utilisées pour effectuer l'imputation.
L'imputation de données MICE est une technique qui vise à remplacer les valeurs manquantes dans un jeu de données en utilisant une série de modèles de régression. Ces modèles sont entraînés sur les observations complètes de l'ensemble de données et sont utilisés pour prédire les valeurs manquantes de chaque variable. L'algorithme répète cette opération itérativement jusqu'à ce qu'il atteigne une convergence, c'est-à-dire un état où les imputations effectuées dans chaque itération sont suffisamment proches les unes des autres. L'un des moyens courants de mesurer cette convergence est le critère de Gelman-Rubin. Lorsque cet écart est suffisamment faible, l'algorithme est considéré comme convergé. Une fois que les valeurs manquantes ont été imputées dans chaque jeu de données, elles sont combinées en utilisant une moyenne pondérée pour obtenir une unique estimation.
Cette méthode d'imputation est particulièrement utile lorsque les valeurs manquantes sont réparties de manière aléatoire dans le jeu de données et qu'il y a plusieurs variables qui peuvent être utilisées pour prédire les valeurs manquante, ce qui est bien notre cas .
La librairie Python que nous avons utilisée pour appliquer la méthode MICE est 'fancycompute' [4].

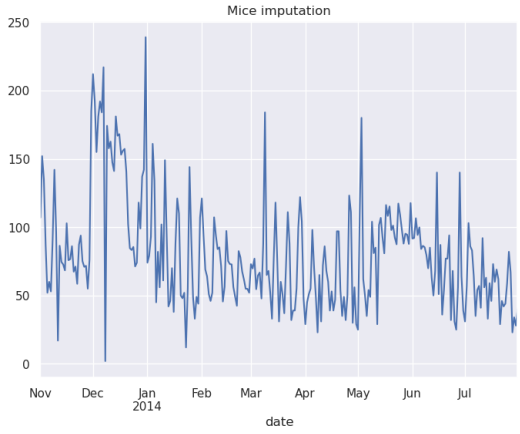

FIGURE 1 – Imputation par MICE

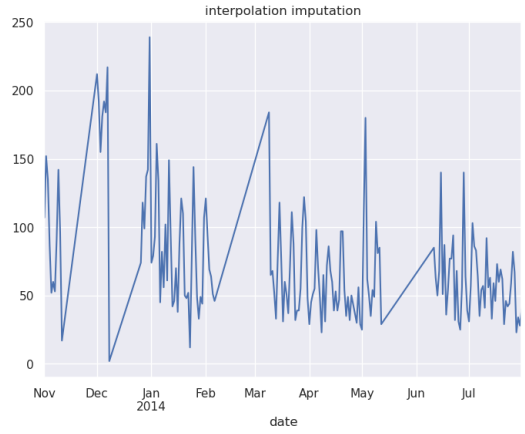

FIGURE 2 – Interpolation linéaire

La technique MICE semble meilleur qu'une interpolation linéaire normale, respectant mieux les fluctuations de prix notamment en fonction du jour de la semaine.

2.3 Division des données

Dans un premier temps, nous avons divisé nos données en un jeu d'entraînement et un jeu de validation. Le jeu de validation est simplement les 28 jours que nous devons prédire. Ce

4. `https://pypi.org/project/fancyimpute/`

dataset ne sera jamais exploré pendant notre étude, il servira juste de comparaison finale avec un benchmark (la prédiction en moyenne).
Ensuite, nous avons divisé nos données entraînement en gardant les 4 premiers mois de 2016 (juste avant les données validations) pour l'évaluation de nos modèles pendant notre étude. Ce choix est motivé par la proximité temporelle de ces données avec celles de validation et leur large spectre temporelle. De plus, ces données ne comportaient aucune valeur manquante.

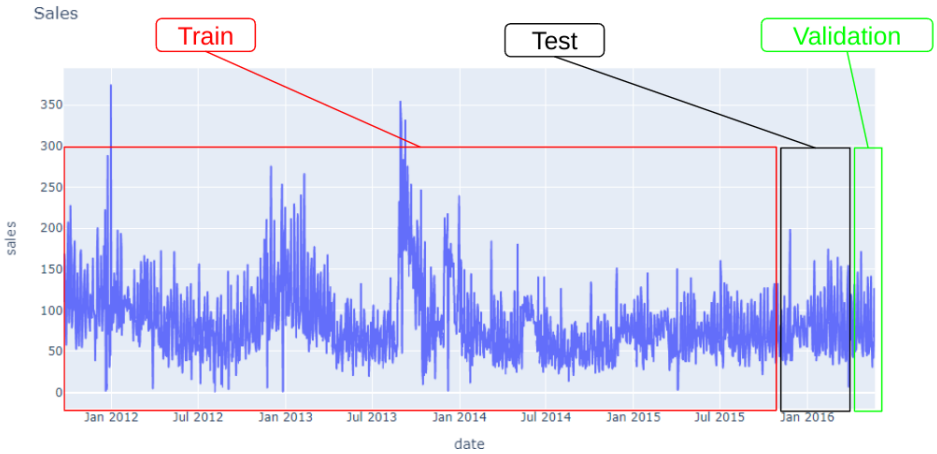

FIGURE 3 – Illustration du découpage des données

3 Exploration et analyse des données

3.1 Présentation des données

Pour pouvoir mieux comprendre notre dataset et avoir plus d'intuition pour la modélisation de modèles, nous explorons ici nos données.
Outre les données temporelles (jours, mois, années), voici les variables explicatives que nous pourrons utiliser pour prédire le nombre de ventes journalier de notre produit :

- prix : prix de l'article
- event_name_1 : Evènement comme le Superbowl, Nouvel An, Halloween
- event_type_1 : Type de l'évènement ('Sporting', 'Cultural', 'National', 'Religious')
- snap_WI : Snap (Supplemental Nutrition Assistance Program) autorisé ce jour (0 : non, 1 : oui)
- temp : Température en Farenheit
- feelslike : Température ressentie en Farenheit
- humidity : humidité de l'air
- precip : Taux de précipitiation
- preciptype : Type de précipitation (neige, pluie)
- windspeed : Vitesse du vent en mph
- visibility : visibilité dehors (détails de la mesure passés)
- cloudcover : Couverture nuageuse
- uvindex : Indice UV

Remarque : Pour ne pas obtenir d'informations supplémentaires par visualisation, les données de validation ont écartées de la visualisation.

3.2 Données m5

3.2.1 Prix

Les données m5 comportaient toutes les variables qui ne sont pas météorologiques. En analysant ces données, nous avons pu voir que le pic de baisse de prix est en accord avec la baisse de prix la plus importante.

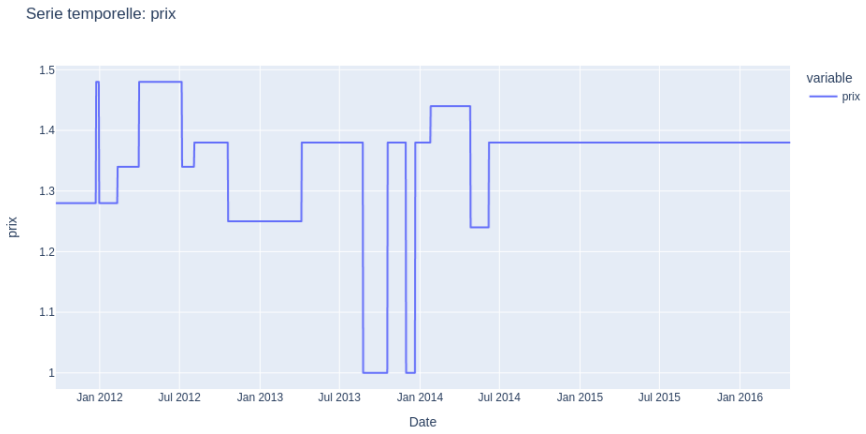

FIGURE 4 – Prix de l'article

3.2.2 Mois et jour de la semaine

En faisant la moyenne des ventes par mois ainsi que par jour de la semaine, on voit une forte importance de ces variables. Notamment pour le jour de la semaine dont l'incertitude de son importance est bine moins faible considérant que l'on fait une moyenne sur plus de 200 semaines.

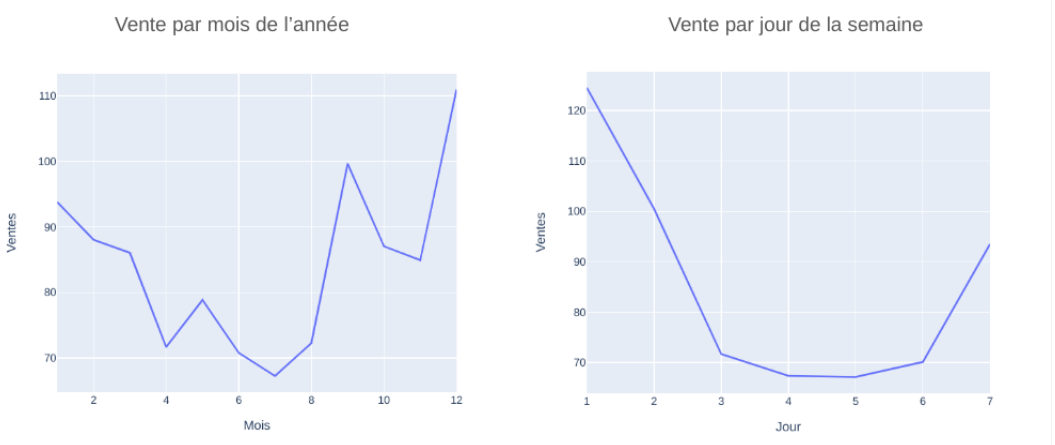

FIGURE 5 – Influence de la semaine et du mois sur les ventes

3.2.3 SNAP

Les données 'SNAP_WI', concerne le programme éponyme d'aide alimentaire. C'est une variable binaire du premier au 15 de chaque mois avec une alternance 2 jours avec, un jours sans. Il est difficile de voir l'influence de ces données, la corrélation n'étant que peu significative pour avec les variables catégorielles. Cependant, nous gardons cette variable étant très souvent présente.

3.2.4 Données événementielles

Notre dataset comporte un grand nombre d'évènements qui pourraient être plus ou moins importants pour notre prévision. Chaque évènement ayant son propre label et n'arrivant qu'une fois par an, il était plus judicieux d'utiliser la variable : 'event_type' qui regroupe les évènements en quatre catégories : culturel, national, religieux et sportif pour notre exploration.
Malheureusement, notre analyse visuel n'a pas démontré de véritable influence de ces variables exogènes. Cela pourrait être dû à la grande diversité des évènements pris en compte. Par exemple, la fête de Noël orthodoxe (7 janvier) est probablement moins importante au Wisconsin que le 25 décembre. Dans une étude plus approfondie, il pourrait être judicieux de considérer un raffinement des évènements à prendre en compte. N'étant pas des experts des Etats-Unis et de leurs évènements et n'ayant pas d'informations spécifiques sur notre produit alimentaire, nous avons décidé de ne pas inclure ces variables pour la modélisation.

3.3 Données météorologiques

Nous avons d'abord regardé la corrélation entre nos données météorologiques.

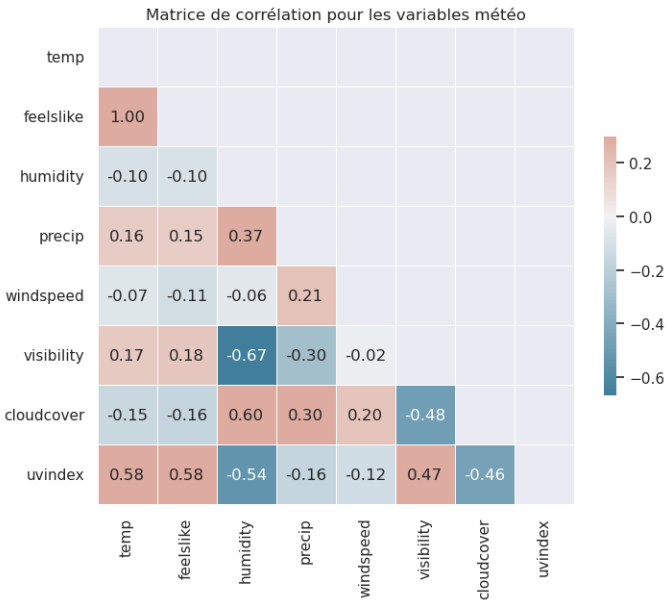

FIGURE 6 – Corrélation des variables météo

Au vu de la matrice de corrélation de la météo et pour allèger notre étude sachant que la météo n'a pas un impact très conséquent sur la consommation de notre produit, on va réduire le nombre de données météos et garder uniquement la pluie et la température.

3.3.1 Création de nouvelles variables

Nous avons rajouté une variable avec la météo normalisée en fonction du mois. Ce choix est motivépar l'intuition que les valeurs extrêmes pendant une saison peuvent influencer notre consommation de produits. Ensuite, nous rajoutons des features dits de "valeurs êxtremes", valant 1 si la valeur a un **z-score** supérieur à notre critère.
Informations sur nos variables :
- température, les valeurs vont de -25 à 32, donc il n'y a pas vraiment de forte chaleur.
- précipitation : valeurs de 0 à 78.
Nous avons décidé de créer les variables binaires "extrêmes" suivantes :
- température extrême : $z < -2$ ou $z > 2.5$ (2.3% des valeurs les plus froides et 0.6% plus chaudes).
- précipitation extrême : $z > 1.3$ (10% des jours avec le plus de pluie/neige).
Le choix de ces critères est assez arbitraire, mais motivé par la volonté d'avoir des variables gardant un nombre conséquent de valeurs.

Remarque : Nous voulions aussi utiliser "windspeed" car le vent peut être dangereux. Cependant, notre jeu de données ne comporte que 8 valeurs qui dépassent les 77 kmh, valeur à partir de laquelle il peut être dangereux d'être dehors, c'est à dire 8 sur l'échelle de Beaufort [5].

3.4 Bilan de l'EDA

Finalement, nous pouvons résumer notre exploration des données en quelques points :

- Les variables prix, jour de la semaine et moins sont fortement corrélées avec les ventes.
- Création de variables normalisées pour la météo, et étude de ces dernières.
- Création de nouvelles variables météorologiques dites extrêmes.
- Suppression des variables 'event_name_1' et 'event_type_1' qui sont trop peu significatives.

Évidemment, ces choix devront être vérifiés et étudiés lors de la création de nos modèles par différents tests statistiques, comme avec les tests de Student lors des modèles de régression linéaire par exemple.

4 Modèles de prévision

Après avoir effectué une analyse détaillée de notre jeu de données et sélectionné les variables qui nous semblent pertinentes, nous allons entamer la phase de modélisation.
Nous commencerons par les modèles paramétriques tel que la régression et nous finirons par des modèles non paramétriques et ensemblistes comme les modèles d'arbres. Cette variété de modèles devraient nous permettre par la suite d'avoir de bons candidats pour une agrégation d'expert.

4.1 Régression linéaire

Pour débuter, nous allons utiliser la régression linéaire avec et sans pénalisation de la bibliothèque scikit-learn. Cela nous permettra de comparer les performances de ces modèles

5. `https://www.weather.gov/pqr/wind`

et de déterminer si nous devons ajuster les hyperparamètres. La régression linéaire est un modèle simple et interprétable qui peut s'avérer être un très bon candidat quand ses hypothèses sur la distribution des données sont respectées.
Les modèles de régression pénalisée ont été choisis pour réduire le sur-apprentissage, la régression Ridge utilise une régularisation L2 qui force les coefficients à devenir plus petits, ce qui réduit la variance des prévisions et améliore la capacité du modèle à généraliser les données.
En revanche, la régression Lasso utilise une régularisation L1 pour réduire la dimension si nécessaire. Cela peut s'avèrer utile dans le cas où nous devons utiliser un OneHotEncoding pour les variables catégorielles, qui augmente considérablement la dimension. Par exemple pour un encodage de la variabke "mois" rajoute 12 variables binaires.
Nous avons utiliser un modèle de référence (benchmark) prédisant toujours la moyenne comme outil de comparaison. Voici les résultats de nos modèles :

	model	MSE	RMSE	MAE	R2	gap less than 0.1	time
0	Bunchmark	1239.401247	35.205131	29.697698	-0.00456	0.156522	0.0
1	Linreg	472.004613	21.725667	17.318727	0.61743	0.286957	0.0
2	Rdige	485.636454	22.037161	17.431472	0.60638	0.295652	0.0
3	LASSO	511.009234	22.605513	17.712572	0.58582	0.313043	0.0
4	ElasticNet	721.969308	26.869487	21.457211	0.41483	0.286957	0.0

FIGURE 7 – Résultats des métriques de performance

Finalement, au vu des résultats, la régression simple a obtenu les meilleurs score et donc la régularisation (Ridge, LASSO ou Elastic-Net) n'était pas nécessaire.
En complément des résultats fournies par nos métriques, nous opérons toujours à une visualisation graphique de nos modèles. Cela nous permet d'avoir une meilleur idée des performances de nos prévisions, tant sur les données d'entraînements que celles de test.
Nous pouvons constater que, lors de l'entraînement, la régression linéaire n'a pas fait de sur apprentissage et que le biais n'est pas nul. De plus, il semble que le modèle ait appris suffisamment bien les tendances des données pour faire des prévisions adéquates.

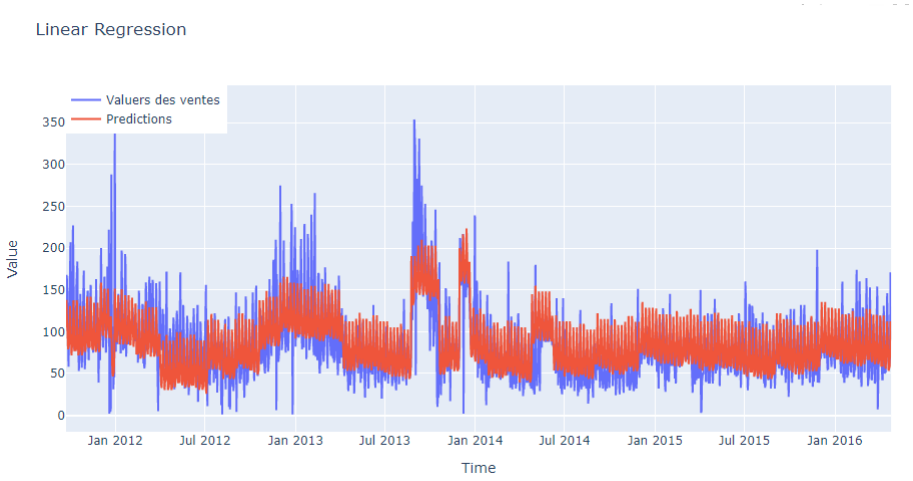

FIGURE 8 – Prédiction par régression linéaire

Nous observons que les prévisions se généralisent bien aux données test en captant adéqua-

tement les variations. Nous pouvons aussi noter que notre modèle ne présente pas de forte variance et est donc assez stable pour généraliser ses prédictions sur des données non-vues.

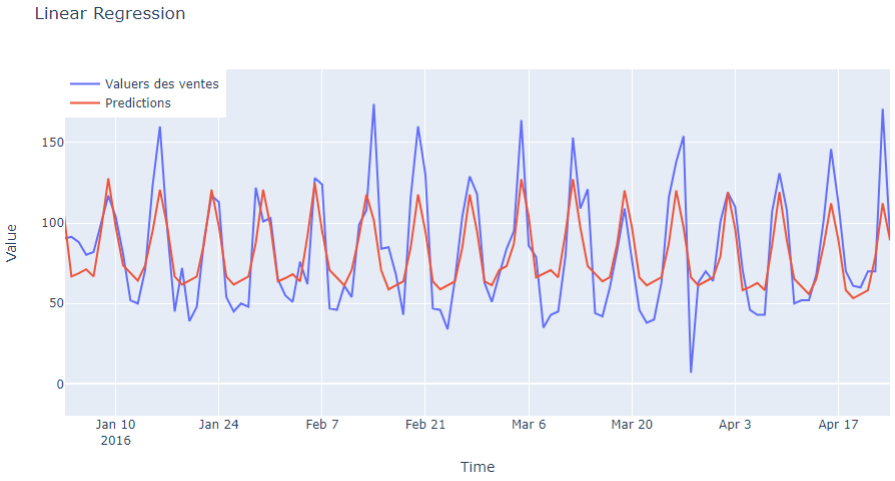

FIGURE 9 – Prédiction de la régression linéaire : données test

4.2 Modèle additif généralisé

Dans cette partie, nous allons utiliser un modèle GAM pour modéliser nots prévisions. Contrairement à la régression linéaire, le GAM ne suppose pas que la relation entre la variable cible et les variables explicatives est linéaire. Au lieu de cela, il utilise des fonctions continues "splines" pour ajuster chaque variable explicative. Ce modèle permet d'exploiter des relations plus complexes entre les variables. Par défaut, le GAM de la librairie "pyGAM" [6] utilise des b-splines penalisés comme fonctions de base et des coefficients de régularisation à 0.6.
la fonction de lien par défaut est "l'identité". C'est une fonction de lien particulière qui ne fait aucune transformation de la variable exogène pour les prédictions du modèle.
Après plusieurs tests, nous avons pris une distribution gaussienne comme supposé distribution de notre variable cible et fixés les coefficients de régularisation à 0.1.
L'illustration suivante montre l'effet spline sur la variable 'prix'.

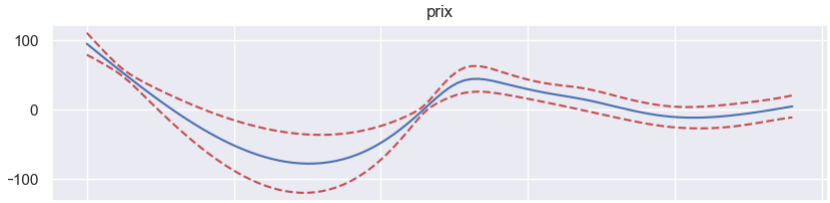

FIGURE 10 – Transformation B-Splin pour : 'prix'

Les résultats obtenus avec le modèle GAM sur les données d'entraînement ainsi que les données test sont satisfaisants mais légèrement moins bon qu'avec une simple régression linéaire.

6. `https://pygam.readthedocs.io/en/latest/`

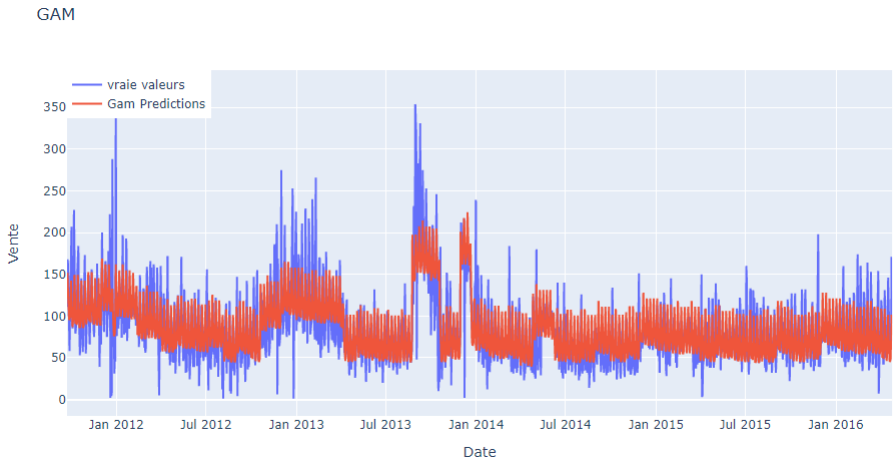

FIGURE 11 – Prédictions du GAM sur les données d'entraînement

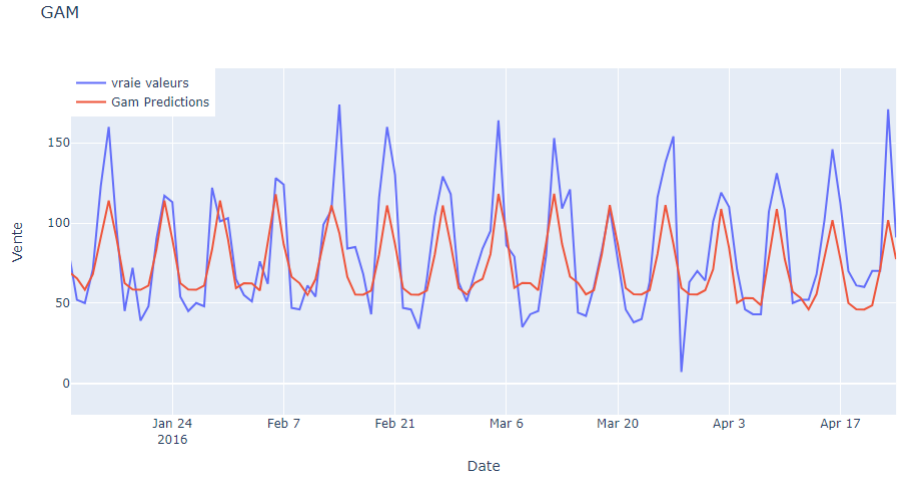

FIGURE 12 – Prédictions du GAM sur les données test

Bien que les résultats obtenus restent satisfaisants, la moins bonne performance de notre modèle face à la régression linéaire simple traduit une complexité supplémentaire non justifiée. La transformation des variables exogènes n'étaient donc pas nécessaire, des relations linéaires avec la variable cible suffisent.

4.3 Modèles de Boosting

4.3.1 Premiers résultats

En utilisant des approches non-paramétriques, comme les modèles ensemblistes basés sur les arbres de décision, il est parfois possible de prédire efficacement sans s'inquiéter de l'interprétabilité ou de la compréhension complète des lois des données. Ces modèles peuvent être un choix judicieux lorsque l'on cherche à obtenir des performances élevées.
Les modèles ensemblistes basés sur des arbres de décisions peuvent s'avérer très performants et sont souvent utilisés lors de compétitions de prévision. Nous étudions dans cette partie certains d'entre eux.
Dans un premier temps, nous avons sélectionné certains candidats et les avons évalués. Ces modèles étant sensibles aux hyperparamètres, nous avons d'abord étudié leurs perfor-

mances avec les valeurs par défaut. Ensuite, nous avons utilisé une technique de recherche d'hyperparamètres (gridsearch) en utilisant une séparation temporelle pour essayer d'optimiser nos modèles
Voici les résultats sans grid search :

	model	MSE	RMSE	MAE	R2	gap less than 0.1	time
0	Bunchmark	1239.401247	35.205131	29.697698	-0.00456	0.156522	0.0
1	Knn	711.038601	26.665307	20.622002	0.42369	0.260870	0.0
2	Arbre	838.467577	28.956305	22.511587	0.32041	0.260870	0.0
3	RF	808.059222	28.426382	21.960997	0.34505	0.260870	0.0
4	ADA	817.307861	28.588597	23.217745	0.33756	0.252174	0.0
5	XGB	806.830436	28.404761	22.036324	0.34605	0.278261	0.0
6	LGBM	718.191389	26.799093	20.578043	0.41789	0.313043	0.0

FIGURE 13 – Performance des modèles ensemblistes

Nous remarquons directement que les résultats sont pour l'instant bien moins bons que ceux de la régression linéaire.
Les modèles ensemblistes sont généralement complexes et peuvent donc facilement surajuster les données d'entraînement, réduisant ainsi les performances sur les données de test.

4.3.2 Amélioration des hyperparamètres

Dans l'espoir d'obtenir de meilleures performances, nous avons effectué une gridsearch sur tous ces modèles en utilisant la méthode de séparation temporelle. Cela consiste à tester différentes combinaisons de paramètres sur les données par validation croisée pour sélectionner les paramètres qui permettent d'obtenir les meilleures performances sur les données de test.

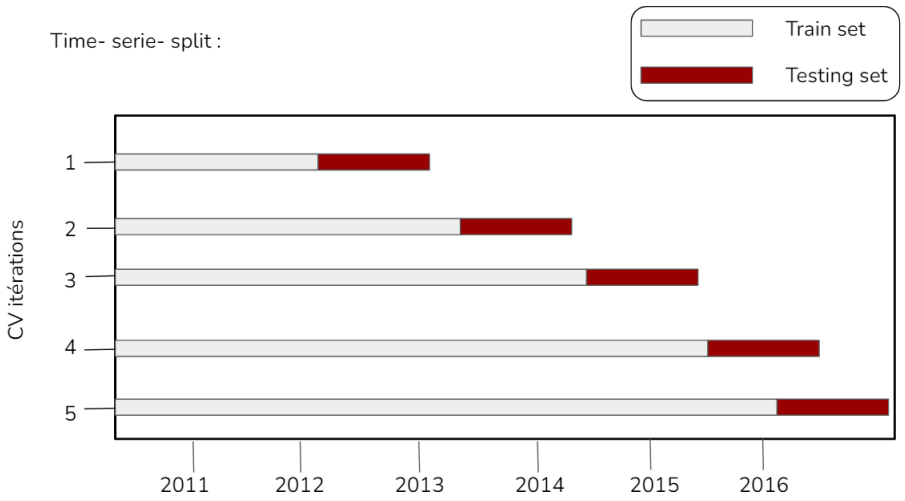

FIGURE 14 – Illustration du découpage temporel pour la validation croisée

Après avoir entraîné les nouveaux modèles avec les hyperparamètres optimaux, nous obtenons de meilleurs résultats.

	model	MSE	RMSE	MAE	R2	gap less than 0.1	time
0	Bunchmark	1239.401247	35.205131	29.697698	-0.00456	0.156522	0.0
1	Knn	617.600505	24.851569	20.032068	0.49942	0.304348	0.0
2	Arbre	838.467577	28.956305	22.511587	0.32041	0.260870	0.0
3	RF	776.886585	27.872685	21.617989	0.37032	0.269565	0.0
4	ADA	802.564619	28.329571	21.967041	0.34951	0.286957	0.0
5	GB	978.271006	31.277324	24.966375	0.20709	0.234783	0.0
6	XGB	724.240289	26.911713	20.811432	0.41299	0.295652	0.0
7	LGBM	674.083971	25.963127	20.136864	0.45364	0.321739	0.0

FIGURE 15 – Performance des modèles ensemblistes après gridsearch

4.3.3 Visualisation des résultats de deux modèles

Pour illustrer les biais et la variances de nos modèles, nous regardons ici le modèle K-NN et Random Forest :

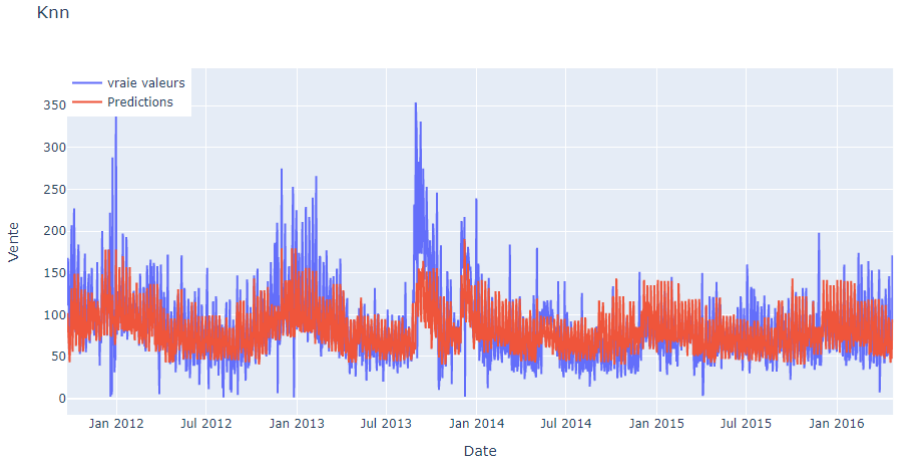

FIGURE 16 – Estimation en train pour K-nn

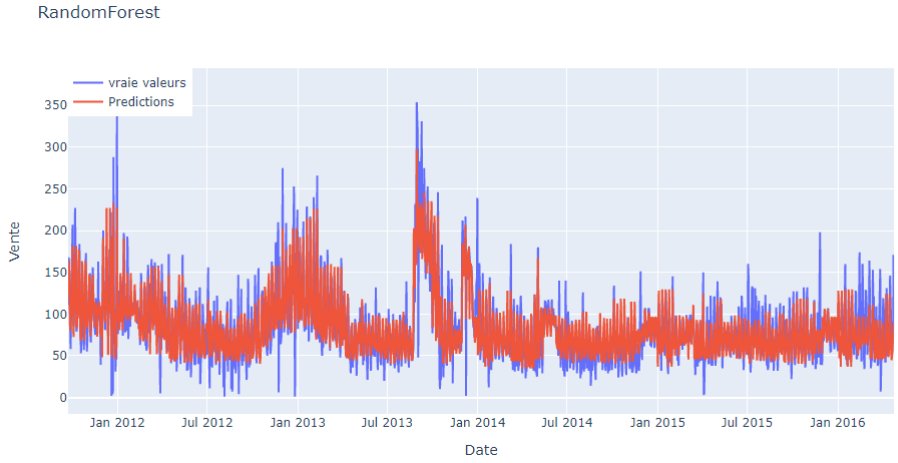

FIGURE 17 – Estimation en train pour les Random Forest

Il semble que les forêts aléatoires ont tendance à avoir moins de biais et d'ajustement excessif que K-nn (avec k = 9) sur certaines parties des données d'apprentissage. Les forêts aléatoires semblent "coller" aux données d'entraînement et peuvent prendre en compte les valeurs éloignées de la moyenne, tandis que K-nn ne semble pas reproduire les données d'entraînement avec autant de précision, mais donne des estimations plus correctes sur les données test.

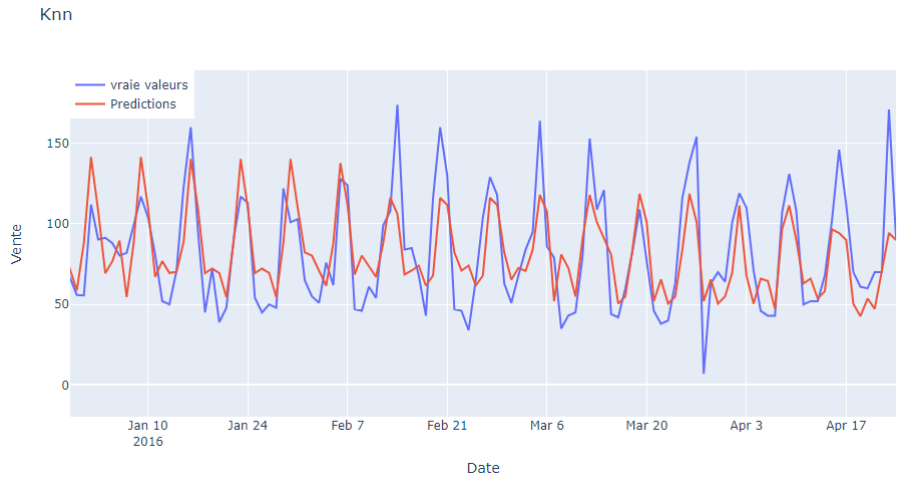

FIGURE 18 – Prédictions K-nn

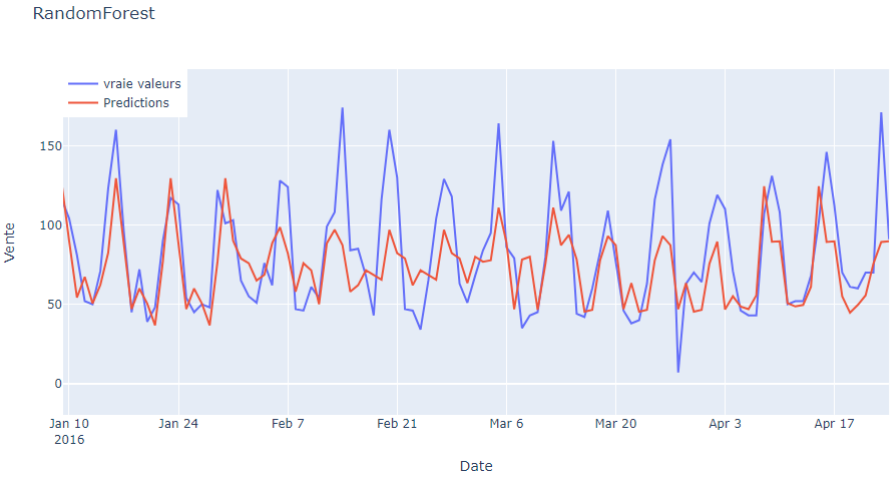

FIGURE 19 – Prédictions Random Forest

4.4 Arimax

Dans cette partie, nous allons essayé de modéliser notre série à l'aide d'un ARIMAX. Nous avons choisi ce modèle qui combine le modèle ARIMA (AutoRegressive Integrated Moving Average) et les variables exogènes pour pouvoir faire une comparaisons correcte avec les autres modèles qui utilisent des variables explicative. Voici la formule générale du modèle ARIMAX :

$$y_t = c + \phi_1 y_{t-1} + ... + \phi_p y_{t-p} + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} + \beta_1 x_{1,t} + ... + \beta_k x_{k,t} + \epsilon_t$$

où :

— y_t est la variable dépendante (série chronologique) à prévoir à la période t

— $\phi_1, ..., \phi_p$ sont les coefficients de l'autorégression

— $\theta_1, ..., \theta_q$ sont les coefficients de la moyenne mobile

— $\beta_1, ..., \beta_k$ sont les coefficients des variables exogènes

— $x_{1,t}, ..., x_{k,t}$ sont les variables exogènes à la période t

— ϵ_t est le terme d'erreur à la période t

— c est une constante

Nous avons fait une décomposition en composantes, tendance ,saisonnalité et bruit pour pour mieux comprendre le comportement de notre série chronologique. A l'aide de la fonction *sm.tsa.seasonal.decompose()* de la librairie *statsmodels*[7]. Nous avons obtenu une saisonnalité d'ordre 7 et un bruit qui semble gaussien. Voici des graphiques illustrant cela :

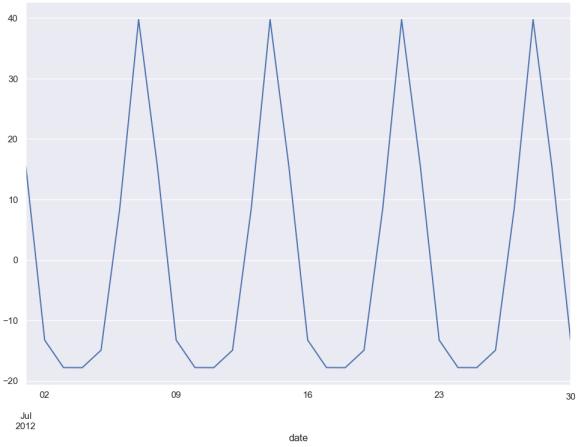

FIGURE 20 – Saisonnalité du mois de Juillet 2012

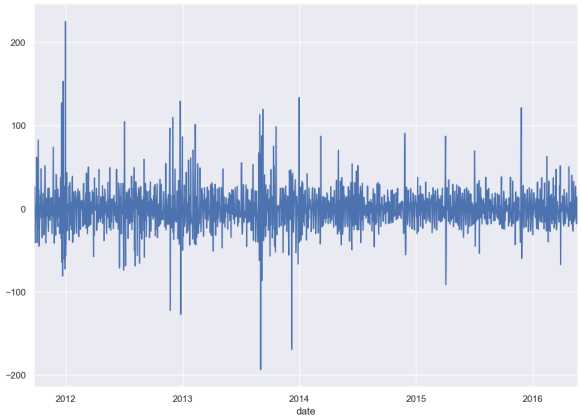

FIGURE 21 – Bruit de la série chronologique

Le bruit semble être gaussien, ce que nous avons vérifié avec un test de Shapiro-Wilk. L'hypothèse nulle de ce test étant la normalité et ayant obtenu une p-value de 0.1285, nous pouvons conclure que le bruit est bien gaussien. Cette information pourrait aussi expliquer

7. `https://www.statsmodels.org/stable/index.html`

pourquoi le modèle de régression linéaire marche bien (ce dernier supposant que le bruit est gaussien).
Finalement, le modèle retenu, maximisant le critères AIC était : ARIMA(4,1,3). Malgré certains passages ou les prédictions semblent bonnes, ce modèle ne parvient pas à battre le benchmark que ce soit pour la RMSE ou la MAE (1543 et 32 contre 1239 et 29).

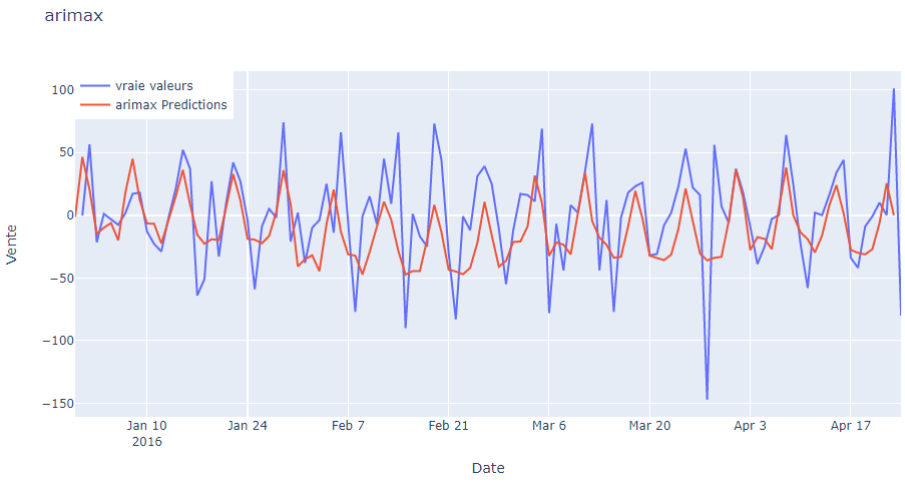

FIGURE 22 – Prédictions Arimax

5 Agrégation de modèles et résultats

Nous allons maintenant essayer d'améliorer nos prédictions en effectuant une agrégation de nos modèles. Pour cela, nous allons tester des combinaisons convexes dans l'ensemble $\{0, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1\}$. Une agrégation en ligne n'est pas nécessaire étant donnée que nous voulons faire une unique prédiction de 28 jours.
Les trois modèles sélectionnées selon leurs RMSE étaient la régression linéaire, le K-NN (K=9) et le GAM. Après avoir fait une minimisation sur notre ensemble de poids et les leurs résultats obtenus sur les mois de test, la meilleur combinaison convexe était de garder uniquement le modèle de régression linéaire (poids 1 et 0 pour les autres). Ainsi, nous ne ferons pas d'agrégation de modèles.

Finalement, notre score RMSE final sur les données de validation avec la régression linéaire est de 460 contre 1267 pour le benchmark prédisant la moyenne. Nous pouvons donc conclure que notre modélisation a été fructueuse, captant beaucoup mieux les fluctuations que qu'une prédiction par moyenne. De plus, après vérification, le modèle de régression linéaire c'est bien avérer meilleur que nos autres candidats (GAM et k-NN) sur les données de validation, ce qui conforte la validité de notre choix pour les données test.

Modèle	MSE	RMSE	MAE	R2
Régression	460.84	24.387	16.293	0.5683
GAM	471.13	28.956	16.135	0.5587
KNN	475.169	28.009	18.628	0.5549