12 March 2019

Alexandre ALLANI
Adrien CLOTTEAU

Rapport projet S5

Challenge Total

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Contents

1 Introduction

Vous trouverez l'ensemble de ce qui a été fait en suivant le lien ci-dessous.
`https://github.com/Syndorik/ChallengeTotal`.

En suivant le lien Github, vous trouverez:

- Challenge-Total-Overlook-of-the-data.ipynb : Jupyter Notebook sur la visualisation des données.
- Data-Engineering-and_Modeling.ipynb : Jupyter Notebook sur la préparation des données et la modélisation.
- Time_series.ipynb : Jupyter Notebook sur la modelisation avec les Time Series.

Dans le cadre de notre projet ISA de 3e année, nous avons travaillé sur un challenge proposé par ARGEDIS, une filiale du groupe TOTAL chargée de la gestion et de l'exploitation des boutiques d'une partie des stations-service du réseau routier français. ARGEDIS prend en charge tout l'approvisionnement de ses boutiques en produits frais, snacking et boissons.
L'objectif du challenge est de réaliser des prédictions de vente de certains produits vendus par deux stations-service entre le 1er et le 15 mai 2017.
Les données mises à disposition sont un historique de vente depuis de 1er janvier 2016 jusqu'au 30 avril 2017, ainsi que certaines données open data (météo, trafic routier). Les jeux de données d'entraînement (01/01/2016 - 30/04/2017) et de test (01/05/2017 - 15/05/2017) sont disponibles directement sur la plateforme mise à disposition par ARGEDIS pour l'occasion.
Nos résultats sont évalués directement par la plateforme d'ARGEDIS. Le fichier test à fournir à cette plateforme au format spécifié et contenant nos prédictions est évalué par la méthode RMSE: $\sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_t - y_t)^2}{n}}$ avec Y_i , la valeur réellement vendue et $\hat{Y}_i$ notre prédiction.
Ne pouvant communiquer directement avec notre client ARGEDIS, les retours sur nos modèles nous étaient donnés par nos encadrants : Mr Laurent Lecornu et Mr John Puentes.
La méthodologie CRISP est appliquée tout au long du projet.

2 Contexte

La grande majorité des stations-essence gérées par Argedis sont situées sur des autoroutes. L'importation des produits coûte cher à cause des frais de stockage et de transports (notamment à cause des tarifs autoroutiers).

L'objectif du projet est donc de prédire de manière précise la consommation des différents produits. Cela permettrait de réduire le gaspillage et donc les dépenses invendues dues à des camions trop remplis et de limiter au maximum les pénuries et donc les pertes de chiffre d'affaires.

3 Visualisation des données

Le code lié à cette partie est disponible sur le lien suivant `https://github.com/Syndorik/ChallengeTotal/blob/master/Challenge-Total-Overlook-of-the-data.ipynb`.

3.1 Description des jeux de données

Le jeu de données d'entraînement contient 39 variables et 198288 entrées. Cela correspond à un type d'article par station par jour. La durée d'observation est de 486 jours et il y a 2 stations-essence différentes.

Le nombre d'entrées correspond donc au nombre d'articles différents possibles à la ventes (204) multiplié par le nombre de stations-essence (2), et le nombre de jours (486).

3.2 Premières variables

La variable id donne un identifiant unique à chaque entrée.
La variable à prédire est nommée *qte_article_vendue*. Celle-ci est discrète et est égale au multiple d'une valeur de base (différente pour chaque article et correspondant à un nombre de palettes, à un prix ou bien à un poids fixé par Argedis). Pour un identifiant donné, elle correspond donc au nombre d'achats d'un article un jour fixé, dans une station-service donnée, multipliée par la valeur de base de l'article.
Pour chaque entrée :

- On dispose d'une variable *article_nom* qui renseigne le nom de l'article.
- La variable date renseigne la date.
- La variable implant indique la station-service.

3.3 Variables numériques

3.3.1 Température

La température est exprimée par 4 variables de la forme *t_Hh_Ville* avec $H \in 9, 15$ et $Ville \in$ Paris, Rouen. Elles donnent la température mesurée à Paris ou Rouen à 9h ou à 15h.
Elle est exprimée en Kelvin. On peut l'obtenir en degrés Celsius en soustrayant 273,15, comme sur la représentation ci-dessous (plus facile à comprendre pour un utilisateur habitué aux Celsius).

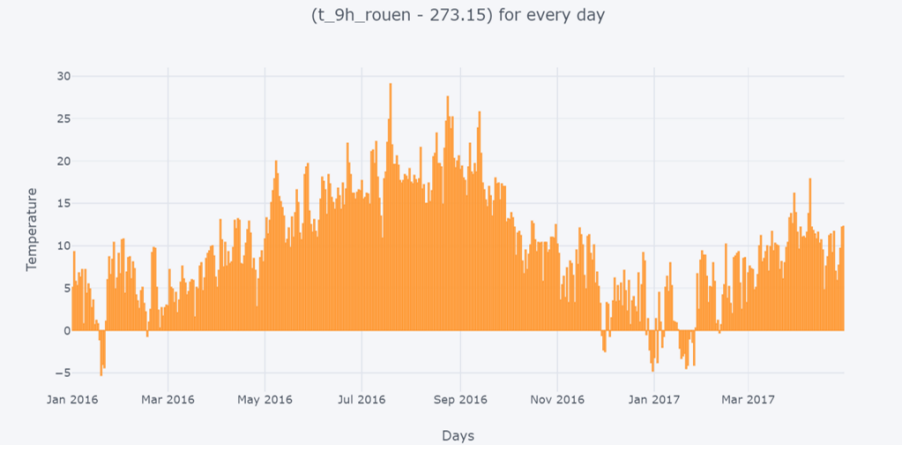


Figure 1: Température pour une des stations Service au cours de l'année

3.3.2 Vent

Le vent est exprimé par 4 variables de la forme *ff_Hh_Ville* avec $H \in 9, 15$ et $Ville \in$ Paris, Rouen. Elles donnent la force du vent mesurée à Paris ou Rouen à 9h ou à 15h.

En représentant le nombre d'occurrences des valeurs de force de vent, on se rend compte que ces valeurs sont comprises entre 0 et 11 et que la majorité de ces occurrences ont lieu pour des valeurs comprises entre 2 et 5. Cela correspond parfaitement avec une échelle de Beaufort et la comparaison des valeurs à dates fixées avec des valeurs relevées dans des registres à disposition sur infoclimat confirment cette hypothèse. **Lien vers info climat**

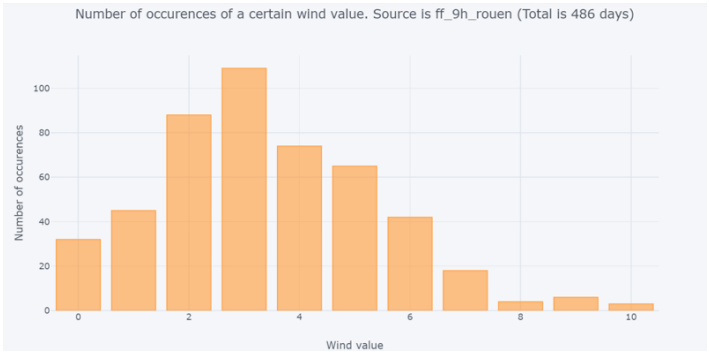


Figure 2: Vent pour une des stations Service au cours de l'année

3.3.3 Nebulosité

La nébulosité est exprimée par 4 variables de la forme *n_Hh_Ville* avec $H \in 9, 15$ et $Ville \in$ Paris, Rouen. Elles donnent la nébulosité mesurée à Paris ou Rouen à 9h ou à 15h.

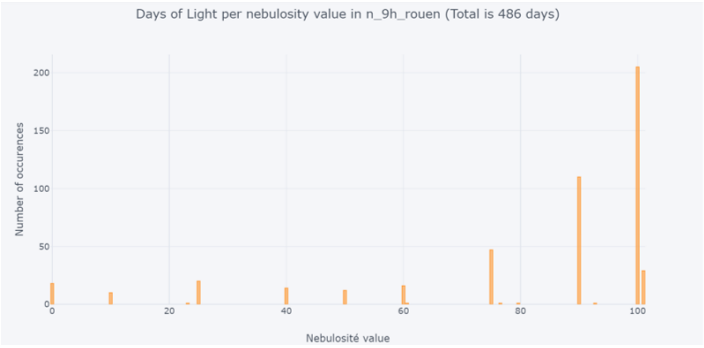


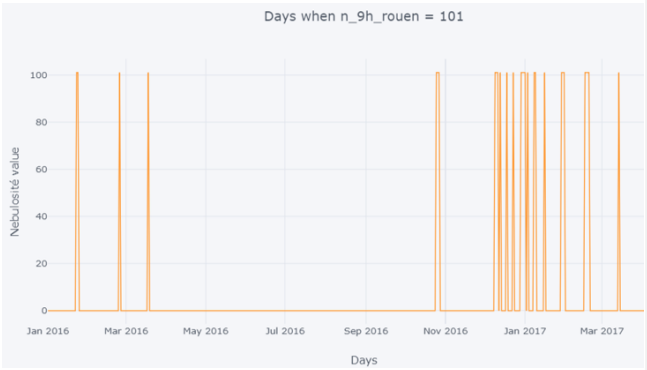
Figure 3: Nebulosité pour une des stations Service au cours de l'année

En représentant le nombre d'occurrences de chaque valeur de nébulosité, on se rend compte que celle-ci ne prend qu'un certain nombre de valeurs discrètes. A part 4 points aberrants, la nébulosité ne peut prendre que 10 valeurs : les neufs octas possibles (de 0/8 à 8/8) affichés en pourcentages et la valeur 101.

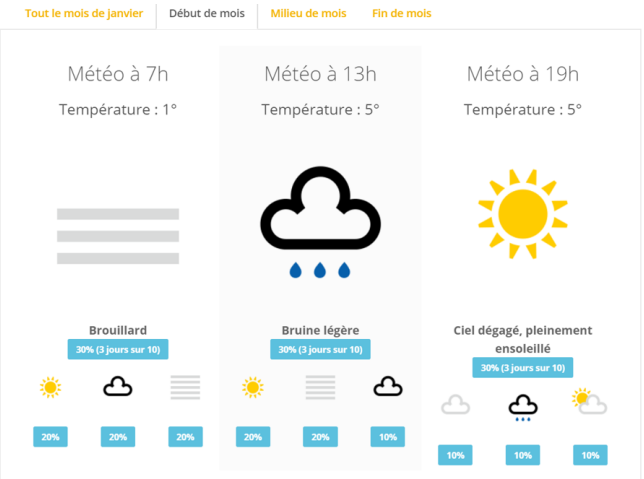
La nébulosité est traditionnellement calculée en octas, un 0 représentant un ciel dégagé, un 8 représentant un ciel couvert.

Cependant, il reste à découvrir si la valeur 101 est significative d'un évènement particulier ou si elle doit être considérée comme une erreur de remplissage du jeu de données.

En n'affichant uniquement que les jours où la nébulosité vaut 101 (**Figure 4a**) pour Rouen à 9h, on se rend compte que ce phénomène a lieu majoritairement pendant l'hiver.



(a) Nébulosité à 101 au cours de l'année



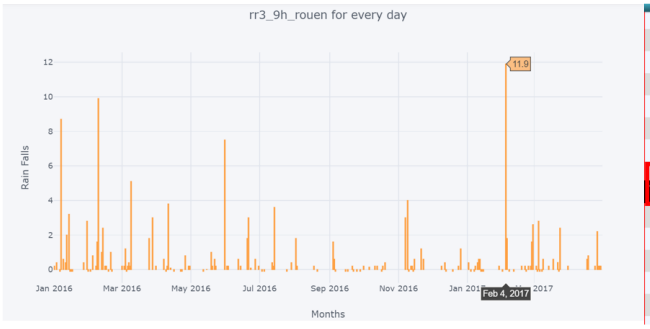
(b) Météo pour un jour de nébulosité 101

Figure 4: Vérification de nébulosité à 101

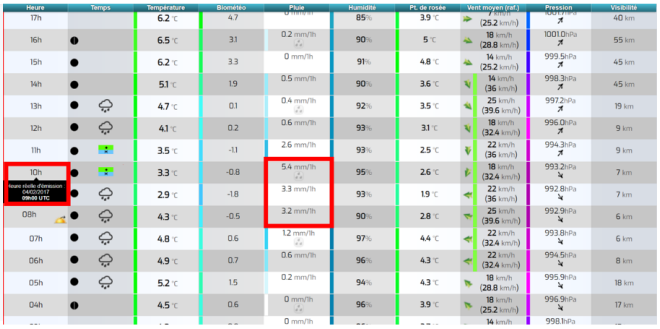
Nous avons regardé quels étaient les relevés météo enregistrés pour le début du mois de Janvier 2017, où les valeurs 101 de nébulosité reviennent très souvent.
Il s'est avéré que cette période avait été forte en brouillard, comme nous l'indique la photo ci-dessous. En comparant les relevés jours par jours, cela correspond.
La valeur 101 indique donc du brouillard donc une visibilité très mauvaise (**Figure 4b**);

3.3.4 Précipitations

Les précipitations sont exprimées par 4 variables de la forme *rr3_Hh_Ville* avec$H \in 9,15$ et $Ville \in$ Paris, Rouen. Elles donnent les précipitations mesurées à Paris ou Rouen à 9h ou à 15h. En affichant les précipitations pour chaque jour de l'année, on remarque que la valeur renseignée est la somme des précipitations (en mm) tombées lors des 3 dernières heures selon infoclimat.([Lien vers info climat](#)) Les valeurs égales à -0,1 ne semblent pas décrire un évènement particulier. Elles doivent être causées par une erreur de mesure entre deux relevés : le relevé précédent a été surestimé de 0,1 et il n'y a pas eu de précipitations entre ces deux relevés.



(a) Précipitation au cours de l'année



(b) Info climat pour le 4 Février 2017 précipitations

Figure 5: Vérification du calcul des précipitations

3.4 Variables catégorielles

3.4.1 Stations-service

D'après le site de Total France, les deux stations sont situées en Normandie, dans l'ouest de Rouen. La station dont l'identifiant est NF078544 est la station de Beuzeville, située la plus à l'ouest de Rouen. (**Lien vers la position de la station sur google maps**) (**Figure 6a**)

La station dont l'identifiant est NF059473 est la station de Bosgouet, plus proche de Rouen. (**Lien vers la position de la station sur google maps**) (**Figure 6a**)

3.4.2 Catégories de produits

Les différents articles sont décrits selon une hiérarchie de 3 catégories : Article à vendre $\subset$ Catégorie 6 $\subset$ Catégorie 5 $\subset$ Catégorie 4;
Chaque catégorie est décrite par les variables *id_categorie_X* et *catX_nom* avec $X \in 4, 5, 6$. Par exemple, une catégorie 4 a pour nom « Boulan Vienn Patiss » et pour id 1001639.

3.4.3 Bison Futé

Pour ce qui est de la circulation, on dispose de 6 variables *aller_zone_X* avec $X \in 1, 2, 3, 4, 5, 6$ et de 6 variables *retour_zone_X* avec $X \in 1, 2, 3, 4, 5, 6$. Leurs valeurs vont de 0 à 3, 0 représentant un trafic fluide et 3 représentant une circulation très difficile. (**Figure 6b**)

(a) Position des stations service

(b) Carte Bison Futé

Figure 6: Différentes cartes

3.5 Corrélations

3.5.1 Corrélations entre variables

Une carte de corrélations a été réalisée entre toutes les variables afin de savoir lesquelles influent fortement sur le résultat et lesquelles contiennent des informations redondantes.
Les variables quantitatives (température, précipitations, vent, nébulosité) sont fortement corrélées entre elles. En particulier les 4 variables de température sont corrélées entre elles au minimum à 95%, elles contiennent toutes les 4 la même information et n'en garder qu'une ou faire une moyenne des 4 devrait être suffisant.
La variable à prédire (*qte_article_vendue*) ne semble être corrélée qu'à la température. Certains produits sont-ils particulièrement dépendant de la température pour être vendus ?

3.5.2 Corrélations entre certains articles et température

La vente de certains articles est fortement corrélée positivement à la température. Par exemple, la vente de « Discoveries Americano 22Cl Starbucks » qui sont des cafés glacés est corrélée à 49% à la température.
Globalement, les sandwichs (particulièrement les sandwichs suédois et ceux à bases de poissons), café

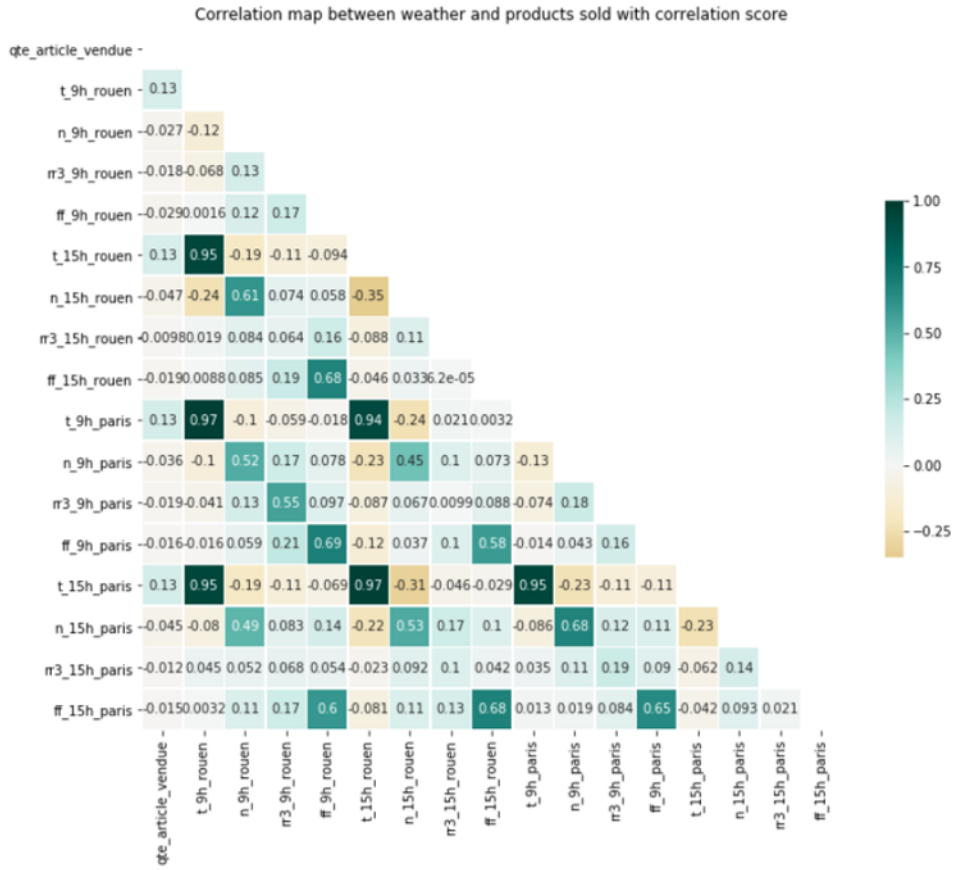


Figure 7: Corrélation entre les différentes variables

glacés et boissons, qui sont des produits frais, sont corrélés à la température alors que les repas, pâtisseries et desserts chauds sont inversement corrélés à la température.
On aurait pu s'attendre à ce que les cafés soient plus consommés en hiver, cependant l'afflux de passager pendant l'été compense cette baisse de consommation moyenne par passagers. Par conséquent, la vente de cafés chauds n'est pas corrélée à la température.

4 Préparation des données (Data Engineering)

Le code lié à cette partie est disponible sur le lien suivant `https://github.com/Syndorik/ChallengeTotal/blob/master/Data-Engineering-and_Modeling.ipynb`.

Dans cette partie, nous allons présenter la préparation des données afin qu'elles obtiennent la forme que nous avons utilisé dans nos dernières modélisations. En suivant la méthodologie CRISP, cette préparation des données est le fruit de plusieurs tests/modélisation. Nous ne présenterons que les résultats fructueux, qui ont mené à une amélioration du modèle.

Modification du jeu de données Le jeu de données étant français et l'interpréteur utilisé étant sous le format anglo-saxon, les valeurs numériques (telles que la température, la nébulosité etc.) étaient considérées comme des chaînes de caractère à cause de la virgule. Nous avons donc remplacé les "," par des "." pour pallier à ce problème.
Par ailleurs, le jeu de données ne contenant aucune valeur non acquises (NA values), il n'y a pas eu de pré-traitement à réaliser.

4.1 Segmentation temporelle

4.1.1 Distinction des vacances

L'une des premières améliorations effectuées au niveau des données est la segmentation du jeu de données en périodes de "vacances" et en périodes de "travail". En effet, il est important de modéliser cette saisonnalité car les ventes sont très différentes pendant et hors des périodes de vacances. Le graphique ci-dessus, **Figure**

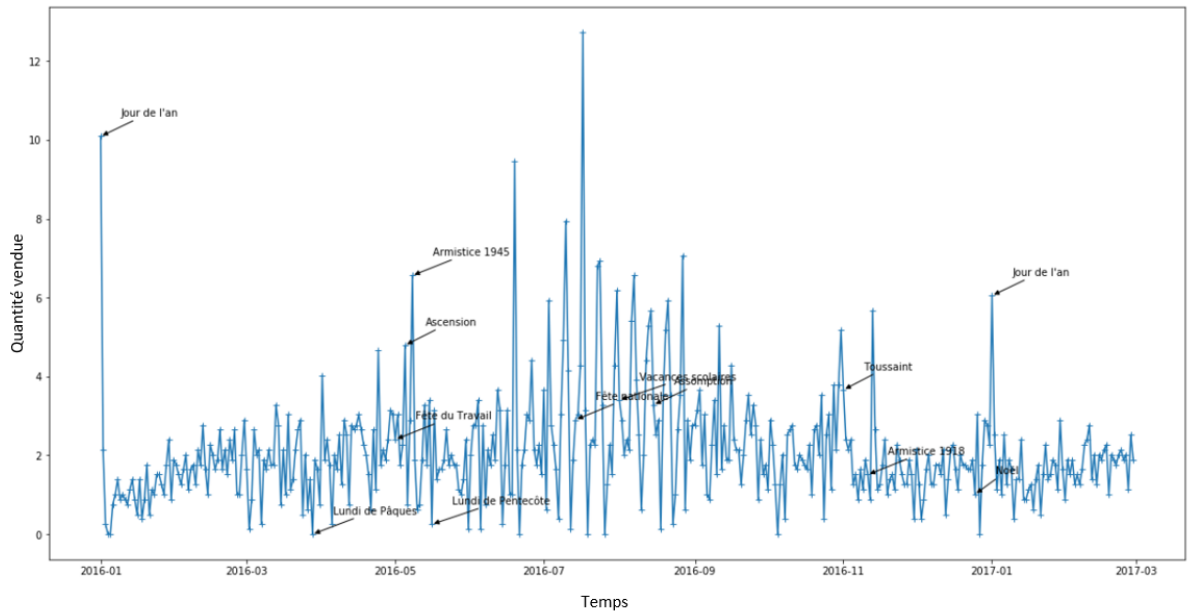


Figure 8: Ventes de baguettes au cours de la période

8 représente les ventes des baguettes sur les deux stations durant la période couverte par le jeu de données. Les baguettes sont les produits les plus vendus, tous produits confondus. Nous pouvons observer que les ventes lors des jours fériés et périodes de vacances sont beaucoup plus élevées. En particulier, les grandes vacances sont plus sujettes à de fortes variations.
Ces observations sont logiques, les périodes de vacances correspondent aux périodes de départs. Les familles vont en général s'arrêter et consommer plus dans les stations-service, ainsi la quantité de produits

vendus est beaucoup plus grande.
Segmenter l'année permet de mettre en évidence les périodes où les ventes sont très variables par rapport à la moyenne et donc permet d'affiner notre modèle. En se basant sur les **vacances 2015-2016** et les **vacances 2016-2017**, nous avons pu segmenter la période selon les vacances scolaires et les zones de départ en créant une nouvelle variable "holidays".

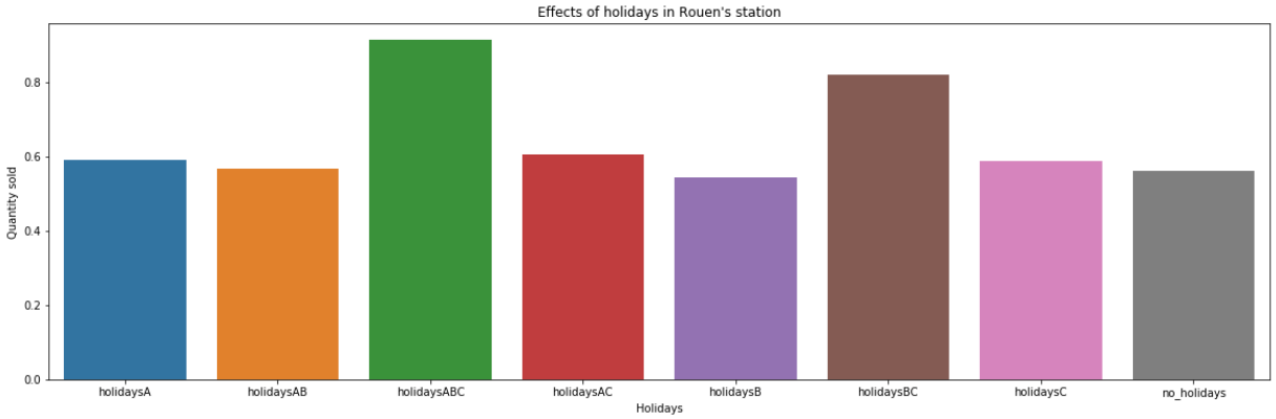


Figure 9: Ventes cumulées par semaine selon la journée

Après création de ces variables, nous pouvons comparer les ventes moyennes entre les différentes périodes de vacances grâce à la **Figure 9**. Il apparait qu'il y a une réelle différence lorsque toutes les zones (A, B, C) sont en vacances par rapport à seulement les zones (B,C). Dans les autres cas, il n'y a pas beaucoup de différences avec les périodes de travail. Cela s'explique par le fait que les stations situées à Rouen sont en zone B, et la plupart des voyageurs se dirigeant vers Rouen doivent passer par Paris (zone C). Ainsi lorsque les deux zones sont en vacances en même temps, il y a bien plus d'aller-retour qu'en période normale. Nous choisissons de garder la discrimination entre les vacances pour les trois zones (A,B,C), les vacances pour les zones B et C, et finalement les périodes de "travail".

4.1.2 Distinction des dates de départs

Nous pouvons voir sur la **Figure 8** que les premières journées et dernières journées de vacances sont particulièrement hautes en terme de quantités vendues. Ce sont, en effet, des journées où beaucoup de familles sont sur les routes, ainsi il y a mécaniquement plus de ventes qui sont effectuées.
Il est donc intéressant de discriminer ces journées afin de mieux les modéliser. Pour cela nous avons créé une nouvelle variable "departure" qui segmente le jeu de données selon les dates de départ.

4.1.3 Segmentation de la période selon les journées et mois

Dans l'optique d'exploiter au maximum les saisonnalités des ventes, nous avons continué à travailler sur les dates. Comme nous pouvons le remarquer sur la **Figure 8**, les ventes atteignent un maximum local de manière périodique environ tous les 7 jours. Il y a une périodicité remarquable au niveau de la semaine. En effet, le trajet Paris-Rouen ne dure que 2 heures, il est envisageable que certaines personnes fassent l'aller-retour chaque semaine pour rentrer le week-end, et donc s'arrêtent aux deux stations-service.

La **Figure 10** représente les ventes cumulées sur la période du jeu de données en fonction des jours de la semaine. Elle met en évidence cette périodicité. Les ventes des journées du Vendredi et Samedi sont bien au-dessus des autres jours de la semaine. De manière générale, les ventes sont faibles pendant la semaine et hautes pendant le week-end. C'est pourquoi nous avons différencié chaque jour de la semaine dans notre jeu de données.

4.1.4 Suppression de la colonne date

Une fois ces transformations réalisées, nous supprimons la colonne date. En effet pour les modèles que nous allons utiliser (excepté pour les Time Series), garder les dates va entraîner du sur-apprentissage. Les modèles

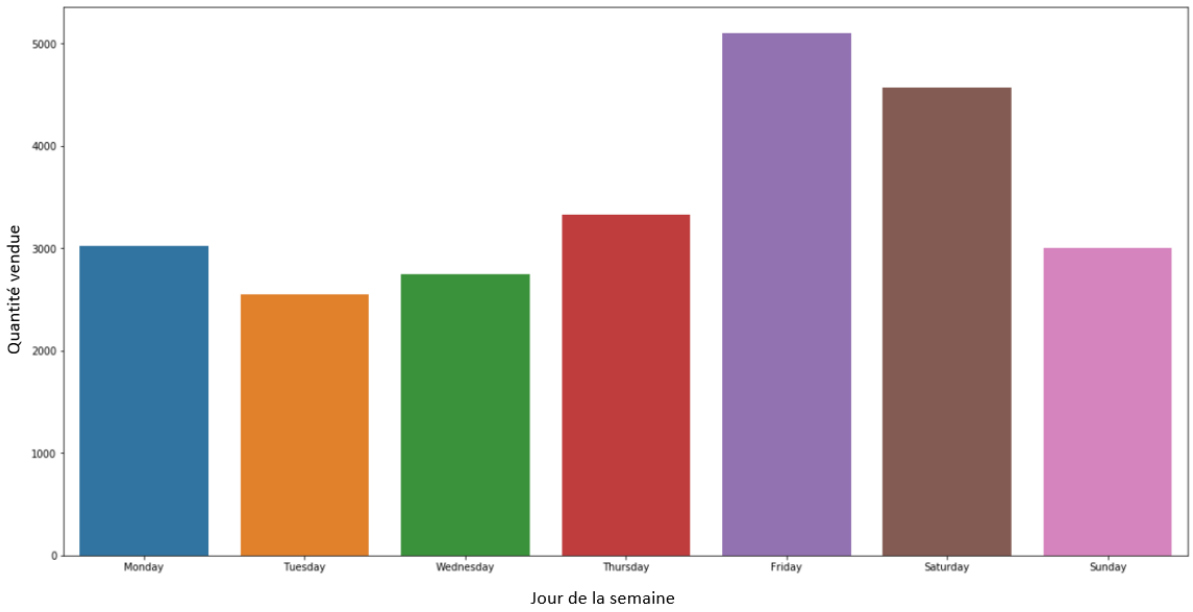


Figure 10: Ventes cumulées par semaine selon la journée

vont apprendre sur la date exacte (jour-mois-année), sachant que ces dates sont passées, cela va entraîner une grande erreur.
Il est à noter que toutes ces segmentations temporelles ont pour but d'affiner le modèle, et retirer le plus d'information de la variable *date*. Cependant pour le modèle basé sur les Time Series, ces segmentations ne seront pas conservées, car incompatibles avec le modèle. Nous expliciterons ce sujet dans la **section 5.2**

4.2 Influence des catégories

Nous voulions savoir si la distinction par catégorie était utile, sachant qu'il y a inclusion des catégories : $cat6 \subset cat5 \subset cat4$. La **Figure 11 et 12** représente les ventes en fonction de la catégorie 5 pendant la période.

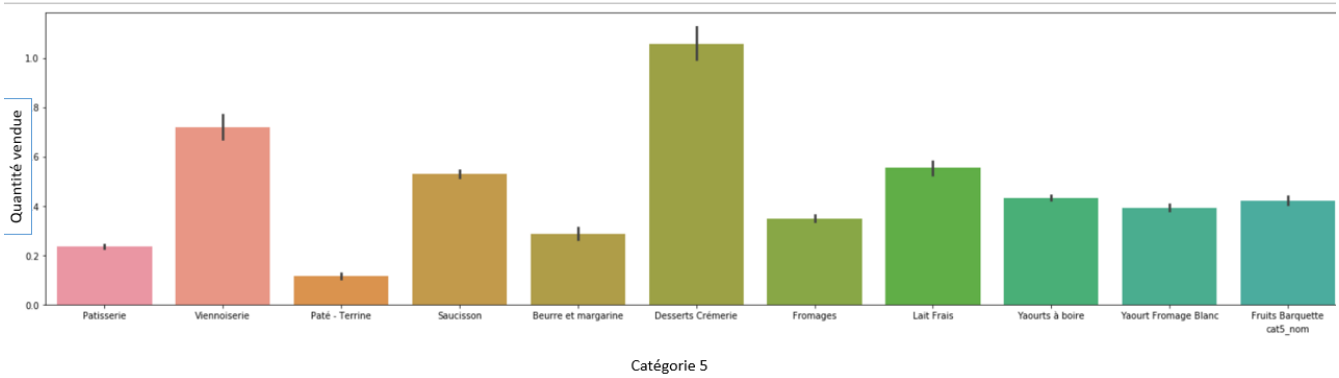


Figure 11: Ventes cumulées en fonction de la catégorie 5 (partie 1)

Il est clair qu'il y a une différence de ventes en fonction de la catégorie. La catégorie *Sandwich* ou encore *Dessert Crémerie* domine, tandis que la catégorie *pâté* et *autre charcuterie* font de faibles ventes. Cependant comme on peut le voir sur la **Figure 18** (Annexe 7.1), il y a peu de différence entre les sous-catégories, en particulier entre la catégorie 5 et 6 qui paraissent être presque les mêmes. Cela s'explique par l'inclusion des catégories.
En réalisant les tests de modélisation, nous avons supprimé la catégorie 5 pour éviter la redondance des données. Cela n'a pas affecté nos modèles. Ainsi on conservait le regroupement par catégorie qui permet

d'avoir une version plus grossière de nos produits, et un certain clustering, tout en limitant le nombre de variables pour éviter le *Fléau de la dimension* (*"curse of dimensionality"*).

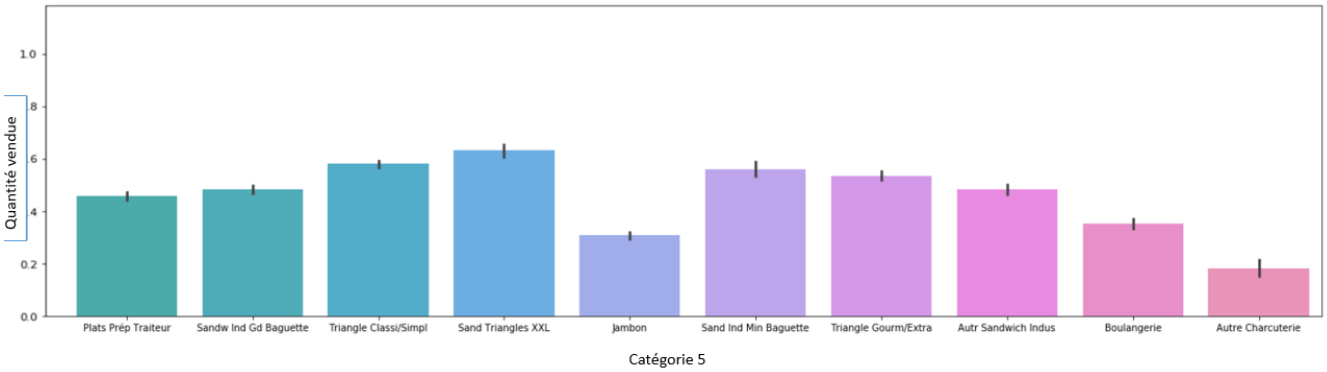


Figure 12: Ventes cumulées en fonction de la catégorie 5 (partie 2)

4.3 Transformation des données

4.3.1 One Hot Encoding

Le One Hot Encoding est une transformation des données catégorielles en vecteur binaire. La **Figure 13** représente une telle transformation. Le principe est de créer un ensemble de nouvelles variables représentant

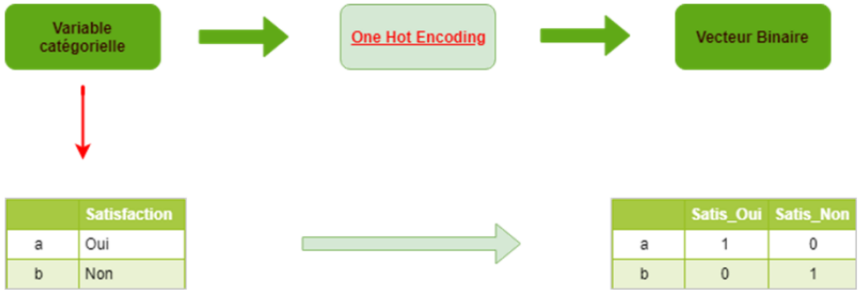


Figure 13: Principe du One Hot Encoding

une variable catégorielle (une variable par possibilité différente) prenant des valeurs binaires (0 ou 1). Il est important de faire cette transformation car plusieurs algorithme de machine learning n'arrivent pas à travailler avec les variables catégorielles. En l'occurrence, concernant notre projet les modèles suivants ne supportent pas les variables catégorielles:

- Modèle linéaire
- Réseau de Neurone

Cependant il y a plusieurs désavantage à cette technique:

- L'augmentation du nombre de dimensions qui est la cause de la *"curse of dimensionality"*. Le traitement de données devient plus difficile et long, car les données prennent plus de place en mémoire
- Les éléments du vecteurs sont équidistants, impliquant que chaque modalité d'une variable catégorielle ait la même importance ce qui n'est pas le cas. (Par exemple, le fait qu'il y a des vacances en zone A, B et C est plus important que le fait qu'il n'y ait pas de vacances).
- Le vecteur binaire contient une information sémantique encodée, ce qui rend plus difficile la compréhension.

Cette méthode étant simple à appliquer et permettant l'utilisation directe de la majorité des algorithme, nous allons l'utiliser pour tous nos modèles dans la suite du projet.

4.3.2 Normalisation des données

De même que pour le One Hot Encoding, il est nécessaire (ou préférable) pour certains algorithmes, en particulier le Réseau de Neurones, d'avoir des données normalisées. En effet le Réseau de Neurones ne prend en entrée que des données normalisées. Nous avons donc normalisé par rapport à la différence minimum/ maximum pour chaque variable numérique.
La normalisation permet aussi d'avoir une convergence plus rapide pour les algorithmes basés sur de l'optimisation par gradient (Gradient boosting). Ce dernier se base sur une step size (le gradient de la fonction d'erreur multiplié par le taux d'apprentissage), cependant si les variables ne sont pas sur la même échelle, les step sizes sont différentes (car le gradient est différent), rendant la méthode de gradient boosting plus longue. De plus elle permet d'avoir de meilleurs résultats pour le modèle linéaire, car celui-ci est moins affecté par les grandes variations. Les variables ayant un grand intervalle de définition vont induire une grande variance au modèle, ce qui leur donne trop d'importance.
Pour toute ces raisons, nous appliquons une normalisation des données.

5 Modélisation

Le code lié à cette partie est disponible sur le lien suivant `https://github.com/Syndorik/ChallengeTotal/blob/master/Data-Engineering-and_Modeling.ipynb`.

Dans cette section, nous présentons l'ensemble des modèles qui ont été testés pour modéliser le problème posé par Argedis.

5.1 Modélisation basée sur le Data Engineering

Le modèle doit répondre à un problème de régression de machine learning supervisé. Ici, on considère la variable comme continue (classement supervisé), bien qu'elle soit en réalité discontinue. En effet, comme précisé en Introduction, la quantité vendue d'un produit est le multiple d'un certain nombre propre au produit. Ainsi dans l'idéal, un modèle aurait été développé par produit. Dans ce cas là, le problème aurait été un problème de classification. Cependant, au vu de la faible taille de notre jeu de données, il peut être compliqué de faire un modèle par produit. On a choisi de garder comme base d'entraînement l'ensemble des produits.
Une autre possibilité aurait été de faire un modèle apprenant sur les périodes de travail (hors vacances) et un modèle apprenant sur les vacances. Comme nous l'avons vu, les périodes de vacances sont anormales : périodes de haute variance de quantité d'articles vendues. Donc cela va avoir tendance à perturber les modèles. Cependant nous n'avions qu'une année et demi de données. La période trop courte pour pouvoir faire cela. Il est à noter que les Time Series vont nous aider sur cette problématique. Nous allons détailler l'ensemble des modèles suivant :

- Régression linéaire
- Decision Tree Regressor
- Extra Tree Regressor
- Random Forest Regressor
- Gradient Boosting Regressor
- XGBOOST
- Multi-layer Perceptron (MLP)

5.1.1 Méthodologie d'évaluation des modèles

Comme nous l'avons énoncé en Introduction, notre modèle est évalué selon le RMSE : $\sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_t - y_t)^2}{n}}$. La méthode que nous avons utilisé pour s'assurer de la validité d'un modèle est la validation croisée (cross-validation). Le principe est d'entraîner le modèle sur une partie du jeu de données, 80% par exemple et d'utiliser les 20% restant pour le tester. Le processus est répété un nombre n de fois, et la moyenne des scores RMSE obtenus nous permet d'évaluer le modèle

5.1.2 Méthodologie d'amélioration des modèles

Chaque modèle, que nous allons expliciter dans les parties suivantes, possède des paramètres qui lui sont propre. Pour chaque modèle nous allons énoncer les paramètres les plus important. Cependant la méthodologie pour optimiser ces derniers est la même. Pour chaque combinaison de paramètre, nous allons lancer l'apprentissage d'un modèle et son évaluation par cross-validation (en RMSE).
Les paramètres du modèle ayant la meilleure évaluation seront considérés comme optimum. Dans la pratique, il existe une multitude de paramètres pour chaque algorithme, en général nous n'appliquons cette méthode que pour certains d'entre eux.

5.1.3 Régression linéaire

Ce modèle essaie de décrire les relations entre la variable à prédire et les variables d'entraînement comme linéaire (du type : $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 ...$ avec Y la variable à prédire et les X_i les variables d'entraînements). La méthode utilisée pour créer la courbe de régression est la méthode des moindres carrés. On obtient après entraînement un score R^2 de 35.4%, ce qui veut dire que ce modèle explique 35.4% de la variance. Le score RMSE est le suivant

RMSE = 0.9568

Ce score est assez faible surtout en comparaison des autres modèles. Cela est assez logique, car les relations au sein de notre jeu de données ne sont clairement pas linéaires, et il est assez difficile de modéliser des relations non linéaires avec ce modèle. C'est possible via l'introduction de termes d'interaction, mais cela reste assez primaire comme modélisation.

5.1.4 Decision Tree Regressor et Extra Tree Regressor

Ces modèles se basent sur la création d'arbres afin de fournir une prédiction. Le principe est que chaque feuille de l'arbre de décision va utiliser une variable pour effectuer une partition des données. Les arbres de décision permettent d'avoir un modèle qui est compréhensible en sortie d'apprentissage. Nous obtenons les résultats suivant :

$RMSE_{Decision-tree-Regressor} = 0.669$ et $RMSE_{Extra-tree-regressor} = 0.646$

La différence entre le Decision Tree Regressor et l'Extra Tree Regressor repose sur la manière séparer la population au sein d'un noeud afin de créer une branche. L'Extra Tree Regressor va prendre un certain nombre de variables au sein d'un noeud, regarder quelle est la meilleure manière de séparer le noeud à partir de ces variables et recommencer le processus plusieurs fois. Le meilleur *split* est alors choisi.
Cela explique pourquoi l'Extra Tree Regressor possède un meilleur score RMSE. Cependant, le grand inconvénient de ces méthodes est qu'elles ont tendance à sur-apprendre ou au contraire pas assez apprendre en fonction de la profondeur de l'arbre. C'est pourquoi nous nous sommes concentrés sur les modèles ensemblistes d'arbre de décision.
Paramètres principaux à optimiser : la profondeur de l'arbre, le nombre minimum de feuilles

5.1.5 Random Forest Regressor

Les forêts d'arbres de décision ou random forest, se basent sur la création de plusieurs arbres de décision. Le principe est de créer plusieurs arbres de décision de manière aléatoire : sélection aléatoire des variable et du jeu d'apprentissage (bootstrap sample). Une fois l'apprentissage des arbres fini, il suffit de moyenner la sortie de chaque arbre de décision pour obtenir la variable à prédire. C'est le principe du bagging.Nous obtenons les résultats suivants :

$RMSE = 0.57997$

Le score obtenu est meilleur que pour les précédents modèles utilisés. Cela s'explique par l'aspect général du modèle qui permet de compenser les problèmes de sur-apprentissage que l'on peut rencontrer avec un arbre de décision unique. L'inconvénient de cette méthode est le temps d'exécution qui peut être assez long.
Paramètres principaux à optimiser : la profondeur de l'arbre, le nombre minimum de feuilles, le nombre d'arbres maximum.

5.1.6 Gradient Boosting et XGBOOST Regressor

Ces deux modèles sont des versions modifiées des forêts d'arbres de décision. Au lieu de se baser sur du bagging pour créer les arbres de décisions, ils se basent sur des méthodes de gradient boosting. Le principe est de créer des arbres de décisions à la chaîne. Cependant entre chaque création d'arbre, l'erreur est minimisée en utilisant la méthode de descente du gradient. Nous obtenons les résultats suivant :

$$RMSE = 0.57909$$

Le score obtenu est meilleur que pour les forêts d'arbres aléatoires. Cependant, la différence se fait au millième du RMSE. Sachant que notre prédiction est continue alors que la quantité vendue par produit prend des valeurs discontinue, la différence obtenue n'est pas significative. En effet l'erreur due à l'échantillonnage est plus grande.
On peut trouver une explication à ces résultats par le fait que les deux algorithmes reposent sur la création d'arbres de décision. Les forêts d'arbres aléatoires vont en général essayer de réduire la variance tandis que les méthodes de boosting vont essayer de réduire le biais. Ici, il apparaît que dans les deux cas nous obtenons une erreur similaire.
Paramètres principaux à optimiser : le taux d'apprentissage, la profondeur de l'arbre, le nombre minimum de feuilles, le nombre d'arbres maximum.

5.1.7 Multi-layer Perceptron (MLP)

Cet algorithme repose sur la création d'un réseau de neurones. Il va apprendre une fonction à partir du jeu de données. L'avantage est la capacité de modéliser les non linéarités. Le réseau de neurones comporte 4 couches dont deux cachées. La première est l'input layer, la seconde est composée de 32 neurones, la deuxième de 16 neurones, et la dernière de un neurone (correspondant à la variable à prédire). La fonction d'activation utilisée est la fonction "relu".

$$RMSE = 0.685$$

Le score obtenu est moins bon que pour les autres modèles testés auparavant. Cela est probablement dû au fait que notre réseau de neurones n'est pas adapté à notre problème et qu'il pourrait être optimisé. Cependant le temps d'apprentissage du réseau étant relativement long, nous n'avons pas insisté sur cette voie.

5.1.8 Stacking

Cette méthode ensembliste permet de combiner les meilleurs modèles précédemment créés afin d'obtenir un modèle plus performant. Dans notre cas, nous avons pris 3 modèles différents, le modèle linéaire, le modèle de forêts aléatoire et le modèle Gradient Boosting. Le principe est décrit en **Figure 14**. Le jeu de données est divisé en différentes bases d'apprentissages pour différents modèles. Les prédictions faites à partir des modèles vont ensuite servir de base d'apprentissage au meta-modèle. Dans notre cas ce sera un XGBOOST. Nous obtenons alors le résultat suivant :

$$RMSE = 0.5735$$

Le score obtenu est meilleur mais ne varie que de quelques centièmes. En effet, le stacking est utile pour combiner des modèles qui ont des zones d'erreurs assez différentes. Les modèles basés sur les forêts d'arbres de décision et les modèles basés sur XGBOOST ont des zones d'erreurs assez similaires. Ainsi, nos modèles sont trop similaires pour bénéficier du stacking.

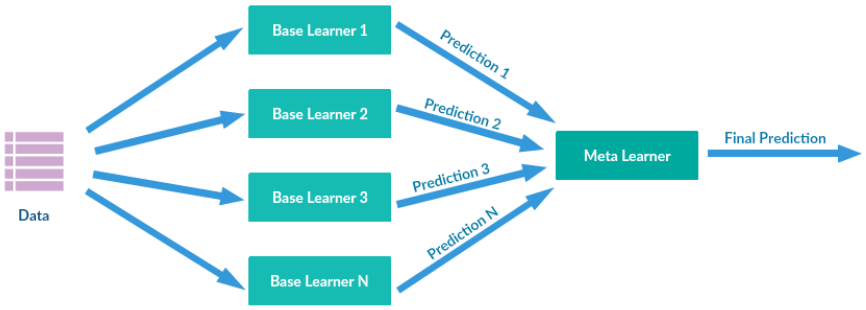


Figure 14: Principe du stacking

5.1.9 Post-traitement

Nous avons implémenté deux post-traitements en vue d'améliorer la qualité de notre modèle.

- Il existe des produits qui étaient disponibles dans une station mais pas dans l'autre, et vice-versa. Nous savons dès lors, que s'il est demandé de prédire le nombre de vente d'un produit qui n'existe pas dans la station, il sera prédit automatiquement à 0.
- Nous savons qu'un produit ne peut prendre comme valeurs de quantité vendue que des multiples d'une quantité de base. Cette quantité de base est différente pour chaque produit. Ainsi, nous avons tenté d'approximer chaque quantité vendue, soit par sa valeur de multiple supérieure, soit par sa valeur de multiple inférieure.

Le premier point du post-traitement est assez efficace et améliore effectivement les prédictions des modèles. Le deuxième point est bien plus problématique. Nous l'avons testé sur les résultats du modèle stacké et nous sommes passés d'un RMSE de 0.5735 à un RMSE de 0.72. Nous nous attendions à une régression légère du score RMSE, cependant ici cela est bien trop grand. Il y a plusieurs explications possibles. Tout d'abord, notre post-traitement se trompe pour chaque produit et retourne toujours la mauvaise valeur (ie il retourne la valeur au-dessus quand la valeur réelle est en-dessous et inversement). Cette première explication semble assez improbable. Une seconde explication serait une mauvaise compréhension de la variable quantité vendue. En effet, aucune information dans le jeu de données ne nous dit ce qu'elle représente exactement.

5.2 Modélisation basée sur les Time Series

> Le code lié à cette partie est disponible sur le lien suivant `https://github.com/Syndorik/ChallengeTotal/blob/master/Time_series.ipynb`.

Ce modèle est spécial dans le sens où il va se baser uniquement sur l'évolution des produits vendus sur le temps. Ainsi, l'ensemble des préparations des données n'est pas nécessaire. Nous utilisons le module *Prophet*. Ce modèle va se baser sur les évènements passés pour prévoir les évènements futurs à partir d'une analyse des séries temporelles. Par ailleurs, nous pouvons indiquer des périodes spéciales au modèle, en particulier les périodes de vacances. Les résultats sont visibles sur la **Figure 15**. Elle représente l'erreur par rapport à la réalité en terme de quantité vendue, sur la période de test. Les résultats suivant sont obtenus:

> RMSE = 1.3479

Nous pouvons remarquer que les grandes vacances, et en soit l'ensemble des vacances indiquées (jour fériés inclus) sont très bien prédites par le modèle, l'erreur est presque nulle. A l'inverse le reste de l'année, n'est pas bien prédit. En effet, les quantités vendues pendant les vacances sont tellement grandes que le

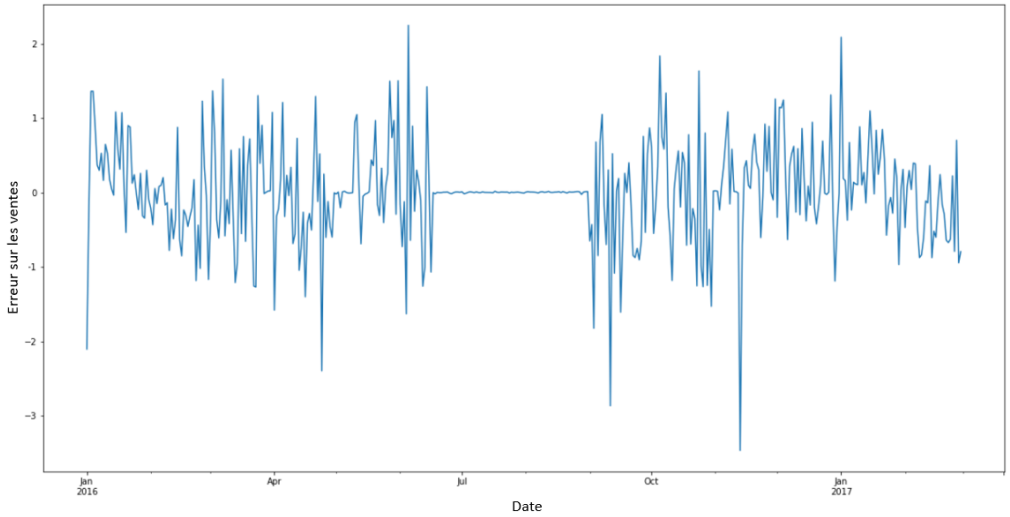


Figure 15: Prévision des ventes

reste de l'année apparaît comme du bruit. Ainsi le modèle n'arrive pas à bien approximer ces périodes de l'année. Même si le score RMSE est assez mauvais, il est très intéressant d'utiliser les Time Series pour prédire les vacances, tandis qu'on peut utiliser une méthode ensembliste vu précédemment pour prédire les période "normale" de l'année.

5.3 Critique des résultats

L'étude produite au cours de cette année aurait pu être plus fructueuse en terme de résultat. Il serait probablement intéressant de voir si la création d'un modèle par produit donne de meilleurs résultats que ce qui a été fait ici (ie la création d'un unique modèle). En effet les ventes sont très variable d'un produit à l'autre. Ainsi en ne se concentrant que sur un seul, nous pourrions éliminer cette erreur de variance.

6 Conclusion

Nous avons cherché tout au long de ce projet à minimiser l'écart RMSE entre nos prédictions et la valeur réelle des ventes de tous les produits. Après avoir visualisé les données puis les avoir préparées, nous avons testé différents modèles. Notre meilleur résultat est obtenu par Gradient Boosting. En réalisant du stacking à partir du modèle linéaire, du modèle de forêts aléatoire et du modèle Gradient Boosting, on arrive à améliorer légèrement nos résultats. L'utilisation de time series pour les vacances et jours fériés puis de post-processing, lorsque l'on sait par avance qu'un produit n'est pas présent, permet d'améliorer encore notre meilleur résultat.

Notre modélisation a été, cependant, réalisée sur l'ensemble de tous les produits disponibles à la vente. Une modélisation par produit pourrait permettre d'améliorer sensiblement notre score. Il est donc possible de prédire la vente de produits d'une station-service avec une précision fine. L'utilisation de nos modèles pourrait aider Argedis à augmenter sensiblement ses résultats financiers tout en réduisant son gaspillage et ses invendus.

7 Annexe

7.1 Vente par Catégorie de produit

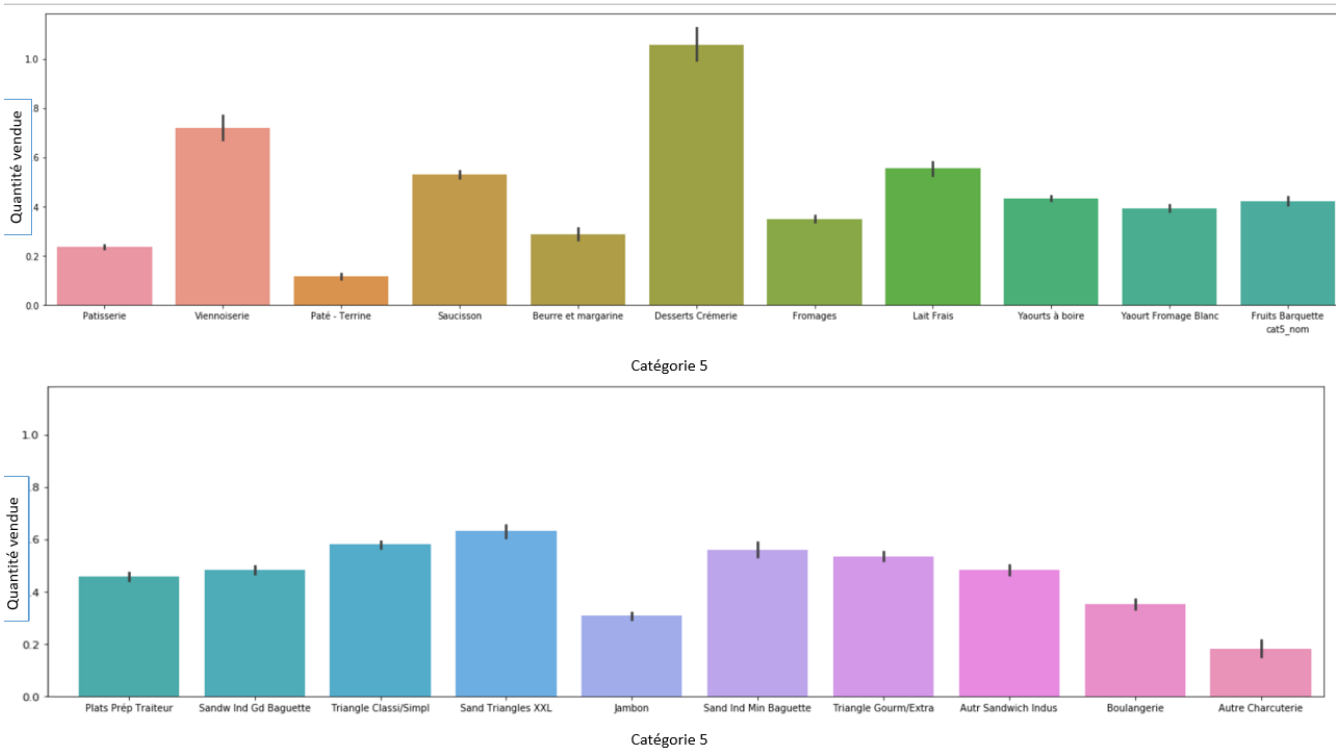


Figure 16: Vente cumulé en fonction de la catégorie 5

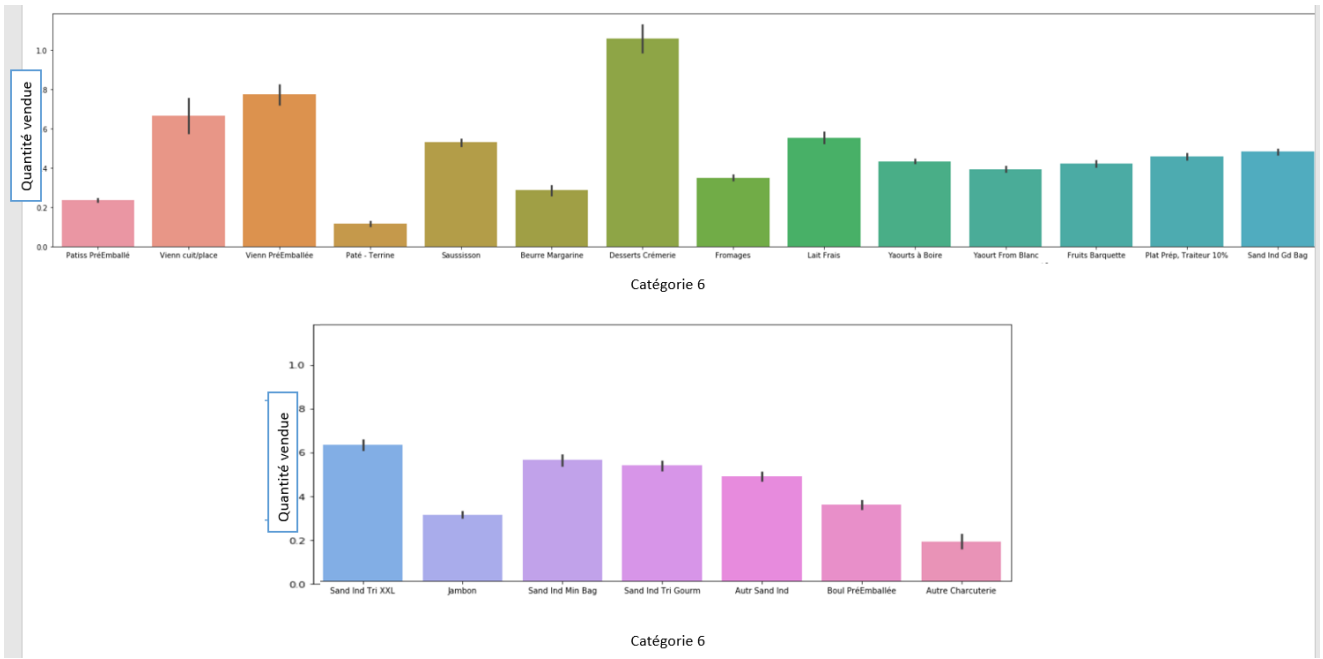


Figure 17: Vente cumulé en fonction de la catégorie 6

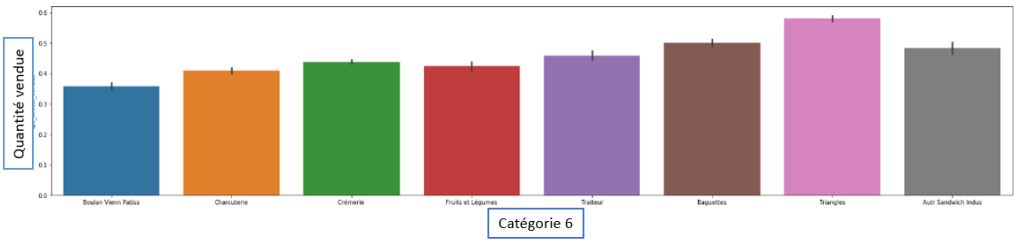


Figure 18: Vente cumulé en fonction de la catégorie 4