DataPic –Prédiction de ventes

Alexandre Brasseur, Matthieu Glorion, Lorys Hamadache

2018-2019

Table des matières

Résumé

Ce document restitue le travail de notre groupe dans le cadre du Projet de Recherche (PR) **DataPic** réalisé à l'Université de Technologie de Compiègne (*UTC*). Notre groupe s'intéresse à la prévision des ventes du foyer étudiant de l'UTC, le **Pic'asso**, afin de pouvoir gérer l'approvisionnement de celui ci. En effet chaque jour de la semaine des centaines d'étudiants passent au Pic'asso et consomment de la nourriture et des boissons. Les stocks se vident alors assez rapidement et il est difficile pour les responsables de l'approvisionnement d'estimer correctement la demande. L'objectif de notre projet est de construire un outil d'aide à la décision par prévision des ventes.

1 Introduction

1.1 Contexte

Le contexte est propre à notre foyer étudiant au début du semestre d'Automne 2019. Les Responsables en Approvisionnement du Pic'asso ont quelques difficultés pour bien gérer les stocks du foyer. En effet lors des premières semaines de prise de poste, il est difficile d'évaluer les quantités nécessaires pour satisfaire les étudiants. De plus, la demande varie, parfois fortement, de semaine en semaine en suivant certaines tendances. Le prédicteur créé par Sylvain Marchienne, lors d'un projet antérieur, permet d'avoir une idée globale des ventes mais n'est pas encore assez précis pour aiguiller correctement les choix d'approvisionnement. De plus il n'est actuellement pas pensé pour être utilisé à des fins de production, puisqu'il ne s'agit encore que d'un prototype.

1.2 Objectifs

Le but principal de ce projet de recherche est de fournir une solution concrète pour les responsables en approvisionnement. Nous allons donc développer un outil d'aide à la décision permettant de les aider pour les commandes et la gestion des stocks. Pour cela plusieurs objectifs sont définis.

D'abord on cherche à améliorer la précision du prédicteur. Des paramètres non étudiées expliquent certains écarts que nous avons pu observer, par exemple lors de la veille d'évènement utcéen comme le Gala. Pour cela plusieurs pistes sont à explorer : utiliser de nouveaux modèles plus adaptés aux séries temporelles, retraiter les données brutes d'une meilleure façon ou bien prendre en compte plus de paramètres influençant les ventes, comme la météo et les événements associatifs, des paramètres difficiles d'accès mais cruciaux car impactant fortement les UTCéens et leurs consommations.

Ensuite on veut créer un outil simple d'utilisation présentant les stocks et les prédictions de ventes de manière efficace. Des indicateurs et des analyses statistiques divers pourront être présents afin d'apporter une meilleure compréhension des données. Cela pourrait aussi nous aider à améliorer, dans le futur, la précision du prédicteur.

Enfin, pour une meilleure intégration dans le système informatique de l'association, il faudrait créer une interface utilisateur commune DataPic, avec tout ces outils, destinée à l'équipe du Pic'asso. D'autres modules DataPic potentiellement créés dans le futur, comme la détection d'anomalie, pourraient alors s'ajouter facilement à l'outil qui pourra et devra être modulaire.

1.3 Moyens à disposition

A notre disposition, nous avons toutes les ressources disponibles sur internet. Nous travaillons à l'aide de nos ordinateurs portable sous nos différents IDE pour Python. Nous avons choisi Python car c'est un language de prototypage rapide, simple, et adapté à la modélisation statistiques grâce aux multiples bibliothèques disponibles. Les bibliothèques que nous utiliserons le plus sont `numpy` et `pandas` pour les structures des données en DataFrame et Series et les différents opérateurs associés, ainsi que `statsmodels` qui comprends des fonctions faciles d'utilisation pour l'apprentissage et la modélisation statistique. Celles-ci sont toutes open-sources ce qui permet de faciliter le développement du projet. Les documentations sont correctes mais manque de profondeur. Elles s'adressent à un public de connaisseur. Nous avons eu quelques problèmes pour la gestion de la bibliothèque `statsmodels` dont la plupart des dépendances n'ont pas été mis à jour, par exemple, causant dans les premiers jours des incompatibilités entre `numpy` et `statsmodels` nous obligeant à réaliser des downgrades de certaines bibliothèques.

Ensuite nous avons accès aux données des ventes du Pic'asso comme nous le verrons dans la partie 2.1.1.

1.4 Répartition du travail

Alexandre ayant déjà eu de l'expérience avec les transaction Weezevent, s'est occupé de la collecte et du traitement des données. Lorys et Alexandre sont en FDD, ils se sont donc occupés de la partie analyse de données et modélisation 3. Matthieu étant en filière ADEL, il s'est attelé à la création de l'interface de présentation des prédictions et de gestion des stocks.

1.5 Plan

Notre rapport se composera des grandes parties suivantes :

1. Cette brève introduction
2. La collecte des données
3. Les séries temporelles et les modèles associés
4. L'interface utilisateur
5. Conclusion

2 Collecte des données

Avant tout, il s'agit de collecter les données utiles à la modélisation et de les nettoyer. Comme nous pensons obtenir de meilleurs résultats en incluant de meilleures variables explicatives, il nous faut les trouver.

Nous collecterons et traiterons les données de ventes dans un premier temps, et les données événementielles dans un second temps.

2.1 Données de ventes

Les données de ventes antérieurs sont essentielles car elles nous servent pour l'apprentissage et les tests de nos modèles et la découverte des paramètres adéquat. Tout cela dans le but de prédire les prochaines ventes.. Un traitement adéquat et des données adaptées sont alors nécessaires pour obtenir de bons résultats.

2.1.1 Source des données et récupération

Le prestataire de gestion des ventes est https ://weezevent.com/. Il s'agit d'une entreprise proposant des services de billetterie, de paiement sans contact pour les événements. Le Pic'asso n'utilise que le paiement sans contact développé initialement à l'UTC et appelé PayUTC.

Cinq ans de données de ventes sont disponibles, d'Automne 2014 à aujourd'hui, Printemps 2019. Chaque vente correspond à une transaction contenant les variables d'intérêt du tableau 1.

Variable	Description
id	L'identifiant de transaction
created	L'horodatage de création
validated	L'horodatage de validation
status	Le statut de la transaction (`'V'` pour validée)
rows	Une transaction peut avoir plusieurs produits achetés
rows.item	L'identifiant du produit acheté
rows.unit_price	Le prix unitaire du produit en centimes
rows.quantity	La quantité achetée

TABLE 1 – Description d'une transaction

Les données sont récupérables de deux façons différentes :

— de manière statique : en téléchargeant un export via l'interface d'administration
— de manière dynamique : via une API web (Interface de Programmation d'Application en ligne)

Sylvain Marchienne récupérait les données par export. C'est une solution pratique pour un prototype. Cependant ici l'un des objectifs étant la mise en production, une récupération des données de façon automatique est nécessaire.

Pour cela nous avons développé :

— un client intéragissant avec l'API PayUTC nommé PayutcClient
— un collecteur de données PayutcCollector utilisant le client précédent

Pour une période de collecte choisie, le client s'occupe de réaliser l'appel HTTP et de transmettre les données brutes au collecteur. Ce dernier traite chaque transaction et compte le nombre de chaque produit vendue par jour. On obtient alors un dictionnaire de quantité vendue par produit pour chaque jour collecté.

Les quantités vendues par jour correspondent à un format idéal. Il s'agit du grain le plus fin qui nous intéresse, en effet il n'y a pas d'intérêt à prédire les ventes sur seulement quelques heures. Cela permet de les agréger facilement en semaine si besoin.

Les ventes de nos produits sont repartis sur 5 ans soit 10 semestres. Concernant les produits, il est aussi possible de les récupérer de la même façon avec des points d'accès à l'API différents. Nous obtenons alors 1153 produits, contenant les variables d'intérêt du tableau 2.

Variable	Description
id	L'identifiant du produit
active	Si le produit est disponible à la vente
alcool	Si le produit est alcoolisé
category	L'identifiant de la catégorie du produit
image_url	L'url de l'image du produit
name	Le nom du produit
price	Le prix de vente
removed	Si le produit a été supprimé

Table 2 – Description d'un produit

Chaque produit appartient à une catégorie qui le définit. Cette catégorie a un identifiant numérique unique et un nom.

2.1.2 Nettoyage des données de ventes

Une fois les données brutes récupérées il faut les nettoyer et les transformer dans un format pratique à utiliser. Pour cela nous réalisons plusieurs pré-traitements.

Choix des catégories de produits intéressantes

Sur les 1153 produits récupérés, tous ne sont pas intéressants à prédire. En effet il y a beaucoup de produits très éphémères, comme les menus, ou alors vendus uniquement lors d'un événement, comme une place pour l'Associathon. Nous pouvons facilement repérer les produits intéressants par leur catégorie.

Ainsi, seuls les produits des catégories suivantes seront retenus :

— Bières bouteilles
— Bières pression
— Softs
— Snacks Salés
— Snacks Sucrés
— Glacé
— Repas
— Fruits & Jus frais
— Petit Dej
— Pampryl

Il reste alors 435 produits de catégorie intéressante.

Regroupement de produits similaires

Ensuite, nous nous apercevons que parmi les produits restants, certains ont le même nom à quelques lettres près et correspondent donc au même produit vendu mais avec des identifiants différents. Afin de récupérer le maximum de données par produit réel, il est préférable de les traiter sous un seul identifiant. Il faut alors détecter les doublons et les regrouper.

Pour détecter les doublons, nous appliquons la distance de Levenshtein à chaque paire de nom. Cette distance correspond au nombre minimal de charactères à changer dans un mot pour obtenir l'autre. Par une simple exploration des données, nous nous apercevons que les paires de mots ayant une distance de 2 ou moins correspondent au même produit. Nous obtenons alors une liste des similarités. Pour chaque couple il suffit alors de garder l'identifiant le plus bas (soit le plus ancien car les ids sont auto-incrémentés) et de garder une liste des ids similaires par produit. Nous obtenons alors 376 produits uniques

Nous créons alors une map `{id_similaire => id_unique}` qui sert lors du traitement

des transactions à utiliser le bon identifiant unique.

2.2 Événements et autres données

Une de nos idées pour améliorer les prédictions est d'apporter de meilleures variables explicatives. Au début du projet, Sylvain Marchienne utilisait des régresseurs avec les données de ventes et le calendrier UTCéen qui était ajouté à la main.

Nous avons alors pensé à plusieurs améliorations possibles :

1. Créer un format plus automatisé pour le calendrier UTCéen
2. Incorporer la popularité des événements à partir du groupe Facebook UTC
3. Ajouter des informations sur les produits, ainsi que leur notation à partir de différents sites spécialisés
4. Intégrer des informations météorologiques

2.2.1 Calendrier UTCéen

Collecter le calendrier UTCéen n'est pas facile. A notre connaissance, il n'y a pas d'API proposant le calendrier universitaire ou associatif. La seule solution est de récolter ces données à la main. Ajouter jour par jour les événements dans un DataFrame python est laborieux, c'est pourquoi il fallait trouver un format plus facile à manipuler. Ainsi nous avons créer un tableur CSV contenant la liste de tous les événements UTCéens avec leur date de début et de fin, et leur type.

Ensuite chaque événement est collecté par un `EventsCollector` qui concatène les événements dans un calendrier par jour sous forme de DataFrame utilisable facilement par la suite.

2.2.2 Météo et Produits

Concernant la météo et les caractéristiques de produits, nous n'avons pas eu le temps de nous en occuper. De plus les APIs sont restreintes et parfois payantes.

2.3 Confidentialité des données

Les données manipulées étant des transactions, elles peuvent être sensibles. Cependant nous ne collectons que ce qui est nécessaire : les ventes de chaque produit par jour. Ce ne sont pas des données personnelles.

De plus les données sont sécurisées. L'accès au code Gitlab est restreint et la base de

données Arango est sécurisée et contrôlée. De même les clés nécessaires pour récupérer les transactions depuis Weezevent sont gardées secrètes.

3 Séries Temporelles et Modèles de prédiction

Commençons par rappeler le format de données actuel. Nos données sont un ensemble de séries temporelles, par jours, du nombre de ventes par produit sur une période de 5 ans, de A14 à P19. Elles sont stockées dans un DataFrame de la bibliothèque python *pandas*. Une colonne correspondant à un produit, identifié par son ID et chaque ligne correspond à un jour de vente, comme présenté ci-dessus.

Sales: (1595, 321)

	36	42	25	34	31	30	37	20	411	18	...	13703	13788	13843	13866	13868	13896	13789	13927	13949	13950
2014-09-08	10.0	0.0	11.0	0.0	1.0	26.0	0.0	35.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2014-09-09	7.0	0.0	17.0	6.0	3.0	14.0	4.0	27.0	0.0	2.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2014-09-10	10.0	0.0	14.0	10.0	10.0	3.0	22.0	31.0	0.0	3.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2014-09-11	5.0	1.0	14.0	12.0	10.0	15.0	15.0	26.0	0.0	2.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2014-09-12	2.0	2.0	8.0	8.0	18.0	9.0	9.0	20.0	0.0	6.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

FIGURE 1 – Aperçu du DataFrame des ventes

Le but est de pouvoir prédire le nombre de vente des différents produits pour la semaine suivante. Pour cela on ne s'intéressera qu'à un ou deux produits pour tester nos modèles. Les résultats suivants correspondent donc aux modèles appliqués à deux produits : le Pampryl Abricot et le Café. Nous les avons choisis car ils sont vendus depuis le début des ventes collectées, sont peu sujet aux ruptures de stock qui seraient difficile à prédire et fausseraient les données. Nous avons également réalisé les tests sur la vente de différentes bières mais les mauvais résultats nous ont poussé à nous retrancher sur des produits dont les ventes ont une moins grande variance et dépendant moins des évènements.

Pour le moment, nous faisons des hypothèses simplificatrices. On considérera par exemple que l'évolution des ventes d'un produit est indépendante de l'évolution des autres produits. Cela peut paraître faux à première vue puisque si les étudiants consomment un produit en particulier, ils consommeront moins un autre mais en prenant des produits tels que le café, produit d'habitué, l'hypothèse n'est pas absurde. On réalisera d'autres hypothèses au fur et à mesure de nos avancée.

3.1 Analyse exploratoire des données

Pour chaque produit, nous avons donc le nombre de vente par jour. Dans un premier temps, on note que tous les jours ne sont pas présents. Ce sont les jours où nous n'avons aucune vente. Or, lors du tracé du nombre de ventes en fonction du temps, on remarque que les ventes sur les dates manquées sont interpolées linéairement alors que les ventes sont censées être nulles.

Nous traitons rapidement ce problème en rajoutant les jours manquants au DataFrame avec un nombre de vente égal à 0 ce qui fait plus de sens.

3.1.1 Observation des ventes globales

Commençons par observer les ventes du Pampryl Abricot et du Café sur tout la période disponible.

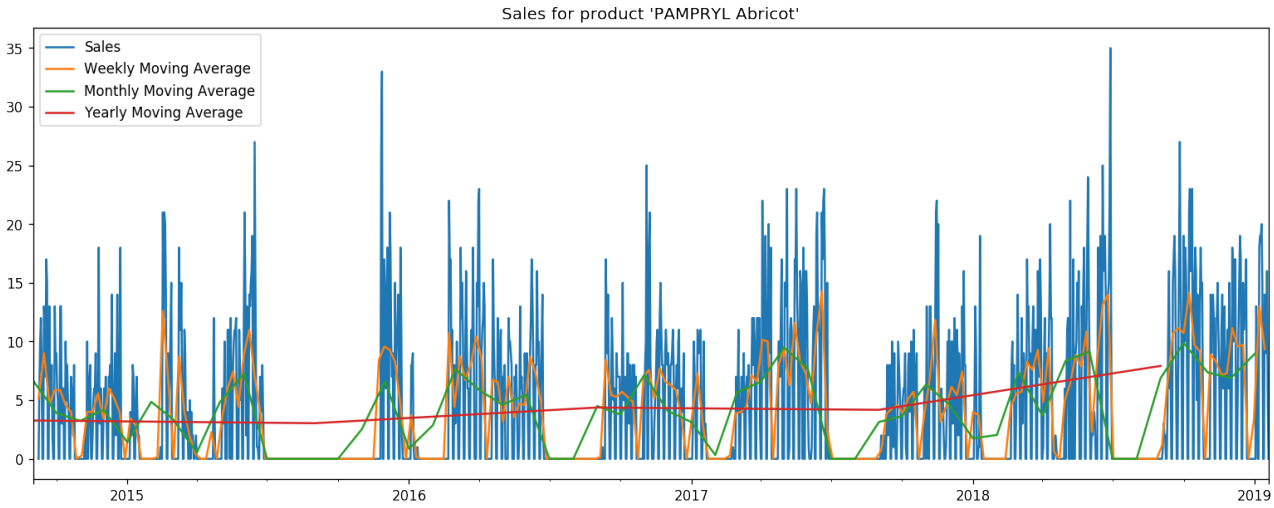

FIGURE 2 – Ventes du Pampryl Abricot

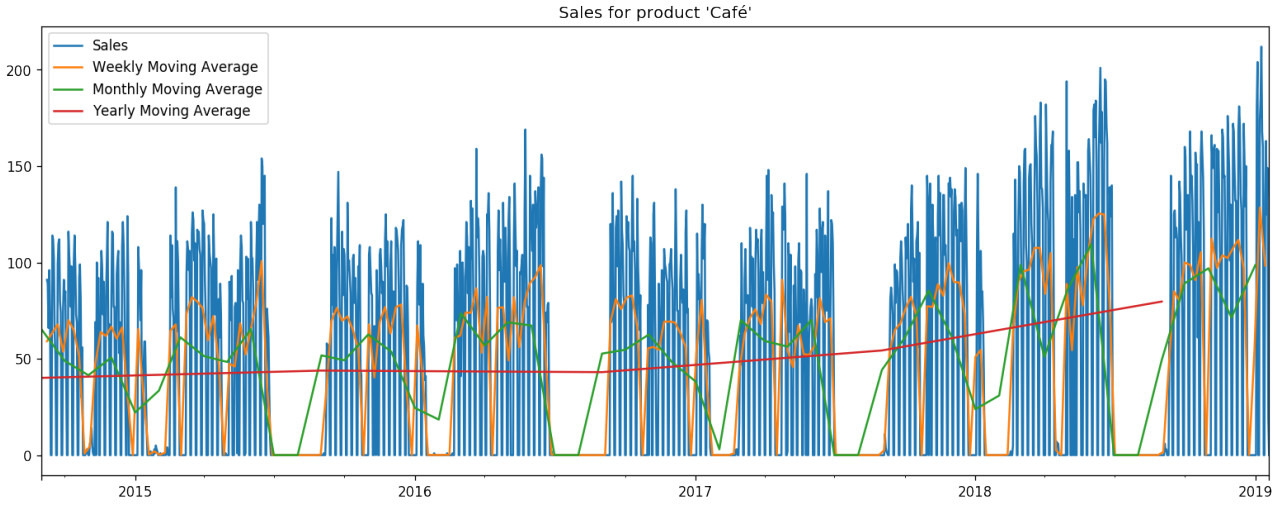

FIGURE 3 – Ventes du Café

Nous avons ajouté les moyennes par semaine, mois et années afin de mieux visualiser le comportement des ventes. Tout d'abord on observe bien sur les deux graphiques des ventes nulles lors des week-ends et des vacances, mais aussi des pics de ventes parfois au début, au

milieu ou à la fin des semestres ainsi qu'une progression du volume de vente par semestre. On note aussi quelques différences : le Café est vendu en plus grandes quantités et varie moins que le Pampryl.

3.1.2 Observation des ventes par semestres

Pour un produit, par exemple ici le Pampryl, nous traçons les ventes en fonction du temps par semestre. Cela a pour but de mettre en évidence des comportements similaires entre les semestres, par exemple s'il y a plus de consommation à l'approche des vacances. On obtient alors le graphique si dessous.

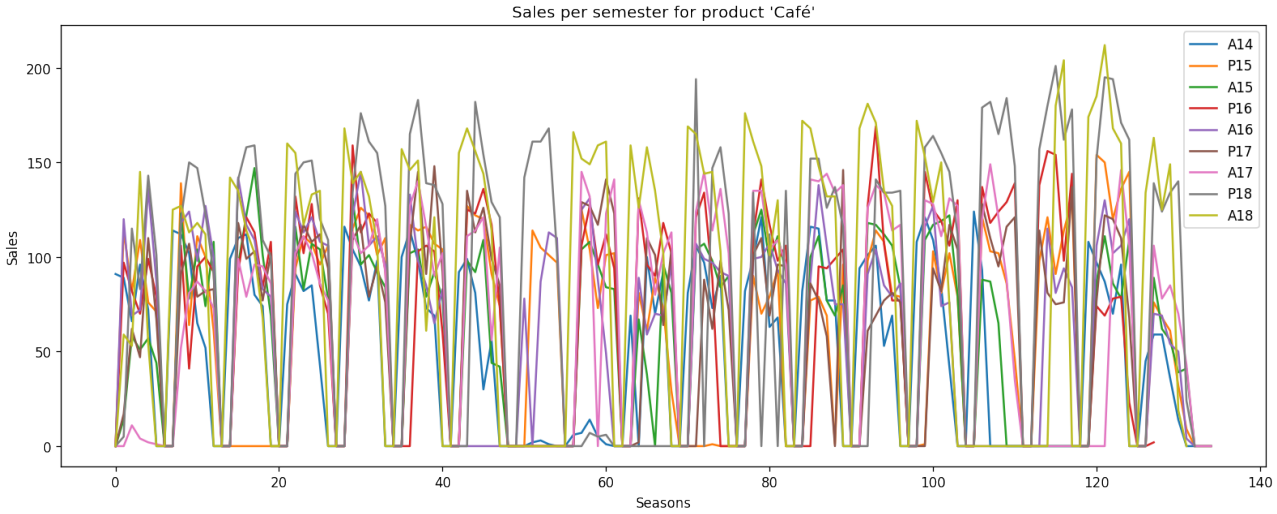

FIGURE 4 – Ventes du Café en fonction des semestres

On remarque qu'un schéma saisonnal se dessine, mais décalé différemment selon les semestres. Cela témoigne alors d'un problème majeur : les semestres n'ont pas tous le même format. Ces différences mineures vont compliquer la modélisation. De plus, un semestre ce compose d'une multitude d'évenements. Ces évenements ne sont pas présents à tous les semestres et encore moins aux mêmes dates. Cela rajoute encore davantage de problèmes.

3.2 Un premier test des modèles ARIMA

Sans connaissance sur les séries temporelles, sur recommandation de nos encadrants nous avons commencé à étudier les modèles de type ARIMA. Le principe de tels modèles est de déterminer la valeur de la série à un instant t_i en fonction des valeurs précédentes, c'est à dire aux instants t_{i-1}, t_{i-2}, ... Pour la suite nous prendrons les ventes de Café 3 comme série temporelle notée `ts`.

3.2.1 Le modèle ARIMA

ARIMAsignifiant *AutoRegressive Integrated Moving Average* est un modèle de prédiction de séries temporelles composé de plusieurs modèles.

La partie *AutoRegression* indique que le modèle comporte une régression par rapport à la série en elle même. Le modèle AR(p) est paramétré par p le nombre de retards pris en compte. On a alors :

$$\text{AR}(p) : x_i = c + \epsilon_i + \sum_{k=1}^{p} \phi_k x_{i-k} \tag{1}$$

avec x_i la valeur de la série à l'instant i, $(\phi_k)_{k=1..p}$ les paramètres du modèles et ϵ_i un bruit blanc.

Un bruit blanc ϵ est une variable aléatoire telle que :

$$\begin{cases} E(\epsilon_i) = 0 & \forall i \\ E(\epsilon_i^2) = \sigma^2 & \forall i \\ E(\epsilon_i \epsilon_j) = 0 & \forall i, j \end{cases} \tag{2}$$

La partie *Moving Average* indique que l'erreur de regression est une combinaison linéaire des erreurs précédentes. Le modèle MA(q) est paramétré par q le nombre de retards pris en compte pour l'erreur. On a :

$$\text{MA}(q) : x_i = \mu + \epsilon_i + \sum_{k=1}^{q} \theta_k \epsilon_{i-k} \tag{3}$$

avec $(\theta_k)_{k=1..p}$ les paramètres du modèles et μ la moyenne de la série.

Enfin *Integrated* indique que la série est différenciée. Pour I(d), les valeurs la séries ont été remplacées par la différence entre la valeur à l'instant i et à l'instant $i - d$.

Ces 3 composantes permettent de former un modèle complet ARIMA(p, d, q) où les paramètres correspond à ceux vus dans les modèles précédents. En revanche, ce type de modèle ne prend pas en compte la saisonnalité. Pour cela il faudra utiliser les modèles SARIMA, variante d'ARIMAsur lesquels nous reviendrons.

3.2.2 Stationnarité

ARIMAest paramétrable mais nécessite une série temporelle stationnaire. Une série temporelle (x_i) est dite stationnaire (au sens faible) si :

$$\begin{cases} E(x_i) = \mu & \forall i \\ Var(x_i) = \sigma^2 < \infty & \forall i \\ Cov(x_i, x_{i-k}) = f(k) & \forall i, k \end{cases} \tag{4}$$

Nous allons alors tester la stationnarité de notre série.

Nous avons commencer par le test **KPSS** (Kwiatkowski-Phillips-Schmidt-Shin) de la librairie *statsmodel*. L'hypothèse H_0 de ce test est que la série est stationnaire. L'application d'un tel test à `ts`pour une tendance constante ou linéraire nous donne une p-value de 0.1. On ne peut donc pas rejeter l'hypothèse de stationnarité de notre série.

Puisque l'on ne peux pas conclure de la stationnarité autour d'une tendance, nous avons eu l'idée d'utiliser un autre test. Il n'existe pas à proprement parler de tests où l'hypothèse H_0 est la non-stationnarité de la série. Un test beaucoup utilisé dans ce cas est le test de **Dickey-Fuller augmenté**. L'hypothèse H_0 est que la série ne contient pas de racine unitaire et donc qu'elle n'est pas stationnaire. L'hypothèse est validée si la série ne varie que par un bruit blanc ($x_t = x_{t-1} + \epsilon_t$). Nous obtenons les résultats suivants :

Tendance	p-value
constante (c)	1.035E-05
linéire (ct)	4.649E-05
quadratique (ctt)	1.769E-04
non constante (nc)	6.602E-03

TABLE 3 – Résultats du test de Dickey-Fuller augmenté pour le Café

On rejette donc H_0, la série temporelle ne possède pas de racine unitaire et peut être stationnaire.

Ainsi, nous ne pouvons pas conclure sur la stationnarité de la série `ts`. Nous pouvons quand même faire l'hypothèse qu'elle l'est et tenter d'appliquer le modèle ARIMAet revenir sur les hypothèses si cela ne fonctionne pas.

3.2.3 Choix des paramètres p, d, q

Pour choisir ces paramètres, plusieurs méthodes existent :

— Estimation graphique

— Recherche automatisée

Estimation graphique

Comme nous nous intéressons au nombre de ventes par jour et non à sa variation, nous fixons $d = 0$. Pour trouver des valeurs de p et q adéquates, nous avons deux outils à notre disposition. Le graphique d'autocorrélation (ACF) et le graphique d'autocorrélation partielle (PACF).

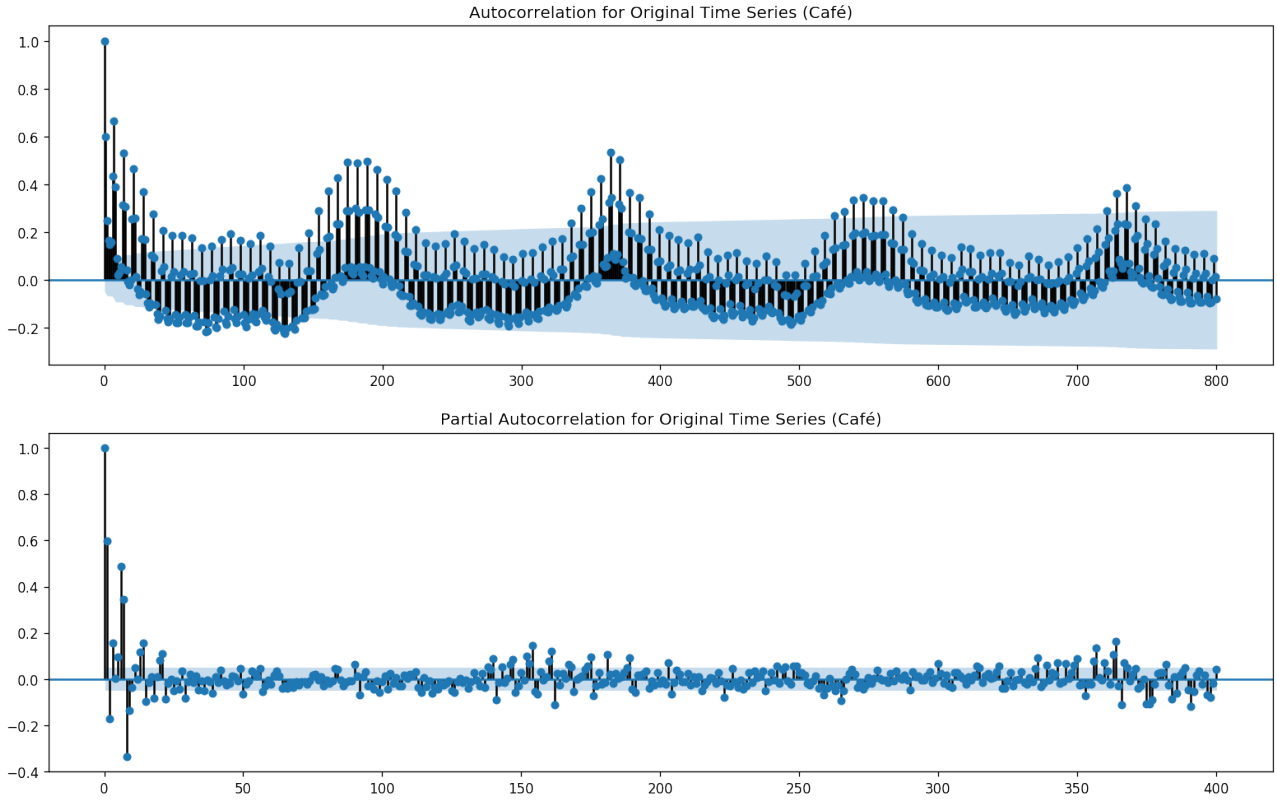

FIGURE 5 – ACF et PACF de la série Café

L'ACF nous donne l'autocorrélation entre la série et ses différents retards. On peut trouver le nombre de retards dont dépend la série et ainsi lui associer l'ordre q du modèle de la Moyenne Mobile.

Le PACF nous donne la corrélation entre la série et ses retards, non expliquée par les retards précédents. Cela nous permet de déterminer l'ordre p du modèle d'autorégression.

Nous avons donc les graphiques suivants :

La zone bleu donne les autocorrélations non significatives à 95%.

On observe sur l'ACF une périodicité par semaine (tous les 7 retards) et une autre par semestre et année (tous les 182 et 365 retards). ARIMAn'est pas capable de prendre en compte cette saisonnalité mais le modèle SARIMAoui.

Nous avons donc tester ARIMAavec toutes les combinaisons p et q égaux à 7 et 182. On remarque que le temps d'apprentissage est extrêmement long lorsque l'on utilise un ordre tel que 182. Cela peut prendre plusieurs heures sans aboutir et ce n'est pas viable pour notre recherche et notre projet. Malgré cette recherche de paramètres, les résultats sont encore mauvais. C'est pourquoi nous sommes passé au modèle SARIMA.

Recherche automatisée

La recherche automatisée consiste à définir une grille de paramètres à tester, puis de retenir la combinaison qui minimise un certain critère comme l'AIC (Critère d'information d'Akaike), le BIC (Critère d'information Bayésien) et la classique MSE (Erreur Quadratique Moyenne).

Pour cela nous avons d'abord créé la fonction `utils.models.ts_grid_search` qui effectue une recherche automatisée parallélisable avec des listes de valeurs pour les paramètres et un compte rendu détaillé des performances des modèles. Puis nous avons découvert le package `pmdarima` qui propose la méthode `auto_arima` recherche et donne le modèle optimisant l'AIC à partir d'intervalles de valeurs pour les paramètres.

Malgré ces différents indicateurs et ces différentes méthodes, nous n'avons pas trouvé de paramètres fonctionnels et encore moins optimaux. Lors de nos visualisations des résultats, les écarts entre les prévisions et les ventes étaient bien trop importantes.

Évaluation des performances

Pour évaluer les performances d'un modèle de manière complète nous avons créé la fonction `utils.models.evaluate_model`. Cette fonction facilement paramétrable prend un modèle entraîné, affiche les prédictions avec des indicateurs tels que AIC, BIC et MSE.

Pour les prédictions, deux méthodes sont possibles. La première consiste à faire une seule prédiction directement pour tout le semestre : rapide mais très peu fiable sur la durée donc peu utile. Le deuxième est un roulement de prédictions (*rolling forecast*). Il s'agit d'une boucle apprentissage/prédiction qui permet avec des données de test, d'avoir une meilleure évaluation des performances du modèle sur la série temporelle. C'est la méthode qui sera appliquée en production pour prédire jour par jour ou semaine par semaine les ventes. Le boucle est la suivante :

1. Apprentissage des données d'entraînement
2. Prédiction d'une période (1 semaine ou 1 jour)
3. Comparaison à la période réelle connue
4. Ajout de la période réelle aux données d'apprentissages
5. Ré-apprentissage du modèle avec les nouvelles données d'apprentissages
6. Retour au point 2.

Une étape supplémentaire serait de recalculer les paramètres optimaux après l'étape 5. Cela sera implémenter pour la production mais pour des raisons de rapidité de calculs nous ne le ferons pas dans nos expérimentations.

Ici nous utiliserons la méthode du roulement de prédictions avec comme données de test le semestre A18 entier.

Résultats

Le meilleur modèle ARIMAobtenu avec `auto_arima` a pour paramètres ($p = 5, d = 0, q = 5$). Avec la méthode décrite précédemment nous obtenons les résultats suivants.

AIC	14436.25
BIC	14504.94
MSE	2401.85

TABLE 4 – Résultats pour ARIMA(5,0,5) sur le Café

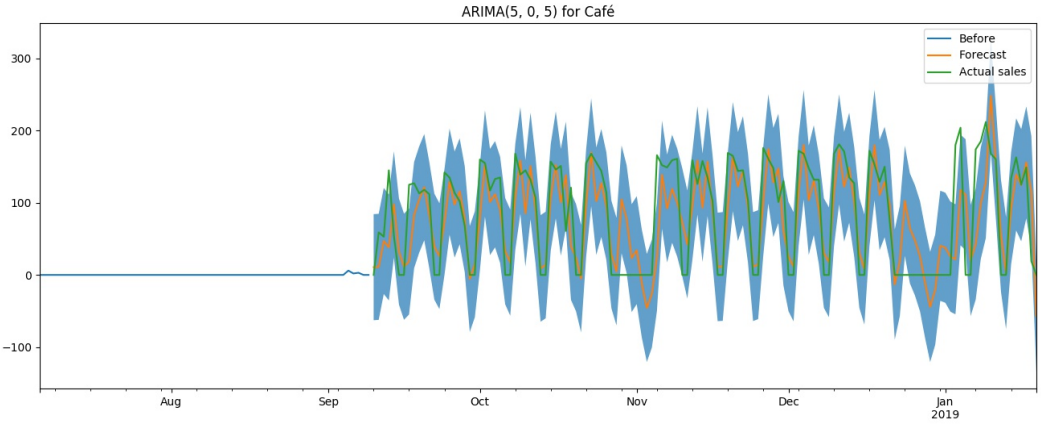

FIGURE 6 – Prédiction ARIMApour le Café

Les résultats semblent cohérents mais sont assez peu précis. Nous allons essayer d'exploiter la saisonnalité observée et d'ajouter des variables exogènes avec SARIMAXpour améliorer les performances.

3.3 Le modèle SARIMAX

3.3.1 Le modèle SARIMA, ajout de la saisonnalité

SARIMAest une variante du modèle ARIMAcomprenant une composante de saisonnalité. Nous avons désormais 4 paramètres supplémentaires :

— P : l'ordre d'autorégression saisonnale
— D : l'ordre de différentiation saisonnale
— Q : l'ordre de la moyenne mobile saisonnale
— S : le nombre de retards pour une période de la saisonnalité

Pour avoir une idée de ces paramètres, il nous faut décomposer la partie saisonnale de la série. Pour cela nous utilisons la fonction *seasonal_decompose* de *statsmodels*.

FIGURE 7 – Décomposition de la série Café

On remarque bien deux saisonnalités, l'une par semaine et l'autre par année.

Une fois la saisonalité observée, nous réalisons une recherche automatisée comme décrit précédemment 3.2.3 mais avec la composante saisonnale. Nous testons uniquement la périodicité par semaine $S = 7$ car $S = 182$ provoque un modèle trop coûteux.

Nous obtenons les résultats suivants.

AIC	14127.09
BIC	14164.00
MSE	2249.31

TABLE 5 – Résultats pour SARIMA(1, 0, 0)(3, 0, 0, 7) sur le Café

On ne note pas de grandes améliorations par rapport au modèle ARIMA6. Il peut s'agir d'un problème de paramètres ou simplement que la saisonnalité n'est pas très intéressante.

Nous allons alors tenter d'ajouter des variables explicatives suivantes au modèle SARIMAX.

3.3.2 Le modèle SARIMAX, ajout de variables exogènes

Afin de tenter d'améliorer les résultats du prédicteur, nous avons penser ajouter des variables explicatives telles que la présence de vacances ou d'examens. Grâce au travail réalisé lors de la collecte des événements 2.2, cela est possible. Nous avons donc créé un DataFrame contenant les variables exogènes pour chaque jour :

— `weekly_sin` et `weekly_cos` : un encodage cyclique du numéro du jour dans la semaine
— `yearly_sin` et `yearly_cos` : un encodage cyclique du numéro de la semaine dans l'année

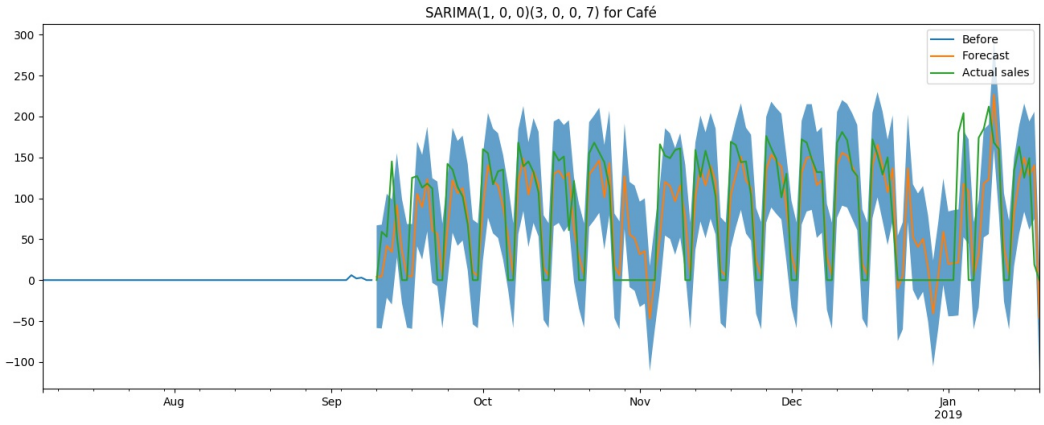

FIGURE 8 – Prédiction SARIMApour le Café

— `holiday` : la présence de vacances ou non en binaire
— `exams` : la présence d'examen ou non en binaire
— `weekofsemester` : le numéro de la semaine dans le semestre

L'encodage cyclique f_P de période P suit la fonction suivante :

$$f_P(t) = \begin{pmatrix} \sin\left(\frac{2\pi t}{P}\right) \\ \cos\left(\frac{2\pi t}{P}\right) \end{pmatrix} \tag{5}$$

Ces variables nous semblaient intéressantes pour la prédiction, nous les avons testées et selectionnées.

Nous obtenons le DataFrame exogène suivant :

	weekly_sin	weekly_cos	yearly_sin	yearly_cos	holiday	exams	weekofsemester
2018-06-20	0.974928	-0.222521	0.196673	-0.980469	0.0	0.0	18.0
2018-11-18	-0.781831	0.623490	-0.674444	0.738326	0.0	0.0	9.0
2018-05-05	-0.974928	-0.222521	0.835925	-0.548843	0.0	1.0	11.0
2015-10-04	-0.781831	0.623490	-0.998435	0.055917	0.0	0.0	4.0
2017-09-22	-0.433884	-0.900969	-0.988678	-0.150055	0.0	0.0	3.0

FIGURE 9 – Aperçu du DataFrame des variables exogènes

Nous testons alors cette configuration, et obtenons les résultats suivants.

On note une légère amélioration des résultats, on passe d'un MSE de 2401.85 pour ARIMA4 à 2015.17.

AIC	14259.78
BIC	14365.37
MSE	2015.17

TABLE 6 – Résultats pour SARIMAX(1, 0, 0)(3, 0, 0, 7) sur le Café

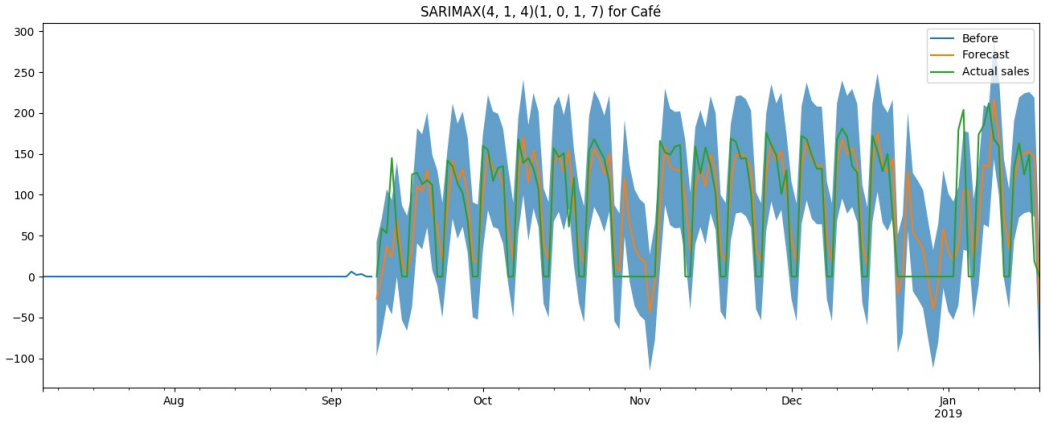

FIGURE 10 – Prédiction SARIMAXpour le Café

On s'aperçoit de plusieurs choses :

— les incertitudes sont toujours grandes
— des ventes négatives sont prédites alors que ce n'est pas possible
— la saisonnalité hebdomadaire est bien prise en compte
— l'évolution au cours du semestre semble juste
— les ventes sont trop basses

SARIMAXa des difficultés avec les ventes nuls lors des jours où le pic est fermé. En réalité, nous avons l'impression que les variables exogènes améliorent un peu les prévisions mais on se rends rapidement compte que le problème général reste. En effet, en combinant de multiples variables, on peu s'approcher d'une prédiction correcte sur notre jeu de test mais cela ne permet en aucun cas de réaliser de bonnes prédictions : c'est de l'overfitting.

3.4 Retour sur les bases et réflexions

Les prédictions semblent correctes mais comment les améliorer encore ?

3.4.1 Décomposition de la tendance

Retournons à la base du modèle ARIMA. Celui-ci fonctionne optimalement sur des séries stationnaires. Comment être sûr de la stationnarité de notre série ?

Repartons de la définition. Une série est dite stationnaire si son évolution ne dépends pas du temps. Notre série est composé d'une tendance, d'une saisonnalité et du "reste".

Dans un premier temps, il nous a paru évident que la tendance doit être étudiée à part. Elle fait évoluer la série en fonction du temps. Pour cela nous testons deux méthodes. La première, qui est la plus simple à mettre en place, consiste à soustraire la tendance de notre série et à traiter la série résultante. On utilise la décomposition comme vue précédemment et on enlève simplement la tendance. Ensuite on utilise un modèle SARIMAsur la série sans tendance. Cela nous permettra de comprendre si le modèle fonctionne mieux dans le cas où la tendance est exclus car la série devrait être davantage stationnaire. Obtenant des résultats un peu meilleur visuellement, mais sans réelle quantification, nous avons appliqué une méthode plus réaliste, plus proche de l'état de production que pourrait avoir l'utilisation d'un tel modèle. Cela s'approche de l'utilisation des modèles en production, semaine par semaine.

Voici la méthode :

1. Décomposition de la série avec 1 semestre de test et le reste pour l'entraînement.
2. Séparation de la série d'entraînement en 2 séries distinctes, la tendance et le reste.
3. Apprentissage du modèle SARIMAsur la série d'entraînement sans tendance (les paramètres sont déterminés comme précédemment)
4. Application d'un modèle de régression linéaire ou quadratique sur la tendance en fonction de sa forme.
5. Prédiction du semestre suivant avec le modèle SARIMA
6. Prédiction de la tendance avec le modèle de régression
7. Assemblage des deux prédictions (par somme) et comparaison avec la série de test.

Avec des résultats médiocres, on se demande si il faut aussi enlever la partie saisonnal. Nous savons que les modèles ARIMA ne doivent pas comporter de saisonnalité mais les modèles SARIMA sont sensés palier ce problème. Nous avons quand même essayer la même démarche en enlevant la saisonnalité (grâce à la décomposition) et en utilisant les modèles ARIMA sur le reste, sans plus de résultats bien qu'il y est moins de paramètre à déterminer.

3.4.2 Transformation de la série temporelles

On s'intéresse à l'agencement de nos données, c'est-à-dire comment sont reparties les ventes sur les différents semestres. Voici un graphique détaillé de la répartition des semestres avec les

vacances et les évènements, ainsi que des statistiques sur la durée des semestres.

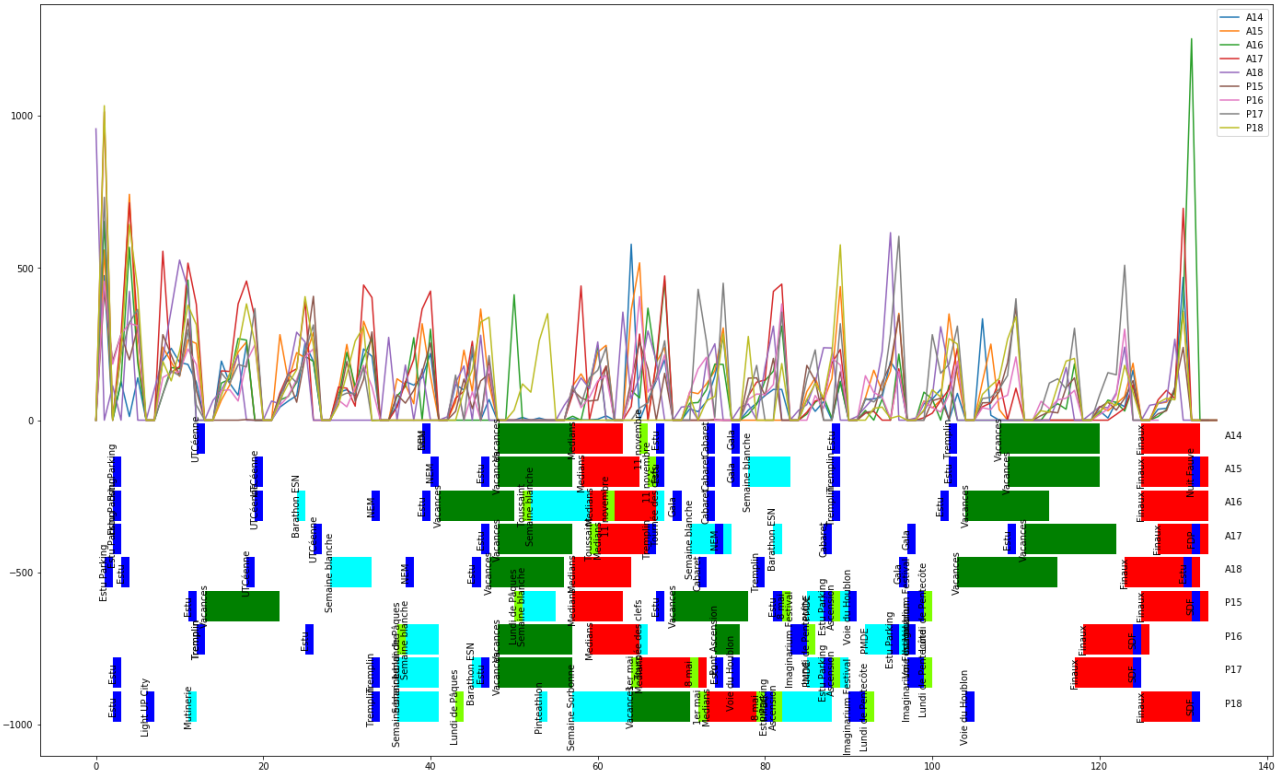

FIGURE 11 – Visualisation de l'agencement des semestres

Semestre	Jours	Semaines
A14	133	19
A15	135	21
A16	135	21
A17	135	21
A18	132	19
P15	135	21
P16	128	20
P17	127	19
P18	133	20
Moyenne	132.5	20.1

TABLE 7 – Statistiques sur la durée des semestres

Les semestres se ressemblent globalement mais ne sont pas identiques. Ils ne possèdent pas le même nombre de jours ni de semaines même si les écarts ne sont pas grands.

Reconstruction des semestres

Avec les semestres d'Automne et de Printemps, nous avons deux types d'organisation. Nous avons deux périodes de vacances sur les semestres d'Automne et une seule sur les semestres de Printemps. Cela peut poser des problèmes pour la modélisation. Ce que l'on peut faire est de séparer notre problème en deux. On pourrait travailler avec deux modèles, un pour chaque type de semestre.

Pour le nombre de semaines pas exacte, on a plusieurs solutions : dans un premier temps, nous pouvons enlever les semaines en trop tant quelle ne sont pas trop près d'évènements tels que les vacances. On peut aussi penser à interpoler lorsque des semaines manquent ou dupliquer des semaines sans évènements.

En faisant cela, il nous reste à gérer le cas du placement des évènements. Nous pourrions transformer à la main tous les évènements des semestres afin qu'ils soient ordonnées similairement. Cela pose deux soucis majeurs. Le premier, qui est le moins grave est le temps nécessaire. Celui ci doit être fait pour chaque semestre lors de l'apprentissage et dois être de nouveau fait pour les semestres qui passent. Ensuite lors du travail à la main, il faudra interpoler et/ou enlever de nombreuses semaines afin d'avoir toujours exactement le même nombre de semaines entre chaque évènement.

Pour les week-ends nous faisons simple dès le début. Nous décidons de passer à la semaine de 5 jours et d'enlever les quelques ventes des samedis. Les vacances sont enlevées tout en réfléchissant aux positionnements des différentes semaines comme expliqué ci-dessus.

Nous pouvons imaginer une mise en production comme sur le diagramme si dessous : ("nous" correspond à l'apprentissage).

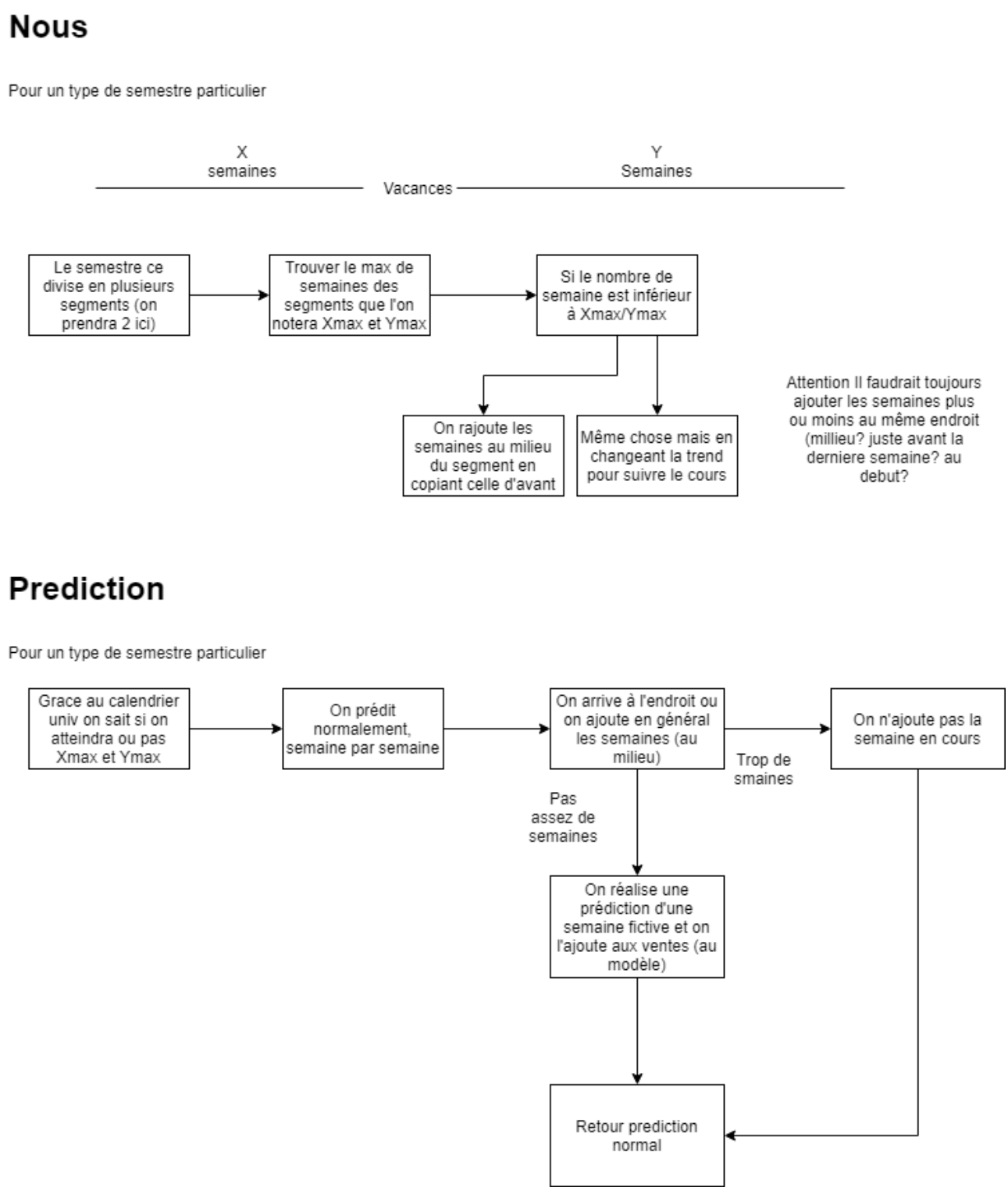

FIGURE 12 – Schéma des transformation possibles des séries temporelles

Cependant cette solution est bien trop compliquée à mettre en place et à maintenir. De plus nous ne sommes pas sûr de son efficacité. Manipuler les semaines poserait forcément des problèmes dans les prédictions et deviendrait vite ingérable.

Découpage des ventes nulles

Les ventes sont souvent nulles (environ la moitié des observations) et tendent à fausser les modèles. Que se passe-t-il si nous retirons ces données ?

Nous avons créé la fonction `utils.datetime.remap_ts` qui transforme la série selon la méthode suivante :

1. Passe les données des week-ends et des vacances à `NaN`
2. Remplace les données des jours fériés par une moyenne du même jour des deux dernières semaines

Cela permet d'enlever les données inintéressantes tout en conservant la série temporelle continue en temps.

Nous obtenons les résultats suivants :

AIC	6507.99
BIC	6587.21
MSE	1239.62

TABLE 8 – Résultats pour SARIMAX(2, 0, 2)(0, 0, 1, 7) sur le ventes transformées de Café

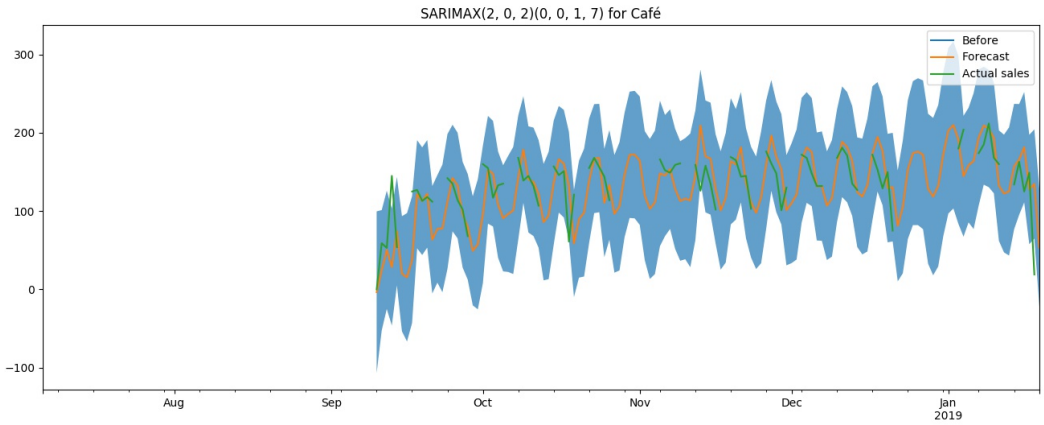

FIGURE 13 – Prédiction SARIMAX(2, 0, 2)(0, 0, 1, 7) sur le ventes transformées de Café

On gagne significativement en AIC, BIC et MSE en les minimisant de moitié, mais aussi en rapidité, les valeurs `NaN` étant tout simplement passées.

3.5 Comparaison des résultats

En comparant les résultats 4, 5, 6 et 8, nous obtenons le classement suivant :

Indicateur	ARIMA	SARIMA	SARIMAX	SARIMAXtransformé
AIC	14436.25	14127.09	14259.78	6507.99
BIC	14504.94	14164.00	14365.37	6587.21
MSE	2401.85	2249.31	2015.17	1239.62
Temps	8min24s	17min13s	46min54s	18min43s
Classement	4	2	3	1

TABLE 9 – Comparaison des résultats

Ainsi chaque étape de notre travail a permis d'améliorer les performances de prédictions. Le modèles SARIMAXtransformée 13 est le plus intéressant car plus précis que les autres tout en étant relativement rapide.

Dans le projet précédent, Sylvain Marchienne a réalisé lui aussi au préalable quelques recherches sur la prédiction des ventes. Il a par exemple utilisé le framework Prophet développé par Facebook. Prophet utilise une approche Bayésienne au problème de prédiction des séries temporelles. L'avantage principal de Prophet est sa facilité d'utilisation directement sur les données. En effet, celui-ci détermine lui-même automatiquement la tendance et la saisonnalité cachées dans la série. Cette simplicité et automatisation est à la fois son plus gros avantage mais aussi son plus grand défaut. En effet, si les prédictions ne sont pas à la hauteur, il devient très difficile d'en connaître la cause et encore plus compliqué de résoudre le problème. Les modèles ARIMAnécessitent une évaluation des paramètres compliquée afin prendre en compte la saisonnalité par exemple mais sont plus robustes.

Les recherches faites par Sylvain sur Prophet et les différents modèles ne sont malheureusement que peu exploitable. En effet ceux ci ont été conçus sans aucune pensée pour la production. Ce sont des modèles à la semaine qui ne peuvent pas être comparé autrement que visuellement à nos modèles. De plus les tests sont réalisés sur des catégories de produits et non un produit en particulier. On s'attend, lorsque l'on réalise un modèle par semaines et par catégories de produits à visualiser moins d'erreurs. Ce n'est pas le cas et nos derniers modèles se comportent plutôt bien comparé aux modèles entraînés aux semestres précédents.

4 Interface Utilisateur

4.1 Une approche centrée utilisateur

Depuis le début du projet Datapic, l'approche a été technique et statistique avant d'être métier. En plus d'améliorer la précision du prédicteur, l'un des principaux objectifs de cette PR était de mieux intégrer les besoins du Pic'Asso, et de **passer d'un proof of concept technique à un produit** utilisable par les équipes.

"Il faut développer l'interface et les prédictions pour qu'elles aient plus de sens pour les responsables"

Le but était de développer une couche applicative, qui puisse rendre les résultats du modèle statistique accessibles et facilement interprétables par les utilisateurs. Respecter une conception centrée sur l'utilisateur était une priorité dans la construction de cette partie, plus importante que ses performances : il faut que l'utilisateur puisse obtenir rapidement les informations dont il a besoin, et qu'il soit satisfait par les fonctionnalités proposée.

La première étape était donc de rassembler des idées de fonctionnalités exploitant les données du modèle statistique. Le besoin qui est apparu en premier pour les équipes du Pic'Asso se rapproche de celui d'un ERP (Entreprise Resource Planning) : gestion du stock, assistance à la création des commandes, suivi de l'inventaire... Les fonctionnalités sont détaillées dans la partie 4.4.

4.2 L'interface existante

Le travail réalisé précédemment sur le projet Datapic a abouti au développement d'une application web, qui permet d'afficher les prédictions sur un intervalle de temps donné. Bien qu'efficace pour afficher les résultats du modèle statistique qui avait été développé, l'application n'était pas utilisable en soi par les équipes du Pic'Asso car elle donnait des prédictions sur les catégories et non sur les articles.

4.3 Les choix techniques réalisés

La priorité a été donnée à la simplicité dans la stack technique. Dans la continuité de l'interface précédemment développée, le choix du langage Python était évident pour garder la cohérence avec le reste du projet, et faciliter l'intégration des différents éléments entre eux. Contrairement à l'interface existante développée avec Python et Flask, nous avons choisi d'utiliser le framework Django pour sa simplicité et rapidité de développement :

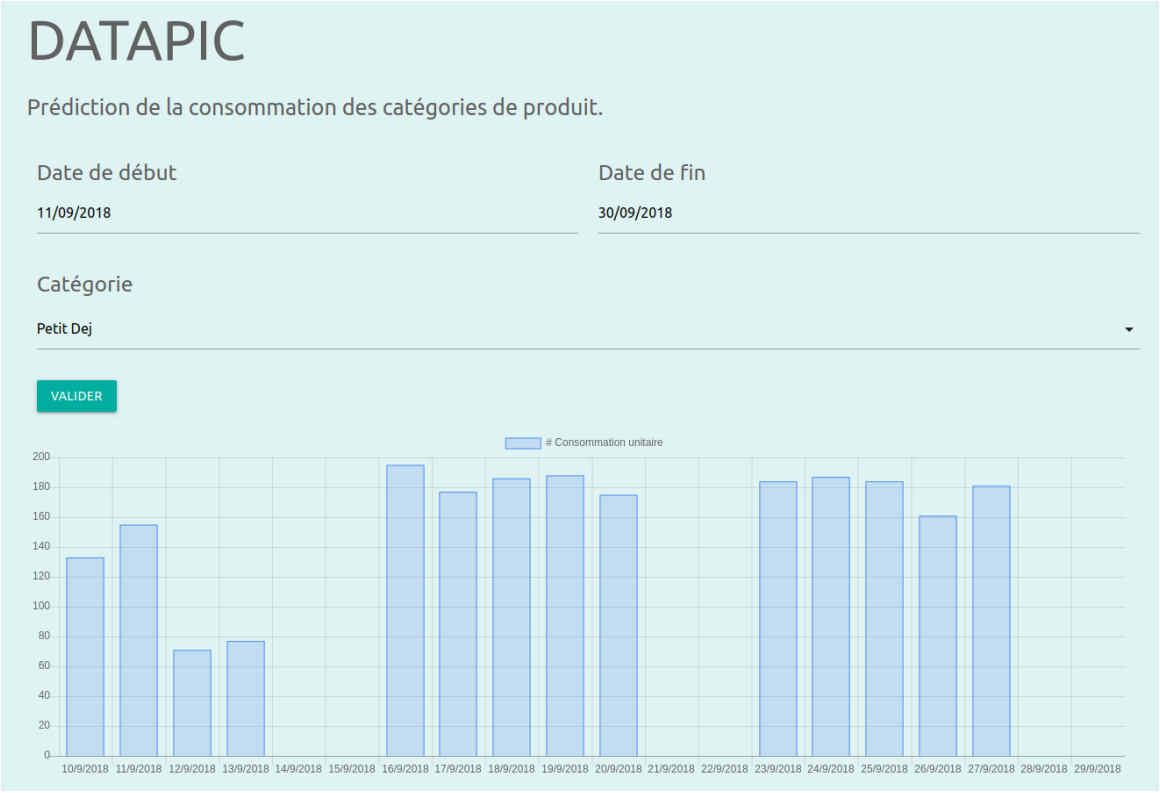

FIGURE 14 – L'application web développée auparavant

— Il permet une grande évolutivité afin d'ajouter plus tard des fonctionnalités au projet, en altérant un minimum le code existant notamment grâce à l'architecture Modèle-Template-Vue (implémentation de MVC). C'est un point important car l'application sera amenée à évoluer avec les retours des équipes d'approvisionnement du Pic'Asso.

— En plus d'être un standard, le framework Django est maîtrisé par plusieurs membres de l'équipe, et est cohérent avec la stack utilisée au Pic'Asso.

— De plus il permet de développer à la fois le front-end et le back-end facilement.

L'interfaçage avec la base de données Arango n'est pas disponible nativement dans le framework, mais l'intégration a été facile grâce au package `python-arango` déjà utilisé pour le modèle.

Étant donné que l'application sera utilisée par un nombre restreint d'utilisateurs, il n'y a pas de problématiques de scalabilité et de besoins élevés en performance. Avec le système de templates de django, les pages sont générées côté serveur, mais cela n'empêche pas d'avoir des temps de chargement rapides, le cache de django étant très efficace. Les seuls composants générés dynamiquement côté client sont les graphes : avec l'outil `chart.js`.

4.4 Fonctionnalités

Le sujet laisse la porte ouverte au développement de nombreuses fonctionnalités, car un modèle de prédiction efficace ouvre beaucoup de perspectives. Les premières discussions ont fait émerger un axe qui était central dans le développement de l'interface : l'assistance à la gestion des stocks. La plupart de fonctionnalités évoquées se rapprochent en fait de celles d'un ERP :

— Réalisation assistée de commandes en fonction de la demande à venir et des stocks restants
— OCR pour les factures et les tarifs avec ajout automatique aux stocks et aux descriptif produit
— Liste et Détail des Fournisseurs et Producteurs (compagnie, nom, mail, tel, produits proposés, actions en cours...)
— Calcul de pertes (e.g. portion du fût perdu)
— Aide à la décision au niveau des commandes (actions fournisseurs / promotions)
— Visualisation de l'évolution des stocks (graphiques)
— Assistance à l'inventaire manuel
— Statistiques sur les ventes en CA, quantité, sur la durée de vie d'un fût... pour améliorer la logistique et mieux évaluer le succès de certains produits
— Statistiques sur la fréquentation
— Comparaison entre différentes périodes

L'ensemble des fonctionnalités implémentées est détaillé dans la documentation de l'interface sur le GitLab du projet DataPic. Voici l'état de développement des fonctionnalités principales :

Fonctionnalité	État
Architecture django & routes	Implémenté
Affichage de la liste des produits	Implémenté
Affichage des détails d'un produit	Implémenté
Affichage des graphes des ventes	Implémenté
Calcul du stock restant d'un produit	Implémenté
Estimation de la durée de vie restante	À améliorer
Outil d'aide à l'inventaire	À implémenter
Assistance à la réalisation de commandes	À implémenter

Table 10 – État du développement des fonctionnalités principales de l'interface

5 Conclusion

Les modèles de la famille ARIMApermettent de modéliser certains phénomènes variant avec le temps comme ici des ventes, et de faire des prévisions pour les valeurs futures du phénomène. Ce sont des outils qui fonctionnent le plus souvent rapidement et avec peu d'hypothèse à vérifier. Ici malgré les particularités des données, avec une bonne prise en main ce sont des outils utiles et efficaces.

Bien sûr il a été difficile d'obtenir des résultats dès le début. Un gros travail était nécessaire pour prétraiter et analyser les données. Les modèles ARIMAet la prédiction de série temporelle en général ne sont pas facile à appréhender. N'ayant jamais eu de cours ni de précédents avec ce domaine, nous étions perdu. Désireux d'obtenir rapidement des résultats et en appliquant ces modèles comme des outils magiques qui fonctionnent du premier coup, nous avons longtemps été dans l'incapacité d'avancer ou même d'utiliser une démarche cohérente. Nous nous sommes lancés dans la recherche d'optimisation de nos modèles ARIMAavec de longues recherches de paramètres ou d'ajout de variables explicatives, sans réel succès.

Avec l'aide de notre professeur encadrant, nous nous sommes réorientés vers une réflexion sur l'organisation des données, c'est à dire sur comment palier aux particularité des données et traiter les semestres de façon similaire. Cette réflexion n'as pas aboutit tout de suite et nous avons dû réfléchir à différentes solutions, plus ou moins fructueux, dont celles présentées dans le rapport. Au final nous sommes assez satisfaits des résultats obtenus bien qu'ils ne soient sûrement pas optimaux et que nous n'avons travaillé qu'avec une petite quantité de produits différents.

De plus, la création d'un site permettant la visualisation des données et des différents modèles est une étape supplémentaire vers la mise en production d'un tel projet.

Malgré la difficulté du projet, notre recherche nous aura dans un premier temps permit de découvrir un domaine qui nous était inconnu. Nous avons appris à force de temps les différentes étapes de réflexions autour des séries temporelles, mais aussi les limites de ces modèles. Enfin l'état du projet DataPic a progressé. Les prédictions semblent correctes et sont facilement intégrables dans l'interface développée, ce qui nous rapproche de la mise en production d'une interface ergonomique complète pour la prédiction des ventes au Pic'Asso.

Merci de votre lecture

Table des figures

Références

[1] Arima - wikipedia. https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average. Accessed : 2019-09-27.

[2] How to create an arima model for time series forecasting in python. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/. Accessed : 2019-09-27.

[3] Introduction to forecasting with arima in r - oracle x datascience. https://www.datascience.com/blog/introduction-to-forecasting-with-arima-in-r-learn-data-science-tutorials. Accessed : 2019-09-27.

[4] Time series forecasting —arima models - toward data science. https://towardsdatascience.com/time-series-forecasting-arima-models-7f221e9eee06. Accessed : 2019-09-27.

[5] R. Shumway and D. Stoffer. *Time Series Analysis and Its Applications With R Examples*, volume 9. 01 2011.