Machine-learning pour la prédiction des prix dans le secteur du tourisme en ligne

Till Wohlfarth

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2013-ENST-0090

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Signal et Image »

présentée et soutenue publiquement par

Till WOHLFARTH

le 17 décembre 2013

Méthodes de fouilles de données pour la prédiction de l'évolution du prix d'un billet et application au conseil à l'achat en ligne

Directeur de thèse : **Stéphan CLÉMENÇON**
Co-encadrement de la thèse : **François ROUEFF**

Jury

M. Stéphan CLÉMENÇON, Professeur	Directeur de thèse
M. François ROUEFF, Professeur	Co-encadrement de la thèse
M. Xavier CASELLATO, VP of engineering, liligo.com	Examinateur
M. Fabrice ROSSI, Professeur des Universités	Rapporteur
M. Nicolas VAYATIS, Professeur des Universités	Rapporteur
M. Thierry ARTIÈRES, Professeur des Universités	Examinateur
M. Patrice BERTAIL, Professeur des Universités	Examinateur

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

T H È S E

Remerciements

Je tiens tout d'abord à remercier Xavier Casellato, qui m'a donné l'opportunité de faire cette thèse CIFRE et qui m'a encadré au sein de liligo.com. Je remercie aussi mes directeurs de thèse, M. Stéphan Clémençon et M. François Roueff, pour m'avoir fait confiance et pour m'avoir guidé pendant quatre années.

Merci également à M. Rossi et M. Vayatis, qui ont accepté d'être les rapporteurs de cette thèse et qui m'ont permis, par leur participation au jury et par leurs remarques, d'améliorer la qualité de ce mémoire.

M. Artières et M. Bertail m'ont également fait l'honneur de participer au jury de soutenance; je les en remercie profondément. Je tiens par ailleurs à remercier tout particulièrement M. Artières qui a eu une influence certaine sur mes choix académiques et professionnels en tant que professeur du Master IAD de l'UPMC.

Pour leurs encouragements et leur assistance qui m'ont permis de faire cette thèse dans de bonnes conditions, je remercie mes collègues (dans le désordre): Sylvain, Emily, Lola, Hakima, Lude-Mia, Nadia, Corinne, Imène, Dimitri, Fatima, Kim, Nicolas, Farell, Clotilde, Julie et les Xavier.

Je tiens aussi à adresser mes plus vifs remerciements à mes amis pour leur bienveillance (dans l'ordre alphabétique): Adeline, Anne-Laure, Jun-yi, Lâm, Laurent, Maxime, Rajiv, Sabine, Sylvain, Vincent et tous les autres.

Merci à ma belle famille, Dr. Margarita, Yoko, Eléonore et Constance pour tous les moments passés ensemble entre rigolades et engueulades.

Je voudrais remercier chaudement ma mère, mon père et mon frère qui m'ont encouragé à chaque étape de ma vie et qui continuent d'être à mes cotés. Merci pour votre amour.

Enfin, ces remerciements ne seraient pas complets sans mentionner celle qui depuis presque dix ans vit en couple avec un étudiant. A toi Natacha, pour ta sincérité et tes attentions, je dédie cette thèse ainsi que tout mon amour et toute mon estime.

Résumé

Ces travaux de thèse portent sur la prédiction de l'occurrence d'une baisse de prix d'un billet d'avion pour fournir un conseil à l'achat immédiat ou reporté d'un voyage sur un site web de comparaison des prix. La méthodologie proposée repose sur l'apprentissage statistique d'un modèle d'évolution du prix à partir de l'information conjointe des attributs du voyage considéré et des observations passées du prix. L'originalité principale consiste à représenter l'évolution des prix par le processus ponctuel inhomogène des sauts du vol en question. Cette représentation servira en particulier à regrouper les vols (issus de la base de données de liligo.com) en comportements similaires et à construire pour chacun des comportements identifiés un modèle commun. Nous mettons alors en œuvre une méthode d'apprentissage d'un modèle d'évolution des prix. Ce modèle permet de fournir un prédicteur de l'occurrence d'une baisse du prix sur une période future donnée et donc de prodiguer un conseil d'achat ou d'attente au client.

Les travaux de recherche présentés dans ce manuscrit s'organisent autour de trois axes. Dans un premier temps, nous introduisons une nouvelle manière de représenter les séries temporelles de prix. Cette représentation nous permettra de comparer les séries entre elles et d'appliquer des algorithmes de fouille de données bien connus. L'approche envisagée repose sur une modélisation statistique préalable de la dynamique des séries temporelles de prix, fondée sur la théorie des processus ponctuels. Nous transformons tout d'abord les séries temporelles par des processus ponctuels dans le plan temps-rendement. Puis nous modélisons ces séries de rendements par une estimation de l'intensité sous la forme de niveau de gris. C'est avec ces niveaux de gris que nous allons travailler tout au long des étapes suivantes.

Dans un deuxième temps, nous proposons la mise en oeuvre d'une segmentation des données d'apprentissage afin d'extraire des comportements types au moyen de techniques d'apprentissage non supervisé. Basés sur notre nouvelle représentation des séries temporelles, nous appliquons des algorithmes de segmentation à partir desquels nous extrayons des comportements moyens nommés centroïdes. Avec chaque centroïde, nous sommes capables de simuler des courbes de prix qui, moyennées, nous donnent une prédiction comportementale. Il s'agira alors de rapprocher les nouveaux vols du centroïde le plus vraisemblable pour construire une prédiction. C'est donc dans une troisième phase que nous appliquons des algorithmes d'apprentissage supervisé sur les attributs des vols de chaque groupe, pour une future attribution des nouveaux vols à un centroïde, basée sur leurs attributs.

Un premier chapitre décrira la topologie de nos données et expliquera la pertinence de ce module d'aide à la décision. Les trois chapitres suivants détailleront le processus de changement de représentation, la segmentation des données et la phase d'apprentissage, en décrivant pour chacune des étapes les différents paramétrages. Nous présenterons ensuite les résultats expérimentaux où nous proposons notamment une étude comparative des différentes configurations des algorithmes et des différentes approches. Enfin le dernier chapitre exposera les problématiques techniques liées à la gestion des gros volumes de données, à la parallélisation des traitements ainsi qu'à nos choix technologiques et s'intéressera aux détails de l'implémentation du service d'aide à la décision.

Table des matières

Introduction

Les travaux de thèse présentés dans ce manuscrit portent sur le développement d'un système de prédiction d'évolutions de séries temporelles finies basé sur le changement de modélisation des séries et l'apprentissage supervisé de comportements types générés par une étape de clustering.

Les compagnies aériennes au premier chef, suivies par l'ensemble des professionnels du tourisme (compagnies ferroviaires, hôteliers, etc.) ont généralisé les politiques de "yield management" afin d'optimiser le prix d'une prestation en fonction de leur niveau d'inventaire et de la date de réservation [45]. Il en résulte une opacité totale dans le processus de formation des prix, qui, pour un même billet et aux mêmes dates, peuvent varier très fortement (i) d'un fournisseur à un autre, et (ii) d'un moment à l'autre. Le consommateur est maintenu dans l'ignorance et l'incertitude, parfois encouragé à réserver longtemps à l'avance, parfois exhorté à réserver précipitamment à la dernière minute pour bénéficier d'offres présentées comme dégriffées.

Liligo.com est un moteur de recherche du voyageur capable de chercher un billet d'avion parmi plus de 250 sites d'agences de voyages et de compagnies aériennes. Afin d'aider l'utilisateur dans son acte d'achat, nous voulons pouvoir afficher pour chaque vol retourné par la recherche, une aide à la décision d'achat basée sur une estimation de la tendance dans l'évolution du prix. A chaque prix sera associé un indice d'évolution échelonné sur 5 valeurs (Forte hausse, Faible hausse, Prix stable, Faible baisse, Forte baisse), ainsi qu'un indice de confiance. Nous voulons un service flexible qui saurait répondre à plusieurs types d'interrogation comme par exemple l'évolution (hausse ou baisse du prix) à 7 jours ou l'intensité de la variation à 3 jours.

La prédiction de séries temporelles est un sujet très répandu et ses applications se retrouvent dans de nombreux domaines : dans la finance pour la prédiction de l'évolution des cours de la bourse, en météorologie pour la prédiction à court terme de la température ou dans un domaine plus lié, pour la prédiction du nombre de réservations d'un vol [31] et l'évolution de l'émission de CO_2 dans l'aviation [24]. Dans ces dernières approches, les séries sont le plus souvent infinies et la prédiction repose sur l'analyse statistique des précédentes évolutions.

Notre approche

Nos séries temporelles représentent l'évolution de prix d'un produit dit périssable. En conséquence, elles possèdent toutes une date d'expiration au-delà de laquelle elles ne sont plus disponibles. Notre base de données est donc constituée d'un ensemble de séries finies d'où nous

tentons d'extraire des comportements moyens. Nous attribuerons par la suite aux nouveaux vols leur comportement le plus probable pour en extraire une prédiction. Nous procédons donc tout d'abord à une étape d'apprentissage non supervisé des séries temporelles menant un regroupement en comportements types, puis à une étape d'apprentissage de ces comportements basée sur les vecteurs d'attributs des vols de chaque groupe.

Nous observons ici une collection de séries temporelles de même longueur, régulièrement échantillonnées, sur un panel de routes[1] représentatif et caractérisées par un ensemble d'attributs statiques (route, horaires, compagnie aérienne, etc.) et contextuels dépendant du point de la courbe (prix à l'instant t, évolutions passées,etc.). Nous regroupons ces séries selon leur trajectoire pour former des groupes de comportements similaires. A chacun de ces groupes sont associés un identifiant (appelé étiquette), un comportement type et une topologie d'attributs correspondant à la population du groupe. L'objectif est alors de prédire pour chaque résultat d'une recherche, l'étiquette la plus probable grâce aux attributs du vol. Nous pouvons alors extraire du comportement type associé à l'étiquette, les informations nécessaires pour prédire l'évolution des prix du vol.

Dans le cadre de l'application décrite, les méthodes de prédiction existantes reposent essentiellement sur des approches de type "data-mining", n'exploitant pas la temporalité des séries de prix et n'utilisant qu'un résumé simpliste des données (moyennes temporelles brutes). Par contraste, nous proposons dans ce projet de mettre en œuvre des méthodes de type machine-learning en tenant compte de la nature et de la complexité des données d'entrée dans le contexte présent. Les méthodes classiques d'apprentissage pour la prédiction (régression/classification) ont généralement été élaborées dans des contextes où les données d'entrée sont indépendantes et identiquement distribuées. Or, elles prennent ici la forme d'un historique de série temporelle et présentent donc une structure de dépendance essentielle (forte auto-corrélation). L'approche envisagée repose sur une modélisation statistique préalable de la dynamique des séries temporelles de prix, fondée sur la théorie des processus ponctuels. Les prix observés évoluant « par sauts », la notion de processus ponctuel, largement utilisée pour la modélisation des « files d'attente » en recherche opérationnelle est adaptée à la représentation des données d'entrée.

L'article [18] pose la question "Est-il possible de développer des techniques de fouille de données capable de prédire l'évolution du prix des billets d'avion" et y répond positivement. Les auteurs ont développé un algorithme nommé HAMLET combinant plusieurs algorithmes de fouille de données (technique appelée "stacking"). Leur base d'apprentissage est constituée de vols échantillonnés toutes les 3 heures pendant 21 jours. Ils veulent prédire l'évolution du point suivant ($t + 3h$) et pour cela ils décident dans un premier temps d'utiliser un simple algorithme à base de règles (RIPPER [12]) : chaque point possède 5 attributs qui sont le numéro du vol, le nombre d'heures avant le départ, le prix courant, la compagnie aérienne et la route. Dans la base d'apprentissage ils attribuent à chaque point une classe "*buy*" ou "*wait*" en fonction de l'évolution du prix suivant. Si le prix augmente la classe sera "*buy*" et si le prix est stable ou baisse, la classe sera "*wait*" . L'algorithme va donc créer un certain nombre de règles de classification,

[1] Couple Aéroport de départ - Aéroport d'arrivée

facilement interprétables, basées sur les attributs précédemment décrits.

Dans un second temps, ils utiliseront un algorithme d'apprentissage par renforcement classique (Q-learning [30]) en modifiant la règle de récompense pour qu'elle prenne en compte les pertes liées à un vol plein et à une mauvaise prédiction à la baisse.

Un troisième algorithme basé sur la moyenne glissante de la série temporelle intervient ensuite. Utilisant l'information des 7 jours précédents et la formule suivante :

$$\frac{\sum_{i=1}^{k} \alpha(i) p_{t-k+i}}{\sum_{i=1}^{k} \alpha(i)}$$

où $\alpha(i)$ est un fonction croissante de i, cette méthode donne une prédiction du prix à l'étape p_{t+1}. La règle de prédiction est ensuite la même que précédemment : si $p_t >= p_{t+1}$ alors prédire d'attendre, sinon prédire d'acheter.

Les prédictions binaires ("*buy*" ou "*wait*") de ces 3 algorithmes sont enfin ajoutées à l'algorithme RIPPER comme attributs supplémentaires pour créer le prédicteur final nommé HAMLET.

Malgré des résultats prometteurs, nous pensons que l'augmentation du nombre de routes et de vols posera rapidement un problème de temps de calcul et de stockage (disque et mémoire). De plus, leur approche est peu flexible car dépendante de la nature de la prédiction à effectuer. Si la prédiction voulue passe soudainement de $t + 3h$ à $t + 8h$ ou comme dans notre cas à $t + 7j$, tout le processus d'apprentissage doit être effectué à nouveau et les mêmes performances ne sont pas garanties.

Nous proposons un apprentissage supervisé des comportements des séries temporelles de prix, basé sur une nouvelle représentation de données (Figure 1). Celle-ci s'effectue en deux étapes : la première consiste à transformer les séries de prix (1(a)) en séries de sauts relatifs (1(b)) permettant de s'affranchir des ordres de prix des vols. La seconde étape va regrouper les sauts en fenêtre de temps et d'intensité pour former un niveau de gris représentant une empreinte du comportement de la série (1(c)). Un juste milieu est à trouver dans la taille des fenêtres pour représenter au mieux l'évolution des prix. C'est donc sur ces nouvelles représentations que nous comptons appliquer nos algorithmes afin d'extraire des comportements types.

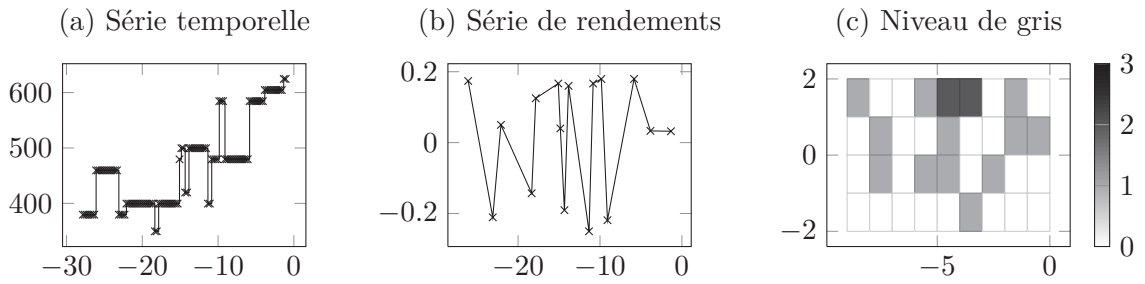


FIGURE 1 – Vol Paris-Marrakech du 31/12/2010 au 03/01/2011 proposé par *Transavia* (Low Cost d'*Air France*)

Plusieurs méthodes ont été proposées dans la littérature pour dégager des comportements d'un ensemble de séries temporelles. Dans l'article de Liao [33] plusieurs méthodes sont décrites, chacune classée dans une des trois premières approches décrites dans la Figure 2. L'approche (a) consiste à ne pas modifier la nature des séries mais à utiliser des mesures de distance appropriées. Une contrainte majeure est la nécessité d'avoir des séries normalisées et également échantillonnées. Ensuite sont listées les méthodes basées sur l'extraction d'attributs liés à la série (b) : de la plus simple extraction de points importants ([20]), à la transformation spectrale ([44]) ou par les ondelettes d'Haar ([50]). Ces méthodes ne correspondent pas au phénomène d'apparition de sauts et à la structure de nos séries temporelles constantes par morceaux. Cependant chaque case du niveau de gris peut être assimilé à un attribut du vol que l'on donne à entrée de l'algorithme des K-Means. La distance entre les vols sera alors la somme des différences case à case et le centroïde de chaque groupe sera la moyenne case à case de tous les vols du groupe. La dernière approche (c) considère que chaque série temporelle est générée par un modèle ou un mélange de distributions de probabilités sous-jacentes. Dans [52], Xiong et Yeung supposent que leurs séries temporelles ARIMA sont générées par k différents modèles ARMA. Ils utilisent alors un algorithme Espérance-Maximisation (EM) pour apprendre les paramètres et les coefficients de ces modèles maximisant la log-vraisemblance. Nous avons suivi la même démarche en appliquant le principe de l'EM à l'estimation des paramètres d'un mélange de densités par classification automatique. Nous supposons alors que les niveaux de gris sont des réalisations d'un processus ponctuel de Poisson d'intensité donnée par les centroïdes représentant les comportements moyens des groupes.

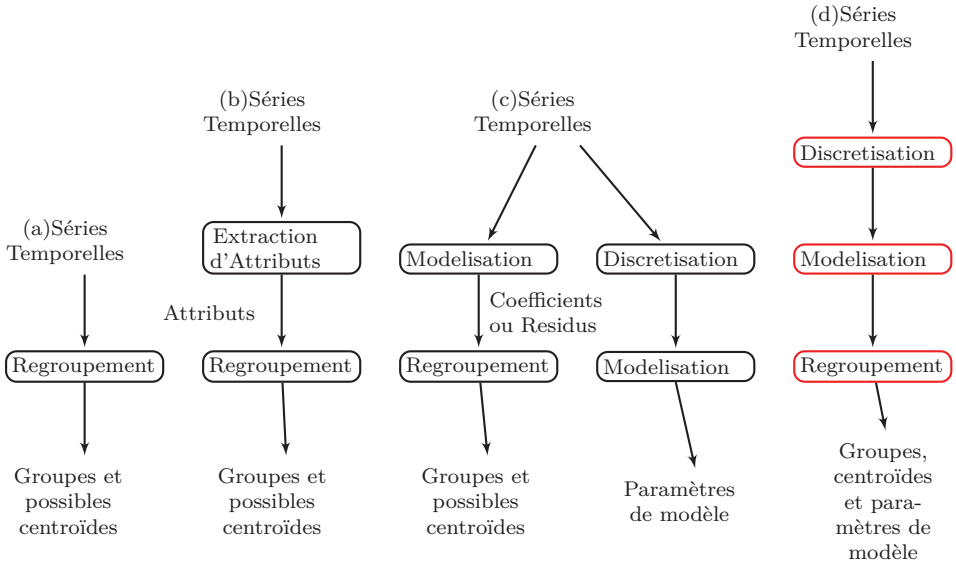


FIGURE 2 – Trois approches de regroupement de séries temporelles proposées dans [33] : (a) raw data based, (b) feature based, (c) model based ainsi que (d) notre approche

Notre nouvelle représentation permet d'utiliser les deux branches des méthodes basées sur la modélisation des séries temporelles (Figure 2(c)) mais notre démarche est plus précisément un mélange des deux (Figure 2(d)). Une première approche consiste à appliquer l'algorithme de K-Means aux niveaux de gris en mesurant la distance entre les séries par différence euclidienne : l'algorithme partitionne l'espace des cases des niveaux de gris pour créer des groupes de comportement similaires en minimisant la distance au centroïde (comportement moyen du groupe). L'autre méthode se base sur les mélanges de modèles maximisant la vraisemblance des vols au centroïde de leur groupe par un algorithme Espérance-Maximisation. Dans les deux cas, nous attribuons à chaque vol $i \in 1, \ldots, n$ de la base d'apprentissage un identifiant de groupe (étiquette) E_i et le centroïde associé.

Nous construisons alors un classifieur qui prend en entrée un trajet i et son vecteur d'attributs $V_i(1), \ldots, V_i(p)$, ainsi que les premiers prix observés pour lui assigner un identifiant de groupe $E_i \in \{1, \ldots, C\}$. Nous utiliserons plusieurs algorithmes d'apprentissage supervisé, à savoir les arbres de classification (CART [5] et C4.5 [41]), Adaboost [19] et les forêt d'arbres décisionnels (Random Forest [9]). Nous améliorerons par ailleurs l'algorithme EM pour qu'il effectue l'étape de segmentation conjointement à l'étape de classification.

Les arbres de décision nous permettent d'observer les règles de classification créées pour en donner directement une interprétation et repérer les attributs importants. De la même manière, les forêts aléatoires classent chaque attribut par ordre d'importance dans la classification en observant leur influence dans la multitude d'arbres créés. Il est alors possible de sélectionner les meilleurs attributs pour construire un prédicteur plus léger et plus performant.

Posons maintenant par convention la date de départ à l'origine, $T_0^i = 0$. Nous définissons la variable $\varphi_t^{(i)} \in \{0, 1\}$ correspondant respectivement aux conseils "achat" et "attendre" à l'instant

t. Ce conseil peut prendre autant de forme qu'il y a de manière d'interpréter un centroïde : il est donc possible de fournir une prédiction binaire, mais aussi en 5 intensités de variation (forte hausse, faible baisse, stable...), pour une prédiction à n jours. Une fois la classe $E_i = j$ obtenue en appliquant le classifieur aux attributs du trajet i considéré, le modèle de la classe j est utilisé pour calculer la probabilité $\mathbb{P}(\varphi_t^{(i)} = 1)$ grâce à la moyenne observée des $_t$ sur un ensemble de simulations issues du centroïde I_j. Cette valeur est alors utilisée pour proposer un conseil d'achat ou d'attente en fonction d'un niveau de confiance fixé à l'avance.

Une dernière approche consiste à apprendre directement $\varphi_t^{(i)}$ durant l'étape d'apprentissage. Il faudra alors créer autant de modèle que de jour avant la date de départ. Un inconvénient majeur de cette approche est qu'il faudra recréer les modèles à chaque changement de définition de φ_t.

L'introduction d'une nouvelle représentation de séries temporelles améliore l'étape d'agrégation et permet l'élaboration d'un modèle de probabilité. Nous avons ainsi réussi (i) à modéliser des comportements généraux nous permettant par la suite de prédire les évolutions possibles et (ii) à prédire de manière directe l'évolution du prix d'un trajet. La mise en place de ce service sur le site liligo.com nécessitera le contrôle constant des prédictions et la mise à jour régulière des modèles. Nous résumons l'ensemble des étapes de notre générateur de modèle sur la Figure 3.

Plan du manuscrit

Les travaux présentés dans ce manuscrit ont été réalisés dans le cadre d'une thèse CIFRE co-dirigée par le laboratoire TSI de Télécom ParisTech et la département Recherche et Développement de l'entreprise Findworks Technologies pour le moteur de recherche liligo.com. Ces recherches ont abouti à la création d'un prototype de service de prédiction en ligne de l'évolution des prix des billets d'avion.

Le Chapitre 1 décrit et analyse les données que de notre base afin d'extraire des informations sur les comportements des utilisateurs et des changements de prix. Nous commençons par décrire la construction de notre base de données et le choix de notre structure. Nous analysons ensuite les recherches utilisateurs et les différents types d'achat afin d'orienter nos choix de paramètres de modélisation. Nous étudions par la même occasion le comportement des courbes de prix en visualisant les différents types de yield management mais aussi en comparant des mêmes billets proposés par des sites différents. Il sera enfin question de la pertinence de notre projet dans l'aide à la décision d'achat de l'utilisateur.

Le Chapitre 2 détaille notre nouvelle représentation des données obtenue par la transformation des séries temporelles de prix par des processus ponctuels dans le plan temps-rendement puis en estimation d'intensité. L'idée est de s'abstraire des écarts d'ordre de prix entre les vols long-courriers et les vols intérieurs par exemple, mais qui sont susceptibles de suivre les mêmes types de variations à des niveaux de prix distincts. Les séries temporelles sont dans une premier temps transformées en séries de rendements où seuls les changements de prix sont représentés en pourcentage de variation. Nous modélisons ensuite cette suite de points par un processus

ponctuel marqué inhomogène dont l'intensité est estimable sous la forme d'une image pixélisée.

Le Chapitre 3 sera consacré à la segmentation des données d'apprentissage. Cette segmentation permet d'extraire un ensemble fini de comportements types de notre base de données de vols. Nous utilisons dans un premier temps l'algorithme des K-Means avec comme mesure de distance la différence case par case des niveaux de gris. Nous tentons ensuite de stabiliser les résultats des K-Means en appliquant le principe de bagging, qui consiste à multiplier l'étape de segmentation sur des sous-ensembles aléatoires de la base d'apprentissage. Cette méthode, appelée Bagged K-Means, permet d'améliorer la convergence vers l'optimal global et d'atténuer l'influence de l'initialisation. Nous appliquons enfin la procédure de mélange de modèle via l'algorithme Espérance-Maximisation où le critère d'optimisation ne sera plus la distance au centroïde mais la vraisemblance global du modèle.

Dans le Chapitre 4, l'ensemble des étapes amenant à la prédiction finale est exposé. La première étape consiste en l'apprentissage supervisé de l'identifiant du groupe E_i, d'après la topologie des attributs des vols de chaque groupe. Nous utilisons différents algorithmes tels que les arbres de classification, adaboost ou l'algorithme des forêts d'arbres décisionnels. Cet apprentissage permet ensuite de classer un nouveau vol dans son groupe le plus vraisemblable, auquel est associé un comportement moyen avec lequel nous simulons un certain nombre de courbes. En moyennant le comportement de ces simulations, nous obtenons une prédiction de l'évolution future des vols appartenant au groupe correspondant. L'attribution d'un vol à un groupe est ensuite améliorée par l'ajout de l'information des premières évolutions de prix.

Nous comparons par ailleurs ces méthodes à une prédiction dite "directe" où l'étiquette apprise n'est plus E_i mais la prédiction à un instant t, c'est-à-dire $\varphi_t^{(i)}$. Le nombre de modèles à calculer est alors équivalent au nombre de jours avant la date de départ.

Enfin nous abordons dans les deux derniers chapitres les résultats expérimentaux et l'implémentation du service. Pour évaluer nos résultats, différentes métriques sont utilisées telles que le pourcentage de bonnes prédictions, le pourcentage moyen de gain et de perte par billet d'avion ou encore la courbe ROC. Il est alors important de savoir quel critère nous souhaitons optimiser entre le pourcentage de bonnes prédictions et le gain moyen économisé par billet. En effet, la prédiction à la baisse faisant économiser moins d'argent, il peut être préférable du point de vue de l'utilisateur de choisir une approche conservatrice qui prendra moins de risque sur la prédiction à la baisse.

Le service final proposé à l'utilisateur doit répondre à 3 critères : la clarté des informations présentées, la rapidité de la réponse et sa qualité. Nous axons notre modélisation sur l'indépendance des composants applicatifs afin de rendre chaque sous-ensemble évolutif sans modifier l'ensemble de l'architecture. Nous décrirons cette architecture et l'enchainement des mécanismes enclenchés lors d'une demande de prédiction par des diagrammes de séquence et de cas d'utilisation.

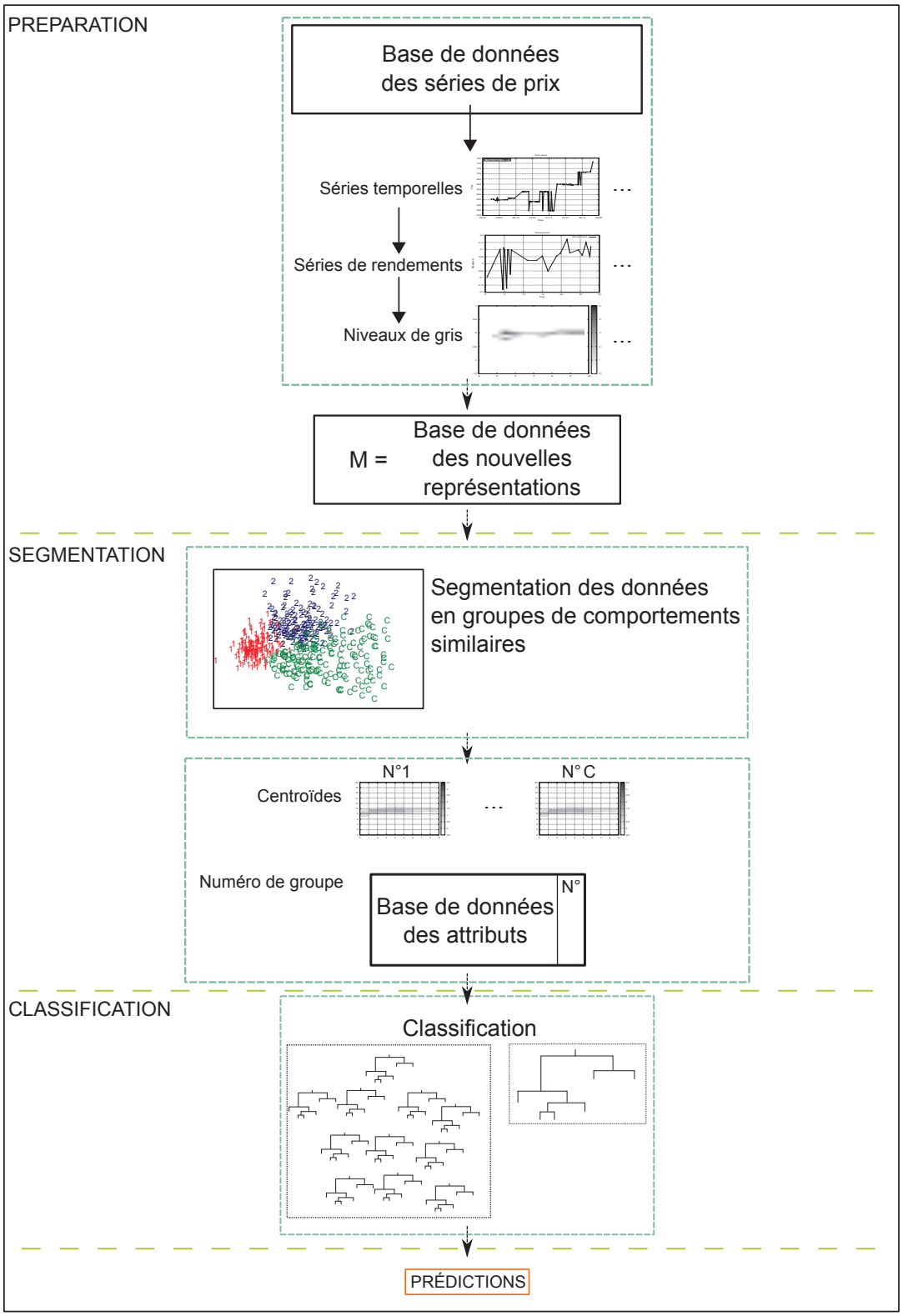


FIGURE 3 – Workflow - Vue Simple

Chapitre 1

Analyse exploratoire

Contents

Introduction

L'objectif de ce chapitre est de présenter la structure de nos données, leur origine et en extraire les statistiques nécessaires à nos futurs choix de paramètres. La bonne connaissance de la nature et du comportement de nos données est essentielle pour la construction de notre infrastructure et pour la validation de notre approche. Elle passe par l'étude des techniques d'optimisation de prix appliquées par les différents sites marchands et par une analyse des effets de ces optimisations sur les courbes de prix.

Les changements de prix dans le domaine aérien suivent des règles régies par des algorithmes de *yield management* ou *revenue management* décrits dans de nombreux ouvrages [2][15] ou [46], laissant entrevoir que certaines de ces règles, communes à tous les marchands, peuvent être apprises pour prévoir leurs évolutions. Le Yield Management est une discipline économique adaptée à des secteurs où la tarification par segments de marché est pratiquée et combinée à une analyse statistique poussée. Cette pratique a pour objectif d'augmenter le revenu de la compagnie par siège disponible. Les paramètres déterminants dans l'optimisation des prix sont donc le taux de remplissage de l'avion et l'évolution de la demande mais d'autres variables peuvent introduire des subtilités dans la maximisation des revenus [51]. Ces informations n'étant pas publiques, nous pouvons uniquement percevoir ces variables cachées par le biais de l'évolution des séries temporelles et de la répartition par destination du trafic de liligo.com.

Nous rappelons que liligo.com est un moteur de recherche de voyages permettant aux utilisateurs, de comparer plus de 250 sites d'agences de voyages et compagnies aériennes. A chaque recherche utilisateur, toutes les informations de la page de résultats sont conservées en base de données représentant une source volumineuse d'informations à traiter. Il est donc nécessaire de faire des choix quant aux vols que nous souhaitons utiliser et quant à l'architecture de notre base de données. Notre base d'apprentissage devra représenter la majorité des comportements existants tout en conservant une taille raisonnable. L'architecture doit permettre de reconstruire les séries temporelles de prix, de comparer les mêmes vols proposés par des sites différents et d'accéder rapidement aux caractéristiques des vols.

Nous introduisons la notion de vol unique, qui décrit un trajet défini par des dates de départ et de retour, des aéroports de départ et d'arrivée ainsi que les codes des vols correspondants (AF5653 par ex.). Ces vols sont vendus par la ou les compagnie(s) qui les affrète(nt) (elles sont plusieurs en cas de vols à escales ou de partage de code) mais aussi la plupart du temps par des agences de voyages. Chaque vol unique possède donc une série temporelle de prix par site marchand. Il est de fait intéressant de constater que les évolutions de prix d'un même vol unique peuvent être similaires ou alors complètement différentes selon le site qui les vend.

La première partie de ce chapitre est consacrée à la description du phénomène à l'origine de ce projet : le yield management. Après en avoir expliqué les tenants et les aboutissants, nous détaillons une des méthodes utilisées par les compagnies aériennes. Dans un second temps, nous décrivons l'origine de nos données (recherches utilisateur, alertes mail) et l'architecture de données qui nous a semblé la plus proche de la réalité du transport aérien. Nous avons notamment pris soin de conserver le lien entre un billet vendu directement par la compagnie aérienne et indirectement par le biais d'une agence de voyages. Nous illustrons ce phénomène

par des exemples, et nous détaillons les différentes approches d'optimisation des prix pratiquées par chacune des parties.

Puis nous expliquons le choix de nos paramètres de construction de la base d'apprentissage par l'analyse statistique du comportement des utilisateurs dans l'achat de leurs billets d'avion. Nous y discutons de la sélection des routes étudiées, de la durée de séjour et de la longueur des séries temporelles ainsi que de leur quantité et de leur qualité.

Enfin nous expliquons les enjeux du service et la pertinence d'un conseil à l'achat du point de vue de l'utilisateur. Si le lieu commun est que les prix ne font qu'augmenter et qu'il faut acheter le plus tôt possible pour avoir le meilleur prix, nous démontrons qu'avec notre service il est possible d'économiser de l'argent quelque soit le jour avant la date de départ. Nous rappelons que notre service ne consiste pas à indiquer le meilleur moment pour acheter son billet mais à fournir à l'utilisateur une indication sur la future évolution de son billet à un instant t.

1.1 Notations

n	:	Nombre de séries temporelles dans la base de données.
i	:	Numéro du vol de la base d'apprentissage $i \in 1, ..., n$
V_i	:	Vecteur d'attributs du vol i.
$p_i(t)$	:	Courbe de prix du vol i.

1.2 Yield Management

Au sein d'une même cabine, la compagnie divise son avion en classes de réservation, ou classes tarifaires, ou encore classes de yield. C'est un découpage purement informatique, invisible pour le passager, et sans conséquence sur le positionnement des voyageurs à l'avant ou l'arrière de l'appareil.

Il ne faut pas confondre ce découpage avec le découpage en classes de transport, que sont la première classe, la classe affaires et la classe économique. Les classes de réservation sont des sous-divisions de l'avion au sein même de ces cabines. Toutes ces classes sont emboîtées à la manière de poupées russes, de la classe la plus basse à la classe la plus haute. Chaque vol est décomposé en 10 à 20 classes de réservations. Elles sont désignées par des lettres de l'alphabet. En général, la première classe de transport contient les classes tarifaires P et F, la classe affaires contient les classes tarifaires J et C, et la classe économique contient le plus de classes tarifaires, dont la Y. L'IATA [1] recommande une certaine codification, mais chaque compagnie a ses propres habitudes.

Ces classes sont emboîtées, au sens où une classe inférieure ne peut pas empiéter sur une classe supérieure, alors qu'une classe tarifaire supérieure peut préempter des sièges prévus pour

[1] Association internationale du transport aérien

une classe inférieure. Les compagnies low-cost[2] appliquent pour la plupart les mêmes principes, mais d'une manière fortement simplifiée. Ainsi, le prix de leurs billets ne varie généralement que suivant deux facteurs : l'achat à l'avance, et l'état de remplissage de l'avion. À un instant donné, il n'existe qu'un seul prix pour le billet d'avion, valable pour tout le monde. Ce système est bien adapté à la clientèle plus homogène (essentiellement loisir) de ces compagnies, et présente également l'avantage d'être bien compris par les passagers, car il se résume par la formule simple "plus on achète tôt, moins c'est cher". Dans la réalité, ce principe est infirmé quotidiennement par les algorithmes de revenue management qui doivent baisser les prix dans diverses situations : annulation de billet, augmentation de la taille de l'avion, retour de places allouées aux agences de voyages, etc. Nous allons d'ailleurs montrer dans la section "Pertinence"(1.5) de ce chapitre que nombre de lieux communs ne sont pas toujours vérifiés.

Les algorithmes utilisés ont pour but essentiel de déterminer quelles classes de réservation seront ouvertes sur un vol, avec quel quota de sièges affecté à chacune. Il s'agit d'un contrôle de l'offre par ajustement des capacités disponibles. Par exemple, il faudra ouvrir beaucoup de sièges dans les basses classes de réservation et n'en garder que peu pour les passagers à haute contribution sur un vol en heure creuse, qui sinon ne sera pas rempli, alors que sur un vol en heures pleines il s'agira de procéder à l'inverse pour obtenir le revenu maximal.

Exemple d'algorithme : Bid-Price Une des méthodes utilisées dans le domaine aérien pour maximiser les revenus est l'optimisation d'un vecteur représentant l'évolution du prix selon le remplissage de l'avion. Ces vecteurs nommés "bid-price vectors" sont des indications de modification de prix par cabine envoyées aux GDS[3] afin qu'ils ajustent les prix annoncés au fur et à mesure du remplissage. Chaque cabine est divisée en classes associées à un tarif. Toutes les classes ayant un tarif inférieur au bid-price seront alors fermées à la vente.

La création de ce vecteur se fait en plusieurs étapes et nécessite un certain nombre de paramètres d'entrée :

1. Les évolutions passées des demandes par cabine
2. Le type d'appareil permettant de connaître la capacité par cabine (première classe, business, économique)
3. Les divisions passées des cabines en classes : nombre de classes, capacité et tarifs de chaque classe
4. L'historique des présences des passagers (Entre 15 et 20% d'absence en moyenne)
5. L'historique des sièges alloués aux agences de voyages/brokers non vendus
6. Le seuil à partir duquel il est préférable de surclasser un passager plutôt que de lui changer son vol

[2]Une compagnie aérienne à bas prix ou compagnie aérienne low cost, est une compagnie aérienne qui s'est positionnée sur le créneau commercial du transport aérien à moindre coût (low cost) en limitant ou en supprimant les services annexes au sol et en vol.

[3]Un système informatisé de réservation, ou global distribution system (GDS) en anglais, est un système informatisé qui centralise les données concernant les vols, les horaires, les places disponibles, les tarifs et les services connexes avec des moyens permettant d'effectuer des réservations et de délivrer des billets.

Avec tous ces paramètres provenant de bases de données d'historiques, de nouvelles données sont calculées :

1. Le placement du rideau séparant la première de l'économique (pour les plus petits avions)
2. Le choix du nombre de sièges dépassant la capacité de l'appareil (surbooking)
3. La division des cabines en classes (capacité, tarif)
4. La prédiction de la demande sur l'ensemble des jours avant la date de départ

A partir de toutes ces données de sortie, il est alors possible de construire de manière certaine par des algorithmes d'optimisation, un vecteur de bid price optimal qui sera ensuite transmis au GDS pour qu'il puisse appliquer les règles de ventes de tous les vols en question (Figure 1.1).

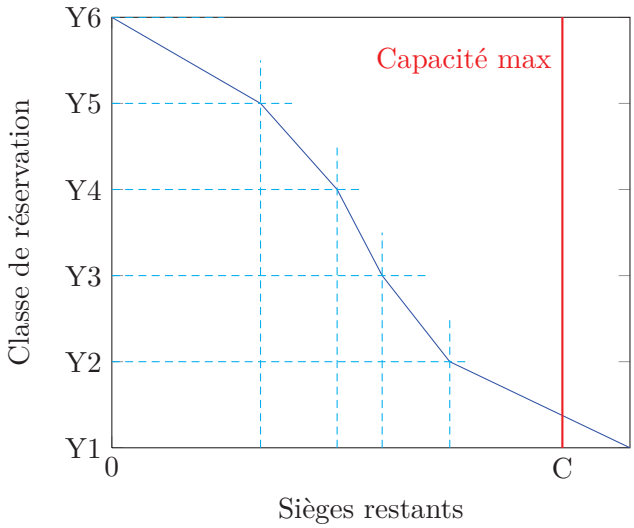


FIGURE 1.1 – Exemple d'évolution de prix en fonction du Bid Price

La modification des prix des classes peut se faire manuellement suite à la détection d'une trop grande différence entre la demande actuelle et la prévision, ou pour coller aux offres de la concurrence. Dans la majorité des cas ce processus d'optimisation est quotidien pour les vols dont la date de départ est proche et moins fréquent à mesure que l'on s'éloigne de celle-ci. La fluctuation des prix aura donc un comportement différent au fur et à mesure du temps et du remplissage. Nous allons voir dans la section suivante quelles sont les conséquences du yield management sur le comportement des séries de prix.

1.3 Structure et description de nos données

Dans cette section nous décrivons l'origine et les propriétés de nos données, et la manière dont elles ont influencé la structure de notre base de données. Cette dernière est composée de

trois tables principales dont chacune aura pour but de faciliter une des étapes de notre travail préparatoire : l'identification d'un vol unique, l'extraction de la ou des série(s) temporelle(s) $p_i(t)$ associée(s) et enfin la création des vecteurs d'attributs V_i correspondants. Ces tables respectent à la fois la notion d'unicité d'un vol et celle d'unicité du billet à vendre.

1.3.1 Description des données

Liligo.com possède une base de données d'historiques de recherche des utilisateurs grâce à laquelle nous avons construit notre base d'apprentissage et de test. Dans cette base de données, un *trajet unique* ou *vol unique*, correspondant à un couple vol aller/vol retour, est défini par 6 attributs : le couple "aéroport de départ" et "aéroport d'arrivée" que nous nommerons "route", la date de départ et la date de retour comprenant les heures et les minutes et enfin, le code transporteur du vol aller et celui du vol retour[4]. Les prix sont collectés depuis des sites marchands tels que les compagnies régulières (*Air France*, *EasyJet*, etc.) et les agences de voyages (*GoVoyages*, *VoyagesSNCF*, etc.) ces dernières pouvant vendre les mêmes vols que les compagnies régulières. Cela implique que pour un même trajet, on peut avoir autant de séries temporelles que de sites proposant le vol. Nous nommerons "*provider*" le site marchand d'où le prix est extrait et "*supplier*" la compagnie aérienne qui affrète l'avion. Si le prix est issu du site de la compagnie aérienne, alors le code *provider* sera identique au code *supplier*. Inversement, si les prix proviennent d'un site tiers tel *GoVoyages* ou *Opodo*, le code *provider* ne sera pas celui du *supplier*. L'association d'un trajet unique et d'un site marchand correspond alors à un résultat de recherche, chaque recherche renvoyant une multitude de résultats.

Les origines de nos données sont diverses : tout d'abord lorsqu'un utilisateur se rend sur le site liligo.com, il lance une recherche avec comme paramètre une ville ou un aéroport de départ et d'arrivée ainsi qu'une date de départ et de retour sans préciser les horaires. Chaque recherche utilisateur renvoie alors en moyenne 300 résultats ayant chacun des paramètres supplémentaires mais différents : le site marchand (*provider*), la compagnie aérienne (*supplier*), les horaires et le tarif. La notion de vol unique est déjà présente sur le site car les vols proposés par plusieurs sites marchands sont regroupés afin de comparer rapidement les offres similaires et d'alléger l'affichage. Cet important volume d'informations doit être correctement stocké et indexé afin de pouvoir y accéder facilement.

Par ailleurs, chaque utilisateur peut définir des alertes : l'alerte est un outil programmable qui effectue une recherche demandée, quotidiennement, à la place de l'internaute. Après avoir renseigné la destination et les dates de vol, le voyageur reçoit chaque jour par email un récapitulatif des meilleurs résultats, classés par prix. Ces alertes peuvent nous donner de longues séries de prix régulièrement échantillonnées mais ne garantissent pas de pouvoir suivre un même vol pendant toute la durée des envois de mails. En effet, seules les cinq meilleures offres sont sélectionnées réduisant les chances d'avoir pendant un mois la même offre dans le top cinq. Cette limitation ayant été mise en lumière, nous pensons faire évoluer le système des alertes pour conserver l'intégralité des résultats. Cependant les alertes peuvent aussi être stoppées à n'importe quel moment par l'utilisateur. Malgré tout, cette source de données nous donne un

[4] ex : AF350 pour un vol sans escale et AF630AF405 pour un vol avec une escale.

grand nombre de séries consistantes. Nous définissons une série consistante comme une série possédant suffisamment de points pour garantir la détection de la majorité des variations de prix.

Les données issues des recherches utilisateurs, sont quant à elles plus éparses lorsque la date de départ est éloignée, et ne nous garantissent pas d'avoir toujours une collecte quotidienne. Heureusement, nous verrons dans le chapitre suivant que la fréquence des changements de prix est en grande majorité supérieure à 24 heures autorisant un échantillonage plus faible. Ces changements de prix par à-coup créent des séries de prix constantes par morceaux, comme le montre la Figure 1.2 avec un faible nombre de plateaux. Pour pouvoir garder des ensembles de séries temporelles consistantes et uniformément échantillonnées, nous divisons notre base de données en 2 sous-ensembles :

1. Un ensemble constitué de séries de 28 jours échantillonnées toutes les 6 heures
2. Un ensemble constitué de séries de 90 jours échantillonnées quotidiennement

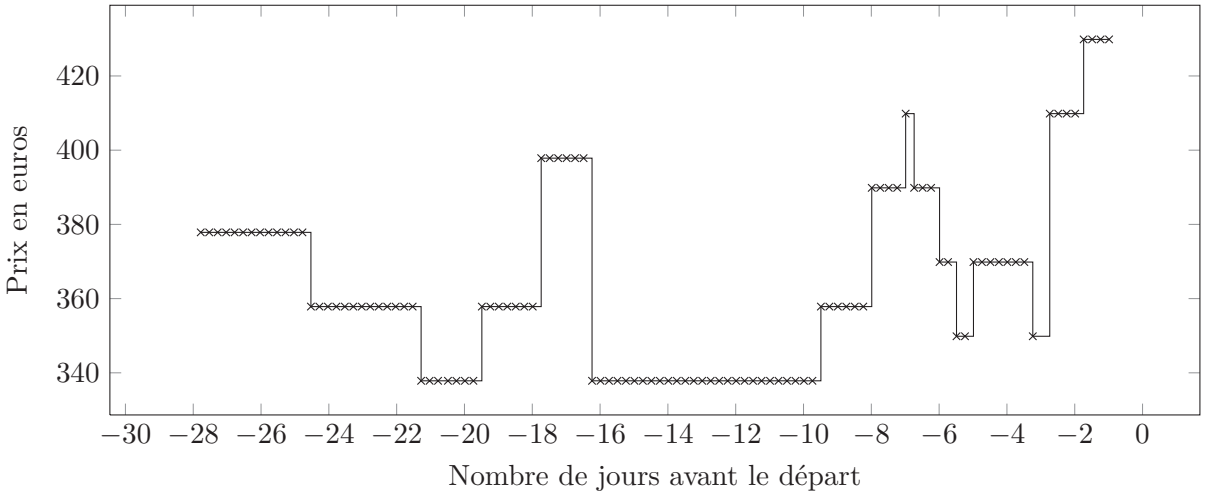


FIGURE 1.2 – Exemple de série de prix pour un Amsterdam-Barcelone du 20/11/2011 au 23/11/2011 proposé par *Austrian Airlines*. Cette série possède un point toutes les 6 heures pendant les 28 derniers jours de sa vente.

Nous allons maintenant décrire les trois grandes tables de notre base de données qui nous permettent d'extraire les séries de prix et leurs caractéristiques.

1.3.2 Structure de la base de données

Comme expliqué précédemment, nous avons choisi de diviser notre base de données en trois tables principales, nous permettant chacune d'extraire les informations nécessaires à différentes étapes de notre processus de prédiction. Une première table stockera les paramètres des trajets

uniques et leur associera un identifiant unique. La seconde collectera les prix pour chaque couple vol unique/provider. Enfin une table contiendra l'ensemble des paramètres associés à une série de prix, tels que le type de compagnie, la ville, le pays et le continent de l'aéroport de départ et d'arrivée etc.

Les vols uniques

Dans le domaine du tourisme aérien, un vol opéré par une compagnie aérienne est défini par un aéroport de départ (*departureStation*) et un aéroport d'arrivée (*arrivalStation*) (couple nommé "route"), un horaire de départ (jour, mois, année, heure, minute), et un horaire de retour, un code transporteur aller (*transportCode*) et un code transporteur retour (*transportCodeRet*). La règle de nommage des codes transporteur est opaque mais les codes correspondent souvent au couple d'aéroports et à un horaire variant selon les périodes de l'année. Par exemple le vol 'HV3014' Paris-Marrakech opéré par *Transavia* (Low Cost d'*Air France*) correspond au vol de 6h40 de mars à juin et de 7h00 de juillet à février.

Ce code est systématiquement composé d'un premier groupe de lettre correspondant au code IATA de la compagnie aérienne opérant le vol et est suivi d'un groupe de chiffre choisi par la compagnie. La Figure 1.3 nous montre 3 vols ayant les mêmes codes transporteur aller et retour à différentes dates et nous montre qu'il n'y a aucune corrélation apparente entre les variations de prix, car comme expliqué dans l'exemple de yield management, le paramètre principal de l'évolution des prix est la courbe de demande prédite, différente en fonction du jour de départ. On peut cependant noter une similarité dans l'ordre de prix des vols décollant à la même période : 17 et 21 février 2012.

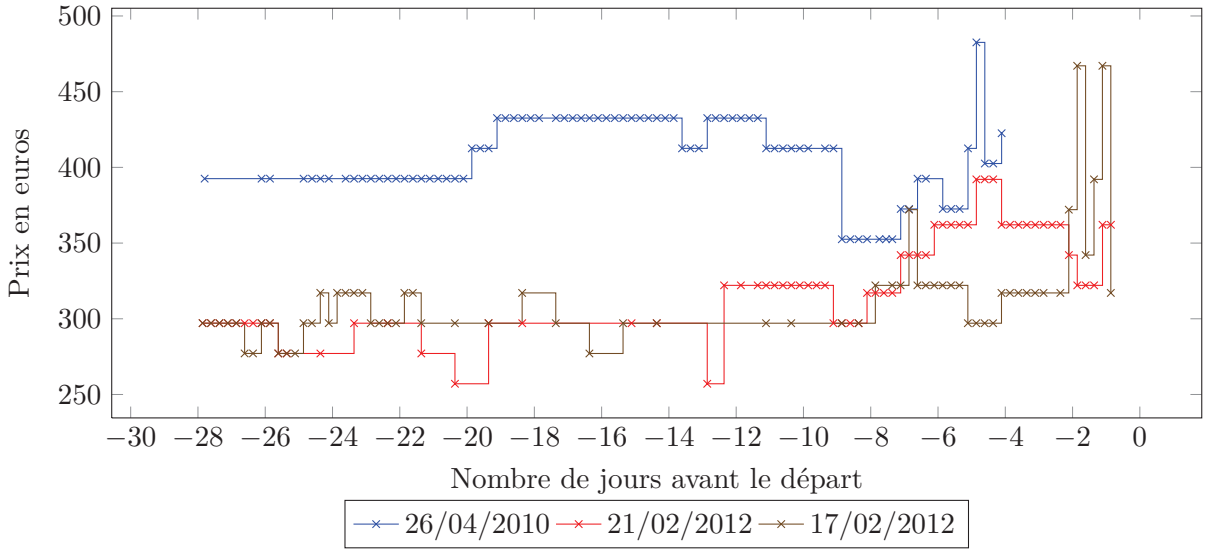


FIGURE 1.3 – Séries temporelles d'un même couple "code transporteur aller" (AT749) et "code transporteur retour" (AT640) (Paris-Marrakech sur *Royal Air Maroc* vendu par *VoyagesSNCF*) à différents moments de l'année.

En ce qui concerne les trajets à escales, nous avons décidé de concaténer les codes transporteur comme pour 'LH4271LH3472' par exemple, qui est un Paris-Budapest opéré par Lufthansa et faisant une escale à Düsseldorf. Pour des problèmes de maintenance de base de données et de volumétrie, nous ne conservons malheureusement aucune information sur les escales. Cependant, dans notre exemple, il est possible d'interroger notre base de données en cherchant le vol "LH4271" afin de retrouver l'aéroport de l'escale. Si le vol n'existe pas, des sites spécialisés peuvent nous fournir cette information, mais elle ne nous est pour l'instant pas nécessaire.

Nous définissons alors un vol unique par le n-uplet suivant : { aéroport de départ, aéroport d'arrivée, date de départ, date de retour, code transporteur aller, code transporteur retour} en excluant la notion de site marchand : ce vol unique représente le trajet indépendamment du vendeur. A chaque n-uplet nous associons un identifiant unique nommé *id_unique_flight* .

La création d'un identifiant unique indépendant du site marchand va nous permettre d'observer des phénomènes de concurrence ou de détecter les changements de prix supplémentaires appliqués par les agences de voyages.

En revanche, dans certains cas, la route et les dates sont identiques mais les codes transporteur différents : nous avons affaire à un vol cobrandé (aussi appelé “partage de code”). Il arrive que certaines compagnies s'accordent à vendre en parallèle des places pour un même vol, chacune sous sa propre marque. Dans ce cas, toutes les caractéristiques du vol sont les mêmes excepté le code du vol (AF312 et IB415 par exemple). Ces accords commerciaux permettent d'augmenter la visibilité des deux compagnies et d'assurer une meilleure rentabilité [10].

Trois types de partage de code existent :

- **En blocs :** la compagnie X achète à la compagnie Y un nombre de sièges fixé à l'avance sur un vol donné, à un prix également fixé à l'avance. Chaque compagnie vend les sièges qui lui reviennent en suivant sa propre politique tarifaire, et au stade de la réservation tout se passe comme si les deux blocs étaient en fait deux avions différents les deux partenaires ne vendent pas forcément leurs billets respectifs au même tarif, car leur clientèle n'est pas la même.

- **Free flow :** dans ce modèle, la répartition des sièges n'est pas fixée à l'avance : la compagnie qui opère l'avion gère un inventaire commun de places, réparti en classes de réservation, et dans lequel les deux compagnies vont puiser. L'autre compagnie paie un prix par billet, qui dépend de la classe de réservation. Il suppose donc une coopération plus poussée, et une certaine harmonisation des conditions tarifaires, si ce n'est des tarifs.

- **Joint venture :** les deux compagnies peuvent décider de mettre en commun toutes leurs ressources sur une ligne ou un ensemble de lignes, et de partager coûts et recettes. La coopération tarifaire est alors totale, et du point de vue tarifaire elle est équivalente à une fusion. Ce type de partage a été mis en place par *Air France* et *Alitalia* sur les lignes reliant la France et l'Italie.

Selon les cas ces vols peuvent être considérés comme des vols indépendants ayant leur propre évolution, ou comme des vols similaires. Dans les faits, nous pensons que chaque compagnie applique son propre yield management et qu'il faut séparer le comportement des deux séries

temporelles. Les séries seront susceptibles de se terminer plus rapidement car chaque compagnie possède moins de sièges et le vol sera plus rapidement complet. Dans l'exemple de la Figure 1.4, on observe deux séries de prix d'un vol à escales dont le deuxième tronçon est cobrandé. Il s'agit d'un vol Amsterdam-Barcelone passant par Londres et vendu par *LastMinute*. Le Londres-Barcelone est à la fois visible sous le code "IB7453" (vol *Iberia*) et sous "BA486" (vol *British Airways*). Quelques recherches nous indiquent que le vol est opéré par *British Airways* mais il nous est impossible de savoir qui opère le vol à partir des informations récoltées sur les sites marchands. Les politiques de prix des deux compagnies sont complètement différentes et on observe des variations beaucoup plus fréquentes de la part d'*Iberia*, alors même que les prix pour un tel trajet sont déjà très élevés. En revanche il est intéressant de remarquer que les plateaux sont quasiment les mêmes, et donc que le mécanisme de changement de prix, comme expliqué précédemment est indépendant de l'étape de définition des prix par classe de réservation.

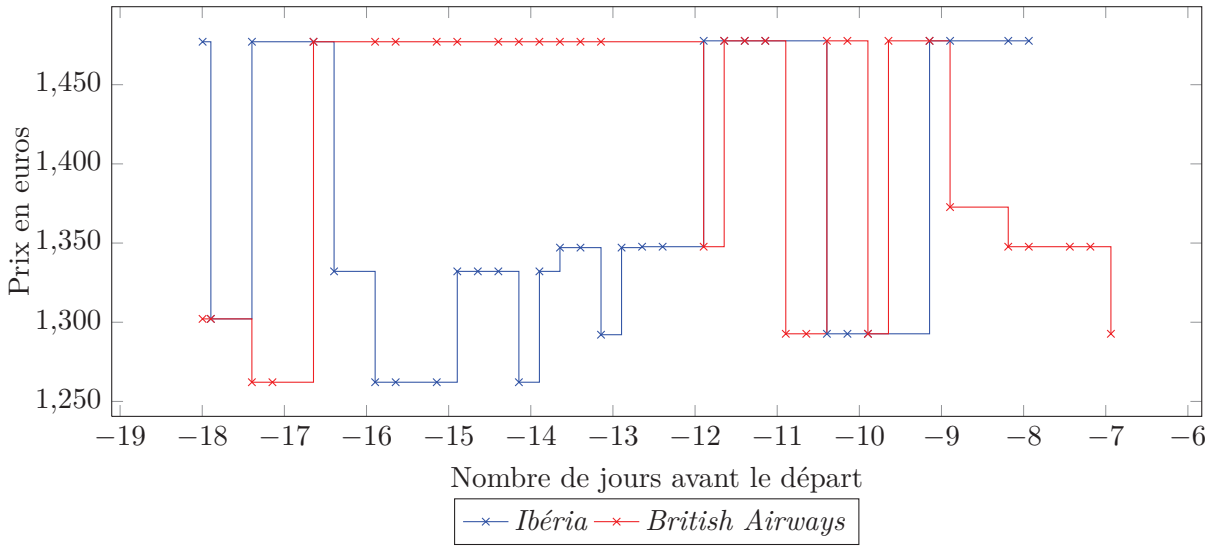


FIGURE 1.4 – Exemple de vols cobrandés proposés par *Ibéria* et *British Airways* (Amsterdam-Barcelone)

Les séries de prix

Chaque vol est ensuite vendu indépendamment, ou non, par un ou plusieurs sites marchands que nous appelons "*provider*". Le couple {*id_unique_flight* , *provider*} (vol unique, site marchand) possède lui aussi un identifiant unique nommé *id_flight* . L'ensemble des prix proposés sous l'identifiant *id_flight* constituera une série temporelle de prix uniques que nous stockons dans une table dédiée. Chaque ligne de cette table correspond à un vol unique proposé par un marchand, à un instant t avec un prix p créant ainsi la série de prix $p_i(t)$. Lorsqu'une recherche utilisateur est effectuée, nous ajoutons donc pour chaque résultat une entrée dans cette table.

La Figure 1.5 nous montre l'utilité d'avoir créé un identifiant différent pour un vol et pour les billets associés : on y observe pour un même *id_unique_flight* , les différentes séries temporelles

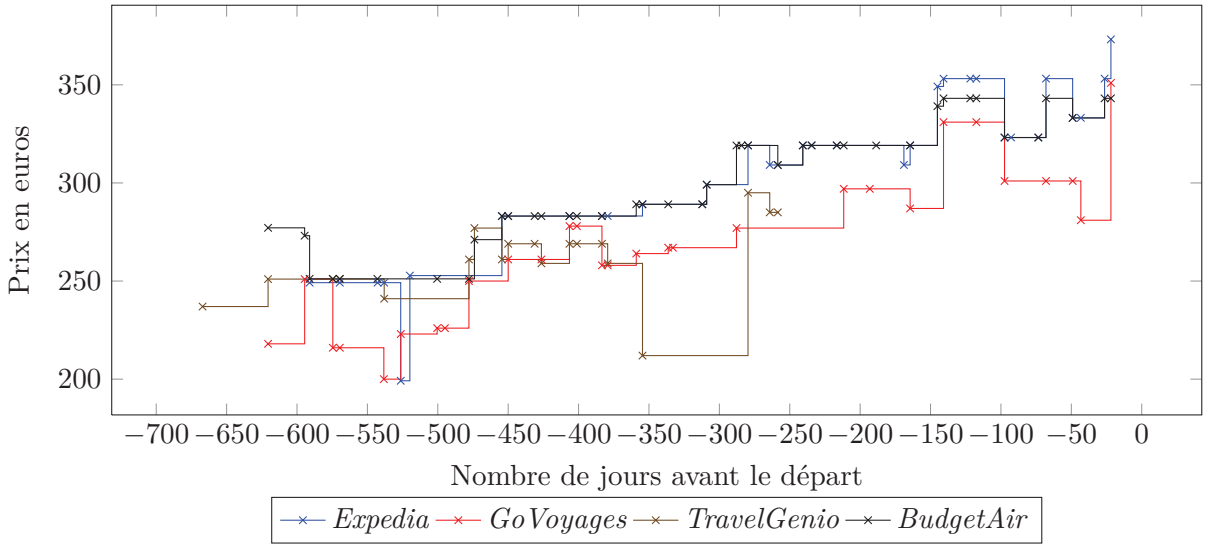


FIGURE 1.5 – Yield Management de différentes agences de voyages pour un même billet d'avion. L'*id_unique_flight* est identique mais les *id_flight* sont différents. Paris-Marrakech opéré par *Royal Air Maroc* du 26/02/2013 au 04/03/2013.

identifiées par un *id_flight* . Le même vol Paris-Marrakech opéré par *Royal Air Maroc* est vendu à la fois par *Expedia*, *GoVoyages*, *TravelGenio* et *BudgetAir*, et possède donc quatre séries bien distinctes. Pour pouvoir identifier ces séries, il suffit de rechercher tous les *id_flight* associés à l'*id_unique_flight* correspondant.

Il aussi intéressant d'observer l'évolution parallèle d'un billet vendu par la compagnie qui affrète le vol et de celui vendu par l'agence de voyage ayant des accords commerciaux avec ladite compagnie.

Observons que les agences de voyages, en plus des variations du billet souvent liées à la variation du billet vendu par la compagnie régulière, ajoutent leur propre couche de yield management afin d'augmenter davantage leurs revenus. Ils optimisent alors leur prix jusqu'à plusieurs fois par jour. Sur la Figure 1.6, nous pouvons voir l'évolution de prix d'un même billet Paris-Bangkok vendu par la compagnie régulière (*Qatar*) et l'agence de voyages (*GoVoyages*). Selon l'heure de la journée certaines agences peuvent ajouter des frais supplémentaires et dans le cas présent, *GoVoyages* applique la tarification du Tableau 1.1.

Horaire	0-2	2-4	4-6	6-8	8-19	19-23	23-24
Supplément(€)	7	6	0/6[5]	4	8	9	11

TABLE 1.1 – Évolution du supplément sur le site *GoVoyages* en fonction de l'heure d'achat (le 27 novembre 2012).

Qatar fait vendre une partie de ses sièges par *GoVoyages* qui propose des tarifs plus avanta-

[5]En fonction des compagnies

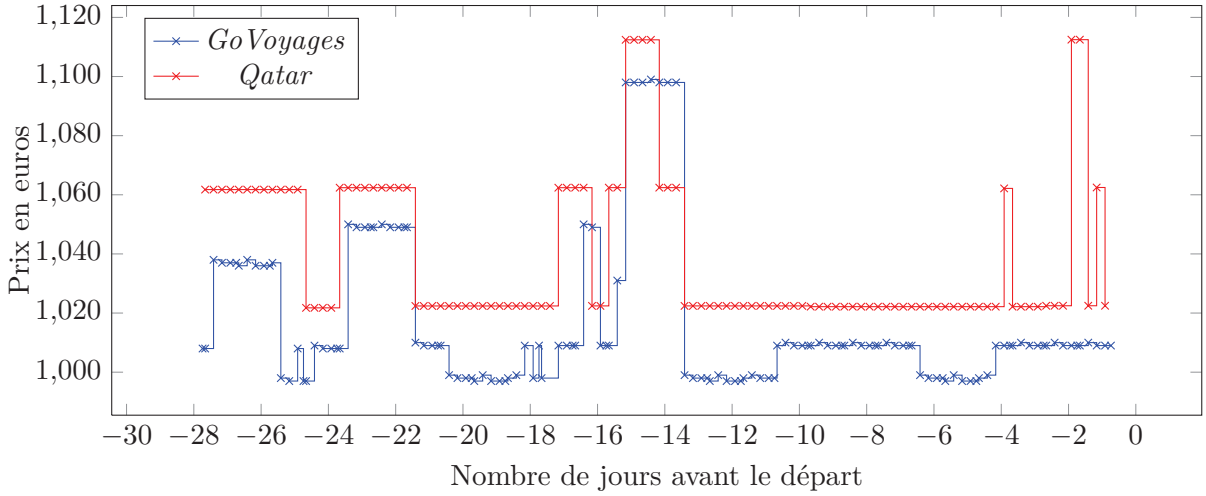


FIGURE 1.6 – Vol Paris-Bangkok opéré par *Qatar*, vendu par *GoVoyages* et *Qatar*

geux en échange d'une visibilité plus importante du vol. *GoVoyages* applique alors ses propres optimisations de prix tout en suivant les variations importantes de *Qatar*. Bien que nous n'utilisons pas cette information, elle pourrait nous permettre de prédire un saut de prix de manière presque certaine.

Les attributs

A chaque *id_flight* est associé un vecteur d'attributs $V_i(1), \ldots, V_i(p)$ qui regroupe toutes les informations possibles sur le billet vendu. Ces attributs sont très utiles pour extraire des statistiques sur notre base d'apprentissage. Ainsi il est facile de trouver la destination la plus populaire en Asie, ou bien le prix minimum observé depuis Paris vers n'importe quelle ville allemande opéré par une low cost. C'est en grande partie grâce à ces attributs que nous comptons prédire l'évolution de la trajectoire des vols, c'est pourquoi il est important de conserver l'intégralité des informations fournies par le site marchand et de créer de nouveaux attributs dérivés.

Les attributs peuvent être divisés en quatre catégories : tout d'abord nous utilisons les attributs définissant un vol unique (Tableau 1.2).

Nom	Type	Description
day	$\in 1-31$	Jour du mois de départ
month	$\in 1-12$	Mois de départ
year	$\mathbb{N}$	Année de départ
departureHour	$\in 0-23$	Heure de départ
departureMinute	$\in 0-59$	Minute de départ
transportCode	Code	transporteur du vol aller
departureStation	Code	Code aéroport de départ
arrivalStation	Code	Code aéroport d'arrivée

TABLE 1.2 – Caractéristiques du vol unique. Les trajets étant des allers-retours, ces attributs sont tous doublés.

Viennent ensuite les attributs dérivés de ces dernières caractéristiques (Tableau 1.3) qui détaillent les attributs temporelles (saison, départ le week end, jour de l'année, période vacances scolaires, etc.), les attributs géographiques (ville, pays, continent) et les attributs liés au trajet (long-courrier, nombre d'escales, etc.).

Nom	Type	Description
season	$\in 1,2,3,4$	Saison (1=printemps,...)
length_of_stay	$\mathbb{N}$	Durée du séjour
day_of_year	$\in 1-365$	Jour de l'année
day_of_week	$\in 1-7$	Jour de la semaine (1=lundi)
dep_on_we_dep	boolean	Départ le week end
dep_periode	$\in 1,2,3,4$	Période de la journée (1=très tot, 2=tot,3=après midi,4=soir)
holiday	$\in 0,1,2$	0=hors vacances, 1=vacances, 2=grandes vacances
stops	$\mathbb{N}$	Nombre d'escales
distance	$\mathbb{R}^+$	Saison
haul	$\in 0,1,2$	0=court-courrier, 1=moyen-courrier, 2=long-courrier
departureCity	Code	Code ville de départ
departureCountry	Code	Code pays de départ
departureContinent	Code	Code continent de départ
arrivalCity	Code	Code ville d'arrivée
arrivalCountry	Code	Code pays d'arrivée
arrivalContinent	Code	Code continent d'arrivée

TABLE 1.3 – Attributs dérivés des caractéristiques du vol unique

Nous avons aussi des attributs liés au site marchand (Tableau 1.4) et enfin les attributs contextuels qui évoluent avec les points de la série temporelle (Tableau 1.5) : à chaque instant t, nous calculons par exemple le nombre de sauts observés précédemment ou la somme des demandes faites pour ce vol. L'utilisation des ces derniers attributs est différente des autres, c'est pourquoi nous stockerons ces informations dans une table à part où la clef sera à la fois

l'*id_flight* et l'instant t.

Nom	Type	Description
provider	Code	Code du site marchand
type	$\in 0, 1, 2$	1=Agence de voyages, 2=Compagnie régulière, 3=Low cost,
directSeller	boolean	Vendeur direct
train	boolean	Trajet en train

Table 1.4 – Attributs liés au site marchand

Nom	Type	Description
volatility	$\mathbb{N}$	Nombre de sauts passés
volatility_hausse	$\mathbb{N}$	Nombre de sauts à la hausse passé
volatility_baisse	$\mathbb{N}$	Nombre de sauts à la baisse passé
demand	$\mathbb{N}$	Nombre de recherches utilisateur passé

Table 1.5 – Attributs contextuels évoluant avec la série temporelle

Nous sommes maintenant capables de construire facilement nos séries temporelles, d'accéder aux attributs d'un vol ou même d'afficher simultanément les séries de vols uniques vendus par des sites différents. Notre seule difficulté est de mettre en évidence les vols co-brandés : il faudrait pour cela créer un identifiant unique supplémentaire composé des attributs d'un vol unique sans les codes transporteurs mais avec les horaires des atterrissages. Cet ajout de complexité dans l'architecture de notre base de données diminuerait les performances pour une utilité limitée, c'est pourquoi nous avons préféré ne pas prendre en compte ce phénomène.

Une fois notre base de données construite nous pouvons commencer à l'explorer pour en extraire les premières informations sur les comportements des usagers et des séries temporelles. Ces comportements influeront sur les paramètres de notre modélisation des séries temporelles.

1.4 Statistiques expliquant le choix des paramètres

L'architecture de notre base de données nous permet maintenant d'extraire de nombreuses informations essentielles à la construction de notre modèle de prédiction : les séries de prix, leurs attributs et toutes les statistiques dérivées possibles. La collecte de nos données ayant débuté il y a plus de 6 ans, l'importante masse d'informations à traiter nécessite des choix dans la sélection des séries à étudier. Cette sélection doit à la fois refléter l'attente des utilisateurs sur un service de prédiction (fiabilité sur l'ensemble des routes, prédiction disponible le plus tôt possible, etc.) et prendre en compte les spécificités du marché du voyage (différences de comportements entre vols touristiques et business, suivant les durées de séjour, etc.).

Il est important de pouvoir proposer un service de prédiction à un maximum d'utilisateurs sans détériorer les performances. Il faut donc utiliser un panel représentatif des comportements de prix avec un échantillonage suffisant et choisir des longueurs de séries couvrant le plus large spectre de recherches utilisateurs.

Nous allons donc dans un premier temps décrire les paramètres des vols étudiés (route, durée de séjour, site marchand), puis nous expliquerons le choix de la longueur des séries temporelles.

1.4.1 Les trajets

Afin d'éviter la manipulation d'un trop grand nombre de séries, nous nous sommes limités dans un premier temps à un ensemble de routes représentatives (Tableau 1.6). Nous avons choisi des destinations touristiques (Paris-Bangkok) et professionnelles (Paris-Budapest) pour des durées de séjours (*Lenght of stay*) touristiques (7 et 14 jours) et professionnelles (3 jours). Nous avons par la même occasion sélectionné des vols long-courriers, moyen-courriers et court-courriers (Paris-Toulouse). Les résultats des agences de voyages nous permettent de récupérer des vols sur un large panel de compagnies aériennes et les sites des compagnies nous garantissent des prix stables et plus fiables. Les deux sites peuvent vendre le même billet mais les agences modifient fréquemment le prix pour ajuster leur marge dynamiquement. L'objectif est de collecter le plus de comportements différents tout en minimisant la taille des données.

Lowcost			
From	*To*	*Provider*	*Length of stay*
Paris (ORY)	Budapest	EasyJet (U2)	3,7
Paris (ORY)	Toulouse	EasyJet (U2)	3,7
Paris (ORY)	Marrakech	Transavia (HV)	3,7
Paris (XGB)	Toulouse	IDTGV (TGV)	3,7
Agences de Voyages			
From	*To*	*Provider*	*Length of stay*
Paris (CDG)	Budapest	VoyagesSNCF (VOY)	3,7
Paris (ORY)	Marrakech	VoyagesSNCF (VOY)	3,7
Paris (CDG)	Toulouse	VoyagesSNCF (VOY)	3,7
Paris (PAR)	Bangkok	GoVoyages (GOV)	7,14
Amsterdam	Barcelone	GoVoyages (GOV)	3,7
Paris (CDG)	New York	GoVoyages (GOV)	7
Compagnies Régulières			
From	*To*	*Provider*	*Length of stay*
Paris (PAR)	Bangkok	Quatar (QR)	7,14
Amsterdam	Barcelone	Austrian (OS)	3,7

TABLE 1.6 – Routes de la base d'apprentissage

Il faut par ailleurs sélectionner des trajets fréquemment recherchés pour pouvoir construire des séries consistantes, c'est pourquoi nous avons choisi les durées de séjour les plus demandées (Figures 1.7 et 1.8).

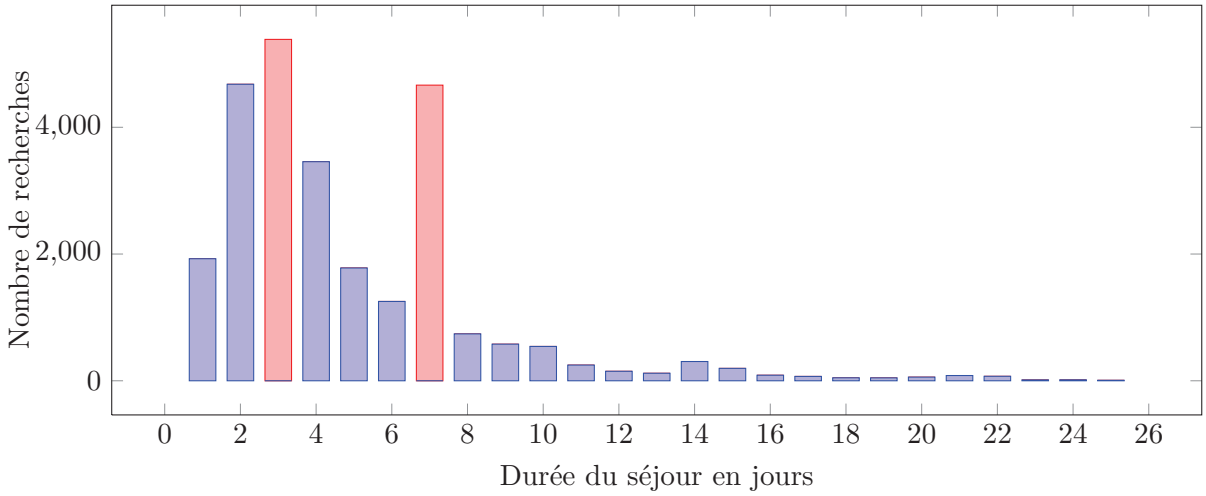


FIGURE 1.7 – Nombre de recherches par durée de séjour pour un Paris-Budapest

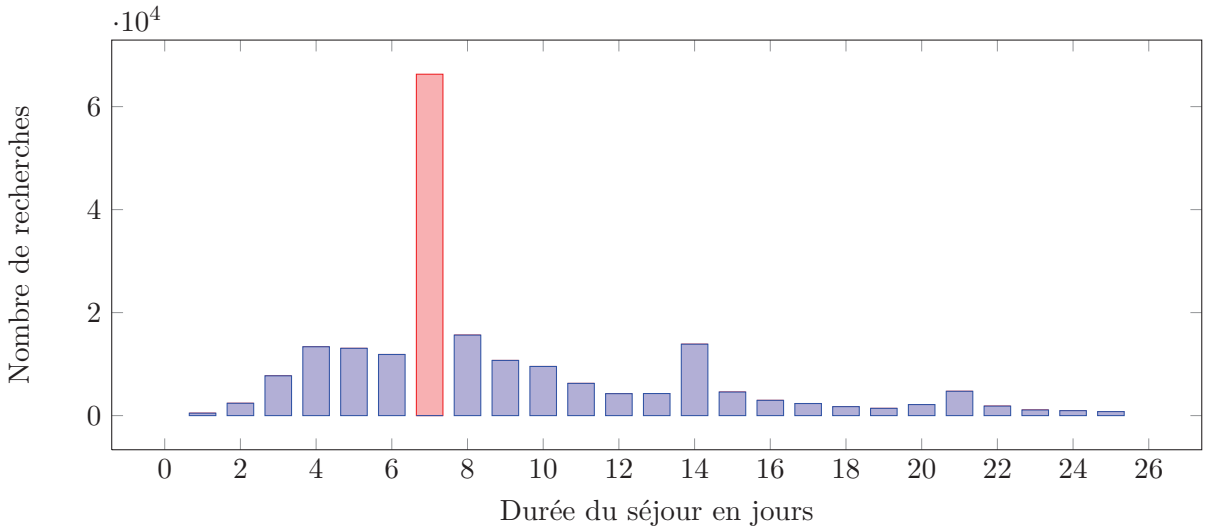


FIGURE 1.8 – Nombre de recherches par durée de séjour pour un Paris-New York

Pour étudier correctement le comportement des séries temporelles, il est souhaitable de détecter le maximum de changements de prix. Nous devons alors sélectionner les vols les plus recherchés pour garantir des séries de prix fidèles à la réalité. Les recherches sur les routes sélectionnées sont fréquentes mais pas nécessairement régulières. Plus la date de départ approche et plus les recherches sont nombreuses, c'est pourquoi nous voulons dans un premier temps étudier les derniers jours de nos séries pour nous assurer une base consistante. Cette réduction de la taille de nos séries implique une réduction du pourcentage d'utilisateurs susceptibles d'obtenir des prédictions. Inversement, en augmentant le nombre de jours étudiés nous augmenterions le spectre d'utilisateurs bénéficiant du service mais les performances et la fiabilité du service

diminueraient. Nous allons donc étudier le comportement des voyageurs pour trouver la durée optimale des séries temporelles à choisir.

1.4.2 La longueur des séries temporelles

Par l'évolution de la demande nous estimons la période à laquelle commencer l'étude des séries de prix. Le but est de couvrir le plus de demandes tout en conservant un jeu de données raisonnable et des séries consistantes. La ventilation représente l'évolution en pourcentage de la demande par rapport au nombre final de recherche. La Figure 1.9 représente les ventilations d'allers-retours pour différents types de destinations et différentes durées de séjours. Pour les long-courriers, la première moitié des recherches se fait entre un an et trois mois avant le départ et entre un an et 1 mois avant le départ pour les moyen-courriers. Dans notre exemple, nous couvrons avec les 28 derniers jours des séries temporelles 15% des utilisateurs pour les long-courriers et 50% pour les moyen-courriers, mais au global un tiers des usagers de liligo.com avec un nombre de recherches suffisant pour avoir des séries correctement échantillonnées.

Par ailleurs, le mois précédant la date de départ correspond à une forte augmentation des sauts comme le montre la Figure 1.10 et donc de la volatilité des prix. C'est alors un moment crucial pour l'utilisateur qui voit les prix changer jusqu'à plusieurs fois par jour et où le conseil à l'achat s'avère déterminant.

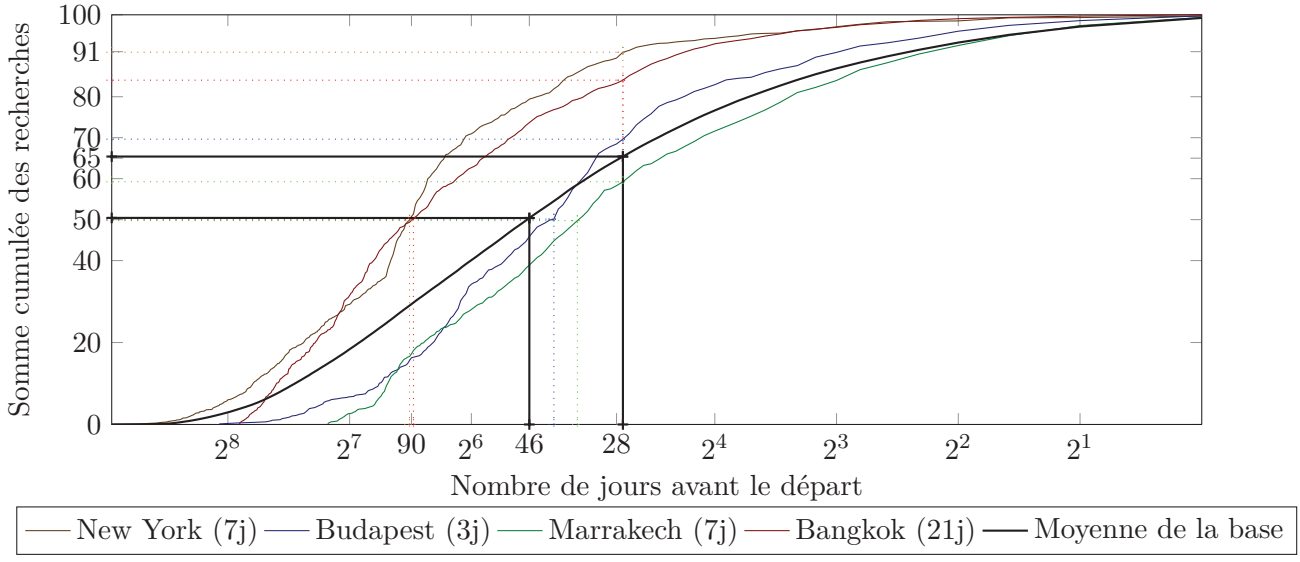


FIGURE 1.9 – Ventilation pour un vol Paris-New York du 14-04-2012 au 21-04-2012, un vol Paris-Bangkok du 28-07-2012 au 18-08-2012, un vol Paris-Marrakech du 21-04-2012 au 28-04-2012 et un vol Paris-Budapest du 01-11-2012 au 04-11-2012

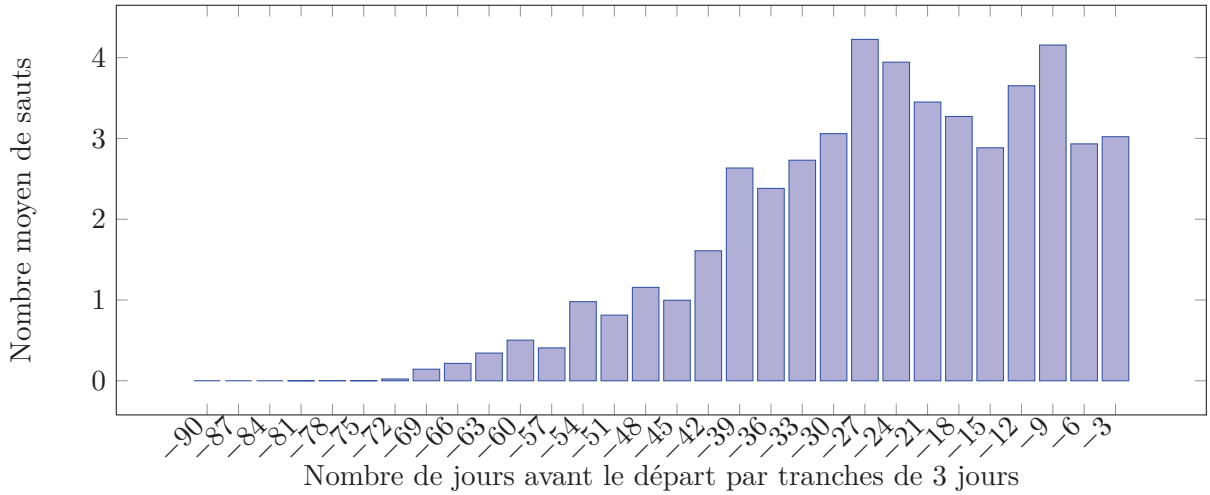


FIGURE 1.10 – Histogramme de l'évolution du nombre de sauts avant la date de départ par tranches de 3 jours.

En accord avec les contraintes d'échantillonnage évoquées dans le chapitre précédent et au vu de la Figure 1.9, nous avons donc décidé de nous focaliser sur les 28 derniers jours des séries de prix, mais nous conservons toutes les informations précédant le dernier mois pour une application étendue à 90 jours de notre méthode. De la même manière, nous avons dans un premier temps restreint notre corpus de vols, mais pour une future extension de notre service, nous continuons d'enregistrer tous les résultats des recherches utilisateurs.

1.5 Pertinence

L'initiative de ce projet est venue de la constatation que le voyageur est volontairement tenu dans l'ignorance des évolutions de prix pour entretenir une confusion quant au meilleur moment pour acheter son billet. Certaines règles sont alors communément utilisées telles que : "Plus on achète tôt, moins on paye cher" et son corollaire "Rien ne sert d'attendre, les prix ne font qu'augmenter".

Tout d'abord, il faut rappeler que nous ne tentons pas d'aider l'utilisateur à choisir la meilleure date pour partir, ni le meilleur moment de l'année pour acheter son billet.

Pour chaque recherche utilisateur, plusieurs offres à différents horaires et proposées par différentes compagnies sont présentées. Dans ces choix l'utilisateur va identifier le vol qu'il souhaite prendre et c'est à ce moment qu'il lui sera conseillé d'acheter son billet immédiatement ou de reporter son achat pour économiser de l'argent.

Sur d'autres sites[6], seule l'évolution du plus bas prix observé quotidiennement est étudiée. Lorsque la prédiction d'une baisse de prix est prodiguée, l'utilisateur n'a pas la garantie que les caractéristiques du vol au plus bas prix à 7 jours corresponderont à celles du vol actuellement le moins cher. La compagnie aérienne affrétant le vol et les horaires étant des critères souvent

[6]Bing.com ou Kayak.com par exemple

très importants pour l'utilisateur, nous avons choisi de faire une prédiction pour chaque billet retourné par la recherche.

Nous évitons ainsi le risque de détériorer la qualité du vol en question en passant, par exemple, d'un vol direct à un vol à multiples escales d'une durée beaucoup plus importante que nécessaire.

Nous montrons donc dans cette section la pertinence de notre service dans l'aide à la décision du voyageur.

1.5.1 Meilleur moment pour acheter

Le comportement le plus courant chez un voyageur est d'acheter ses billets d'avion en avance pour s'assurer un prix raisonnable et éviter les hausses successives de prix. Des études ont montré que les prix n'étaient pas strictement croissants et qu'une période située aux alentours de 8 semaines avant le départ était optimale [37]. Sur la Figure 1.11 représentant l'évolution des prix des billets de l'intégralité de notre base de données, il existe bien une période où les prix sont généralement au plus bas et qui correspond à environ 50 jours avant la date de départ (soit 7 ou 8 semaines avant le départ).

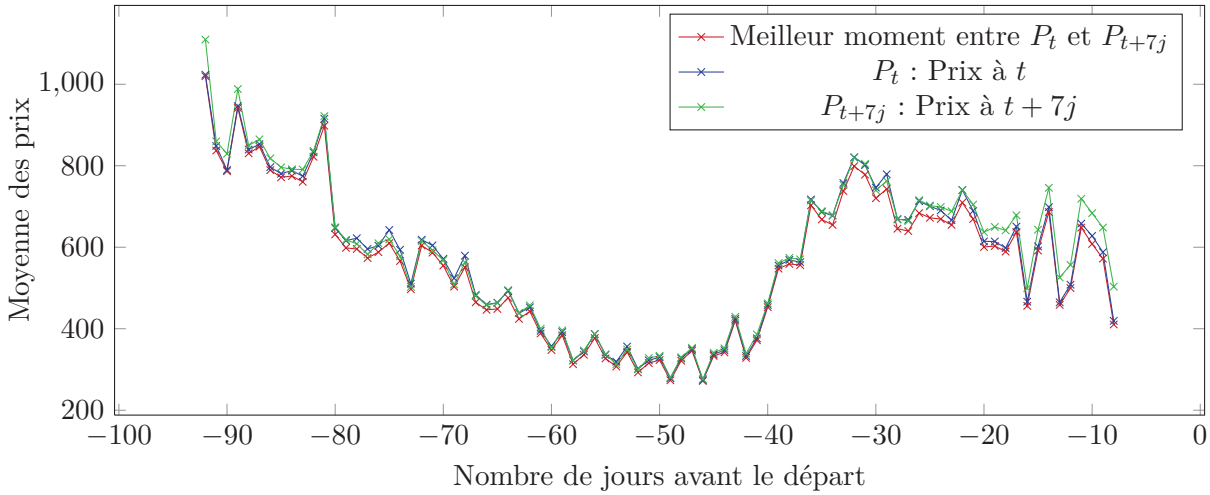


FIGURE 1.11 – Évolution de la moyenne des prix en fonction du moment d'achat

Cependant, comme nous l'avons vu sur la Figure 1.9, la moitié des usagers du site font leurs recherches au delà du 46e jour avant le départ, ce qui correspond à la période où les prix commencent à augmenter. Et c'est à cette période qu'il est crucial de conseiller au voyageur d'acheter son billet avant que la hausse ne soit trop importante ou bien d'attendre, car malgré le comportement globalement haussier de l'ensemble des séries, il est probable qu'une baisse apparaisse.

Toujours sur la Figure 1.11, on observe l'évolution du prix à l'instant t et celui 7 jours après et l'évolution du meilleur des deux prix (optimum). Ainsi on constate que la période la plus propice au conseil d'achat, c'est-à-dire lorsque la courbe optimale s'éloigne le plus des autres

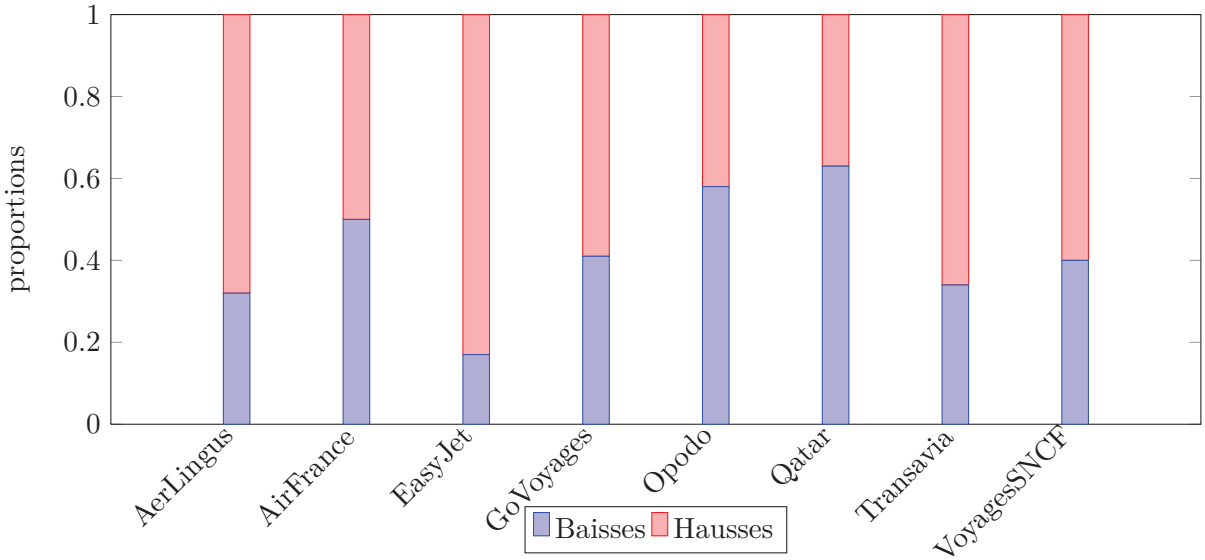


FIGURE 1.12 – Proportion de hausses et de baisses par provider

courbes, se trouve entre 35 et 20 jours avant la date de départ. Dans cette période le bon conseil d'achat ferait économiser de l'argent à l'utilisateur. En ce qui concerne les derniers jours, le conseil est aussi primordial mais évitera surtout au voyageur de perdre de l'argent. La hausse des prix est alors beaucoup plus importante et il donc plus risqué d'attendre une baisse de tarif.

On peut enfin remarquer les très faibles différences de prix entre 60 et 40 jours avant le départ. Les sauts de prix sont alors moins fréquents ce qui prouve encore que c'est une période propice à l'achat de billet d'avion.

1.5.2 Proportion des baisses

Nous insistons sur la nécessité de notre service en montrant que la proportion des baisses de prix est beaucoup plus importante qu'on ne le pense. Sur la Figure 1.12, on constate par exemple qu'*Air France* et *Qatar* pratiquent autant voire plus de baisses de prix que de hausses. En revanche, les compagnies low cost pratiquant des optimisations de prix plus agressives et ne vendant généralement pas de billets remboursables ou échangeables, affichent peu de baisses de prix. Certains attributs (ici la compagnie opérante le vol, *supplier*) sont donc plus discriminants que d'autres dans la prédiction des évolutions de prix. Ainsi un billet *EasyJet* aura plus de probabilité d'augmenter dans les 7 jours qu'un vol d'une autre compagnie.

Nous allons maintenant montrer qu'il est possible d'économiser de l'argent quelle que soit la date avant le départ en prenant les bonnes décisions.

1.5.3 Gain optimal

Nous définissons un gain ou une perte comme valeur absolue de la différence entre le prix initial et le prix à 7 jours qui sera additionnée si la prédiction est bonne et soustraite dans le cas contraire. L'optimal ou oracle est le prédicteur qui sait dans tous les cas quelle décision prendre. Sur la Figure 1.13, nous calculons l'évolution de l'écart relatif à l'oracle en fonction de la date de prédiction pour une personne qui prendrait systématiquement le billet le jour de la recherche, pour une personne qui prendrait systématiquement le billet 7 jours après la recherche et enfin pour une personne qui choisirait au hasard.

Trois périodes se dégagent de ce graphique : avant 80 jours avant la date de départ il est conseillé d'acheter tout de suite. Puis de -80 jours à -50 jours, les prix ont tendances à baisser pour atteindre le prix minimal. Enfin après les 50 jours avant la date de départ il est de nouveau conseillé d'acheter ses billets le jour de la recherche.

La zone rouge représente le gain que nous pourrions apporter à l'utilisateur en lui donnant un conseil plus précis que le précédent et représente donc l'objectif à atteindre.

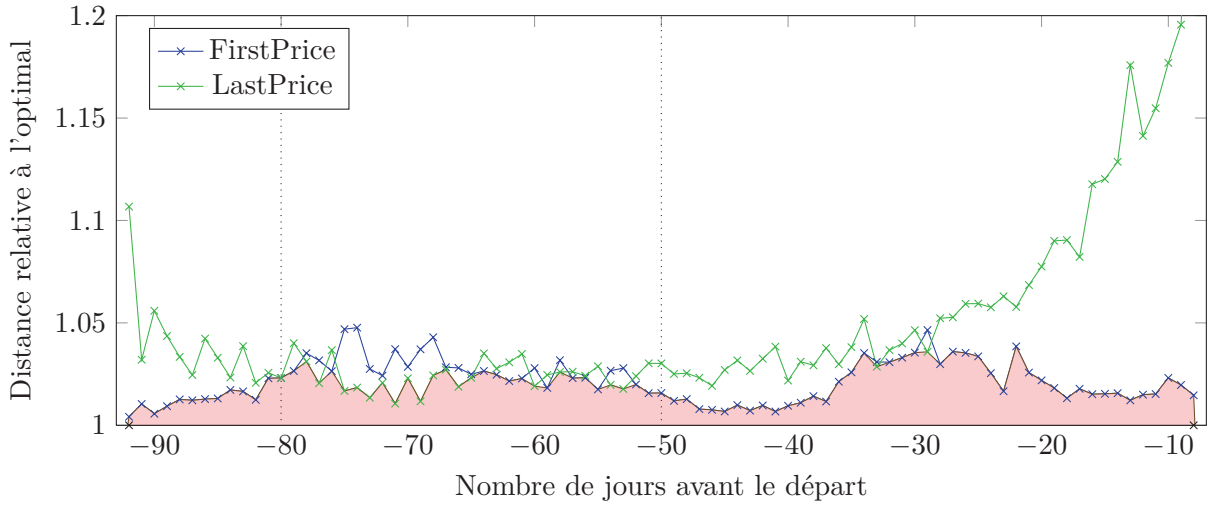


FIGURE 1.13 – Distance relative à l'optimal

1.6 Conclusion et perspectives

Dans ce chapitre nous avons tenté de décrire l'environnement dans lequel nous travaillons, et le contexte économique du monde du voyage. Le "revenue management", technique qui consiste à optimiser le prix de chaque siège d'un avion grâce à la prédiction de la demande, rend le processus d'achat du voyageur complexe et obscur. Le consommateur est maintenu dans l'ignorance et l'incertitude, parfois encouragé à réserver longtemps à l'avance, parfois exhorté à réserver précipitamment à la dernière minute pour bénéficier d'offres présentées comme dégriffées.

Dans un second temps nous avons décrit la structure de notre base de données contenant les recherches utilisateurs et les alertes mails. Elle nous permet d'extraire facilement les séries de

prix, de mettre en parallèle les mêmes billets, vendus par différents sites marchands et d'afficher les attributs correspondants aux vols.

Nous avons tenté d'avoir une base représentative de la demande des utilisateurs ainsi que des différents types de voyageurs, d'avoir des séries de prix régulièrement échantillonnées, tout en limitant la taille de nos données. Pour cela nous avons construit une base sur des destinations et des longueurs de séjour limités, et nous nous sommes focalisés sur les 28 derniers jours avant le départ, garantissant ainsi un nombre de points suffisant pour détecter la grande majorité des sauts de prix.

Enfin nous avons montré qu'il était nécessaire d'apporter une information sur l'évolution du prix aux usagers de liligo.com qui ne possèdent pas la vision globale du marché du tourisme. Il est courant de penser que les prix dans le milieu du tourisme aérien ne font qu'augmenter, mais nous avons démontré qu'il en était autrement et qu'avec notre service nous pourrions détecter une baisse de prix dans une courbe au comportement globalement haussier.

Avec l'accroissement du trafic de liligo.com, il sera possible d'élargir le nombre de destination et la durée des séries temporelles, permettant ainsi de proposer le service à un plus grand nombre d'utilisateurs dans des périodes durant lesquelles ils en ont le plus besoin. Il sera notamment intéressant d'élargir le système d'alerte pour enregistrer l'intégralité des résultats de recherche afin de créer des séries temporelles complètes.

Dans le chapitre suivant nous allons aborder en détail la modélisation des séries temporelles par des processus ponctuels poissonniens dans le but de regrouper les séries de même comportements. Une analyse des comportements des séries temporelles validera notre choix de transformation et nous permettra de choisir les paramètres optimaux de notre nouvelle représentation.

Chapitre 2

Représentation des trajectoires

Contents

Introduction

Dans ce chapitre, nous abordons l'étape de transformation des trajectoires en une représentation adaptée à la comparaison des comportements. C'est avec cette nouvelle représentation que nous allons utiliser nos différents algorithmes. En effet, nous tentons de regrouper les évolutions de prix similaires entre elles afin d'en extraire des comportements types. Plusieurs problématiques se posent alors quant au calcul de similarité entre deux courbes de prix : il faut choisir la métrique la plus adaptée à nos données et pouvoir l'appliquer à des vols de toutes natures. Pour cela nous devons nous détacher des ordres de prix des séries qui sont pour la plupart complètement différents et approximer les moments des "sauts" de prix pour prendre en considération les décalages temporels de comportements similaires.

Lorsqu'une agence de voyages vend un billet d'avion, lui même vendu par la compagnie qui opère le vol, la courbe de prix du billet de l'agence de voyages évolue souvent en suivant les modifications de la courbe de prix du billet de la compagnie aérienne. Nous obtenons alors deux courbes très similaires avec un décalage de quelques heures entre les changements de prix du site de la compagnie aérienne et ceux de l'agence de voyages. S'ajoute à ce phénomène, la temporisation de la mise à jour de la base de données des prix mis à disposition de liligo.com par les agences de voyages. L'approximation temporelle des courbes nous permet alors de rapprocher ces séries similaires.

Dans un premier temps nous étudions le comportements des trajectoires observées. Bien comprendre la nature de nos séries de prix va nous permettre de mieux reconstituer les séries mal échantillonnées et inversement de les approximer sans perdre d'information afin de réduire la taille de notre base de données. Comme dans le chapitre précédent, nous nous servons des statistiques issues de notre base pour justifier nos choix.

Nous décrivons ensuite la première étape de transformation des courbes de prix en une représentation par des processus ponctuels dans le plan temps-rendement basé sur le framework de Cox [13]. Nous ne conservons que les sauts de prix relatifs éliminant ainsi la problématique de différence d'ordre de grandeur entre les séries. Cette étape est totalement bijective conservant ainsi toutes les informations quant aux variations de prix (seul l'échantillonnage des séries sera perdu). Durant cette étape il sera possible de supprimer le bruit causé par certaines variations de prix non pertinentes.

Enfin nous détaillons la seconde étape de la transformation de nos données qui consiste à modéliser les séries de rendement par des processus ponctuels poissonniens. Nous supposons que le nombre d'apparitions de changements de prix suit une distribution de Poisson dont le paramètre $I_y(s,t)$ évolue dans le temps. Nous sommes dans le cas d'un processus de Poisson inhomogène, dont on estime l'intensité $I_y(s,t)$ par des noyaux. Nous approximons donc les moments d'apparition des sauts et leur rendement par un niveau de gris où l'intensité des cellules représentera le nombre de sauts dans l'intervalle de temps et de rendement correspondant. En créant des fenêtres de temps-rendements, nous réalisons une empreinte du comportement global d'un vol, qu'on utilisera par la suite pour le rapprocher de celle des autres vols. Nous détaillerons par ailleurs l'étape qui consiste à simuler une courbe de prix à partir de ce niveau de gris, nous permettant ainsi de retrouver une courbe similaire à la série temporelle initiale.

2.1 WorkFlow

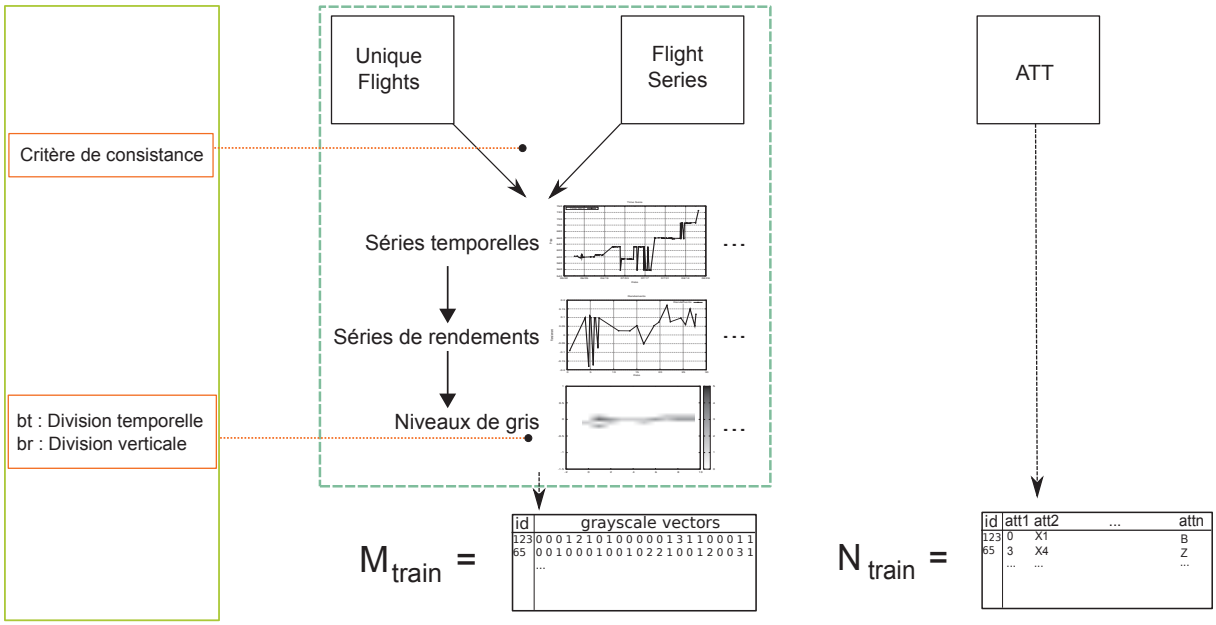


FIGURE 2.1 – Workflow - Etape de préparation des données

2.2 Notations

i	:	Numéro du vol de la base d'apprentissage $i \in 1, ..., n$
V_i	:	Vecteur de p attributs du vol i parmi l'ensemble des attributs $\mathcal{V}$
$p_i(t)$	:	Courbe de prix du vol i
$(P_i(1,k), P_i(2,k))$	:	Données brutes composées de $P_i(1,k)$ la date du k-ième échantillon collecté pour le vol i et $P_i(2,k)$ le prix correspondant observé
$T_{init}^{(i)}$	:	Date du premier point de collecte du vol i
$T_0^{(i)}$	:	Date de départ du vol i
$s_i(t)$	:	Séries de rendements du vol i
$(T_k^{(i)}, s_k^{(i)})$	:	k-ième point temps-rendement de la représentation en processus ponctuel de la trajectoire du vol i
$\mathcal{R}$	:	Ensemble des cases du plan temps-rendement
b_r	:	Dimension verticale des cases (rendement)
b_t	:	Dimension horizontale des cases (temps)
$X_i(s,t)$	:	Niveaux de gris du plan temps-rendement pour le vol i à la case (s,t)

2.3 Séries temporelles

Nous rappelons qu'un vol unique (*id_unique_flight*), représentant un trajet indépendemment du site marchand, est défini par un aéroport de départ (departureStation), un aéroport d'arrivée (arrivalStation), une date de départ (jour-mois-année heure : minute), une date de retour, un code de transport aller [1] (ex : **AF**315, **IB**412**U2**650) et un code de transport retour.

Nous rappelons aussi que chaque vol possède une série temporelle par site marchand sur lequel il est vendu : la série de prix issue du site de la compagnie aérienne, et celles créées grâce aux prix des agences de voyages proposant le même vol. L'attribut *provider* qui défini le site marchand (la compagnie aérienne ou l'agence de voyage) est donc associé à l'*id_unique_flight* pour identifier une série temporelle unique à laquelle nous attribuons un identifiant unique *id_flight* .

2.3.1 Problèmes d'échantillonnage

L'échantillonage de nos séries est un problème important pour la construction de notre base d'apprentissage. Les résultats des recherches utilisateurs, qui nous servent a construire notre base d'apprentissage, sont erratiques et ne nous donnent pas des séries avec des points régulièrement espacés. L'enjeu est donc de savoir si un vol peu être considéré comme consistant ou non, c'est-à-dire déterminer si la perte d'information est trop importante pour ne pas utiliser la série dans notre base d'apprentissage.

Pour ce faire, il est intéressant d'observer le temps moyen d'attente entre deux sauts de prix en fonction du nombre de jours avant la date de départ. Sur la Figure 2.2, on observe un temps moyen de 2 jours et demi entre deux sauts sur l'ensemble des dates de recherche mais on note une forte augmentation de la fréquence des changement de prix dans les 20 derniers jours.

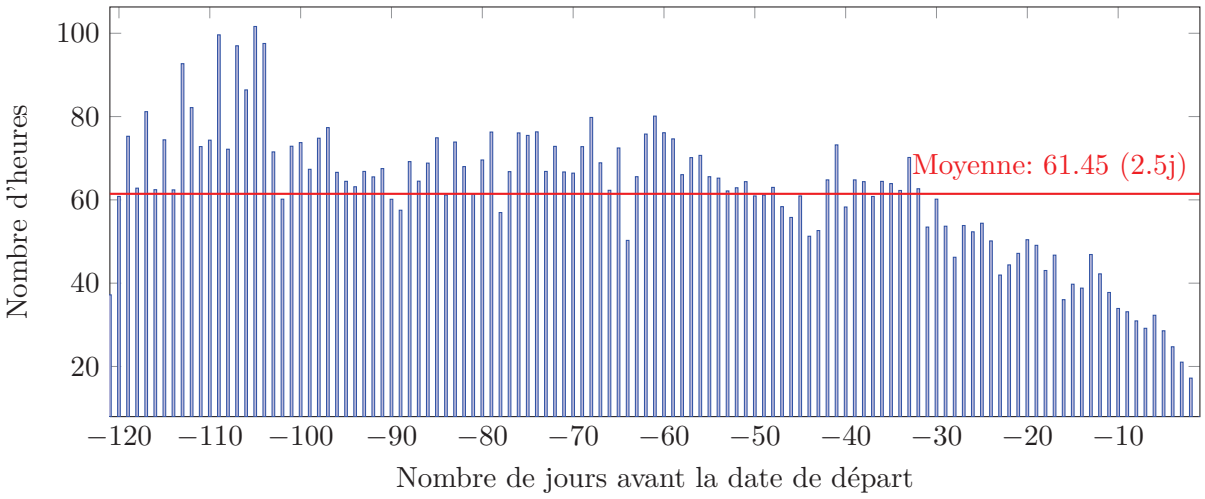


FIGURE 2.2 – Temps moyen d'attente entre deux sauts en heures. En abscisses nous avons le moment du premier saut et en ordonnées le temps d'attente moyen avant le saut suivant.

[1] Dans le cas de vols avec escales, le code sera la concaténation des codes de chaque vol intermédiaire

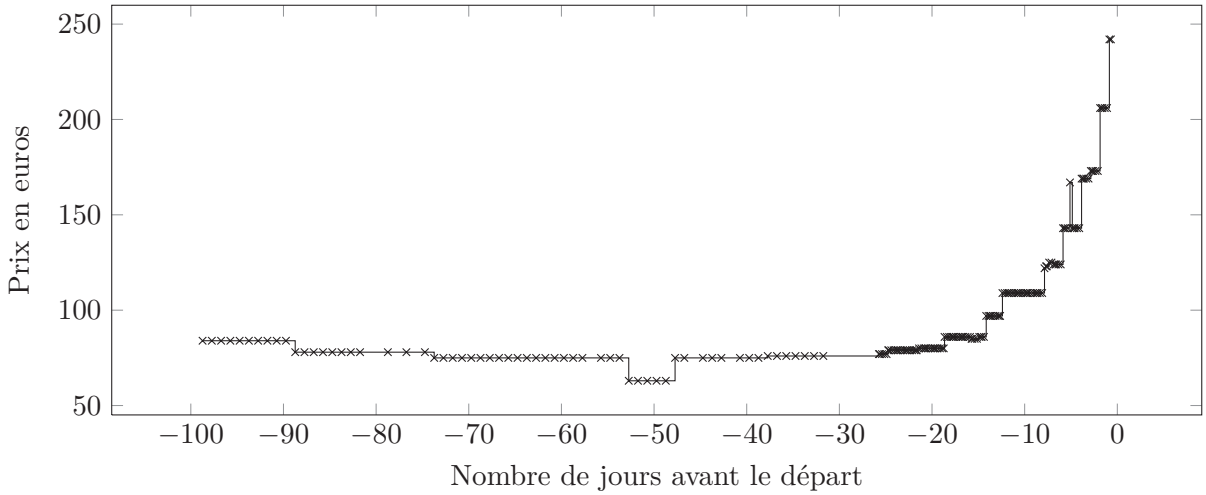


FIGURE 2.3 – Série temporelle : Paris-Toulouse, départ le 06/11/2012 pour 7 jours par *EasyJet*

Notre corpus est composé de vols ayant un historique de 28 jours maximum sur un panel de destinations restreint (voir chapitre précédent) et de vols ayant un historique de 90 jours sur l'ensemble des recherches utilisateurs. Dans le premier groupe, nous avons décidé de conserver les vols échantillonnés toutes les 6 heures et possédant plus de 80% des $4 \times 28 = 112$ points maximum. Le nombre de recherches utilisateurs augmentant significativement dans le dernier mois avant la date de départ, ce critère de sélection nous fournit un nombre satisfaisant de courbes consistantes. Pour une question de taille de base de données, nous ne conservons que les points espacés de 6 heures et supprimons les prix intermédiaires. La Figure 2.9 est un très bon exemple de série correctement échantillonnée sur l'ensemble de la période étudiée. Pour le second groupe, l'échantillonage est seulement quotidien car il est impossible de trouver un vol où un utilisateur à vérifié les prix toutes les 6 heures pendant 90 jours. La Figure 2.2 nous confirme par ailleurs qu'il n'est pas nécessaire d'avoir plus de points car cet échantillonage nous permet de détecter une grande majorité des sauts de prix. L'exemple de la Figure 2.3 représentant une trajectoire issue des deux corpus confirme notre choix.

Enfin le problème des vols complets reste un point qui n'est pas pris en compte. Lorsqu'une cabine est remplie et que le taux de sur-réservation (overbooking) est atteint, le vol n'est plus proposé à la vente et disparaît donc de nos résultats. La détection de ces vols est problématique car nous ne sommes pas en mesure de différencier un vol complet, d'un vol retiré de la base d'une agence ou même d'une mauvaise collecte des prix. C'est pourtant un point important dans l'aide à la décision d'achat de l'utilisateur. Faire perdre de l'argent à un usager en lui conseillant d'attendre lorsque le prix augmente peut avoir un impact négatif sur sa confiance dans le service, mais lui faire rater un vol peut s'avérer beaucoup plus néfaste. Malheureusement les vols complets sont naturellement ignorés lors de l'étape de sélection des vols consistants et la requête de sélection nous permettant d'identifier des vols complets s'avère complexe et jamais totalement fiable.

Les grandes compagnies régulières ont un taux de remplissage moyen tournant autour de

85% et les low costs sont autour de 90%. D'après le groupe Aéroports de Paris, en 2012, le taux moyen de remplissage des vols en partance de Paris a été de 79.7%[2]. Nous pensons donc qu'une grande majorité des vols possèdent des places jusqu'aux derniers jours de vente, mais c'est un axe que nous souhaitons clairement améliorer.

2.3.2 Comportements des trajectoires

A titre d'illustration, considérons la série temporelle de la Figure 2.4. Elle décrit l'évolution des prix d'un vol Paris-Marrakech partant le 25 Février 2011 sur *Royal Air Maroc* (le *supplier*) et proposé par le site *VoyagesSNCF* (le *provider*), pour un voyage de 3 jours. Nous constatons tout d'abord que la trajectoire est constante par morceaux : le prix varie d'un plateau à un autre, déclenché par le système de "yield management" comme décrit dans le chapitre précédent. Selon les compagnies et les techniques de revenue management qu'elles utilisent, ces plateaux peuvent être en nombre fixe et limité ou bien complètement dynamiques. Il est intéressant d'observer l'évolution du nombre de plateaux en fonction du provider, de la destination, du jour de l'année (jour de départ) ou même de comparer entre les agences de voyages, les compagnies régulières et les low cost.

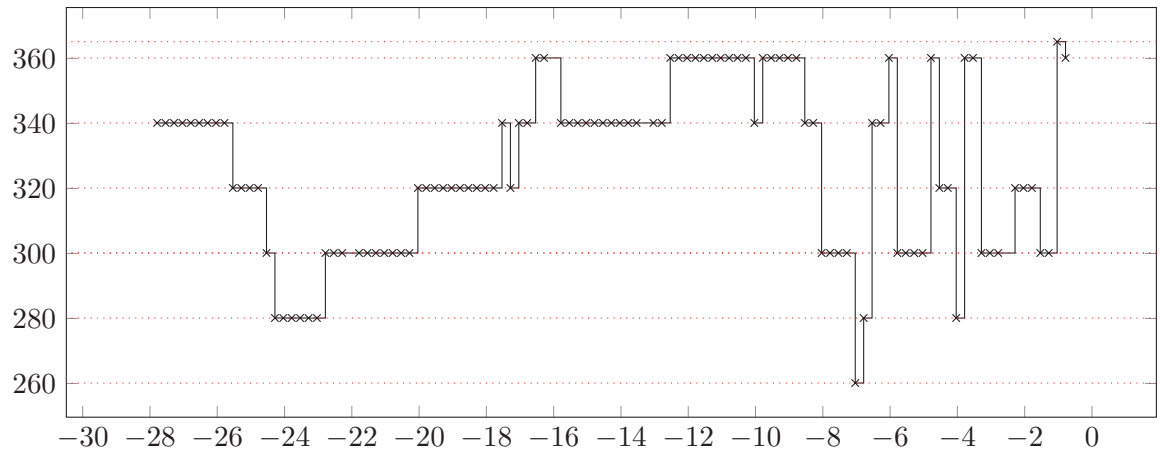


FIGURE 2.4 – Vol Paris-Marrakech, départ le 25/02/2011 pour 7 jours opéré par *Royal Air Maroc* et vendu par *VoyagesSNCF*. Nous observons la présence de 7 plateaux distincts.

Dans la méthode d'optimisation du *bid-price vector* décrite dans le chapitre précédent, le nombre de plateaux est défini par le nombre de classes par cabines, chacune ayant un prix supérieur à la classe inférieure. Cependant il arrive que la demande ne corresponde pas aux prévisions et qu'une classe (celle courante généralement) voit son tarif modifié (à la baisse généralement). Sans modification manuelle du prix des classes, le nombre de plateaux par courbe serait sensiblement le même pour des vols similaires (même route, même période de l'année, etc.) et ne dépasserait pas le nombre de classes tarifaires de la cabine.

[2] http://www.aeroportsdeparis.fr/ADP/Resources/06c3db89-8afd-401e-9882-64775db4fd66-ADP_TraficDecembre2012.pdf

Sur la Figure 2.5, on observe la densité du nombre de plateaux sur l'ensemble de vols ayant des points durant les 90 derniers jours. Cette base contient des vols échantillonnés quotidiennement jusqu'à la date de départ afin d'avoir les statistiques les plus proches de la réalité mais excluant ainsi les vols complets. La majorité des vols possèdent un nombre de plateaux inférieur à 12 et la moyenne est inférieure à 8. La connaissance du nombre de plateaux et de leur valeur, nous permettrait de prédire de façon certaine une baisse si le prix atteignait le plateau supérieur et une hausse inversement.

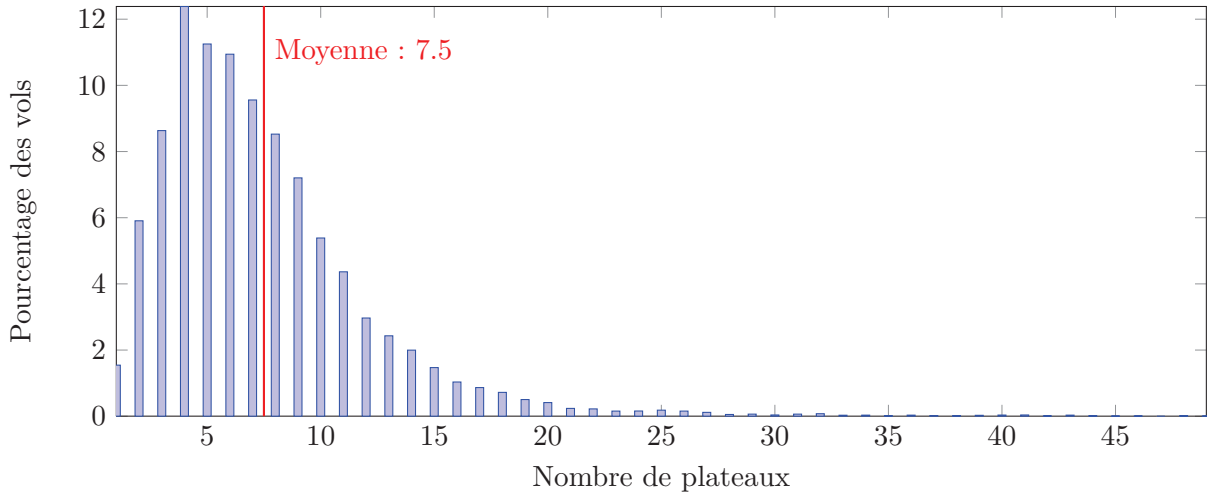


FIGURE 2.5 – Pourcentage des vols par nombre de plateaux

Nous avons divisé les sites marchands (*provider*) en trois catégories : les agences de voyages, les compagnies régulières et les low cost. Les Figures 2.6 et 2.7 représentent l'évolution du pourcentage de vols par nombre de plateaux en fonction du type du site marchand. Nous étudions ces distributions sur la base des vols échantillonnés toutes les 6 heures durant les 28 derniers jours (Figure 2.7) et sur la base des vols à 90 jours (Figure 2.6).

Il est intéressant de noter que l'échantillonage quotidien sur la base de 90 jours n'est pas suffisant lorsque que l'on se rapproche de la date de départ. En effet sur la base possédant 4 points par jours, les low cost ont en moyenne 18 plateaux en 28 jours indiquant une fréquence très importante de changements de prix, loin devant les compagnies régulières qui ont environ 8 plateaux. Les plateaux correspondant souvent au nombre de classes par cabines, celles-ci sont bien plus importantes chez les low-cost.

On remarque une augmentation similaire pour les agences de voyages, mais celle-ci s'explique par la taxe supplémentaire que certaines agences appliquent en fonction de l'heure d'achat du billet. Ces multiples changements durant la journée augmentent alors fictivement le nombre de plateaux. Nous sommes cependant tentés de penser que le nombre de plateaux devrait être très proche de celui de la compagnie régulière et que des ententes existent afin d'accorder les prix entre les deux parties.

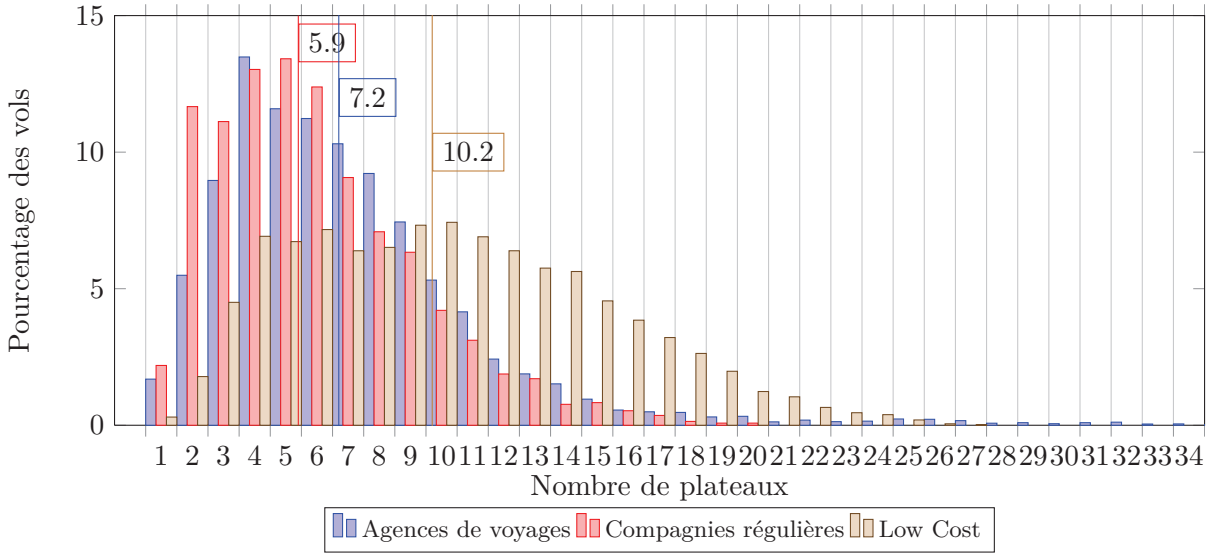


FIGURE 2.6 – Base des vols à 90 jours

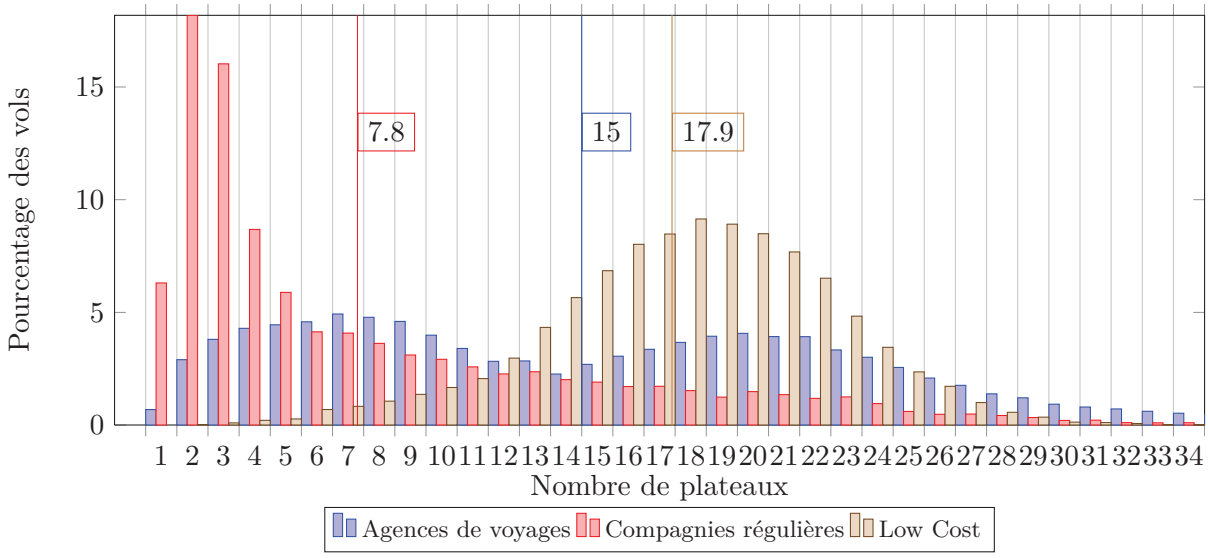


FIGURE 2.7 – Base des vols à 28 jours

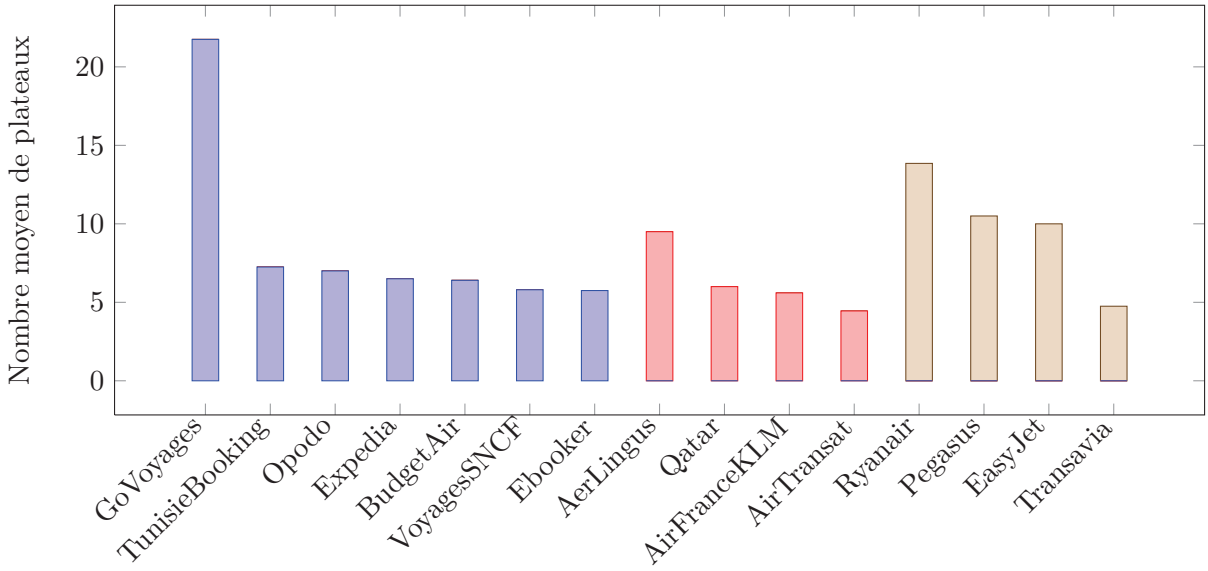


FIGURE 2.8 – Nombre moyen de plateaux par *provider*. En bleu les agences de voyages, en rouge les compagnies régulières et en brun les low cost

La Figure 2.8 nous indique le nombre moyen de plateaux par provider regroupés en catégories : les agences de voyages, revendeurs de billets achetés aux compagnies régulière ; les compagnies régulières appliquant généralement des prix plus élevés avec un service supplémentaire et yield management plus constant ; et enfin les Low Cost compagnies basant leur revenu sur la vente à bas prix de billet sans options[3] et optimisant le prix de chacun des sièges de leurs avions. Ces graphiques nous donnent une indication sur la technique utilisée pour l'optimisation des revenus ainsi que sur l'agressivité quant aux modifications des prix. Même si cette Figure ne nous indique pas la fréquence ni le nombre de changements de prix, on peut remarquer que *GoVoyages* possède un très grand nombre de plateaux toujours dû à leur taxe quotidienne et qu'ensuite vient *Ryanair*, compagnie Low Cost célèbre pour ses techniques de yield management poussées et d'ajouts multiple de taxes.

Le fort taux de plateaux d'*Aer Lingus* qui est pourtant la compagnie nationale Irlandaise s'explique par l'offre hybride qu'elle propose : afin de répondre à la menace des offres low cost, certaines compagnies régulières propose un modèle de vente hybride où coexistent un produit à l'image des low cost avec tous les services additionnels en option et un produit "premium" où tous les services sont inclus [39].

Inversement *VoyagesSNCF*, comme le souligne la Figure d'exemple 2.4, semble ne pas ajouter de frais ni de yield management supplémentaires, offrant ainsi des courbes de prix semblable à celles des compagnie régulières. Il s'agit alors d'une commission de distributeur fixée.

[3]Billets non transférable, non échangeable, non remboursable ; Limitation du nombre et du poids des bagages en cabine ; Limitation en nombre et en poids, voire facturation, des bagages en soute ; Repas ou prestations payants etc.

2.3.3 Interpolation des trajectoires

Il est pratique de considérer l'évolution des prix jusqu'à la date de départ de l'élément i comme une fonction constante par morceaux $p_i(t)$ du temps continu $t \in [T_0^{(i)} - T_{init}, T_0^{(i)}]$, où $i \in I$ désigne l'identifiant *id_flight* de la série, $T_0^{(i)}$ est l'instant du départ (en jours) de l'élément i et T_{init} est le nombre de jours séparant la date de départ et le premier point collecté. On introduit alors les instants de sauts $T_k^{(i)}$ numérotés dans l'ordre croissant de telle sorte que $T_0^{(i)}$ est la date de départ (les indices k sont donc négatifs). Pour chaque i, P_i représente le prix d'achat du voyage à l'instant t. Plus précisément, ces données seront représentées sous la forme d'une matrice P_i, $n_i \times 2$, tel que $P_i(1,k)$ représente le temps (négatif) séparant la date du départ de la date d'achat du $k$ième prix observé (en seconde) et $P_i(2,k)$ le prix correspondant.

On suppose par convention que le prix est continu à droite, d'où la courbe de prix interpolée définie pour tout $t < T_0^{(i)}$ par :

$$p_i(t) = \sum_{k \leq 0} p_i(T_{k-1}^{(i)}) \mathbb{1}_{[T_{k-1}^{(i)}, T_k^{(i)})}(t) \ .$$

avec pour tout $k \leq 0$, $T_{k-1}^{(i)} = T_k^{(i)} - \delta_{-k+1}$. On note $p_i(t-)$ la limite de $p_i(s)$ quand $s \uparrow t$. Sur l'exemple de la Figure 2.9 (Paris-Bangkok, départ le 11/01/2013 pour 14 jours avec *Qatar Airways*), on observe deux zones (en rouge) où aucune donnée n'a été trouvée. Etant donnée la nature de nos courbes, nous décidons donc de considérer que le prix à été constant durant ce laps de temps. Cette interpolation des trajectoires est applicable uniquement pour un nombre de points manquants faible et espacé pour minimiser les erreurs d'approximation. C'est pourquoi le bon échantillonage de nos vols est un point crucial dans l'identification des comportements des trajectoires.

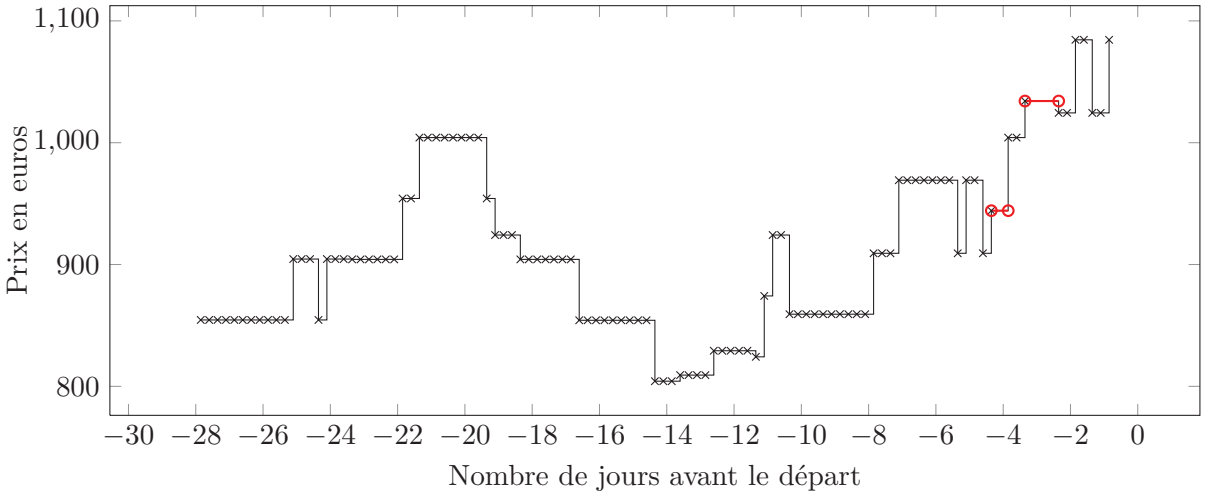


FIGURE 2.9 – Time Series : Paris-Bangkok départ le 11/01/2013 pour 14 jours par *Qatar*

2.4 Représentation par des processus ponctuels

Une fois les séries temporelles constituées, nous voulons être capable de les comparer afin d'en extraire des comportements type. L'échelle des prix n'étant pas la même pour toutes les routes, nous allons transformer les séries de prix en séries de variations relatives.

On définit alors la suite des rendements :

$$s_k^{(i)} = \{P_i(T_k^{(i)}) - P_i(T_k^{(i)}-)\}/P_i(T_k^{(i)}-), \quad k \leq 1 \ .$$

où $p_i(t-)$ désigne le prix juste avant l'instant t. Il est clair que la courbe des prix peut être entièrement reconstruite à partir du prix initial et de la suite des points $(T_k^{(i)}, s_k^{(i)})$ (seul l'échantillonage sera perdu). Le calcul à t_1 d'un prix à t_2 se formule ainsi :

$$P_i(2, t_2) = P_i(2, t_1) \prod_{k \in [t_1, t_2]} (1 + s_k^{(i)})$$

Sur la Figure 2.10, nous observons la transformation de la série temporelle de la Figure 2.9 : un point sous l'axe des abscisses correspond à une baisse de prix alors qu'un point au-dessus, représente une hausse.

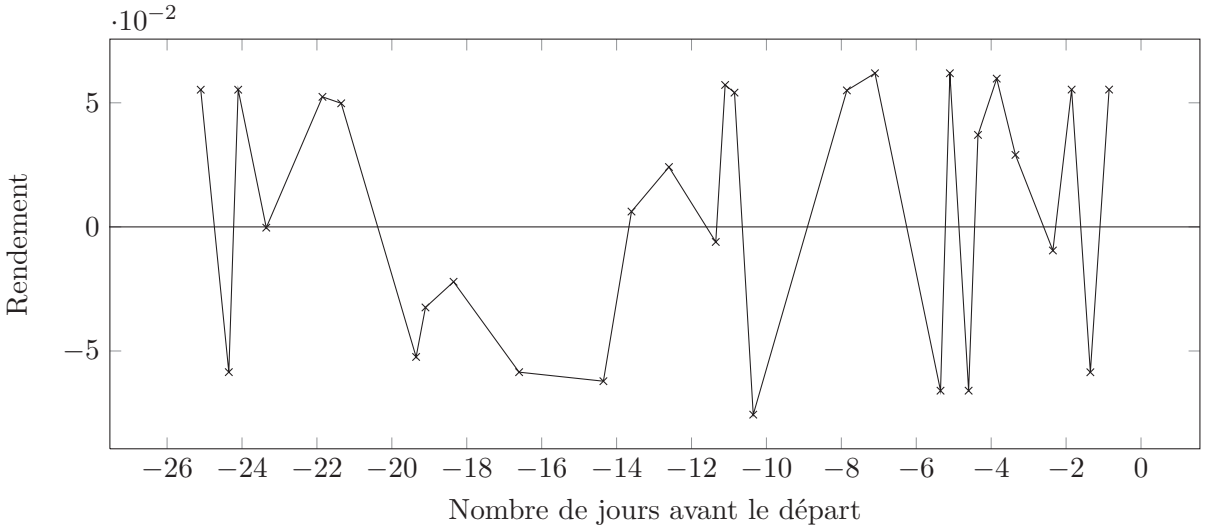


FIGURE 2.10 – Série de rendements : Paris-Bangkok départ le 11/01/2013 pour 14 jours par *Qatar*

Cette suite nous permet dans un premier temps de nous affranchir des valeurs absolues des prix : en ne nous intéressant qu'aux sauts relatifs des prix, nous pourrons donc comparer des voyages ayant des ordres de prix complètement différents en vue d'une agrégation des séries de comportements similaires.

Lors de cette étape il sera possible de “corriger” les fluctuations de prix anormaux : celles de faibles amplitudes et fréquentes et celles ponctuelles de forte intensité. Les premières sont des micro-variations dues à des commissions supplémentaires qui peuvent varier plusieurs fois par jour et que l'on rencontre souvent chez les agences de voyages qui appliquent un second yield management afin d'optimiser leurs marges : sur *GoVoyages* par exemple, pour inciter les internautes à venir à des heures “creuses”, la taxe évolue comme expliqué dans le chapitre précédent (Tableau 1.1). Nous rappelons que les-dites agences peuvent négocier des tarifs avec les compagnies aériennes et proposent des places qui peuvent se révéler être beaucoup moins chères que celles vendues par la compagnie aérienne elle-même. En revanche, ces places subissent de nombreuses variations quotidiennes de très faible valeur (souvent inférieur à 1%) et peuvent bruiter inutilement les courbes de prix. La Figure 2.11 est un parfait exemple du bruit engendré par ces micro-variations : la courbe noire représente la série de prix d'un vol Amsterdam-Barcelone opéré par *Ibéria* mais vendu par *GoVoyages*. Les variations sont quotidiennes et de faible amplitude, et la courbe rouge représente le comportement globale de la trajectoire. Le bruit créé lors de la construction de la série de rendement réduirait la capacité de nos algorithmes à comparer les séries. Par ailleurs, nous souhaitons donner à l'utilisateur un conseil pertinent où la prédiction à la baisse représenterait une baisse significative du prix. Hors, en conservant ces faibles variations, nous risquons de détériorer l'expérience utilisateur. En conséquence, nous appliquons un filtre sur les sauts qui enlève toutes les variations inférieures à 1% transformant la courbe exemple de la Figure 2.10 en celle rouge de la Figure 2.13.

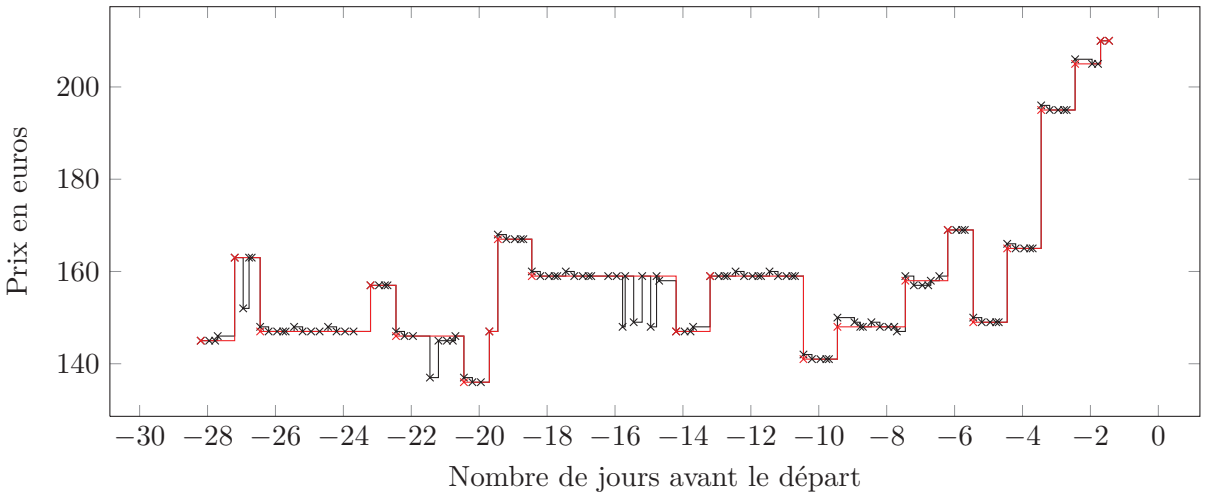


FIGURE 2.11 – Exemple de prix fluctuant de manière artificielle : *GoVoyages*. Amsterdam-Barcelone opéré par *Ibéria* 05/06/2011 pour 3 jours.

Les secondes fluctuations pouvant altérer les trajectoires sont les sauts de prix importants, ponctuels, suivis d'un retour au prix précédent dans l'intervalle qui suit. Ces “variations” sont souvent dues à une information érronée ou “obsolète” du prix que nous fourni le site marchand. Sur la Figure 2.12, nous observons clairement ce phénomène sur un vol Paris-Bangkok vendu et

opéré par *Qatar*. Là encore, nous avons tracé en rouge ce qu'aurait dû être la courbe, même si il est impossible de vérifier que la première baisse de prix à 900€ qui a duré moins de 6 heures n'a pas vraiment existé. Il est beaucoup plus difficile de repérer à la volée ces erreurs.

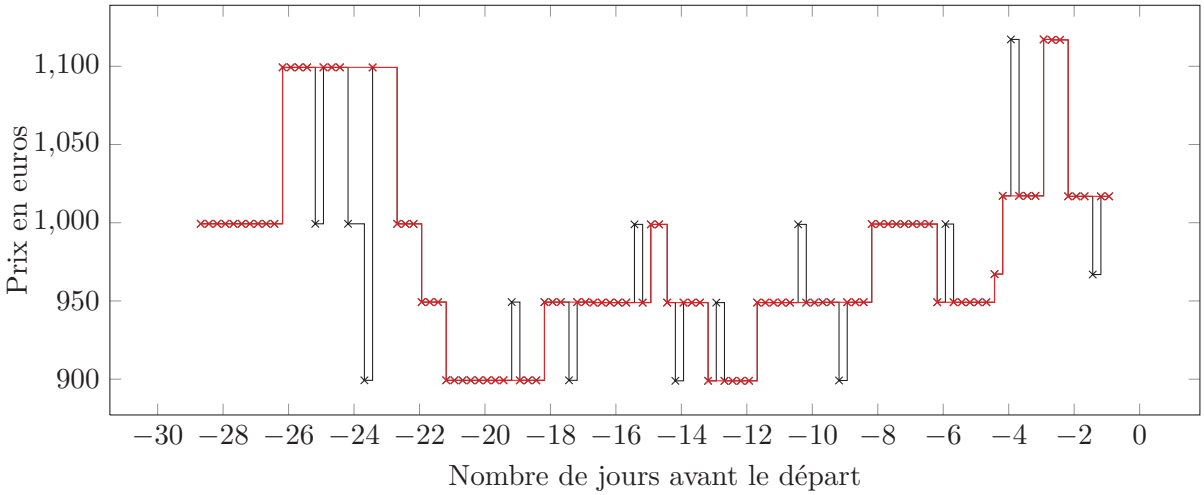


FIGURE 2.12 – Exemple de changements de prix perturbés par des mises à jour différées du site marchand : *Qatar*. Paris-Bangkok départ le 29/04/2011 pour 7 jours

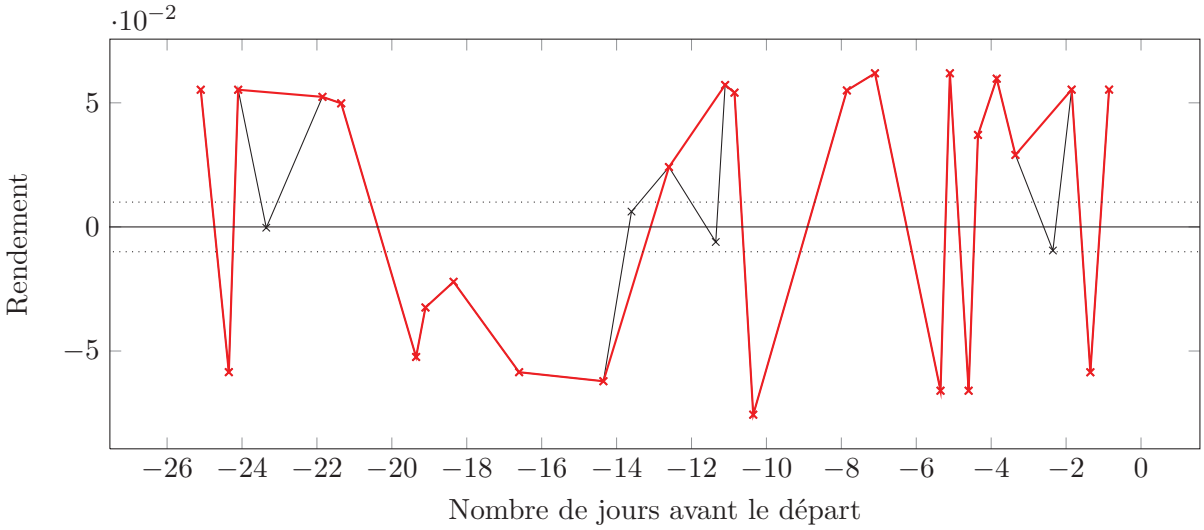


FIGURE 2.13 – Série de rendements après filtre : Paris-Bangkok départ le 11/01/2013 pour 14 jours par *Qatar*

2.5 Modélisation par de processus ponctuels poissonniens

Après s'être abstrait des ordres de prix, il nous faut maintenant approximer le moment des sauts de prix afin de rapprocher des vols au comportement similaire mais décalés dans le temps. L'exemple le plus classique est l'agence de voyage qui vend un billet d'une compagnie régulière et qui va suivre les variations de cette dernière avec un certain décalage. Nous nous intéressons aux comportements de courbes, c'est pourquoi l'ordre de prix et le moment des sauts peut être approximé. Nous décidons donc d'approximer les temps des sauts ainsi que leur rendement pour créer une "empreinte" du comportement des séries temporelles.

2.5.1 Estimation de l'intensité - visualisation par niveaux de gris

Nous modélisons donc la précédente représentation par un processus ponctuel marqué inhomogène (processus de poisson dont les évènements sont pondérés [14]), dont l'intensité $X_i(s,t)$ peut être estimée sous la forme d'une image pixélisée qui prend les valeurs

$$\widehat{X}_i(s,t) = \frac{1}{b_t b_r} \sum_{k \leq -1} 1_R(T_k^{(i)}, s_k), \quad (s,t) \in R\,, \tag{2.1}$$

pour un pixel rectangulaire R de taille $b_t \times b_r$. Le plan temps /rendement est partitionné par une grille régulière de tels pixels dont les intensités sont égales au nombre de sauts par unité de surface dans le plan temps /rendements. Sur la Figure 2.14, on observe la transformation de la série de rendements exemple en un niveau de gris. Plus le nombre de sauts est important dans l'aire de la case, plus celle-ci est foncée.

Un estimateur de l'intensité des sauts et de la densité des rendements s'écrit, pour une bande passante $b = (b_t, b_r) \in (0,\infty)$ et un noyau $K : [0,1] \mapsto \mathbb{R}_+$ tel que $\int K = 1$,

$$X_i(s,t) = \frac{1}{b_t b_r} \sum_{k \leq 1} K(\{s - s_i(-k)\}/b_r) K(\{T_k^{(i)} - t\}/b_t), \quad s \in [0,1], t \leq T_0^{(i)}\,.$$

Remarquons que la limite d'appartenance à un pixel de trouve au coin inférieur gauche de telle sorte qu'un rendement exactement égal à un multiple de b_r sera considéré comme appartenant à la cellule supérieure. De la même manière, un rendement apparaissant à un instant t multiple de b_t sera inclu dans la cellule la plus à droite ($3^{\text{ème}}$ point de la Figure 2.14). On évite les effets de bords en $t \sim T_0^{(i)}$ et $s \sim 0$ (K a support sur $[0,1]$).

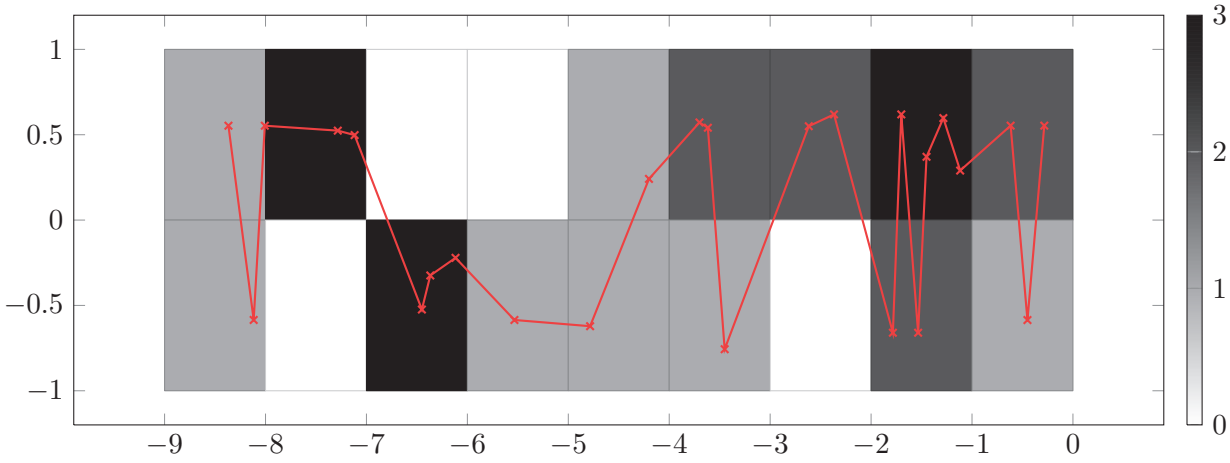


FIGURE 2.14 – Niveau de gris après filtre : Paris-Bangkok départ le 11/01/2013 pour 14 jours par *Qatar*

Le lissage des faibles variations pendant l'étape de transformation des séries temporelles en séries de rendements limite l'intensité des cases autour de 0 évitant ainsi une confusion entre un saut faible et un bruit négligeable. Comme expliqué plus haut, du point de vue du service final, il est important de distinguer une baisse significative de prix d'une faible variation afin de proposer une aide à la décision pertinente.

De la même manière, il arrive que le prix d'un vol double soudainement causant la présence d'un "pixel" coloré à une ligne élevée. Cette situation peut intervenir lorsqu'un billet en classe économique n'est plus disponible et que le même billet en business est proposé. Le niveau de gris étant un quadrillage de l'espace temps-rendement, cela engendre l'apparition de nombreuses cases vides et propage le redimensionnement à toutes les images pixélisées. En effet, notre utilisation des niveaux de gris nécessite des matrices de dimensions égales impliquant un remplissage par des cases vides des vols à faibles rendements.

Les matrices utilisées pour le clustering deviennent alors beaucoup trop grandes et inutilement éparses. Ces évènements étant rares, nous pouvons facilement conserver les sauts de faibles rendements dans un même ordre de grandeur et réduire les sauts à fort rendement en appliquant le logarithme népèrien. On redéfinit alors la suite des rendements ainsi :

$$s_k^{(i)} = ln(P_i(T_k^{(i)})) - ln(P_i(T_k^{(i)}-)), \quad k \leq 1 \ .$$

transformant alors le calcul à t_1 d'un prix à t_2 jours en :

$$P_i(2, t_2) = P_i(2, t_1) \prod_{k \in [t_1, t_2]} e^{s_k^{(i)}}$$

2.5.2 Choix de la bande passante

b_r et b_t sont des critères primordiaux qui vont permettre de généraliser les comportements des séries sans les uniformiser.

b_r représente l'intervalle d'intensité dans lequel seront regroupé les sauts d'intensité similaire. $br = 0.1$ signifie qu'on regroupe tous les sauts relatifs par intervalle de 10%.

b_t, quant à lui, représente le découpage temporel des niveaux de gris. Choisir $bt = 72heures$ va créer des fenêtres de 3 jours dans lesquelles tous les sauts seront regroupés.

La Figure 2.15 donne un exemple de subdivisions plus ou moins fines de b_r et donc des intervalles de prix. Le graphique 2.15(a) est une division binaire de l'axe des rendements où seul le sens de variation est pris en compte ($b_r = 1$). Ce découpage, lorsqu'il s'agira de prédire uniquement une hausse ou une baisse aura son importance car en simplifiant les niveaux de gris, on simplifie le regroupement en comportements similaires. En revanche, cette représentation binaire est très sensible aux bruits de faibles variations, c'est pourquoi là encore l'étape de filtre est importante. Puis sur 2.15(b) ($b_r = 0.1$) et 2.15(c) ($b_r = 0.05$) les intervalles sont affinés réduisant progressivement les approximations des niveaux de gris.

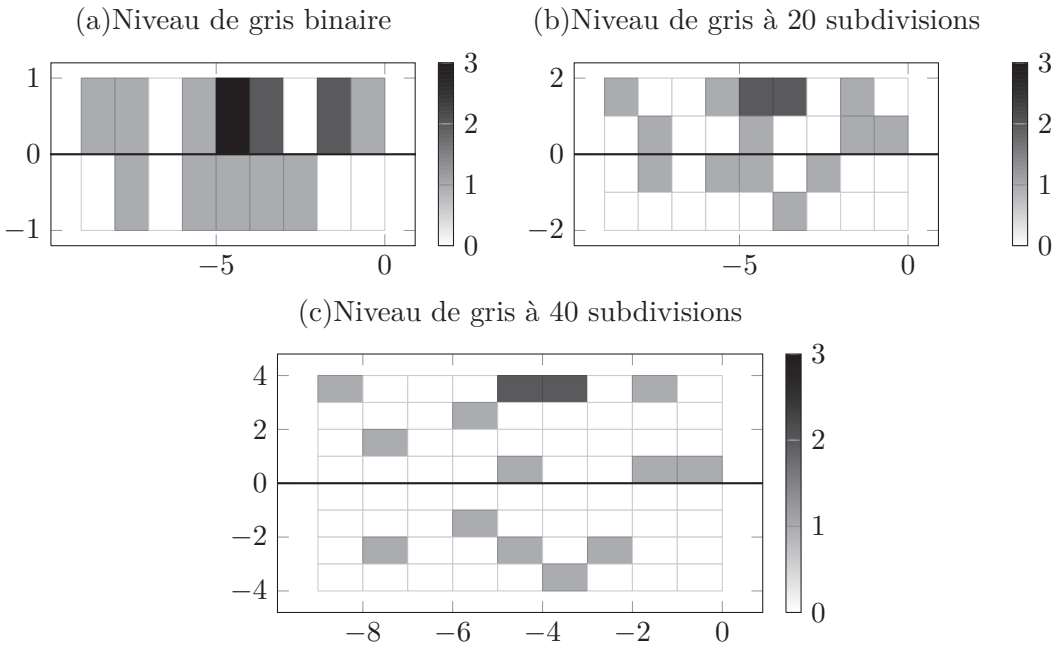


FIGURE 2.15 – Évolutions des niveaux de gris avec l'agrandissement des subdivisions verticales

Cette nouvelle représentation est surjective ce qui implique une perte d'information. Il est toutefois important de pouvoir reconstituer une série de prix à partir d'un niveau de gris. Nous allons donc voir comment simuler une série de rendements grâce à un niveau de gris.

2.6 Simulation

Une trajectoire peut être reconstruite à partir d'une valeur initiale du prix p_t à un instant donné t (avant T_0) et de la connaissance des points (T_k, s_k) tels que $t \leq T_k < 0$.

On part de X que l'on connaît sur un domaine D donné et on veut donc simuler le processus de Poisson ponctuel (PPP) $N = \sum_k \delta_{(T_k,S_k)}$ sur D. Dans le cadre de notre application on peut supposer que

$$\int_D X(s,t)\, ds\, dt < \infty \ .$$

La méthode générale consiste à simuler $M \sim \text{Poi}(\int_D X(s,t)\, ds\, dt)$ puis $(Y_k)_{k\leq -1}$ i.i.d. de densité $\frac{X(s,t)}{\int_D X(s,t)\, ds\, dt}$ sur $(s,t) \in D$, et indépendants de M. Alors, le processus

$$N = \sum_k \delta_{Y_k}$$

est un PPP d'intensité X sur D, cf. [43]. La difficulté repose donc dans la simulation d'une suite de v.a. i.i.d. de densité donnée. Pour un X "général", la méthode du rejet permet d'effectuer cette simulation.

Dans le cas simple où X est constant par morceaux sur des pixels de D, la méthode suivante donnera une simulation à moindre coût. Prenons

$$D = \bigcup_i D_i \ ,$$

avec des D_i disjoints 2 à 2 et $X(s,t)$ constant pour $(s,t) \in D_i$. On note X_k la valeur associée au pixel D_i. Il suffit alors de simuler la restriction $N^{(i)}$ de N à D_i pour chaque i. Chaque $N^{(i)}$ est en effet un PPP homogène sur le domaines D_i. On peut donc le simuler comme suit, et de façon indépendante pour chaque i,

$$N^{(i)} = \sum_{k=1}^{M^{(i)}} \delta_{(T_k^{(i)}, s_k^{(i)})}$$

avec $N^{(i)} \sim \text{Poi}(X_i)$ et $(T_i^{(k)}, S_i^{(k)})_i$ i.i.d. de loi uniforme sur D_i et indépendants de $N_0^{(i)}$.

Sur la Figure 2.16, nous observons un exemple de transformation d'une série temporelle en une série de rendements puis en une image pixelisée. A partir de cette dernière image, nous avons simulé une courbe de rendement, puis appliqué la formule de reconstitution d'une série temporelle pour afficher en rouge la série ainsi simulée.

Nous sommes donc en mesure de créer, à partir d'un niveau de gris, des courbes au comportement statistique similaire à la série temporelle initiale. Nous verrons plus tard qu'il est possible de moyenner les comportements d'un ensemble de simulations afin d'extraire d'un niveau de gris probabilité d'évolution.

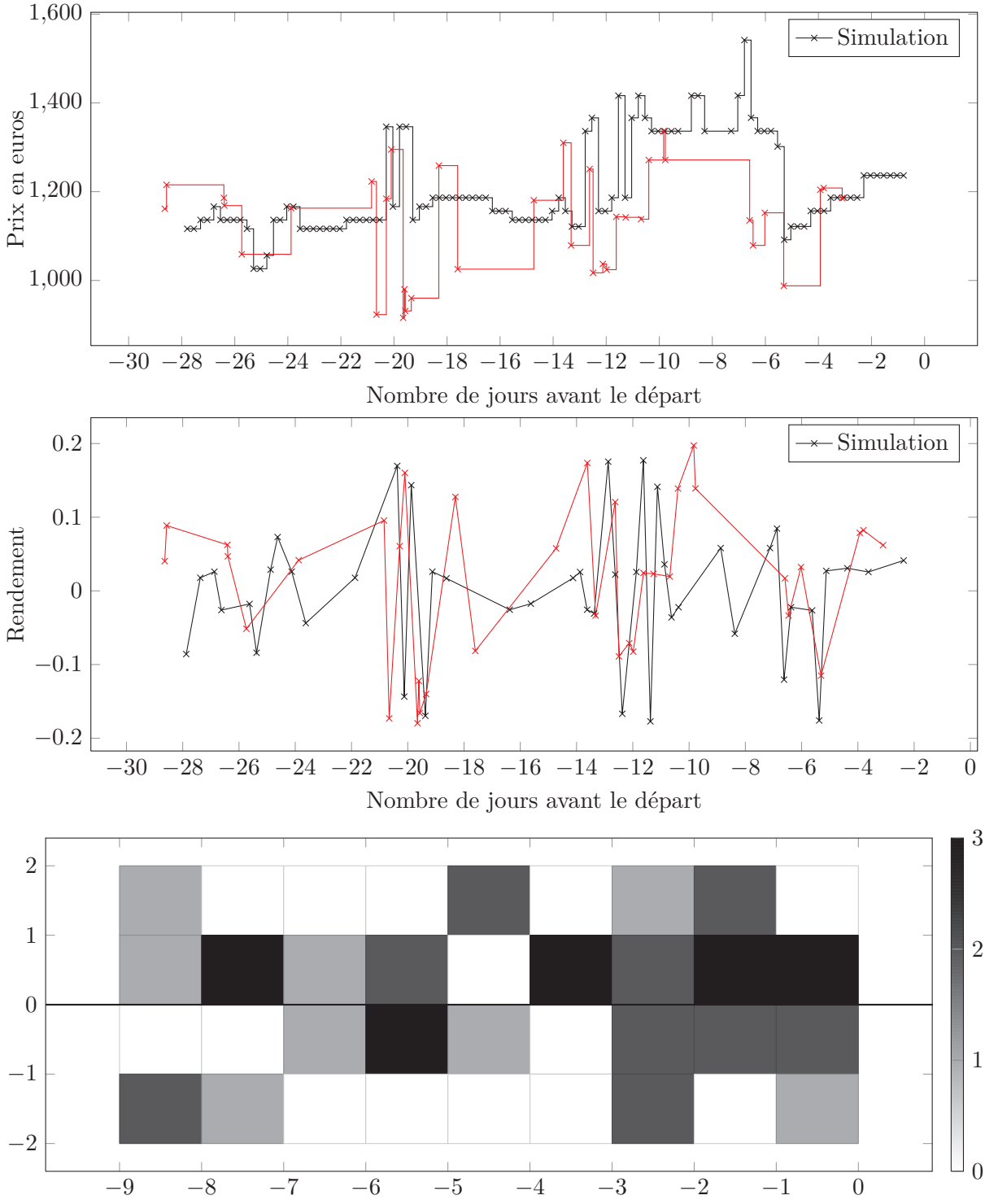


FIGURE 2.16 – Transformations de la série initiale et simulation par le niveau de gris. Paris-Bangkok départ le 16/01/2013 pour 14 jour

2.7 Conclusion et perspectives

Dans ce chapitre, nous avons dans un premier temps décrit la nature de nos séries temporelles. Constantes par morceaux avec un nombre fini de plateaux, elles ne nécessitent pas d'avoir un nombre de points de collecte trop importants pour être bien échantillonnées. Nous avons observé l'évolution du temps moyen d'attente entre deux sauts en fonction du nombre de jour avant la date de départ et nous en avons conclu que dans l'intervalle $[-28, 0]$, la construction d'une série temporelle fidèle à la réalité nécessite la collecte d'un point toutes les 6 heures et que dans l'intervalle $]-90, -28]$, un point de collecte tous les jours permettait la détection de la majorité des variations de prix.

Il est à noter qu'une piste d'amélioration consisterait à prendre en compte les vols complets dans la mesure où ceux-ci peuvent engendrer un lourd préjudice à l'utilisateur s'il ne sont pas détectés.

Nous avons ensuite présenté une nouvelle manière de représenter des séries temporelles de prix en vue d'une agrégation des comportements. Nous avons transformé nos séries de prix en séries de rendements afin de nous extraire des ordres de prix des vols. Les agences de voyages appliquant des changements de prix très fréquents pour une moyenne de prix stable, nous avons décidé de ne pas comptabiliser les variations de moins d'1% lors de cette étape. Nous avons ensuite modélisé ces séries de rendements par des processus ponctuels marqués inhomogènes : nous visualisons cette représentation à l'aide de niveaux de gris dont les pixels représentent l'intensité des sauts dans le découpage de l'espace temps-rendement. Pour éviter de créer des images de trop grande dimension, du fait des quelques séries à forts rendements, nous avons transformé logarithmiquement l'axe des rendements. Le paramétrage des images pixélisées, c'est-à-dire la taille des fenêtres, joue un rôle importante dans la généralisation ou non des comportements. Nous avons enfin montré que cette dernière représentation n'est pas bijective, mais qu'il est possible de reconstituer une série de prix similaire en effectuant une simulations à partir du niveau de gris.

Notre approche consiste à prédire le comportement global d'un vol pour en extraire une prédiction d'évolution de prix à l'instant t. Pour cela, nous devons dans un premier temps identifier les différents comportements existants dans notre base d'apprentissage grâce à notre nouvelle représentation des données. Nous allons donc utiliser des algorithmes d'apprentissage non supervisé afin de segmenter l'espace des niveaux de gris en groupes homogènes et d'en extraire des comportements types que nous associerons plus tard aux vols de test pour établir une prédiction d'évolution à l'instant t.

Chapitre 3

Segmentation des données d'apprentissage

Contents

Introduction

Avec notre nouvelle représentation des trajectoires $X_i(s,t)$, nous nous affranchissons des ordres de prix et nous pouvons donc comparer les comportements de tout type de vols : vols long-courriers et de vols intérieurs par exemple. Nous allons évaluer la similarité de deux séries temporelles en calculant, pour commencer, la distance euclidienne case à case des niveaux de gris. Il est alors possible d'appliquer des algorithmes d'apprentissage non-supervisé sur l'ensemble de notre base de données de vols, afin de regrouper les séries temporelles de même comportement entre elles (clustering). Nous segmentons ainsi nos données d'apprentissage afin de créer des ensembles de comportements similaires auxquels nous associons un comportement représentatif. Ce dernier sera un niveau de gris "moyen", nommé centroïde et noté I_y, $y \in \{1, ..., C\}$, avec lequel nous serons notamment capable de simuler des courbes $p_y(t)$ appartenant au groupe y. Nous verrons plus tard que ces simulations nous serviront dans la prédiction de l'évolution d'un vol de test après lui avoir attribué son comportement moyen le plus vraisemblable.

Dans ce chapitre, nous allons décrire les différents algorithmes que nous avons appliqués afin d'extraire des groupes de comportement similaire basés sur les images pixélisées des intensités. Nous avons dans un premier temps utilisé l'algorithme de partitionnement de données K-Means [28][29] avec comme mesure de distance entre les vols la somme des distances euclidiennes case à case des images pixélisées. Nous étudions ensuite une approche basée sur le mélange de modèles en appliquant l'algorithme Espérance-Maximisation (EM) [38]. Les K-Means tenterons de minimiser la distance des vols d'un groupe au centre de celui-ci, alors que l'EM maximisera la vraisemblance d'appartenance au groupe grâce aux paramètres de ce dernier.

Les deux paramètres principaux des deux approches sont le nombre de groupes et le nombre d'initialisations. Le nombre de groupes est une variable importante dans l'étape de segmentation mais nous verrons plus tard que son influence sur la prédiction finale n'est pas nécessairement corrélée. Il existe des métriques tel que la statistique de GAP [48] qui permettent de choisir le nombre optimal de groupes, et nous comparerons dans le Chapitre 5 le choix théorique de ce nombre avec le choix empirique. Le second paramètre détermine la manière d'initialiser les deux algorithmes, c'est-à-dire d'amorcer la phase d'optimisation récursive. Pour éviter de trouver un minimum local, il est important de multiplier les départs aléatoires et de choisir à l'issue de l'optimisation la meilleure segmentation.

Pour réduire l'influence de l'initialisation dans la convergence des K-Means et pour maximiser les chance d'atteindre l'optimum global, nous utilisons un algorithme qui agrège un ensemble de segmentations appliquées sur des sous-ensembles aléatoires de la base d'apprentissage. Cette technique appelée "bagging", appliquée aux K-Means est nommée "Bagged K-Means".

3.1 WorkFlow

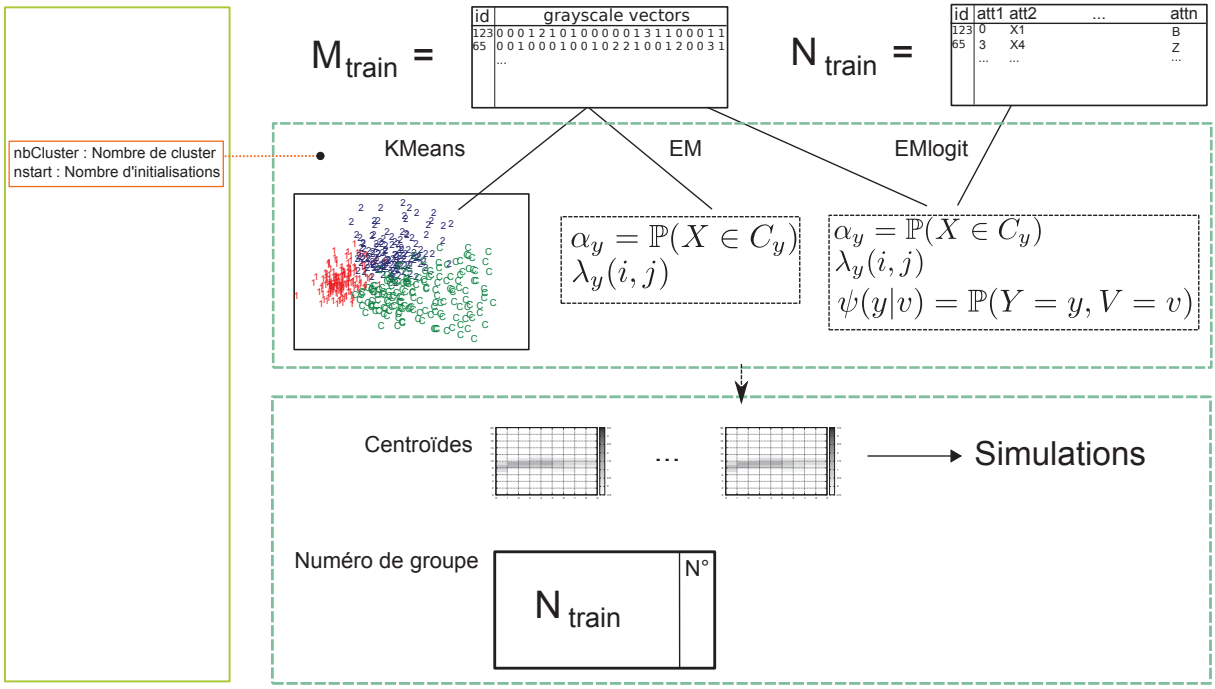


FIGURE 3.1 – Workflow - Étape de segmentation de la base d'apprentissage

3.2 Notations

i	:	Numéro du vol de la base d'apprentissage $i \in 1, ..., n$
p	:	Nombre d'attributs
V_i	:	Vecteur de p attributs du vol i parmi l'ensemble des attributs $\mathcal{V}$
N_{train}	:	Base d'apprentissage des vecteurs d'attributs de taille $n \times p$
n	:	Nombre de vols
$\mathcal{R}$	:	Ensemble des cases du plan temps-rendement
b_r	:	Dimension verticale des cases (rendement)
b_t	:	Dimension horizontale des cases (temps)
$X_i(s,t)$	:	Niveaux de gris du plan temps-rendement pour le vol i à la case (s,t)
M_{train}	:	Base d'apprentissage des niveaux de gris de taille $n \times (s \times t)$
$I_1, ..., I_C$	:	Centroïdes des clusters sous la forme d'image pixelisées : $I_y(s,t), (s,t) \in \mathcal{R}$
C	:	Nombre de clusters
α_y	=	$\mathbb{P}(Y_i = y), \quad y = 1, 2, \ldots, C$.
$\psi(y\|V_i)$	=	$\mathbb{P}(Y_i = y\|V_i), \quad y = 1, 2, \ldots, C$.

3.3 Les algorithmes

Dans cette section nous allons décrire les trois algorithmes utilisés pour partitionner nos données en détaillant les paramètres et leur importance. Deux de ces algorithmes, les K-Means et l'algorithme d'Espérance-Maximisation, font partie de la famille des segmentations à optimisation itérative suivant un schéma itératif en deux étapes : création de modèles et ré-attribution des données (Figure 3.2). L'algorithme s'arrête au bout d'un nombre d'itérations prédéfini, ou lorsque la ré-attribution n'évolue plus. Le dernier, les Bagged K-Means, est l'agrégation de plusieurs K-Means appliqués sur des sous-ensembles aléatoires de la base d'apprentissage.

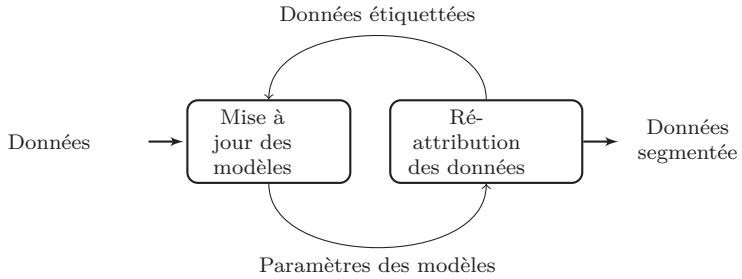


FIGURE 3.2 – Procédure d'optimisation itérative [27]

L'algorithme des K-Means représente chaque classe par un centroïde calculé en moyennant l'ensemble des niveaux de gris du groupe. Ces centroïdes sont notés I_y pour $y \in [1, ..., C]$ avec C le nombre de groupes fixé à l'avance. L'étape de ré-attribution consiste alors à choisir le centroïde le plus proche grâce à une mesure de distance elle aussi fixée à l'avance. L'algorithme d'Espérance-Maximisation va, quant à lui, modéliser chaque classe par une distribution de probabilités créée lors de l'étape de création des classes. La ré-attribution se fait en maximisant la vraisemblance des vols au centroïde.

Nous rappelons que nous possédons un corpus de niveaux de gris $X_i(s,t)$, $i \in [1,n]$ notés M_{train}, inclus dans l'ensemble des cases du plan temps-rendement $\mathcal{R}$. Les cases des centroïdes I_y seront noté $I_y(s,t), (s,t) \in \mathcal{R}$.

3.3.1 K-Means

Nous voulons donc créer des groupes ayant des comportements similaires en appliquant l'algorithme de segmentation K-Means [28] en se basant sur les images pixélisées d'intensité $\widehat{X}_i$ pour i parcourant la base d'apprentissage.

Les K-Means nécessitent une initialisation permettant d'enclencher la phase d'optimisation itérative. C'est une étape importante de l'algorithme qui peut être communément effectuée de deux manières différentes : par la méthode de Forgy et par les partitions aléatoires. La première méthode consiste à choisir aléatoirement C points de la base d'apprentissage comme étant les centroïdes puis d'attribuer aux autres vols le groupe correspondant au centroïde le plus proche. L'approche des partitions aléatoires consiste à attribuer à chaque vol un cluster aléatoirement.

Empiriquement, Hamerly et al. [22] ont montré qu'il est préférable d'utiliser la méthode des partitions aléatoires pour les algorithmes de la famille des "k-harmonic means" et "fuzzy K-Means". Pour les algorithmes d'Espérance-Maximisation et les K-Means standards l'approche de Forgy est conseillée.

Les centroïdes sont ensuite construits en moyennant les vols de chaque groupe et l'algorithme suit alors le processus d'optimisation itératif en ré-attribuant son groupe le plus proche à tous les vols puis en calculant à nouveau les centroïdes.

On obtient ainsi C classes d'indices $I_1, \ldots, I_C$ permettant de regrouper chaque vol i de la base d'apprentissage par comportements similaires d'évolution du prix. Ce nombre C de groupes est fixé arbitrairement et n'évolue pas avec les itérations. Il doit être choisi selon certains critères de performance tels que la densité des groupes ou le taux de bonne prédictions final. Cette étape est donc primordiale dans la construction de notre modèle, un mauvais choix de C pouvant dégrader rapidement l'étape de clustering et donc la prédiction. Il est possible de trouver le nombre optimal de classes en utilisant des critères tels que la statistique de GAP [48] ou l'indice de Calinski-Harabasz [11] que nous décrirons plus tard.

Nous allons maintenant revenir en détail sur le fonctionnement des K-Means appliqué à nos données.

Données d'entrée

L'ensemble de nos niveaux de gris $\widehat{X}_i$ est l'unique point d'entrée de l'algorithme. Afin de minimiser leur taille, nous utilisons les images pixélisées issues de la transformation par le logarithme népérien des séries de rendement. Ces niveaux de gris doivent être de dimensions égales pour pouvoir les comparer. La taille verticale des $X_i(s,t)$ est calculée en utilisant le rendement maximum à la hausse et à la baisse trouvé dans la base d'apprentissage. Les niveaux de gris de plus petites tailles sont alors mis à la même dimension par l'ajout de cases vides. Les paramètres de l'algorithme sont le nombre de groupes voulu C, le nombre d'initialisations aléatoires $nstart$ et le nombre d'itérations maximum $iter.max$. Ce dernier paramètre est un critère d'arrêt en cas de non convergence : l'algorithme des K-Means continuant son optimisation tant que les groupes évoluent, il est nécessaire de pouvoir arrêter l'itération en cas de boucle infinie.

Description détaillée de l'algorithme

Comme décrit précédemment, la première étape consiste à créer un point de départ aléatoire au processus itératif d'optimisation. Nous utilisons l'approche de Forgy qui consiste à choisir aléatoirement C centroïdes parmi les données d'entrée. La segmentation par la méthode des K-Means est une méthode itérative qui, quel que soit son point de départ converge vers une solution. La solution obtenue n'est pas nécessairement la même selon l'initialisation. Pour cette raison, il est important de répéter plusieurs fois ($nstart$) les calculs pour ne retenir que la solution optimale pour le critère choisi.

L'étape d'optimisation itérative commence alors par l'attribution d'un groupe aux autres vols de la base de données. Le centroïde le plus vraisemblable est calculé en minimisant la distance euclidienne, le but des K-Means étant de minimiser la variance à l'intérieur de chaque groupe :

$$E_i^{(t)} = \arg\min_{\mathbf{Y}} \sum_{i=1}^{n} \sum_{X_i \in Y_i^{(t)}} ||X_i - I_y||^2$$

avec $\mathbf{Y} = \{Y_1, ..., Y_C\}$

La distance obtenue entre les trajectoires repose sur la distance L^2 des densités correspondantes (pour b_t et b_r donnés) :

$$d(i_1, i_2) = \left(\int_{s=0}^{1} \int_{t=T_{\min}}^{0} \left[X_{i_1}(s, T_0^{(i_1)} + t) - X_{i_2}(s, T_0^{(i_2)} + t) \right]^2 \, \mathrm{d}s\mathrm{d}t \right)^{1/2},$$

où $T_{\min}$ est la date relative (à T_0) du premier prix disponible.

Nous redéfinissons maintenant les centroïdes à partir des vols qui ont été affectés aux différentes classes.

$$\mathbf{I}_y^{(t+1)} = \frac{1}{|Y_i^{(t)}|} \sum_{X_i \in Y_i^{(t)}} X_i$$

On ré-affecte ensuite les vols en fonction de leur distance aux nouveaux centres. Et ainsi de suite jusqu'à ce que la convergence soit atteinte, ou que nous soyons parvenu au nombre d'itérations maximum $iter.max$.

Données à la sortie

Nous obtenons à la fin de l'algorithme des K-Means une segmentation de l'espace des niveaux de gris, c'est-à-dire, le regroupement des vols en groupes de comportements similaires. Nous attribuons donc à chaque vol de la base d'entraînement le numéro du groupe dont il est le plus proche $E_i \in 1, ..., C$ qu'on appellera "étiquette". Pour chaque groupe, nous créons une représentation du comportement "type" en moyennant case par case les niveaux de gris de l'ensemble des vols du groupe. Ces comportements "type" sont appelés centroïdes et décrivent le comportement moyen de chacun des groupes créés.

Il nous est alors possible d'évaluer la qualité de la segmentation de nos données grâce à plusieurs métriques sachant que la qualité d'un algorithme de clustering se définit par une grande similarité intra-groupe et une faible similarité inter-groupe.

1. **La distribution des vols** sur l'ensemble des groupes est une première information qui valide l'homogénéité des groupes. Une répartition déséquilibrée indique un mauvais choix de nombre de clusters ou une mauvaise représentativité des comportements dans la base d'apprentissage.
$$n = \sum_{i=1}^{C} n_i$$

2. **La somme des distances aux centroides** pour chaque groupe, w_y, quantifie la distance globale entre les vols du groupe et le centroïde associé. Cette information, couplée à la visualisation du centroïde, permet d'identifier les centroïdes denses où l'ensemble de vols du groupe sont de même nature, et les centroïdes plus dispersés qui regroupent des

vols de natures plus variées. Nous verrons que cette métrique est liée au choix du nombre de clusters. La réduction du nombre de clusters créera des groupes moins denses et donc moins représentatifs et la multiplication des clusters aura un effet de dispersion des comportements empêchant la création de "familles" de vols cohérentes. Cette quantité s'écrit donc :

$$w_y = \sum_{X_i \in y} \|X_i - I_y\|^2$$

$$W = \sum_{i=y}^{C} \sum_{X_i \in y} \|X_i - I_y\|^2$$

C'est avec cette métrique que l'initialisation la plus performante est sélectionnée.

3. **La distance entre les centroïdes** nous informe sur la distance entre les différents centre des groupes. Elle est définie par :

$$b_y = n_y \|I_y - \mathbf{I}\|^2$$

$$B = \sum_{y=1}^{C} n_y \|I_y - \mathbf{I}\|^2$$

où $\mathbf{I}$ est le comportement moyen de toute la base de données.

L'algorithme 1 décrit formellement chacune de ces étapes.

3.3.2 Bagged K-Means

Les "Bagged K-Means" applique le principe du bagging (bootstrap aggregating) à l'algorithme des K-Means. Le bagging consiste à appliquer un certain nombre de fois le même algorithme sur un sous-ensemble de la base d'apprentissage toujours différent (*bootstrap*) et d'agréger ensuite les résultats (*aggregating*). Pour améliorer les résultats des K-Means nous tentons donc d'appliquer cette méthode en agrégeant nos segmentations à l'aide d'un algorithme de segmentation hiérarchique ascendante. Ce dernier s'applique non pas à un jeu de données, mais à une matrice de distance entre les centroïdes issues de l'étape de bootstraping.

Algorithmique

Nous décrivons l'algorithme des Bagged K-Means proposé par [32] :

1. Construire $\mathcal{B}$ sous-ensemble de la base d'apprentissage M_{train}, $M^1_{train}, ..., M^{\mathcal{B}}_{train}$ par tirage aléatoire avec remise (bootstrap)
2. Appliquer un algorithme de segmentation comme les K-Means sur chaque sous-ensemble pour obtenir $\mathcal{B} \times C$ groupes $c_{11}, c_{12}, ..., c_{1C}, c_{21}, ..., c_{\mathcal{B}C}$ où C est le nombre de clusters voulu et c_{ij} le j-ème groupe du sous-ensemble M^i_{train}
3. Créer une nouvelle base $\mathcal{Y}^{\mathcal{B}}(C)$ constituée de l'ensemble des centres $\mathcal{Y}^{\mathcal{B}} = \mathcal{Y}^{\mathcal{B}}(C) = c_{11}, ..., c_{\mathcal{B}C}$

Algorithm 1 Partitionnement de l'espace par l'algorithme des K-Means

Pour un nombre d'initialisations voulu *nstart* **faire**
Initialisation de Forgy : Sélection aléatoire de C centroïdes parmi les n niveaux de gris
Tant que l'attribution des vols évolue **OU** l'itération max non atteinte *iter.max* **faire**
Étape d'attribution : On attribue un groupe à chaque vol en fonction de sa proximité au centroïde du groupe à l'itération t.

$$Y_i^{(t)} = \{E_i^{(t)} : \|X_i - I_y^{(t)}\| \leq \|X_i - I_y^{(t)}\| \ \forall \ 1 \leq y \leq C\},$$

On attribut au vol X_i l'étiquette $E_i^{(t)}$ correspondant au groupe le plus probable $Y_i^{(t)}$ basé sur la distance euclidienne.
Etape de mise à jour : Calcule les nouveaux centroïdes

$$\mathbf{I}_y^{(t+1)} = \frac{1}{|Y_i^{(t)}|} \sum_{X_i \in Y_i^{(t)}} X_i$$

Fin Tant que
Fin Pour
return E_i Étiquette associée à chaque vol, ie. identifiant du centroïde le plus proche
return n_y Nombre de vols par groupe
return w_y Somme des distances au centroïde par groupe
return I_y Centroides des groupes

4. (Optionnel) Elaguer $\mathcal{Y}^{\mathcal{B}}$ en appliquant l'algorithme de partitionnement à M_{train} avec les centres $\mathcal{Y}^{\mathcal{B}}$ et en supprimant les centres vides (ou ne dépassant pas un certain seuil θ) :

$$\mathcal{Y}^{\mathcal{B}}_{prune}(C, \theta) = \{c \in \mathcal{Y}^{\mathcal{B}}(C) | \#\{x : c = c(x)\} \geq \theta\}$$

5. Appliquer l'algorithme de clustering hiérarchique décrit plus bas sur $\mathcal{Y}^{\mathcal{B}}$ (ou $\mathcal{Y}^{\mathcal{B}}_{prune}$)
6. Soit $c(x) \in \mathcal{Y}^{\mathcal{B}}$ le centre le plus proche de x. Une segmentation de la base originale peut être obtenue en "coupant" le dendrograme à un certain niveaux. On créé ainsi une partition $\mathcal{Y}^{\mathcal{B}}_1, ..., \mathcal{Y}^{\mathcal{B}}_m$, $1 \leq m \leq \mathcal{B}C$ de $\mathcal{Y}^{\mathcal{B}}$ où chaque point $x \in M_{train}$ est associé à son cluster le plus proche $c(x)$.

Segmentation hiérarchique ascendante

La classification ascendante hiérarchique est dite ascendante car elle part d'une situation où tous les individus représentent une classe (il y a donc autant de classe que d'individus), puis sont rassemblés en classes de plus en plus grandes. Le qualificatif "hiérarchique" vient du fait qu'elle produit une hiérarchie H qui vérifie les propriétés suivantes :

1. $\Omega \in H$: au sommet de la hiérarchie, tous les individus sont regroupés au sein d'une même

classe.

2. $\forall \omega \in \Omega, \{\omega\} \in H$: en bas de la hiérarchie, tous les individus se trouvent seuls
3. $\forall (h, h') \in H^2, h \cap h' = \emptyset$ ou $h \subset h'$ ou $h' \subset h$

Nous décrivons dans l'Algorithme 2 les différentes étapes de la segmentation des centroïdes créés lors de l'étape de bootstrapping.

Algorithm 2 Algorithme de segmentation hiérarchique ascendante

Pré-conditions : M_{train} : liste d'individus X_i, $i \in 1, \ldots, n$
Pré-conditions : C : nombre de classes qu'on veut obtenir
Pré-conditions : Cur_y avec $y \in [1, ..., nbC]$ avec nbC le nombre de classes courant.
Pour $i \in [1, ..., n]$ **faire**
$Cur_i = X_i$ devient une nouvelle classe
$nbC++$
Fin Pour
Tant que $nbC > C$ **faire**
Calcul des dissimilarités entre classes dans une matrice triangulaire supérieure
Pour $i \in [1, ..., nbC]$ **faire**
Pour $j \in [i+1, ..., nbC]$ **faire**
$matDissim[i][j] = dissim(Cur_i, Cur_j)$;
Fin Pour
Fin Pour
Recherche du minimum des dissimilarités
Soit (i, j) tel que $matDissim[i][j] = min(matDissim[k][l])$ avec $1 \leq k \leq nbC$ et $k+1 \leq l \leq nbC$;
Fusion de Cur_i et Cur_j
Pour tout element dans $classes[j]$ **faire**
$Cur_i.ajouter(element)$;
Fin Pour
supprimer(Cur_j) ;
Fin Tant que
return classes : liste de classes initialement vide, une classe est vue comme une liste d'individus

L'idée principale est de stabiliser la segmentation des K-Means en répétant le partitionnement et en combinant les résultats. Les K-Means sont instables car chaque application de l'algorithme donnera un optimal local différent pour les mêmes données et les mêmes paramètres. L'étape d'initialisation a une grande influence sur la convergence de l'algorithme surtout si le nombre de clusters idéal n'est pas connu et une legère modification des données d'apprentissage peut aussi faire converger les K-Means vers un tout autre optimum.

En répétant l'apprentissage sur des sous-ensembles, nous obtenons différentes solutions qui, une fois agrégées, devraient être indépendantes de la base d'apprentissage initiale et du nombre d'initialisations.

3.3.3 EM

L'algorithme d'Espérance-Maximisation (EM) est une classe d'algorithmes qui a pour but de maximiser la vraisemblance des paramètres de modèles probabilistes comportant des variables latentes (c'est-à-dire non-observées). Nous appliquons le principe de l'EM à l'estimation des paramètres d'un mélange de densités par classification automatique. Dans ce problème, nous considérons que nos vols $i \in \{1, ..., n\}$, caractérisés par leur niveau de gris $X_i(s,t)$, sont issus de C différents groupes. On supposera que si un vol i appartient au groupe $y \in \{1, ..., C\}$, on écrira $Y_i = g$, alors les $X_i(s,t), (s,t) \in \mathcal{R}$, sont des réalisations d'un processus ponctuel de Poisson d'intensité donnée par les centroïdes $I_y(s,t), (s,t) \in \mathcal{R}$. Par ailleurs les proportions des groupes sont données par un vecteur $(\alpha_1, \ldots, \alpha_C)$ où $\alpha_y = \mathbb{P}(Y_i = y) \quad \forall y \in \{1, ..., C\}$. On notera $\alpha = (\alpha_1, ..., \alpha_C)$ l'élément du simplexe $S_C = \{(\alpha_1, ..., \alpha_C) \in [0,1]^C, \sum_{y=1}^{C} \alpha_y = 1\}$. Le problème est posé de façon à maximiser la vraisemblance du modèle qui supposera bien que les données sont des observations d'un mélange de densités. Les paramètres du modèle sont regroupés sous la notation $\theta = ((\alpha_y)_{y \in \{1,...,C\}}, (I_y(s,t))_{(s,t) \in \mathcal{R}}) \in [0,1]^C \times (\mathbb{R}^+)^{\mathcal{R}}$

Les α_k représentent la probabilité d'appartenance à un groupe ($\alpha_y = \mathbb{P}(X \in C_y)$) et I_y représente le paramètre de la loi des observations, sachant qu'elles appartiennent au groupe d'étiquette y.

Données d'entrée

Les données d'entrée sont les mêmes que pour les K-Means c'est-à-dire l'ensemble de nos niveaux de gris $X_i(s,t), i \in \{1, ..., n\}, (s,t) \in \mathcal{R}$.

Description détaillée de l'algorithme

Initialisation Comme pour les K-Means, l'initialisation est une phase importante de l'algorithme. Il s'agit içi d'attribuer un valeur initiale aux $I_k(s,t)$ et aux α_k. Nous avons tenté deux approches : la première consiste à attribuer des valeurs aléatoirement à chacun des paramètres et la deuxième initialise les valeurs en appliquant l'algorithme des K-Means. La création aléatoire des $I_k(s,t)$ se fait par l'attribution aléatoire d'un numéro de groupe à chaque X_i, puis en moyennant l'ensemble des vols de chaque groupe exactement comme pour K-Means. En ce qui concerne les α_k, dans les deux méthodes d'initialisation, nous procédons de la manière suivante :

$$\alpha_y = \frac{card(X_i \in y)}{n}$$

Nous verrons dans le Chapitre 5 que l'initialisation aléatoire donne une meilleure vraisemblance qu'un départ par les K-Means, mais que les résultats de prédiction n'en sont pas nécessairement améliorés en pratique. Nous privilégierons donc une initialisation par les K-Means pour des raisons pratiques.

Fonction de vraisemblance La loi du tableau $X_i = (X_i(s,t))_{(s,t)\in\mathcal{R}}$ sachant que l'étiquette de son groupe associé $Y_i = y$ est :

$$\mathbb{P}_{I_y}(X_i(s,t) = x(s,t), (s,t) \in \mathcal{R} | Y_i = y) = \prod_{(s,t)\in\mathcal{R}} \frac{(I_y(s,t))^{x(s,t)} e^{-I_y(s,t)}}{x(s,t)!},$$

où $x(s,t) \in \mathbb{N}$, $y \in \{1, ..., C\}$ et $I_y(s,t) \in ((\mathbb{R})^{\mathcal{R}})^C$.

Ici C représente toujours le nombre de groupes et pour une étiquette $y \in \{1, ..., C\}$ et une case $(s,t) \in \mathcal{R}$ données, $I_y(s,t)$ représente la valeur du centroïde du groupe y à la case (s,t).

Cette loi provient du modèle poissonnien : I_y est une version pixélisée de l'intensité du processus de Poisson pour les vols du groupe y et par suite les $X_i(s,t)$ sont les variables de Poisson indépendantes d'intensité $I_y(s,t)$.

On calcule alors la vraisemblance du couple (X_i, Y_i), pour le paramètres $\theta = (\alpha, I) \in S_C \times (\mathbb{R}_+^{\mathcal{R}})^C$. On obtient :

$$\mathbb{P}_\theta(X_i = x, Y_i = y) = \mathbb{P}_{I_y}(X_i = x, Y_i = y)\mathbb{P}_\alpha(Y_i = y) = \alpha_y \prod_{(s,t)\in\mathcal{R}} \frac{I_y^{x(s,t)}(s,t)e^{-I_y(s,t)}}{x(s,t)} =: p_\theta(x,y),$$

où $x \in \mathbb{N}^{\mathcal{R}}$ et $y \in 1, ..., p$.

Étape Espérance L'étape "E" consiste à calculer l'espérance de la densité jointe au paramètre θ' pour la loi conditionnelle des variables latentes Y_i sachant les variables observées X_i sous le paramètre θ :

$$Q(\theta', (x_i)_{i\in\{1,...,n\}}|\theta) = \sum_{i=1}^{n} \sum_{y_i=1}^{C} [\log p_{\theta'}(x_i, y_i)] \; p_\theta(y_i|x_i) \; .$$

On a alors pour $\theta' = (\alpha', I'_y)$:

$$\log p_{\theta'}(x,y) = \sum_{i=1}^{n} \log(\alpha'_{y_i}) + \sum_{i=1}^{n} \sum_{(s,t)\in\mathcal{R}} [x(s,t) \log (I'_y(s,t)) - I'_y(s,t)]$$

La probabilité conditionnelle s'écrit :

$$p_\theta(y|x) = \mathbb{P}_\theta(Y_i = y|X_i = x) = \frac{\mathbb{P}_\theta(X_i = x, Y_i = y)}{\sum_{k=1}^{C} P_\theta(X_i = x, Y_i = k)} = \frac{p_\theta(x,y)}{\sum_{k=1}^{C} p_\theta(x,k)} \tag{3.1}$$

On a donc en notant $X = (X_i)_{i=1,...,n}$:

$$Q_n(\theta'|\theta) := Q(\theta', X|\theta) = \sum_{l=1}^{n} \sum_{y=1}^{p} \log (\alpha'_y) p_\theta(y|X_l)$$

$$+ \sum_{l=1}^{n} \sum_{(s,t)\in\mathcal{R}} [X_l(s,t) \sum_{y=1}^{C} \log I'_y(s,t) p_\theta(y|X_l) - \sum_{y=1}^{C} I'_y(s,t) p_\theta(y|X_p)]$$

Pour simplification posons $\forall y \in 1...p$:

$$A_n(\theta, y) = \frac{1}{n}\sum_{i=1}^{n} p_\theta(y|X_l) \tag{3.2}$$

$$B_n(\theta, y, s, t) = \frac{1}{n}\sum_{i=1}^{n} X_l(s,t) p_\theta(y|X_l) \tag{3.3}$$

Notons que B_n dépend de s et de t, et donc possède une valeur pour chaque case alors que A_n est uniquement indexé par $y \in \{1, ..., C\}$.

On obtient alors :

$$Q_n(\theta'|\theta) = \sum_{y=1}^{C} \log(\alpha'_y) A_n(\theta, y) + \sum_{(s,t)\in\mathcal{C}} \sum_{y=1}^{C} \log(I'_y(s,t)) B_n(\theta, y)$$

$$- \sum_{(s,t)\in\mathcal{R}} \sum_{y=1}^{C} I'_y(s,t) A_n(\theta, y)$$

Étape Maximisation L'étape "M" consiste à maximiser $Q_n(\theta'|\theta)$ en θ' pour un θ donné. L'optimisation des α_k et des I_k peut se faire séparément.

Pour θ, A_n, B_n fixés, on obtient facilement :

$$\alpha'_y = \frac{A_n(\theta, y)}{\sum_{k=1}^{C} A_n(\theta, k)}, \forall y \in 1...C$$

et

$$I'_y(s,t) = \frac{B_n(\theta, y, s, t)}{A_n(\theta, y)}, \forall y \in 1...C, \forall (s,t) \in \mathcal{R}$$

Données à la sortie

L'EM est un algorithme de calcul numérique des paramètres $(\alpha, I) = \theta$ qui maximisent au moins localement la vraisemblance. A l'arrêt de l'algorithme, on dispose d'un estimateur $\hat{\theta} = (\hat{\alpha}, \hat{I})$. On peut d'autre part facilement obtenir un calcul de la densité jointe $p_\theta(x, y)$ duquel on déduit

1. la log-vraisemblance $L_n(\theta) = \log \sum_{y=1}^{C} p_\theta(X, y)$
2. la log-vraisemblance au cluster $L_n(\theta, y) = \log(p_\theta(X|y)) = \log(p_\theta(X, y)) + Cte$, où la constante additive ne dépend pas de y.

Contrairement aux K-Means, l'EM ne retourne pas une étiquette estimée $\hat{Y}_i$ pour chacun des vols d'apprentissage. Nous calculons ce groupe grâce à la fonction de vraisemblance $p_\theta(x, y)$ et aux paramètres précédemment optimisés. Elle est appliquée aux $X_i(s,t)(\alpha_y$ et $I_y(s,t))$ afin de déterminer pour chaque vol i l'étiquette $\hat{Y}_i$. Les mêmes métriques peuvent être utilisées pour

évaluer la segmentation des données et choisir les bons paramètres, mais la pertinence des w_y n'est pas la même que pour les K-Means : l'EM ne tente pas de minimiser la distance aux centroïdes, mais de maximiser la vraisemblance d'appartenance aux clusters. Il sera alors utile d'observer la vraisemblance finale selon les paramètres.

3.4 Choix des paramètres

Dans cette section nous allons expliquer les différents paramètres qui peuvent influencer les résultats de la segmentation et par conséquence la prédiction de l'évolution de vols de test. Nous verrons seulement dans le Chapitre 5 les graphiques correspondants aux modifications de ces paramètres.

3.4.1 Initialisation

Tout d'abord, il est important de souligner que "la meilleure initialisation" est une notion mal définie, car il n'y a pas de délimitation formelle entre la recherche initiale et la recherche. Par exemple, lorsque l'on considère l'algorithme EM, le point de départ idéal se situerait relativement proche de l'optimum global. Cependant pour trouver un tel point, il faudrait trouver l'optimum global avec une certaine exactitude, ce qui se révèle être un problème en soi peut être plus difficile même que la résolution de l'EM [34].

1. **Le nombre de départs** comme décrit plus haut multiplie les initialisations aléatoires des paramètres internes aux algorithmes de clustering afin d'éviter à ceux-ci de converger vers un optimum local. Il est nécessaire de choisir un nombre relativement important car les algorithmes décrits sont très sensibles à leur initialisation et convergent différemment à chaque itération. En multipliant le nombre de départs, on augmente les chances d'obtenir l'optimal global.
2. **Le type d'initialisation** participe aussi à la qualité de la segmentation. Dans [34], plusieurs algorithmes de clustering sont comparés sur un jeu de données de grande dimension, puis, observant les très bonnes performances de l'EM, différents types d'initialisations sont alors testés sur ce dernier algorithme : (i) une initialisation totalement aléatoire des paramètres, (ii) des perturbations aléatoires de la loi de probabilité des données et (iii) le résultat d'une segmentation préalable par un autre algorithme de clustering. L'article suggère d'utiliser la deuxième méthode, tout en affirmant qu'elle ne présente pas d'énorme différence avec la dernière. Dans notre cas, nous initialiserons l'EM par une première segmentation des K-Means et nous la comparerons à l'initialisation aléatoire dans le chapitre 5.

3.4.2 Nombre de groupes

Le choix du nombre de groupes est le point le plus sensible de nos algorithmes car il n'y a pas vraiment de moyen de connaître exactement sa valeur optimale. Différentes techniques existent afin d'estimer ce nombre comme l'évolution de la distance aux centroïdes ou la visualisation par une projection sur un plan des nuages de points.

Milligan and Cooper [36] ont étudié 30 méthodes d'estimation du nombre de groupes optimal sur des données simulées décrivant des groupes bien distincts. D'après leurs recherches, étudier l'évolution de l'indice de Calinski and Harabasz [11] est la méthode la plus efficace. Dans sa thèse, Yan [53] étudie ces différentes méthodes et propose de les comparer à l'utilisation de l'indice de GAP introduite par Tibshirani et al. [48]. Ses résultats tendent à montrer que l'indice de GAP est un meilleur estimateur du nombre optimal de groupes. Nous allons décrire ici les différentes méthodes que nous avons testées.

1. **Evolution de l'indice de RAND**

   L'indice de Rand (d'après William M. Rand [42]) est une métrique couramment utilisée dans le regroupement des données. C'est une mesure de similarité entre partitions d'un ensemble. Le principe consiste à vérifier pour chaque paire d'objets, si dans les deux partitionnement, ils ont été classé dans le même groupe.

   Soit deux partitionnements de l'espace des niveaux de gris P_1 et P_2. On note

   (a) a le nombre de paires présent dans le même groupe dans P_1 et dans P_2
   (b) b le nombre de paires dans le différents groupes dans P_1 et dans P_2
   (c) c le nombre de paires présent dans le même groupe dans P_1 et dans différents groupes dans P_2
   (d) d le nombre de paires dans différents groupes dans P_1 et dans le même groupe dans P_2

   L'indice de Rand s'écrit alors :

   $$R = \frac{a+b}{a+b+c+d}$$

   En faisant évoluer le nombre de clusters ou le nombre d'initialisations il est possible d'observer l'évolution de l'indice de Rand. En théorie, celui-ci va diminuer linéairement puis ralentir sa diminution vers un plateau à faible pente. C'est à la jonction entre ces deux phases, le coude, que le meilleur compromis se trouvera.

2. **L'indice de Calinski-Harabasz** [11], lorsqu'il est maximisé, permet de déterminer le nombre de clusters optimal. Celui-ci ce calcule ainsi :

   $$CH^{(y)} = \frac{b_y/(C-1)}{w_y/(n-C)}$$

   où b_y et w_y sont les métriques décrites dans la section 3.3.1. Comme pour l'indice de RAND, il s'agira de détecter le "coude" dans la courbe d'évolution de l'indice en fonction du nombre de groupes et d'en déduire le nombre idéal de groupes.

3. **La statistique GAP**

   Tibshirani et al. [48] propose de déterminer le nombre de clusters optimal en se basant sur la statistique GAP. Cette méthode peut s'appliquer à n'importe quel algorithme de segmentation utilisant n'importe quelle mesure de distance. L'idée est de comparer l'évolution de $W^{(k)}$ (décrit dans 3.3.1) à mesure qu'on augmente le nombre de clusters k par

rapport à celui attendu sur des données générées sous une distribution de référence décrivant l'hypothèse nulle. La meilleure valeur de k, $\hat{K}$ est alors le moment où $\log(W^{(k)})$ s'éloigne le plus de la courbe attendue.

4. **Visualisation**

   Il est possible de visualiser le partitionnement des données en effectuant une projection sur l'axe de deux variables issues de l'analyse en composante principale (ACP) : les deux nouvelles variables seront une combinaison linéaire des $I_y(s,t)$. L'idée fondamentale de l'ACP est la suivante : considérant le nuage de n points en P dimensions (dans cet espace, 1 point = 1 vol et $P = i \times j$), on cherche à trouver le plan (donc, une représentation bi-dimensionnelle que l'on va pouvoir voir et comprendre) dans lequel la projection des points du nuage est la moins déformée possible. Pour cela, on commence par rechercher la droite δ_1 qui minimise la somme des distances au carré entre chaque point et la droite. Cette droite a la propriété d'être la direction selon laquelle la variance (l'étalement) des points est la plus grande. Puis, on cherche une deuxième droite δ_2 perpendiculaire à δ_1 qui possède la même propriété ; et on continue jusqu'à obtenir P droites qui forment un repère orthogonal.

   Avec cette nouvelle représentation, nous pourrons observer l'évolution des nuages de points en fonction du nombre de clusters et avoir une idée du nombre de groupes à partir duquel la séparation n'est plus pertinente. De la même manière que les autres métriques, une bonne séparation n'implique pas une prédiction finale de meilleure qualité, mais assure la présence de comportements discriminants.

3.4.3 Dimensions des niveaux de gris

Dans le Chapitre précédent, nous avons décrit les étapes de transformation des séries de prix $p_i(t)$ en séries de rendements $s_i(t)$ puis en images pixélisées $X_i(s,t)$. Cette dernière étape consiste à quadriller l'espace temps-rendement pour créer un niveaux de gris représentant le comportement global de la série $p_i(t)$ initiale. Le paramètre b_t représente la division temporelle de l'espace et regroupe les sauts par fenêtre de temps. Le paramètre b_r divise l'axe des rendements pour assimiler les variations de même ordre de prix. Le choix de ces paramètres va influencer directement la segmentation des données : choisir de diviser l'axe des ordonnées de façon binaire (une ligne pour les hausses et une autre pour les baisses) peut s'avérer pertinent pour une prédiction binaire mais beaucoup moins efficace si l'on souhaite avoir plus de précisions sur l'évolution des prix.

3.5 Conclusion et perspectives

Nous venons de décrire l'étape de partitionnement de l'ensemble de nos vols en groupes de comportements similaires en se basant sur notre nouvelle représentation des séries temporelles. Nous avons utilisé deux approches très connues dans le domaine du clustering qui sont les K-Means et l'algorithme Espérance-Maximisation.

L'algorithme des K-Means représente chaque classe par un centroïde construit en moyennant tous les vols appartenant à celle-ci. A l'initialisation ces centroïdes sont choisis aléatoirement

parmi l'ensemble des niveaux de gris de la base d'apprentissage et la ré-attribution des vols se fait alors en minimisant la distance euclidienne aux centroïdes. On itère ensuite ces deux étapes jusqu'à ce que les groupes ne bougent plus, ou que le nombre limite d'itérations fixé à l'avance soit atteint. La mesure de distance entre les images pixélisées est la somme case à case de la différence euclidienne. Il serait intéressant d'utiliser une métrique prenant compte de la temporalité des niveaux de gris et qui pénaliserait plus une différence temporelle qu'une différence de rendement.

Une méthode consiste à appliquer les K-Means sur un grand nombre de sous-ensembles boostrapés de la base d'apprentissage et d'agréger ensuite les résultats grâce à un algorithme de segmentation hiérarchique. Cette approche nommé "Bagged K-Means", augmente les chances d'obtenir le maximum global en effaçant l'influence de l'initialisation et stabilise les résultats face aux changements de paramètres et de base d'apprentissage.

L'algorithme d'Espérance-Maximisation modélise chaque classe par une distribution de probabilité et groupe les niveaux de gris afin de maximiser la vraisemblance aux centroïdes. L'initialisation des paramètres peut se faire aléatoirement ou par des K-Means augmentant ainsi les chances de finir sur l'optimal global.

Pour les K-Means et l'EM, l'initialisation est une étape très importante qu'il faut répéter un grand nombre de fois pour ne pas converger dans un minimum local. De même le choix du nombre de clusters, défini avant le lancement des algorithmes est primordial et peut nécessiter d'observer des critères tels que la statistique de Gap ou l'indice de Rand afin de choisir le nombre optimal.

A mesure que notre base de données augmente, de nouveaux comportements se créent et nécessitent de mettre à jour notre segmentation des données. Afin d'éviter la réiteration d'un algorithme couteux en ressources, il serait judicieux de faire appel à des algorithmes d'"online clustering" [3][55] ou [1] , c'est-à-dire de segmentation évoluant au fur et à mesure de l'ajout de nouveaux vols. Nous assurerions par la même occasion une stabilité dans le regroupement des comportements et donc une reproductibilité des prédictions.

Après avoir créé des familles de comportement à partir des niveaux de gris de notre base d'apprentissage, nous voulons maintenant attribuer à chaque billet de la page de résultats sa famille la plus vraisemblable au vue de ses attributs dans un premier temps, puis grâce à ses premières évolutions de prix. Nous serons alors en mesure d'évaluer le comportement futur du vol grâce aux simulations faites à partir du centroïde associé à la famille.

Pour cela nous allons appliquer un algorithme d'apprentissage supervisé sur nos données segmentées afin d'en extraire des règles de classification. Par la suite, si les informations sur les premières évolutions des vols de test sont fournies, nous pourrons les utiliser pour affiner notre classification.

Chapitre 4

Apprentissage supervisé - Classification - Prédiction

Contents

Introduction

Chaque recherche utilisateur retourne une liste de résultats correspondants aux paramètres d'entrée : jour de départ, jour de retour, ville de départ et ville d'arrivée. Chaque résultat possède une liste d'attributs propre à son vol et au site proposant le vol : horaires de départ et de retour, compagnie aérienne, aéroport de départ et d'arrivée, site marchand etc. L'information que nous souhaitons fournir aux usagers de liligo.com est une prédiction d'évolution de prix par ligne de résultats, c'est-à-dire pour un trajet vendu par un site marchand. Les seules informations que nous possédons sur les vols sont celles que nous venons de décrire, c'est pourquoi nous voulons pouvoir associer à ces vols une prédiction uniquement grâce à leurs attributs.

La base d'apprentissage des niveaux de gris est à présent segmentée en groupes de vols similaires représentés par un comportement type (moyenne des niveaux de gris du groupe) ou par un ensemble de paramètres (α_k, I_k) suivant l'algorithme de segmentation utilisé. Nous voulons maintenant attribuer aux résultats de la recherche le comportement le plus probable parmi ces groupes en se basant sur la répartition des attributs des vols d'apprentissage.

Nous appliquons donc un algorithme d'apprentissage supervisé sur les données segmentées : chaque vol possède une série d'attributs V_i et une étiquette E_i correspondant au groupe dans lequel il a été placé. Nous allons alors créer des règles de classification uniquement basées sur l'ensemble des attributs $V_i(1), \ldots, V_i(p)$ afin d'affecter aux vols de test leur groupe $k \in \{1, \ldots, C\}$ le plus vraisemblable. Pour cela nous avons utilisé plusieurs algorithmes usuels qui sont les arbres de classification CART [5] et C4.5 [41], Adaboost [19] qui utilise CART comme classifieur faible dans son étape de boosting et les Forêts d'arbres décisionnels [9] (Random Forest) qui utilisent aussi CART comme classifieur faible dans son étape bagging. Avec CART et C4.5, nous pourrons observer la segmentation de l'espace des attributs et les règles de prédiction de comportement. Adaboost va exécuter itérativement CART sur la base d'apprentissage pondérée différemment à chaque itération. La pondération donne plus de poids aux vols mal classés afin d'améliorer la prédiction des vols dit "difficiles" à prédire correctement. Les forêts aléatoires, dont le principe est de multiplier les arbres de décision CART et d'agréger leurs résultats par vote, améliorent les résultats tout en fournissant un classement des attributs les plus influents dans la classification. Nous utiliserons ensuite ce classement pour ne conserver que les attributs les plus pertinents, technique nommée *feature selection*. Nous décrivons aussi une extension de l'algorithme Esperance-Maximisation qui prend en compte les attributs et combine l'étape de segmentation à l'étape de classification.

Nous attribuons donc à chaque résultat de la recherche utilisateur son identifiant de cluster le plus probable auquel est associé un centroïde représentant son comportement global. Dans le Chapitre 2, nous avons montré qu'il était possible de simuler une courbe de prix à partir d'une image pixélisé pour reconstituer un série de prix similaire à l'originale. Il est donc possible de simuler un ensemble de séries appartenant potentiellement à un groupe en utilisant le centroïde du cluster en question. En moyennant un nombre prédéfini de courbes simulées, nous sommes capable d'estimer l'évolution d'un vol assigné à ce groupe. Par exemple, si nous sommes à 20 jours de la date de départ et que le premier résultat de la recherche est associé au groupe numéro 2, nous pouvons simuler des courbes issues du deuxième cluster, se placer à -20 jours pour chacune d'elles et observer l'évolution de celles-ci. Il est possible d'extraire de ces simulations

autant d'informations qu'une série de rendements peut fournir : la hausse ou la baisse du prix à n jours, le rendement de cette variation mais aussi l'existence d'un baisse pendant plus de 24h dans un intervalle de m jours ou la probabilité d'une forte hausse ou d'une faible baisse. Cette flexibilité est très importante car elle se répercute directement sur la flexibilité du service final.

Dans un second temps, nous explicitons ce que nous avons nommé la prédiction "directe" : une prédiction basée sur l'apprentissage direct de l'évolution de prix à l'instant t et non plus d'un comportement global. Cette prédiction peut-être binaire (hausse ou baisse du prix dans 7 jours), par cadran (forte hausse, faible baisse etc.) ou continue (pourcentage d'évolution). L'étiquette E_i que nous tentons d'apprendre n'est donc plus le groupe le plus probable attribué aux vols mais la prédiction à l'instant t, impliquant la création d'un modèle par nombre de jours avant la date de départ. Nous nous passons ainsi de la création de comportements types et de la simulation de courbes. Nous étudierons les avantages et les inconvénients de cette approche.

4.1 WorkFlow

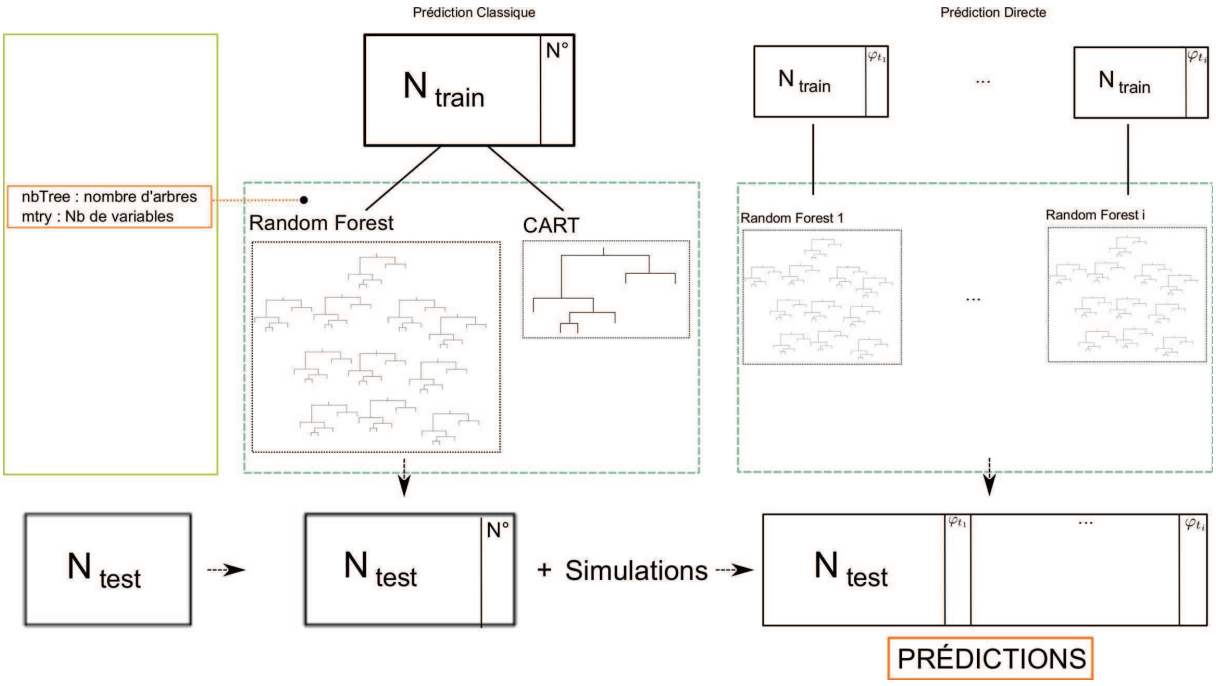


FIGURE 4.1 – Workflow - Étape de classification de la base d'apprentissage et de prédiction de la base de test

4.2 Notations

i	:	Numéro du vol de la base d'apprentissage $i \in 1, ..., n$
n_{test}	:	Nombre de vols de test
p	:	Nombre d'attributs
V_i	:	Vecteur de p attributs du vol i parmi l'ensemble des attributs $\mathcal{V}$
N_{train}	:	Base d'apprentissage des vecteurs d'attributs de taille $n \times p$
N_{test}	:	Base de test des vecteurs d'attributs de taille $n_{test} \times p$

Posons par commodité la date de départ à l'origine, $T_0^{(i)} = 0$. Nous définissons la variable $\varphi_t^{(i)} \in \{0, 1\}$ correspondant respectivement aux conseils "achat" et "attendre" à l'instant t pour le vol i. Nous définissons pour le reste du Chapitre, $\varphi_{t_1}^{(i)} = 1$ si et seulement si le prix $p_i(-t_2)$ à 7 jours est inférieur ou égal à $p_i(-t_1)$. Il est possible de calculer $\varphi_{t_1}^{(i)}$ de plusieurs autres manières mais nous avons décidé de nous concentrer sur une prédiction simple pour ensuite la raffiner.

Nous rappelons q'une fois la classe j obtenue en appliquant le classifieur aux attributs du trajet i considéré, le modèle de la classe j est utilisé pour calculer la probabilité $\mathbb{P}(\varphi_{t_1}^{(i)} = 1)$ grâce aux simulations précédemment décrites. Cette valeur est utilisée pour proposer un conseil d'achat ou d'attente pour un niveau de confiance donné. Dans le cas de la prédiction directe, nous verrons qu'elle servira d'étiquette lors de la phase d'apprentissage.

4.3 Apprentissage

Avant de pouvoir prédire l'évolution des prix d'un vol à un instant t, c'est-à-dire $\mathbb{P}(\varphi_t^{(i)} = 1)$, il nous faut prédire le groupe auquel il appartient. Les seuls informations dont nous disposons sont la liste de ses attributs $V_i = V_i(1), \ldots, V_i(p)$ dans l'espace des attributs $\mathcal{V}$, et éventuellement les premières évolutions de prix. Nous tentons dans un premier temps d'utiliser uniquement les attributs du vol pour prédire son comportement le plus vraisemblable. Pour cela nous utilisons des algorithmes de la famille des arbres de classification, qui vont segmenter l'espace des attributs de la base d'apprentissage selon l'étiquette E_i correspondant au groupe associé à chaque vol, afin de créer des régles de classification :

1. **CART** (Breiman, 1984 [5]) est un algorithme de classification basé sur les arbres de décision. Il va découper l'espace des attributs récursivement en maximisant à chaque étape le gain d'information. A chaque nœud, il choisit l'attribut le plus discriminant et sépare l'ensemble des vols en deux groupes qui, à leur tour, seront divisés en deux sous-ensembles. L'arbre final est un arbre binaire très facilement interprétable puisqu'il suffit de remonter l'arbre depuis chaque feuille pour obtenir les règles de classification. Nous décrivons par la même occasion un autre algorithme basé sur les arbres de décision et très largement utilisé, nommé C4.5 (introduit par Quilan [41]) qui est une amélioration de l'algorithme ID3 [40], et dont les récentes améliorations (il existe maintenant l'algorithme C5.0) le font se rapprocher de CART [23].

2. **Adaboost** (Freund, 1995 [19]) est une méthode de boosting dont le principe est de sélectionner des classifieurs faibles afin de minimiser l'erreur en classification selon des exemples d'apprentissage pondérés itérativement en fonction de leur difficulté à être correctement classés. Nous utiliseront les arbres CART comme classifieurs faibles.

3. **Les forêts aléatoires** (Breiman, 2001 [9]) sont une variante du bagging (également introduit par Breiman [6]) dont le principe est de construire un grand nombre d'arbres décorrélés pour ensuite les moyenner. Dans la plupart des cas d'utilisation, les forêts aléatoires ont des performances similaires au boosting (Adaboost), tout en étant plus facile à entraîner et à paramétrer. C'est pourquoi les forêts aléatoires sont populaires, et sont implémentées dans de nombreux langages. Genuer([21], 2010), fait une description complète et détaillée de la théorie et de la pratique des forêts aléatoires, détaillant notamment la possibilité de sélectionner les variables les plus pertinentes afin d'améliorer la classification.

4.3.1 Arbres de décision : CART & C4.5

L'algorithme de CART peut être divisé en 4 étapes. La première étape consiste à construire un arbre de manière récursive en séparant l'ensemble des vols d'un nœud en deux sous-ensembles selon un des attributs : il est important de noter qu'à chaque nœud, terminal ou non, est associée une classe prédite en fonction de la distribution des vols du groupe correspondant et de la matrice de coût. Cette construction doit ensuite être stoppée suivant différents critères d'arrêt. A l'issue de la construction, l'arbre est de grande taille et surapprend les données d'apprentissage. La troisième étape consiste donc à élaguer l'arbre pour le rendre plus simple et donc plus généralisable. Enfin il s'agira de retenir l'arbre élagué qui minimise le taux d'"erreur estimé" mesuré sur un échantillon test.

1. **La construction de l'arbre** commence à la racine où tous les vols sont regroupés. L'algorithme va alors trouver le meilleur attribut avec lequel séparer l'ensemble des vols en deux : il faut alors tester toutes les valeurs possibles de chaque attribut afin d'optimiser un critère prédéfini. Pour éviter un temps de calcul trop important il est possible de préciser le nombre de niveaux à tester pour les attributs qualitatifs.

   A chaque itération, nous cherchons à maximiser la pureté moyenne des deux sous-nœuds. Pour un nœud m représentant une sous-partition N_m de l'ensembles des attributs de la base d'apprentissage N_{train} et possédant n_m vols posons :

   $$\hat{p}_{mk} = \frac{1}{n_m} \sum_{i \in N_m} \mathbb{1}_{y_i = k},$$

   la proportion des vols de la classe k au nœud m. A chaque nœud m, nous classons l'ensemble des vols dans la classe $k(m) = \arg\max_k \hat{p}_{mk}$. Nous définissons alors différentes métriques de l'impureté d'un nœud :

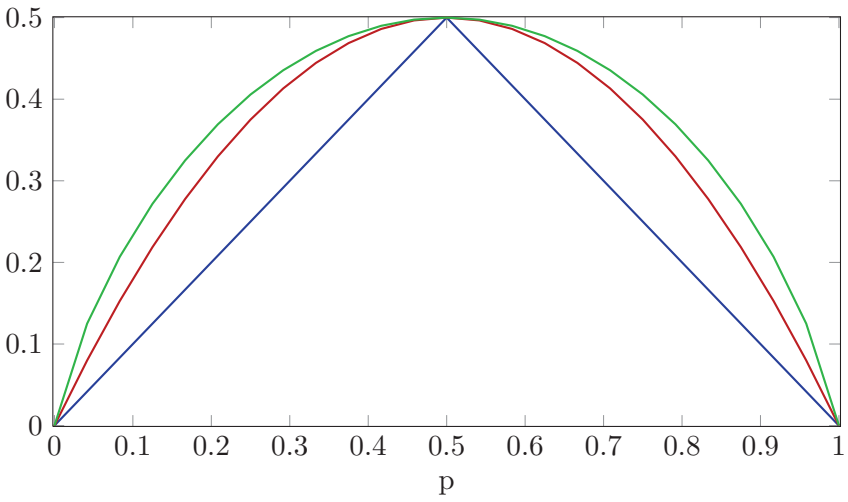


FIGURE 4.2 – Mesures d'impureté d'un nœud pour une classification binaire en fonction de la proportion p de la classe 2.

Erreur en classification :	$\frac{1}{n_m} \sum_{i \in N_m} \mathbb{1}(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}$.
Indice de Gini :	$\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{C} \hat{p}_{mk}(1 - \hat{p}_{mk})$.
Entropie croisée ou déviance :	$-\sum_{k=1}^{C} \hat{p}_{mk} \log(\hat{p}_{mk})$.

Dans le cas d'une classification à deux classes, si p est la proportion dans la deuxième classe, ces trois mesures s'écrivent respectivement $1-(max(p, 1-p)$, $2p(1-p)$ et $-p \log p - (1-p) \log(1-p)$. La Figure 4.2 nous montre graphiquement les différences entre ces trois mesures d'impureté.

Nous n'utiliserons que l'indice de "Gini", introduit par Breiman dans son étude de CART, et qui servira aussi de mesure d'impureté dans l'algorithme des forêt aléatoires. L'impureté d'une population pour une classification multiclasse est mesurée par la formule suivante :

$$I(T) = 1 - \sum_{j=1}^{c} (\frac{n_{j.}}{n_{..}})^2$$

où n_{ji} représente l'effectif de la classe j au nœud i ; le point symbolise alors la sommation selon l'indice correspondant. Cette mesure peut être vue comme l'erreur de classification attendue si la classe prédite était choisie aléatoirement en suivant la distribution des probabilités de classes, approximée par les effectifs du nœud T courant.

Breiman suggère que, dès lors que les mesures utilisées possèdent de bonnes propriétés de spécialisation, c'est-à-dire tendent à mettre en avant les partitions avec des feuilles pures,

elles ne jouent pas un rôle majeur dans la qualité de la prédiction.

Un des inconvénients est que l'algorithme de partitionnement favorise les variables qualitatives avec le plus de niveaux : plus ce nombre augmente, plus le nombre de partitions augmente rapidement ainsi que le choix de segmentation. Il est alors plus probable de trouver une séparation du nœud en utilisant les variables avec le plus de niveaux.

2. **L'arrêt de la construction** intervient lorsque :

   (a) Il n'y a plus qu'une seule observation dans les feuilles
   (b) La distribution des attributs dans toutes les feuilles est identique rendant la séparation impossible
   (c) La profondeur limite définie a été atteinte

   L'arbre final est souvent trop fidèle aux données et n'est donc pas capable de généraliser correctement sur la base de test. C'est pourquoi il est nécessaire de passer par l'étape d'élagage.

3. **L'élagage** consiste à supprimer a posteriori des branches de l'arbre jugées inutiles. Si aucun élagage n'est réalisé l'arbre sur-apprend inévitablement les données d'apprentissage et devient par conséquent très mauvais en généralisation. Le principe de la démarche consiste à construire une suite emboîtée de sous-arbres de l'arbre maximum par élagage successif puis à choisir, parmi cette suite, l'arbre optimal au sens d'un critère. La solution ainsi obtenue par un algorithme pas à pas n'est pas nécessairement globalement optimale mais l'efficacité et la fiabilité sont préférées à l'optimalité.

   Pour un arbre A donné, nous notons K le nombre de feuilles ou nœuds terminaux de A ; la valeur de K exprime la complexité de A. La mesure de qualité de discrimination d'un arbre A s'exprime par un critère

   $$D(A) = \sum_{k=1}^{K} D_k(A)$$

   où $D_k(A)$ est le nombre de vols mal classés de la $k$ème feuille de l'arbre A.

   La construction de la séquence d'arbres emboîtés repose sur une pénalisation de la complexité de l'arbre :

   $$C(A) = D(A) + \gamma K$$

   Pour $\gamma = 0$, $A_{max} = A_K$ minimise $C(A)$. En faisant croître γ, l'une des divisions de A_K, celle pour laquelle l'amélioration de D est la plus faible (inférieure à γ), apparaît comme superflue et les deux feuilles obtenues sont regroupées (élaguées) dans le nœud père qui devient terminal ; A_K devient A_{K-1}. Le procédé est itéré pour la construction de la séquence emboîtée :

   $$A_{max} = A_K \supset A_{K-1} \supset ...A_1$$

où A_1, le nœud racine, regroupe l'ensemble de l'échantillon. L'arbre optimal correspond donc au k qui minimise $D(A_k)$.

Sur la Figure 4.3, nous observons un exemple d'arbre de décision créé à partir de notre base de données. Nous remarquons que des variables peuvent être utilisées plusieurs fois sur différents nœuds, ici *initialPrice*, et que tous les groupes peuvent ne pas apparaître dans les nœuds terminaux (le groupe 4 dans notre cas). Comme expliqué précédemment, il est facile d'interpréter ces arbres et de créer des règles de classification en remontant les feuilles jusqu'à la racine. Les deux règles de la branche de gauche peuvent alors s'écrire :

(a) Si le prix initial est inférieur à 148€ et que la distance entre les aéroports est **inférieure** à 913km alors le vol fait partie du group 1
(b) Si le prix initial est inférieur à 148€ et que la distance entre les aéroports est **supérieure** à 913km alors le vol fait partie du group 2

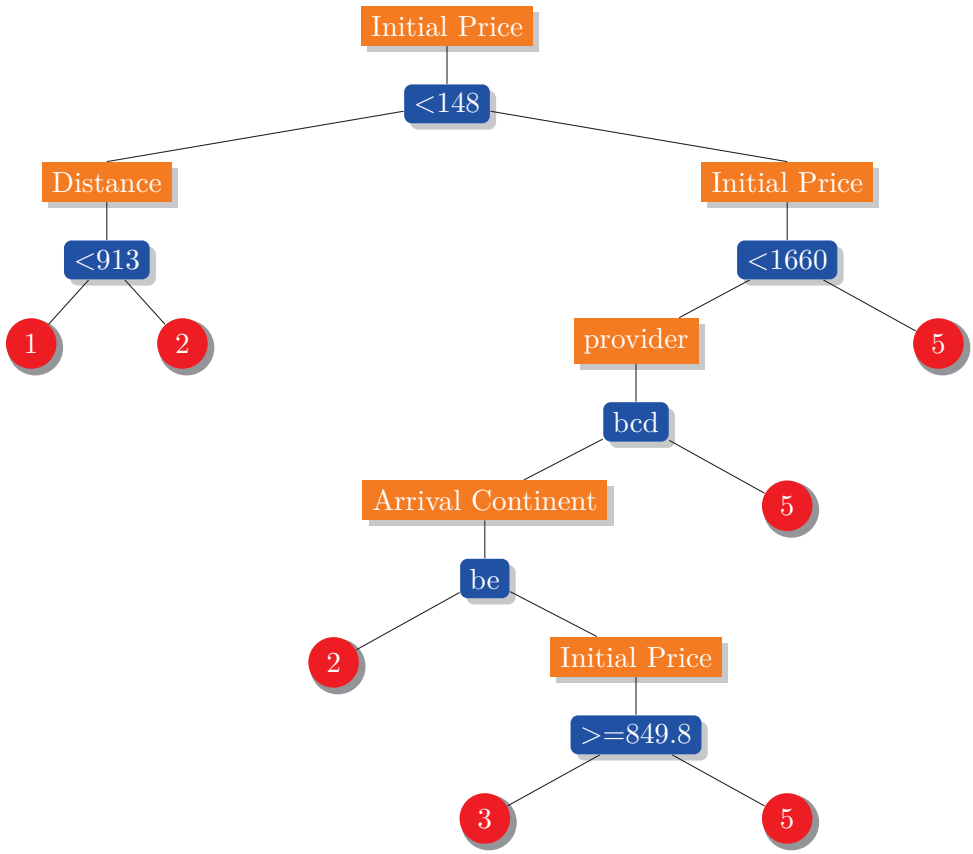


FIGURE 4.3 – Test de représentation d'un arbre binaire : Modèle 348 (CART)

Ce ne sont pas réellement les critères d'évaluation des règles de partitionnement qui conditionnent la structure des arbres, mais la topologie de la base d'apprentissage. En effet CART

est un algorithme qui sur-apprend les données d'apprentissage, et dépend par conséquent inévitablement de la représentativité de ces données vis à vis du domaine de définition du problème. L'avantage des arbres de décision est qu'ils sont simples à comprendre et à interpréter, qu'ils supportent à la fois les données quantitatives et qualitatives et que le temps de calcul sur un grand volume de données reste faible sans dégrader les performances.
Cependant les arbres peuvent devenir trop complexes, ne généralisant pas correctement les données et l'utilisation de variables qualitatives favorisant celles avec le plus de niveaux.

L'algorihtme C4.5 et par extension C5.0, qui est une amélioration des performance de l'ancien algorithme, reprend les principes de construction d'un arbre de décision de CART. Toutefois certains points diffèrent :

1. Alors que traditionnellement CART utilise l'indice de GINI comme critère de sélection de l'attribut suivant lequel séparer le nœud courant, C4.5 utilise l'entropie croisée.

2. L'arbre généré n'est pas binaire : C4.5 créé plusieurs branches à chaque séparation suivant le nombre de niveaux pour une variable qualitative. Pour une variable quantitative, une branche est créée pour chaque intervalle de la version discrétisée de la variable.

Les algorithmes de CART et C4.5, comme tout autre algorithme unique, possède des limitation décrites dans [16] et résumées dans [4] qui expliquent les faibles performances de celui-ci :

La limite statistique concerne les situations pour lesquelles l'espace de recherche du classifieur est particulièrement grand proportionnellement au nombre de données disponibles en apprentissage. Il est en revanche possible de réduire l'erreur d'un classifieur peu performant en généralisation, en combinant les prédictions de plusieurs classifieurs du même type, entraînés sur les mêmes données, et qui se sont montrés performants en apprentissage.

La limite de représentation de la "véritable" fonction de prédiction $y = f(x)$ décrit l'impossibilité de représenter celle-ci par un classifieur unique. Nous pouvons étendre l'espace de recherche en ajoutant des classifieurs combinant plusieurs classifieurs individuels. Il est par exemple possible de résoudre des problèmes de classification multi-classes à l'aide d'algorithmes d'apprentissage binaires.

La limite computationnelle représente le coût pour un algorithme d'apprentissage d'effectuer une recherche plus "large" afin d'éviter les optima locaux. Il est alors également possible de réduire le risque de choisir un mauvais classifieur en combinant plusieurs classifieurs sous-optimaux (localement optimaux).

En résumé, la combinaison de plusieurs classifieurs permettrait (i) de fiabiliser les décisions grâce à l'avis de plusieurs experts face à un seul, (ii) d'élargir l'ensemble des solutions possibles, (iii) d'éviter les optima locaux, et (iv) de traiter des problèmes trop complexes pour un simple classifieur.

Dans la domaine des système multi-classifieurs (MCS), les Ensemble de Classifieurs est une des approches les plus populaires et les plus efficaces, et consiste à combiner un ensemble de classifieurs de même type. L'objectif est de créer de la diversité au sein d 'un ensemble de classifieurs performants et d'établir le meilleur consensus possible entre ces classifieurs. La diversité est la propriété importante pour un MCS.

4.3.2 Adaboost

Adaboost [19] est une technique de boosting basée sur la sélection itérative de classifieurs faibles (en l'occurence CART). L'idée est d'améliorer les compétences d'un faible classifieur c'est-à-dire celle d'un modèle de discrimination dont la probabilité de succès sur la prévision d'une variable qualitative est légèrement supérieure à celle d'un choix aléatoire.

Introduit par Freund et Schapire, le boosting est une des méthodes d'ensemble les plus performantes à ce jour. Étant donné un échantillon d'apprentissage $\mathcal{V}_n$ et une méthode de prédiction faible (ici CART) qui construit un prédicteur $\hat{h}(\mathcal{V}_n)$. Un premier échantillon $\mathcal{V}_n^1$ est tiré aléatoirement (bootstrap), où chaque observation a une probabilité $1/n$ d'être tirée. Nous appliquons alors le classifieur de base à cet échantillon pour obtenir un premier prédicteur $h(\mathcal{V}_n^1)$. Ensuite, l'erreur de $\hat{h}(\mathcal{V}_n^1)$ sur l'échantillon $\mathcal{V}_n$ est calculé. Un deuxième échantillon bootstrap $\mathcal{V}_n^2$ est alors tiré en se basant sur les prédictions de $\hat{h}(\mathcal{V}_n^1)$. Le principe est d'augmenter la probabilité de tirer une observation mal prédite, et de diminuer celle de tirer une observation bien prédite. Nous réitérons alors le processus pour obtenir un ensemble de classifieur que nous agrégeons ensuite en faisant une moyenne pondérée. Le Boosting est donc une méthode séquentielle, chaque échantillon étant tiré en fonction des performances de la règle de base sur l'échantillon précédent. L'idée du Boosting est de se concentrer de plus en plus sur les observations mal prédites par la règle de base, pour essayer d'apprendre au mieux cette partie difficile de l'échantillon, en vue d'améliorer les performances globales.

Avantages : ce type d'algorithme permet de réduire sensiblement la variance mais aussi le biais de prévision comparativement à d'autres approches. Cet algorithme est considéré comme la meilleure méthode "off-the-shelf" [8], c'est-à-dire ne nécessitant pas un long pré-traitement des données ni un réglage fin de paramètres lors de la procédure d'apprentissage. Le boosting adopte le même principe général que le bagging : construction d'une famille de modèles qui sont ensuite agrégés par une moyenne pondéré des estimations ou un vote. Il diffère nettement sur la façon de construire la famille qui est dans ce cas récurrente : chaque modèle est une version adaptative du précédent en donnant plus de poids, lors de l'estimation suivante, aux observations mal ajustées ou mal prédites. Intuitivement, cet algorithme concentre donc ses efforts sur les observations les plus difficiles à ajuster tandis que l'agrégation de l'ensemble des modèles permet d'échapper au sur-ajustement. Les algorithmes de boosting existants diffèrent par différentes caractéristiques :

1. la façon de pondérer c'est-à-dire de renforcer l'importance des observations mal estimées lors de l'itération précédente
2. leur objectif selon le type de la variable à prédire Y : binaire, qualitative à k classes, réelles
3. la fonction perte, qui peut être choisie plus ou moins robuste aux valeurs atypiques, pour mesurer l'erreur d'ajustement

4. la façon d'agréger, ou plutôt pondérer, les modèles de base successifs.

La littérature sur le sujet présente donc de très nombreuses versions de cet algorithme et il est encore difficile de dire lesquelles sont les plus efficaces.

Il est important de noter tout de même que dans le cas de forêts boostées, les arbres de décision doivent impérativement être élagués. En effet le boosting est réalisé sur la base des erreurs de prédiction en apprentissage des classifieurs faibles. Or, sans élagage, les arbres de décision sur-apprennent les données d'apprentissage jusqu'à avoir une erreur en apprentissage nulle dans la plupart des cas.

Dans notre cas, chaque vol d'apprentissage est pondéré à chaque itération afin de minimiser l'erreur de classification. Il n'y a donc aucune part d'aléatoire dans cette approche contrairement aux Random Forest.

Algorithm 3 Adaboost

Entrée : $\mathcal{S} = (V_i, E_i)_{i=1,...,n}$, avec E_i la classe obtenue en appliquant le classifieur
Pour $i = 1...n$ **faire**
  $p_1(V_i) \leftarrow \frac{1}{n}$
Fin Pour
Pour $j = 1...nbClassifieur$ **faire**
  Tirer un échantillon d'apprentissage S_j dans $\mathcal{S}$ selon les probabilités p_j
  Entraîner un classifieur C_j sur ce sous-échantillon.
  Soit q_j l'erreur apparente de C_j sur $\mathcal{S}$
  Calculer $\alpha_j = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$
  Pour $i = 1...n$ **faire**
    si $h_j(V_i) = y_i$ (bien classé par h_t) **alors**
      $p_{j+1}(V_i) \leftarrow \frac{p_j(V_i)}{Z_t} e^{-\alpha_t}$
    sinon
      $p_{j+1}(V_i) \leftarrow \frac{p_j(V_i)}{Z_t} e^{+\alpha_t}$
    Fin si
    Avec Z_t normalisé tel que $\sum_{i=1}^{m} p_j(V_i) = 1$
  Fin Pour
Fin Pour
return L'ensemble des arbres $\{T_b\}_1^B$

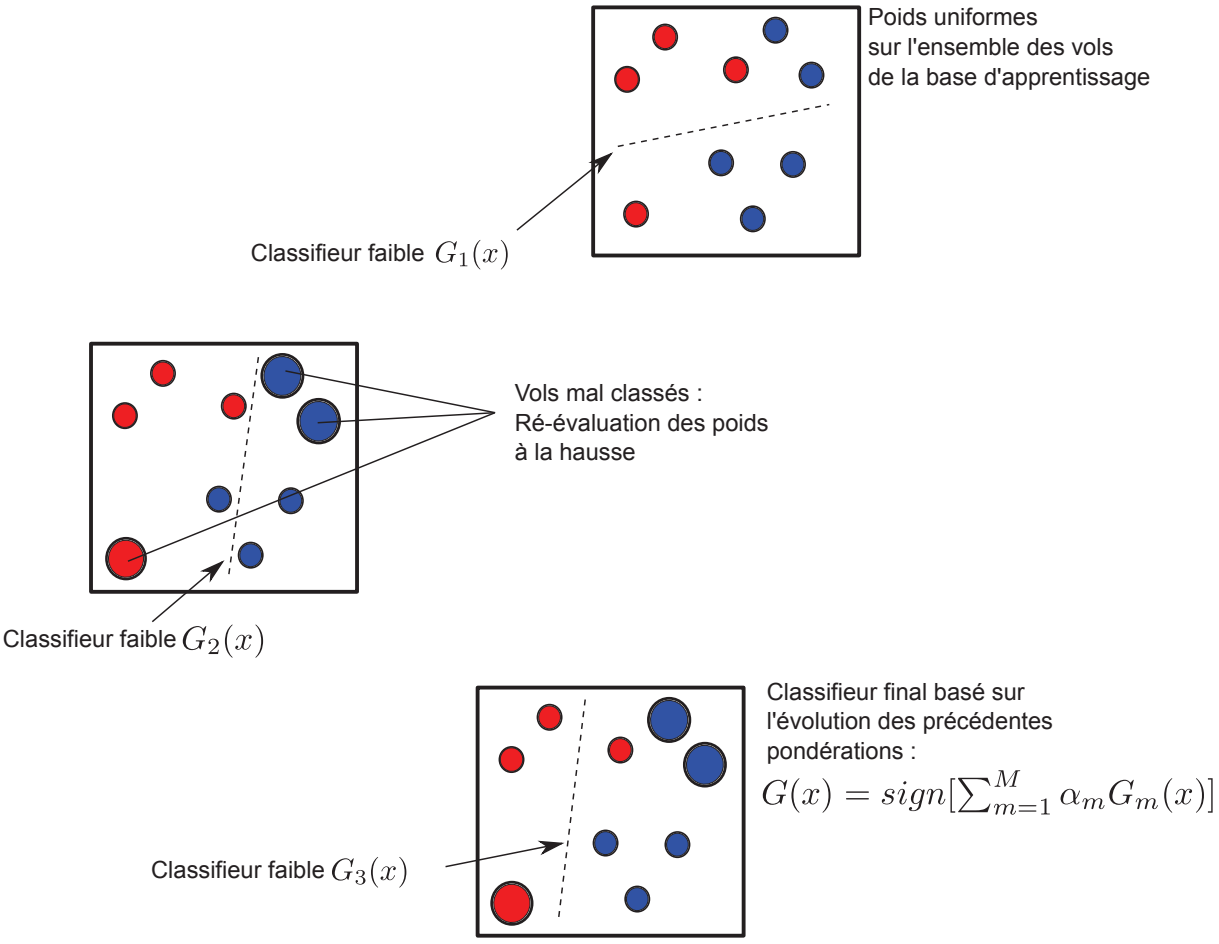


FIGURE 4.4 – Adaboost

Les méthodes de Boosting sont en fait très fréquemment citées à titre de comparaison dans les travaux qui étudient les forêts aléatoires, puisqu'en plus d'être particulièrement efficaces pour l'induction de forêts de décision, elles s'opposent aux forêts aléatoires dans l'idée principale : le Boosting s'appuie sur un principe déterministe pour la création de la diversité dans les ensembles alors que les forêts aléatoires par définition le font via les principes de randomisation.

4.3.3 Forêts aléatoires (Random Forest)

Les méthodes de forêts aléatoires sont basées sur la combinaison de classifieurs élémentaires de types arbres de décision. Individuellement, ces classifieurs ne sont pas réputés pour être particulièrement efficaces, mais ils possèdent une propriété intéressante à exploiter : leur instabilité. Un petit changement dans la base d'apprentissage provoque un changement important dans la structure de l'arbre et donc dans ses performances en généralisation. La spécificité des arbres utilisés dans les forêts aléatoires est que leur induction est perturbée d'un facteur aléatoire, et

ce dans le but de générer de la diversité dans l'ensemble. C'est sur la base de ces deux éléments — utiliser des arbres de décision comme classifieurs élémentaires et faire intervenir l'aléatoire dans leur induction — qu'a été introduit le formalisme des forêts aléatoires.

Dans [7] (1996), Breiman propose une méthode pour construire de multiples classifieurs indépendants sur des sous-ensembles aléatoires de la base d'apprentissage. Chaque classifieur est un arbre de décision CART précédemment décrit mais qui n'a pas été élagué. Les prédictions de chaque arbre sont regroupées et la prédiction finale est choisie par un système de vote à la pluralité : chaque arbre participe au vote de la classe la plus populaire pour une donnée d'entrée x. Cette méthode est basé sur un principe déjà connue nommé bagging.

Le Bagging (**B**ootstrap **Agg**regat**ing**) applique le principe de Bootstrap à l'agrégation de classifieurs. Le Bootstrap est un principe de ré-échantillonnage statistique [17] traditionnellement utilisé pour l'estimation de grandeurs ou de propriétés statistiques. L'idée du bootstrap est d'utiliser non plus une unique distribution empirique, mais plusieurs ensembles de données ré-échantillonnées à partir de l'ensemble des données observées à l'aide d'un tirage aléatoire avec remise. Le bagging est alors l'agrégation de classifieurs appliqués chacun sur des sous-ensembles de données ré-échantillonnées.

Breiman [9] (2001) décide avec les random forest d'ajouter une couche d'aléatoire dans la construction de ces classifieurs en plus de la sélection aléatoire du sous-ensemble sur lequel apprendre l'arbre. Dans le cas du bagging, l'algorithme de construction des arbres est le même que CART et divisent les nœuds en choisissant la meilleure variable parmi toutes les variables possibles. Dans le cas des forets aléatoires, la sélection de la meilleure variable se fait sur un sous-ensemble aléatoire de l'ensemble des variables. Puisque les classifieurs sont de même type, il est évidemment important de générer des "différences" entre chacun d'eux, le critère essentiel dans les système multi-classifieurs étant, encore une fois, de créer de la diversité.

Breiman démontre la propriété importante de convergence des forêts aléatoires. Plus précisément il démontre que le taux d'erreur en généralisation, d'une forêt aléatoire converge nécessairement vers une valeur limite, quand le nombre d'arbres qui la composent augmente. Concrètement, cela signifie une chose importante : avec un nombre suffisamment élevé d'arbres, une forêt aléatoire évite le sur-apprentissage.

Cette approche donne de meilleurs résultats que la majorité des autres classifieurs, est robuste face au sur-apprentissage et ne nécessite que deux paramètres : la taille du sous-ensemble de variables et le nombre d'arbres à construire.

L'algorithme 4 [21] décrit les différentes étape de la construction de la forêt.

La principale force du bagging est donc de réduire l'instabilité pour augmenter les performances en généralisation. Mais il y a un autre point qui fait la force du bagging, ce sont les mesures out-of-bag :

Les forets aléatoires ne nécessitent pas de validation croisée, ou même d'une base de test indépendante pour estimer l'erreur en test. Cette erreur est calculée tout au long du processus comme suit : Pour chaque arbre construit sur le sous-ensemble Z^*, une prédiction est faite sur tous les vols de l'ensemble $Z^{*\complement}$. En agrégeant les prédictions faites par tous les arbres (en moyenne un vol sera prédit une fois sur trois), nous pouvons calculer l'erreur en prédiction que nous nommons alors l'estimation de l'erreur OOB.

Algorithm 4 Forêt d'arbres décisionnels pour la classification

Entrée : $\mathcal{S} = (V_i, E_i)_{i=1,...,n}$, avec E_i la classe obtenue en appliquant le classifieur
Entrée : $ntree$ le nombre d'arbres dans la forêt
Entrée : $mtry$ le nombre de caractéristiques à sélectionner aléatoirement à chaque nœud
Pour $j = 1...ntree$ **faire**
  $\mathcal{S}_j \leftarrow$ ensemble boostrap, dont les données sont tirées aléatoirement (avec remise) de $\mathcal{S}$
  $arbre \leftarrow$ un arbre vide composé de sa racine uniquement
  $arbre.racine \leftarrow RndTree(arbre.racine, \mathcal{S}_j, mtry)$
  $foret \leftarrow foret \cup arbre$
Fin Pour
return $forest$

Algorithm 5 RndTree

Entrée : n le nœud courant
Entrée : $\mathcal{S}$ l'ensemble des données associées au nœud n
Entrée : $mtry$ le nombre de caractéristiques à sélectionner aléatoirement à chaque nœud
si n n'est pas une feuille **alors**
  $C \leftarrow mtry$ caractéristiques choisies aléatoirement
  pour tout $A \in C$ **faire**
    Procédure CART pour la création et l'évaluation (critère de Gini) du partitionnement produit par A, en fonction de $\mathcal{S}$
  Fin Pour
  $partition \leftarrow$ partition qui optimise le critère Gini
  $n.ajouterFils(partition)$
  pour tout $fils \in n.noeudFils$ **faire**
    $RndTree(fils, fils.donnes, mtry)$
  Fin Pour
Fin si
return

Importance des variables Il est très utile, en pratique, d'avoir des informations sur les variables des données que l'on étudie. Quelles sont les variables vraiment nécessaires pour expliquer notre classification ? De quelles variables peut-on se passer ? Ces informations peuvent être d'une grande aide pour l'interprétation des données. Elles peuvent également servir à construire de meilleurs prédicteurs : un prédicteur construit en utilisant uniquement les variables utiles pourra être plus performant et moins complexe qu'un prédicteur construit avec des variables susceptibles de générer du bruit.

Deux méthodes de calcul de l'importance des variables existent dans l'algorithme des forêts aléatoire qui nous permettent de classer les variables par ordre d'importance dans l'apprentissage :

1. La "moyenne des baisses en précision" d'une variable se calcule grâce à l'erreur OOB. Pour mesurer l'importance du j-ième attribut après l'apprentissage, les valeurs du j-ième attribut sont permutées dans la base de données d'apprentissage et l'erreur OOB est à nouveau calculée sur cet ensemble de données "perturbée". L'importance du j-ième attribut est alors calculée en faisant la moyenne de la différence entre l'erreur OOB avant et après la perturbation sur tous les arbres. Le score est normalisé par l'écart type de ces différences. Plus la précision de la classification de la forêt décroît par l'ajout d'une variable supplémentaire, plus celle-ci est importante.

2. La "moyenne des baisses du coefficient de Gini" mesure l'influence d'une variable dans l'homogénéité des nœuds et des feuilles des arbres de la forêt. Dès qu'une variable est sélectionnée pour diviser un nœud, le coefficient de Gini est calculé pour les deux enfants et comparé au nœud initial. La différence des coefficient est sommée pour chaque variable et normalisée à l'issue du processus : plus la baisse est importante, plus l'influence de la variable sur la pureté des nœuds l'est aussi.

Ces méthodes présentent certains inconvénients : elles favorisent les attributs quantitatifs et les attributs qualitatifs possédant le plus de niveaux [47] (comme pour CART). De plus, elles se sont révélées être fortement corrélées. D'autres articles ajoutent que si certains attributs sont corrélées ou ont la même pertinence dans la prédiction, alors les petits groupes sont favorisés par rapport aux grands groupes [49]. Dans le papier [47], Strobl et al. proposent d'utiliser une autre implémentation des forêts aléatoires où les classifieurs faibles sont des arbres de classification non-biaisés (une estimation non-biaisée du taux d'erreur en classement lors de la phase d'élagage) basés sur une approche conditionnelle (package R [25]). De plus, Strobl et al. préconisent l'utilisation d'un boostrap sans remise afin d'obtenir une mesure d'importance des variables sans biais.

Cardinalité des attributs Un autre inconvénient des forêts aléatoires est sa limitation dans le nombre maximum de niveau pour les attributs qualitatifs. L'étape de permutation interdit un trop grand nombre de niveaux en raison de sa nature combinatoire. L'implémentation R, par exemple, nous limite à 32 niveaux par attributs. De plus, la multiplication des niveaux perturbe la classification aussi bien qualitativement que computationnellement [5]. Une solution proposée dans [35] lors de la compétition KDD 2009, est de regrouper les niveaux peu représentés entre

eux et garder les plus fréquents. Nous avons donc décidé d'utiliser pour les variables qualitatives les niveaux apparaissant plus de 2% du temps dans une limite de 30 niveaux. Un nouveau niveau "-1" est alors créé afin de regrouper l'ensemble des autres valeurs.

4.4 Prédiction d'un comportement

Dans cette section nous décrivons le passage de la prédiction d'un groupe à partir d'un vecteur d'attributs à la prédiction d'une évolution de prix. Grâce aux algorithmes précédemment décrits, nous sommes capable d'associer à un vol un numéro de groupe E_i à partir du vecteur d'attributs V_i en suivant des régles de classification. Une fois ce groupe assigné au vol, nous utilisons les simulations du centroïde associé (décrites dans le Chapitre 2) pour prédire le comportement futur de la série de prix à l'instant t de la demande de prédiction.

Apprentissage d'un comportement Chaque vol de test possède un vecteur d'attributs V_i de même nature que les vols de la base d'apprentissage. Nous avons vu précédemment la limitation de certains algorithmes à traiter des vecteurs d'attributs avec des niveaux absents de la base d'apprentissage. Nous avons donc créé pour chaque attribut qualitatif un niveau supplémentaire nommé "-1". Lors de la construction des V_i de N_{test}, nous prenons soin de vérifier l'existence des niveaux des attributs qualitatifs dans la base d'apprentissage et dans le cas contraire de lui attribuer la valeur "-1". Nous appliquons par la même occasion la selection des valeurs les plus présentes et le regroupement des plus rares dans le même niveau "-1".

Les modèles de classification précédemment créés génèrent pour chaque vol de test, représenté par son vecteur d'attribut V_i, un vecteur de sortie correspondant à la probabilité d'appartenance à chacun des cluster : $\mathbb{P}(V_i \in I_k), \forall k = 1, ..., C$. L'étiquette E_i correspondant au numéro du cluster le plus probable est alors associée aux vols de test. Le cas d'une équiprobabilité d'appartenance à plusieurs groupes pourra être traité de différentes manières que nous décrirons plus tard.

Prédiction d'une évolution Nous avons maintenant un modèle capable d'attribuer un numéro de cluster à un nouveau vol uniquement grâce à ses attributs. Ce numéro de cluster nous permet d'associer au vol de test un centroïde et donc un comportement global au vol représenté par le centroïde du groupe. Nous rappelons qu'à partir de n'importe quel niveau de gris nous sommes capable de reconstituer des vols issues de celui ci. Dans le cas des centroïdes, cela signifie que nous créons des séries temporelles synthétiques qui seraient issues de ce cluster. En d'autres termes, la courbe simulée par le centroïde I_{E_i}, si elle faisait partie de la base d'apprentissage, devrait être classée dans le cluster E_i.

Avant même de simuler les courbes de prix, nous devons définir la nature de la prédiction, *i.e.* choisir le calcul à appliquer sur les séries pour obtenir φ. Il ne faut pas oublier qu'une prédiction se fait à un instant t et qu'elle sera différente à $t+1$: φ_t répond à une interrogation sur l'évolution du prix d'une série de prix à un moment précis de celle-ci. Il y aura donc autant de φ_t que de jours avant la date de départ.

φ_t peut être binaire lorsque l'interrogation sur l'évolution du prix l'est aussi : le prix va-t-il augmenter dans 7 jours ? y aura-t-il une baisse de prix d'au moins 24 heures dans les 10

prochains jours ? Dans ce cas, nous simulons N courbes issues du cluster assigné au vol de test, et nous calculons pour chacune d'elles $\varphi_t \in [0, 1]$. En moyennant les prédictions, nous obtenons $\mathbb{P}(\varphi_t = 1)$ qui sera la prédiction finale du vol.

Mais on peut aussi vouloir prédire l'évolution des prix selon des cadrans tels que "Forte Hausse","Faible Hausse","Stable","Faible Baisse" et "Forte Baisse". Dans ce cas, la procédure reste inchangée à la différence près que φ_t prendra 5 valeurs correspondant aux différents cadrans. Pour toutes les simulations, chaque cadran est incrémenté lorsque l'évolution de prix correspond au critère de celui-ci et les résultats de tous les cadrans sont ensuite normalisés. Nous obtenons ainsi autant de $\mathbb{P}(\varphi_t = 1)$ que de cadran. Nous pouvons alors interpréter comme nous le souhaitons les résultats et même retrouver un φ_t binaire en fusionnant les cadrans.

4.5 Prédiction directe

Une autre méthode de prédiction de φ_t est d'appliquer la phase d'apprentissage à la fonction de calcul de φ_t. De cette manière, l'étape de segmentation n'est plus nécessaire et il suffit seulement de calculer φ_t pour tous les vols de la base d'apprentissage et pour tout t. L'étiquette à apprendre n'est donc plus le numéro du groupe mais directement φ_t. Avec cette méthode, il n'y a plus de perte d'informations due à la transformation en niveau de gris et aux simulations à partir de centroïdes. De plus si l'apprentissage se fait sur une valeur binaire de φ_t, les chances d'erreurs diminuent naturellement avec la diminution de la complexité du problème. L'inconvénient principal de cette approche est l'obligation d'avoir un modèle par nombre de jours avant la date de départ, et surtout de devoir recalculer les modèles à chaque nouvelle définition de φ contrairement à l'approche classique où seules les simulations sont à mettre à jour.

Une manière d'évaluer la pertinence de notre prédiction dite "classique" est donc de la comparer à l'apprentissage dit "direct" de l'évolution du prix. Comme expliqué précédemment, un φ_t correspond à une situation temporelle unique confrontant ainsi un modèle (celui classique) à une multitude d'autres (les modèles directs).

4.6 Approches séquentielles

Dans la section 1.2 nous parlions des techniques de yield management utilisant l'évolution de la demande comme paramètre principal dans la modification des prix. Nous pensons donc que les premières évolutions de prix peuvent nous indiquer la tournure que vont prendre les futurs changements de prix. L'utilisation de cette information est cependant soumise aux recherches des utilisateurs, qui sont notre seule source d'information. Les destinations populaires pendant les périodes scolaires seront les uniques séries pouvant bénéficier de ce complément d'information, car le volume de recherche est suffisamment conséquent pour que les combinaisons de dates, de routes et de durées de séjour se recoupent.

L'utilisation des premiers points peut se faire de différentes manières. Dans la prédiction directe par exemple, l'apprentissage de φ_t par les random forest peut se faire en utilisant les attributs contextuels décrit dans le Chapitre 1 : ces attributs dépendent comme φ_t du nombre de jours avant la date de départ et peuvent représenter la volatilité observée des prix, le prix courant,

le nombre de hausses ou de baisses observées. Ajoutés aux attributs statiques, ils apportent un plus non négligeable comme nous le verrons dans le Chapitre 5.

Il est aussi possible d'utiliser les premiers points comme unique paramètre de prédiction créant ainsi un nouveau type de classifieur. En utilisant le calcul de vraisemblance de l'algorithme Espérance-Maximisation du Chapitre 3, nous attribuons le cluster le plus probable à chacun des vols de test.

Enfin nous utilisons l'information des premiers points dans la phase d'apprentissage des comportements en ajoutant un paramètre à l'algorithme d'Espérance-Maximisation. L'apprentissage est alors couplé à l'étape de segmentation et permet de calculer la vraisemblance d'appartenance à un cluster grâce au V_i et aux niveaux de gris partiels des vols de test.

4.6.1 Classification uniquement par les premiers points

En utilisant le calcul de vraisemblance de l'EM aux premiers points d'une courbe, il est possible d'associer un vol à un cluster sans l'utilisation de ses attributs. Supposons que la courbe du vol i est observée jusqu'à une date T, on peut alors reconstruire le niveau de gris partiel du vol $X_i(s,t)$ pour $(s,t) \in \tilde{\mathcal{R}}$ où $\tilde{\mathcal{R}}$ est l'ensemble des cases dans le plan temps-rendements correspondant aux prix observés jusque là. On note alors $\tilde{X}_i$ l'ensemble des $X_i(s,t)$ pour $(s,t) \in \tilde{\mathcal{R}}$. Connaissant les $\tilde{X}_i$, les centroïdes I_y ainsi que les α_y, nous calculons alors la vraisemblance d'appartenance à chaque cluster des vols de test ainsi :

$$\mathbb{P}_{\alpha,I}(Y_i = y|\tilde{X}_i) = \frac{p_{\alpha,I_y}(\tilde{X}_i, Y_i = y)}{\sum_{j=1}^{C} p_{\alpha,I_j}(\tilde{X}_i, Y_i = j)} \propto \alpha_y \prod_{(s,t)\in\tilde{\mathcal{R}}} \frac{(I_y(s,t))^{X_i(s,t)} e^{-I_y(s,t)}}{X_i(s,t)!},$$

pour $y \in 1, \ldots, C$.

La classification se fait en choisissant le y qui maximise cette vraisemblance. Notons que le calcul des $\tilde{X}_i(s,t)$ est d'autant plus précis que la fréquence d'échantillonnage est élevée (voir section 2.3.1).

4.6.2 EM logit

L'EM logit est une méthode de classification accompagnée d'une étape de segmentation similaire à l'EM décrit dans la section 3.3.3. Contrairement à l'EM de la section 3.3.3, la classification et la segmentation sont indissociables. Les données d'entrée sont l'ensemble des niveaux de gris $\widehat{X}_i$ auquel nous ajoutons l'information des attributs $\widehat{V}_i$.

L'hypothèse principale sur laquelle cette méthode repose consiste à dire que les niveaux de gris X_i du vol i dépendent des attributs, uniquement à travers l'appartenance à un cluster particulier. En termes probabilistes, la loi conditionnelle de X_i sachant Y_i ne dépend pas des attributs. En revanche, la loi de Y_i dépend des attributs à travers une fonction $\psi(\cdot,\cdot) : \mathbb{P}(Y_i = y) = \psi(y|V_i)$.

Comme Y_i n'est pas observé, il s'en suit que la loi des X_i dépend des attributs sous la forme : $\mathbb{P}(X_i = x) = \sum_{y=1}^{C} \mathbb{P}(X_i = x|Y_i = y)\psi(y|V_i)$. Comme à la section 3.3.3, la probabilité $\mathbb{P}(X_i = x|Y_i = y)$ dépendra du paramètre $I_y \in (\mathbb{R}_+^*)^{\mathcal{R}}$ et sera donnée par la loi de Poisson.

Nous serons donc encore en présence d'un modèle de mélange de Poisson. La nouveauté est dans le calcul des probabilités d'appartenance aux clusters qui, cette fois, vont être calculées en utilisant les attributs : les paramètres (α_y) sont remplacés par les fonctions $(\psi(y|v))$. Pour pouvoir estimer ces fonctions, nous utiliserons une paramétrisation de type logit (cf équation (4.7)).

Description algorithme

L'algorithme s'apparente à l'algorithme d'Espérance-Maximisation de la section 3.3.3, à cela près qu'il introduit l'information des attributs V_i dans l'étape de segmentation.

Données d'entrée Les données d'entrée sont celles de l'EM décrit dans la section 3.3.3 ainsi que les attributs V_i de l'ensemble des vols $i \in 1, \dots, n$.

Initialisation L'initialisation est quasiment identique à celle de l'EM de la section 3.3.3, mais ajoute un troisième paramètre à optimisé, donné par la fonction ψ définie plus haut. Une première segmentation est donc effectuée par les K-Means, initialisant les centroïdes I_y ainsi que les couples $(X_i, \hat{Y}_i)$. On utilise alors les couples $(V_i, \hat{Y}_i)$ pour optimiser la fonction ψ.

Fonction de vraisemblance La log densité des variables observées X_i, $i = 1, \dots, n$, sachant les attributs observés V_i, $i = 1, \dots, n$ s'écrit :

$$\begin{aligned}
\log p_\theta(x_1, \dots, x_n | v_1, \dots, v_n) &= \sum_{i=1}^{n} \log p_\theta(x_i | V_i) \\
&= \sum_{i=1}^{n} \log \sum_{y_i=1}^{C} p_\theta(x_i | V_i, y_i) p_\theta(y_i | V_i) \\
&= \sum_{i=1}^{n} \log \sum_{y_i=1}^{C} p_\theta(x_i | y_i) p_\theta(y_i | V_i) \ ,
\end{aligned}$$

où, pour $\theta = (I, \psi)$,

$$p_\theta(y_i | V_i) = \psi(y_i | V_i) \tag{4.1}$$

et

$$p_\theta(x_i | y_i) = \prod_{r \in \mathcal{R}} \frac{\mathrm{e}^{x_i(r)(\log I(r|y_i)) - I(r|y_i))}}{x_i(r)!} \ .$$

En factorisant tous les termes factoriels dans le log de la log-vraisemblance, on obtient :

$$\log p_\theta(x_1, \dots, x_n | v_1, \dots, v_n) = (\dots) + \sum_{i=1}^{n} \log \sum_{y_i=1}^{C} \exp(\ell(y_i | V_i, I)) \psi(y_i | V_i) \ , \tag{4.2}$$

où la constante $(\dots)$ ne dépend pas de θ et où

$$\ell(x|y,I) = \sum_{r\in\mathcal{R}} \mathrm{x_i(r)}(\log(\mathrm{I(r|y_i)}) - \mathrm{I(r|y_i)}) \ . \tag{4.3}$$

Etape E L'étape E consiste à calculer, connaissant les attributs $v_1,\dots,v_n$ et les deux paramètres $\theta = (I,\psi)$ et $\theta' = (I',\psi')$, l'espérance conditionnelle de la log-densité jointe au paramètre θ' sachant les variables observées X_i sous le paramètre θ :

$$Q(\theta', x_1,\dots,x_n|\theta, v_1,\dots,v_n) = \sum_{i=1}^{n}\sum_{y_i=1}^{C} \left[\log p_{\theta'}(x_i,y_i|V_i)\right]\ p_\theta(y_i|x_i,V_i) \ .$$

Par commodité, nous allons par la suite omettre $x_1,\dots,x_n$ et $v_1,\dots,v_n$ dans la notation de Q et ecrivons simplement $Q_n(\theta'|\theta)$

Nous obtenons alors :

$$\begin{aligned} Q_n(\theta'|\theta) = (\dots) + \sum_{y=1}^{C}\left(\sum_{r\in\mathcal{R}} B_n(y,r|\theta)\log I'(r|y) - A_n(y|\theta)\sum_{r\in\mathcal{R}} I'(r|y)\right) \\ + \sum_{y=1}^{C}\sum_{i=1}^{n}\left(\log\psi'(y|V_i)\right)\ p_\theta(y|x_i,V_i) \ , \end{aligned}$$

où $(\dots)$ est une constante indépendante de θ', et où l'on a :

$$p_\theta(y|x,v) = \frac{p_\theta(x,y|v)}{\sum_{y'} p_\theta(x,y'|v)} \ , \quad \text{and} \quad p_\theta(x,y|v) = \psi(y|v)\ \exp\ell(x|y,I) \ ,$$

avec ℓ définie dans (4.3) et

$$A_n(y|\theta) = \sum_{i=1}^{n} p_\theta(y|x_i,V_i) \quad \text{and} \quad B_n(y,r|\theta) = \sum_{i=1}^{n} x_i(r)\, p_\theta(y|x_i,V_i) \ . \tag{4.4}$$

Etape M Cette étape consiste à maximiser $Q(\theta'|\theta)$ en θ' pour un θ donné.

Comme pour l'EM de la section 3.3.3, l'optimisation en I' avec le poids ψ peut être traité séparément.

On obtient :

$$I'(r|y) = \frac{B_n(y,r|\theta)}{A_n(y|\theta)}, \quad r\in\mathcal{R}, y = 1,\dots,C \ . \tag{4.5}$$

L'optimisation en ψ dépend de la fonction utilisée dans l'ensemble des fonction Ψ. On obtient :

$$\psi' = \arg\max_{\phi\in\Psi} \sum_{i=1}^{n}\sum_{y=1}^{C}\left(\log\phi(y|V_i)\right)\ p_\theta(y|x_i,V_i) \ . \tag{4.6}$$

La classe Ψ représente un ensemble de fonctions possibles pour ψ. Dans le cas particulier où Ψ est constant en $\mathcal{V}$, c'est-à-dire $\Psi = S_C$, on retrouve

$$\psi'(y) = \frac{A_n(y|\theta)}{\sum_{y'=1}^{C} A_n(y'|\theta)} \ .$$

Ce cas ne nous intéresse pas ici puisque l'on souhaite faire intervenir les attributs. Nous avons choisi la classe des fonctions de type logit. Dans ce cas :

$$\psi(y, v) = logit_\varphi(y|V) = \frac{e^{\varphi_y^T V}}{\sum_{y=1}^{C} e^{\varphi_y^T V}} \tag{4.7}$$

La fonction ψ est entièrement décrite par le paramètre $\varphi \in \mathbb{R}^p$ (espace des attributs). Le calcul de ψ' revient donc à calculer le nouveau paramètre φ'

On obtient alors :

$$\varphi' = \arg\max_{\varphi}(\sum_{i=1}^{n}\sum_{y=1}^{C}(\varphi_y^T V_i)p_\theta(y|x_i, V_i) - \sum_{i=1}^{n}\sum_{y=1}^{C}\log(\sum_{j=1}^{C} e^{\varphi_j^T V_i})p_\theta(y|x_i, V_i))$$

$$= \arg\max_{\varphi}(\sum_{i=1}^{n}\sum_{y=1}^{C}(\varphi_y^T V_i)p_\theta(y|x_i, V_i) - \sum_{i=1}^{n}\log\sum_{y=1}^{C} e^{\varphi_y^T V_i})$$

En posant $\varphi_C = 0$,

$$\varphi' = \arg\max_{\varphi}(\sum_{i=1}^{n}\sum_{y=1}^{C-1}\varphi_y^T V_i p_\theta(y|x_i, V_i) - \sum_{i=1}^{n}\log(1 + \sum_{y=1}^{C-1} e^{\varphi_y^T V_i})$$

En dérivant, on obtient que φ' est solution de l'équation :

$$0 = \sum_{i=1}^{n} V_i p_\theta(y|x_i, V_i) - \sum_{i=1}^{n} \frac{V_i e^{\varphi_y^T V_i}}{(1 + \sum_{j=1}^{C-1} e^{\varphi_y^T V_i})} \quad , \varphi_1, \ldots, \varphi_{C-1} \in \mathbb{R}$$

Données à la sortie

A l'arrêt de l'algorithme, on dispose d'un estimateur $\hat{\theta} = (\hat{I}, \hat{\psi})$ (ou $\hat{\theta} = (\hat{I}, \hat{\varphi})$ dans la paramétrisation logit).

Comme pour l'EM, l'EM logit calcule le groupe le plus probable pour chaque vol i grâce à la fonction de vraisemblance $p_\theta(x, y|v)$ et aux paramètres précédemment optimisés. Elle est appliquée aux $X_i(s, t)$ afin de déterminer pour chaque vol i l'étiquette $\hat{Y}_i$.

$$\hat{Y}_i = \arg\max_y \mathbb{P}_\theta(Y_i = y|V_i) = \arg\max_y \psi(y|V_i)$$

Il est aussi possible après la collecte de quelques point de rajouter l'information des premières variations dans le calcul de la vraisemblance. On définit alors $X_i = (\tilde{X}_i, \hat{X}_i)$ avec $\tilde{X}_i$ les points

observés, et $\hat{X}_i$ les futures évolutions. On aurait donc pour tout événement $\hat{\varphi}$ dépendant de $\hat{X}_i$:

$$\mathbb{P}_\theta(\hat{\varphi}|V_i, \tilde{X}_i) = \sum_{y=1}^{C} \mathbb{P}_\theta(\hat{\varphi}|V_i, \tilde{X}_i, Y_i = y)\mathbb{P}_\theta(Y_i = y|V_i, \tilde{X}_i)$$

$$= \sum_{y=1}^{C} \mathbb{P}_\theta(\hat{\varphi}|Y_i = y)\mathbb{P}_\theta(Y_i = y|V_i, \tilde{X}_i)$$

avec

$$\mathbb{P}_\theta(Y_i = y|V_i, \tilde{X}_i) = \frac{p_\theta(\tilde{X}_i|y)\psi(y|V_i)}{\sum_{j=1}^{C} \mathbb{P}_\theta(\tilde{X}_i|j)\psi(j|V_i)} \propto p_\theta(\tilde{X}|y)\psi(y|V_i).$$

Dans ce cas, les métriques d'évaluation de la segmentation de l'EM peuvent être utilisées en même temps que les performances en prédiction.

Choix des attributs

Pour limiter le temps de calcul et l'utilisation de ressources, nous devons réduire le nombre d'attributs utilisé. Pour cela nous utilisons la technique de sélection de variables précédemment décrite et choisissons les 10 variables les plus discriminantes. Nous devons ensuite transformer les variables qualitatives en des variables quantitatives. Pour cela nous créons une variable par niveaux existant, créant ainsi un grand nombre de variables binaires : dans le cas de la compagnie aérienne, nous allons créer une variable booléenne nommée "*Air France*", puis une autre nommée "*EasyJet*" etc.

4.7 Conclusion et perspectives

Dans ce chapitre nous avons détaillé les différentes étapes qui nous permettent de fournir une prédiction d'évolution de prix à tous les résultats d'une recherche utilisateur. La première étape consiste à associer à chaque résultat sa classe la plus probable : après avoir segmenté les comportements des vols de la base des niveaux de gris, nous avons appliqué un algorithme d'apprentissage supervisé sur la bases des attributs. Cette base est constituée de l'ensemble de caractéristiques des vols V_i et l'étiquette associée E_i correspondant à la classe attribuée lors de l'étape de segmentation. L'algorithme d'apprentissage supervisé va alors créer des régles de classification nous permettant de classer les vols de test dans leur groupe le plus probable uniquement grâce à leurs attributs.

Pour cela, nous avons donc utilisé des algorithmes bien connus tels que CART et C4.5, des arbres de décision permettant l'extraction de règles faciles à interpréter, Adaboost une méthode de boosting basé sur la sélection itérative de classifieurs faibles et les Random Forests, mélange des sous-espaces aléatoires et du Bagging qui effectue un apprentissage sur de multiples arbres de décision entraînés sur des sous-ensembles de données tirés aléatoirement et tous différents. Nous avons par ailleurs introduit une extension de l'algortithme EM défini dans le chapitre précédent, permettant d'effectuer l'étape de segmentation et de classification simultanément.

Nous obtenons donc pour chaque vol de test, une étiquette E_i à laquelle est associé le centroïde du cluster en question. Grâce aux simulations (Chapitre 2) issues de ce centroïde nous calculons alors les probabilités $\mathbb{P}(\varphi_t^{(i)} = 1), \forall t \in [T_{init}, T_0 - 7]$ (si l'on considère une prédiction à 7 jours). Il faut alors définir un seuil au-delà duquel $\mathbb{P}(\varphi_t^{(i)} = 1)$ est considéré comme une prédiction à la hausse ou à la baisse.

Nous avons par ailleurs introduit la notion de prédiction directe qui consiste à prédire directement l'évolution de la série, c'est-à-dire $\mathbb{P}(\varphi_t = 1)$, sans passer par l'étape de segmentation en comportements similaires. Cette méthode impliquant un modèle par jours avant la date de départ devient coûteuse en temps de calcul, mais impose surtout le renouvellement de l'apprentissage à chaque changement de définition de φ_t.

Enfin nous avons défini les différentes approches possibles quant à l'utilisation des premiers points dans l'affinement de la prédiction. Tout d'abord, nous avons utilisé les attributs contextuels décrit dans le chapitre 1 comme attributs dans la prédiction directe. Les informations sur la demande à l'instant t, sur la volatilité des premières variations de prix et la valeur actuelle du prix permettent d'affiner l'apprentissage supervisé. Ensuite, nous avons montré comment il était possible d'utiliser l'information des premiers points pour créer un niveau de gris partiel et calculer la vraisemblance partielle d'appartenance aux clusters.

Toutes ces méthodes impliquent d'avoir un modèle par jour avant la date de départ et donc sont consommatrices de ressources (temps de calcul et mémoire). En revanche, elles permettront d'estimer l'utilité d'une telle information et de comparer les résultats obtenus à ceux de l'approche par les modèles.

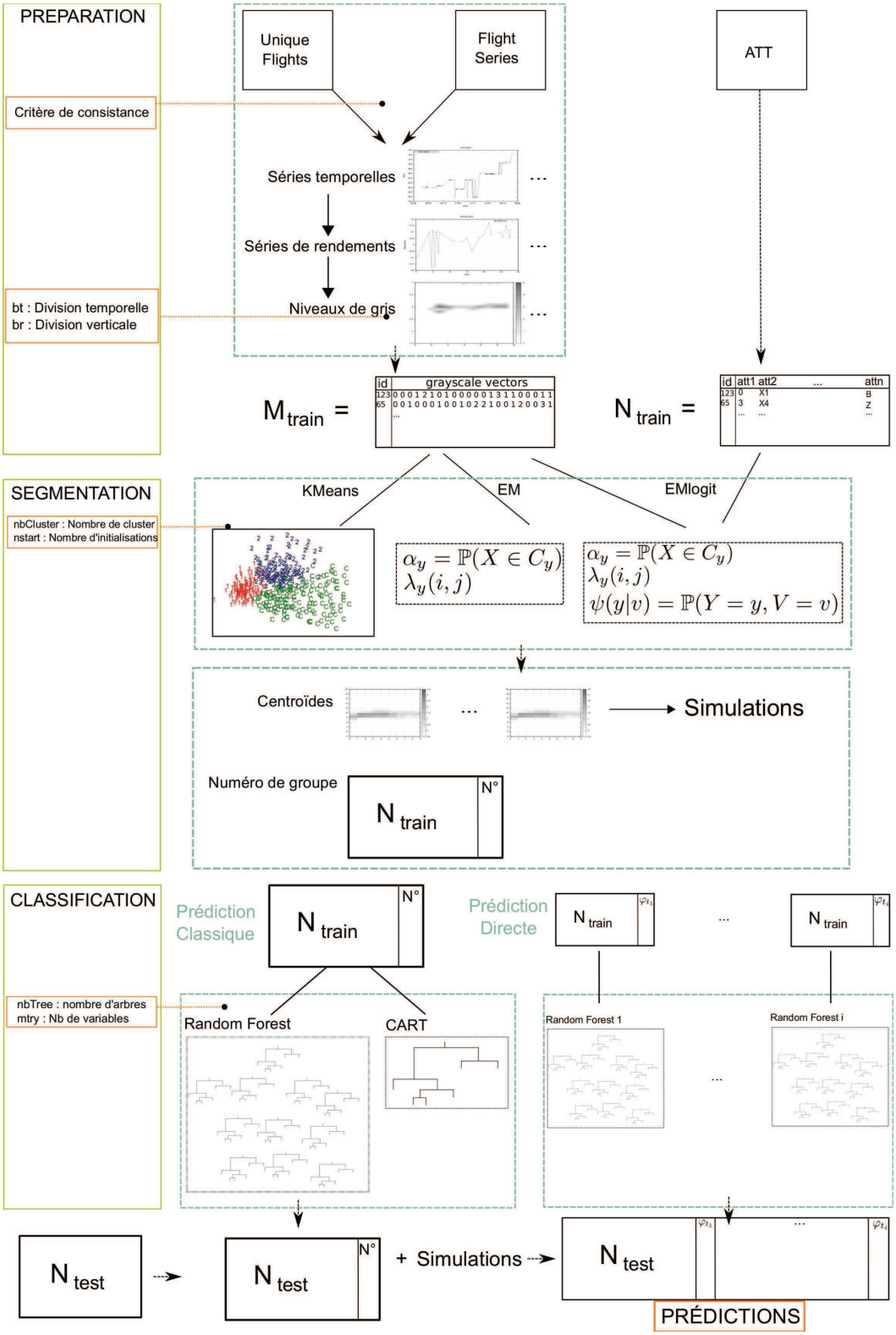


FIGURE 4.5 – Workflow - Vue Détaillée

Chapitre 5

Résultats expérimentaux

Contents

Introduction

Ce chapitre est consacré à l'étude empirique des performances de nos différentes approches et se décompose en deux parties. Dans la première, nous comparons nos différents algorithmes et l'influence de leurs paramètres sur la prédiction. Nous présentons tout d'abord nos résultats en terme de bonnes prédictions des comportements : pour une prédiction binaire ($\varphi_t \in [0, 1]$), par exemple la prédiction d'une hausse ou d'une baisse à 7 jours, nous utilisons les notions de Vrais Positifs, Vrais Négatifs, Faux Positifs et Faux Négatifs, respectivement bonne détection d'une baisse, bonne détection d'une hausse, mauvaise détection d'une baisse et mauvaise détection d'une hausse. Nous comparons d'abord ces quatre critères grâce aux matrices de confusion pour s'assurer de la bonne prédiction à la fois des vols à la hausse (en majorité) et des vols à la baisse. Nous voulons notamment minimiser le nombre de vols à la hausse prédit à la baisse (les faux positifs) pour faire perdre le moins d'argent possible à l'utilisateur. Nous visualisons ensuite l'évolution du taux de vrais positifs (vols à la baisse correctement identifiés) en fonction du taux de faux positifs (vols à la baisse mal prédis) lorsque le seuil de décision de φ_t varie (seuil au-delà duquel φ_t sera considéré comme une prédiction à la baisse). Pour cela nous utilisons la courbe ROC (de l'anglais Receiver Operating Characteristic) qui nous permet de comparer les performances des différentes approches à un instant t. La courbe ROC nous servira par ailleurs à déterminer le seuil optimal à chaque instant t. Enfin à seuil fixe, nous visualiserons l'évolution du taux de bonnes prédictions en fonction du nombre de jours avant la date de départ. Le seuil est déterminé en effectuant une prédiction sur la base d'apprentissage et en sélectionnant le seuil optimal.

Nous visualisons dans le même temps les gains pécuniaires pour l'utilisateur et nous visualisons parallèlement le pourcentage de bonnes prédictions. La façon de calculer les gains et les pertes d'une prédiction est un point sensible que nous discuterons : en effet, l'erreur à la hausse n'a pas le même impact sur l'utilisateur que celle à la baisse. Il est alors délicat de calibrer l'application afin qu'elle prenne en compte les différentes attentes possibles : un sondage effectué au sein de liligo.com montre qu'il existe plusieurs profils d'utilisateur ayant chacun leur vision du service. Nous verrons donc que la définition d'un vol à la baisse et le seuil à partir duquel nous considérons qu'un vol est à la baisse peuvent être sensiblement différents selon les personnes.

A chaque étape (modélisation, segmentation, classification), les différents paramètres étudiés sont comparés pour choisir la configuration optimale. Pour l'étape de clustering, nous comparons le nombre de clusters, le nombre d'initialisations pour toutes les approches précédemment citées : K-Means, Bagged K-Means et EM. Pour l'étape d'apprentissage, nous comparons les différents algorithmes (CART, Adaboost, Forets d'arbres décisionnels et EMlogit) et discutons de l'influence des attributs utilisés pour la classification. Nous vérifions ensuite que l'étape de "feature selection" permet d'optimiser les résultats en éliminant les attributs corrélés ou inutiles.

Dans un deuxième temps, une fois les configurations de chaque algorithme fixées, nous nous intéressons à l'influence de la base d'apprentissage et du paramétrage des niveaux de gris dans les performances : il est importante de savoir s'il est nécessaire d'utiliser l'intégralité de la base de liligo.com pour prédire correctement ou si les 10 derniers pourcents suffisent, permettant ainsi d'alléger le processus de création du modèle. De même, nous vérifions si le degré de généralisation des comportements par l'affinement ou non des niveaux de gris a une influence sur le taux de

bonnes prédictions.

Nous discutons de l'évolution du taux de bonnes prédictions dans le temps afin de connaître la fréquence de renouvellement des modèles. Nous décrivons les effets de l'ajout de l'information des premiers sauts dans la prédiction, et nous comparons l'ensemble de nos résultats à la prédiction directe. Enfin nous décrivons les résultats de notre approche sur la base étendue des vols à 90 jours.

5.1 Mesures de performance

5.1.1 Matrice de confusion

La matrice de confusion est une métrique qui mesure la qualité d'un système de classification. Habituellement utilisée dans le cas d'une prédiction binaire, elle peut être étendue à un nombre quelconque de classes. Dans le cas d'une prédiction binaire, il faut définir 4 valeurs liées aux décisions prises par rapport à la réalité : les vrais positifs (True Positive, TP), faux positifs (False Positive, FP), vrai négatifs (True Negative, TN) et faux négatifs (False Negative, FN).

Dans notre cas, $\varphi = 1$ correspond au conseil "attendre avant d'acheter", c'est-à-dire à l'évolution à la baisse quelque soit le critère (prix inférieur à 7 jours, prix plus bas pendant 24 heures dans les 5 jours, etc.). Nous considèrons que l'événement $\varphi = 1$ est l'événement positif et $\varphi = 0$, conseil à l'achat, celui négatif. Le nombre de vrais positifs est alors le nombre de fois que le prédicteur à prédit une baisse de prix correctement. De la même manière, les vrais négatifs sont le nombre de vols à la hausse correctement identifiés. Enfin le faux positif sera une prédiction à la baisse alors qu'en réalité le prix a augmenté et inversement un faux négatif sera un vol à la baisse prédit à la hausse.

La matrice de confusion est donc le résumé de ces 4 valeurs en un tableau comme celui-ci :

		Réalité	
		0	1
Prédit	0	TN	FP
	1	FN	TP

Table 5.1 – Matrice de confusion

avec en ligne le label prédit et en colonne le label réel.

Un des intérêts de la matrice de confusion est qu'elle montre rapidement si le système parvient à classifier correctement. Dans notre cas, elle sera toujours déséquilibrée en raison d'un plus faible nombre de baisses de prix, et l'importance d'un faux négatif ne sera pas la même que celle d'un faux positif. Un vol prédit comme haussier qui s'avère être à la baisse ne fera pas perdre d'argent à l'utilisateur. L'inverse en revanche implique une perte réelle et une frustration plus importante. Il est donc important de minimiser le nombre de faux négatifs.

5.1.2 Évolution dans le temps

Grâce aux matrices de confusion, nous sommes capables de visualiser le pourcentage global de bonnes prédictions évoluer au cours du temps. Nous définissons alors le taux global de bonnes prédictions :

$$f_{global}(t) = \frac{TP + TN}{TP + TN + FP + FN}$$

mais aussi la spécificité, ou la probabilité de détecter une hausse correctement :

$$f_{spec}(t) = TN/(FP + TN)$$

et enfin la sensibilité, c'est-à-dire la proportion de baisses détectées parmi les vols à baisse :

$$f_{sens}(t) = TP/(TP + FN)$$

Nous visualisons l'évolution de ces critères en fonction du nombre de jours avant la date de départ. Il est important d'étudier ces trois courbes simultanément car une bonne prédiction globale n'assure pas un équilibre dans les prédictions à la hausse et à la baisse : le pourcentage de vols à la hausse étant d'environ 70%, un classifieur strictement haussier serait alors considéré comme performant alors même qu'il ne détectera jamais de baisse. L'enjeu est donc de conserver un bon taux de bonnes prédictions global tout en équilibrant la détection des hausses et des baisses. Pour cela nous allons utilisé la notion de seuil de décision. En effet ces courbes dépendent du seuil $s \in [0, 1]$ au-delà duquel $\mathbb{P}(\varphi = 1)$ est considéré comme une baisse. Nous rappelons qu'à la suite d'une prédiction de cluster, un vol se verra associer une probabilité d'évolution à un instant t calculée par simulation. Pour $s = 0.5$, si $\mathbb{P}(\varphi = 1) >= 0.5$ nous prédisons un baisse, sinon une hausse. Pour toutes les valeurs de $s < 0.5$, nous favorisons les prédictions à la baisse et inversement pour tout $s > 0.5$ les prédictions seront plus souvent à la hausse.

Chaque courbe correspond donc à l'évolution des bonnes prédictions en fonction de la date avant le départ et pour un seuil fixé. Nous verrons comment il est possible de choisir le seuil optimal grâce aux courbe ROC et donc de créer une fonction $f(s, t)$ afin d'optimiser le seuil à chaque jour.

5.1.3 Courbe ROC

La courbe ROC (Receiver Operating Characteristic) mesure l'évolution de la performance d'un classifieur binaire en fonction d'un seuil de discrimination. Dans notre cas, le seuil est celui que nous venons de décrire. Fixer un seuil élevé, par exemple 0.8, correspond à un comportement conservateur, ne prenant pas de risque face à l'erreur à la hausse. Inversement, lorsque le seuil est faible, 0.3 par exemple, la prédiction à la baisse sera privilégiée.

La courbe ROC peut se formuler ainsi :

$$ROC = \frac{\mathbb{P}(x|Positif)}{\mathbb{P}(x|Negatif)}$$

avec $\mathbb{P}(x|C)$ la probabilité conditionnelle que le vol d'entrée fasse partie de la classe C

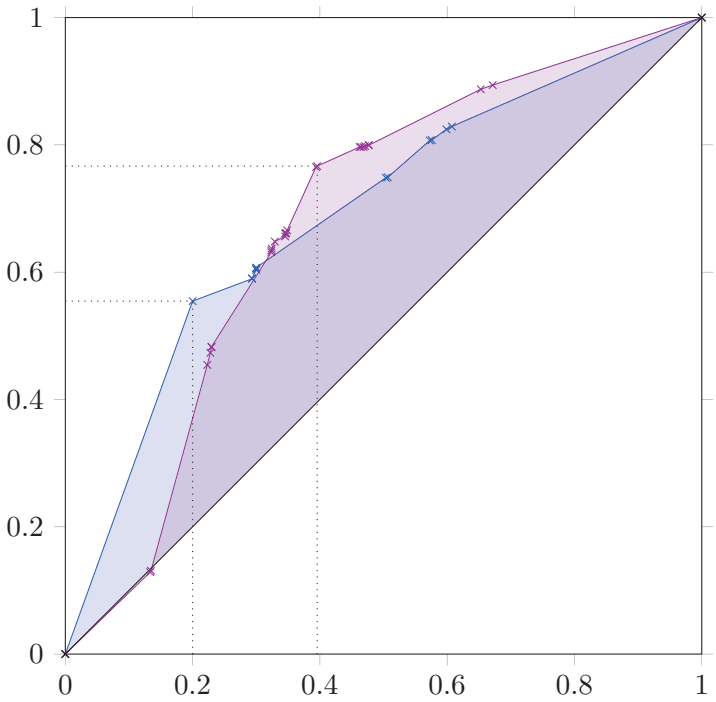


FIGURE 5.1 – Comparaison de deux courbe ROC

Sur la Figure 5.1 nous montrons un exemple de mise en compétition de deux approches par leur courbe ROC. En fonction du problème posé, différents critères de sélection peuvent être utilisés : aire sous la courbe (AUC), le meilleur compromis entre sensibilité et spécificité etc. Dans notre cas, nous voulons minimiser l'erreur à la hausse (prédiction à la baisse alors qu'en réalité le prix va augmenter) et donc minimiser le nombre de FP. La meilleur performance à la hausse tout en minimisant les FP est donc sur la courbe bleue aux coordonnées $(0.2, 0.55)$.

Il est possible de choisir la valeur optimal $\hat{s}_t$ au temps t en utilisant l'index de Youden [54] ou la distance à la diagonale qui sont au final les mêmes mesures :

La distance à la diagonale qui mesure l'endroit de la courbe ROC où celle-ci est la plus éloignée de la diagonale.

$$\frac{\sqrt{2}(f_{sens}(t) - (1 - f_{spec}(t)))}{2} = \frac{\sqrt{2}}{2}(f_{sens}(t) + f_{spec}(t) - 1);$$

L'index de Youden L'index de Youden évalue la capacité du classifieur à éviter l'erreur de prédiction et mesure de manière égale les performances de prédictions à la hausse et à la baisse.

$$f_{sens}(t) + f_{spec}(t) - 1$$

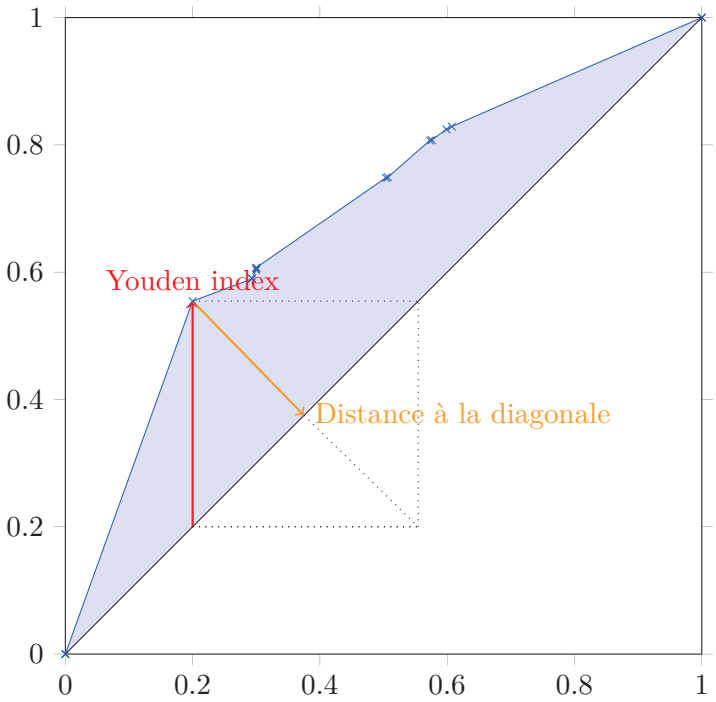


FIGURE 5.2 – Illustration du lien entre la distance à la diagonale et l'index de Youden

Il est à noter que dans notre cas il y aura autant de points d'inflexions que de nombre de clusters : en effet pour minimiser le temps de calcul nous n'effectuons les N simulations qu'une seule fois par cluster. Il en résulte que deux vols de test assignés au même cluster auront le même $\mathbb{P}(\varphi_t = 1)$.

Évolution des gains et pertes : définition

De la même manière que nous avons construit une matrice de confusion des prédictions nous pouvons créer une matrice de confusion des gains et pertes. Pour chacun des quatre cas de la matrice de confusion classique, nous définissons un gain ou une perte pécuniaire comme suit : soit $d = firstPrice - lastPrice$, avec $firstPrice$ le prix du billet au moment de la prédiction et $lastPrice$ le prix réel à la fin de la fenêtre de prédiction. Nous définissons alors une nouvelle matrice de confusion :

		Réalité	
		0	1
Prédit	0	$G_{TN} = -d$	$P_{FN} = d$
	1	$P_{FP} = -d$	$G_{TP} = d$

TABLE 5.2 – Matrice de confusion des gains et pertes

On affichera ensuite 3 courbes qui seront $g_{global} = \frac{\sum_n G_{TP} - G_{TN} + P_{FP} - P_{FN}}{n}$ pour évaluer le gain ou la perte moyenne par vol, $g_{gain} = \frac{\sum G_{TP} - G_{TN}}{n}$ pour évaluer le gain moyen par vol et $g_{perte} = \frac{\sum -P_{FP} + P_{FN}}{n}$ pour la perte moyenne par vol. Nous verrons que l'évolution de ces courbes n'est pas nécessairement similaire à celles des bonnes prédictions. Nous avons en effet montré dans le Chapitre 1 que les valeurs moyennes de baisse étaient inférieures à celles des hausses, rendant ainsi la mauvaise prédiction à la hausse plus coûteuse.

Nous introduisons par ailleurs deux courbes comparatives qui sont le gain effectué en achetant systématiquement au jour de la prédiction, i.e. $\forall t, \mathbb{P}(\varphi = 1) = 0$, $g_0 = \frac{\sum_n -G_{TN} - P_{FP} - G_{TN} - P_{FN}}{n}$ ainsi qu'une courbe optimum qui décrit le gain pour quelqu'un qui connaîtrait l'avenir, c'est-à-dire si l'on fait systématiquement le bon choix, $g_{opt} = \frac{\sum_n G_{TP} + P_{FN} - G_{TN} - P_{FP}}{n}$. La différence positive entre le gain effectué en suivant la prédiction et le gain à l'achat immédiat permet d'affirmer qu'il est parfois judicieux d'attendre et valide ainsi la pertinence de notre service. La distance à la courbe du gain optimum sera alors le critère à améliorer.

Nous devons donc trouver un second équilibre entre l'optimisation du taux de bonnes prédictions et l'augmentation du gain moyen par billet.

5.2 Résultats

Dans cette section nous allons étudier l'influence des paramètres et des algorithmes sur la qualité des prédictions. Nous observerons pour chaque étape les métriques spécifiques associées mais aussi les mesures de performance précédemment décrites. Nous commencerons par étudier les paramètres de la segmentation des données (nombre de groupes, initialisations), puis nous étudierons ceux de l'étape de classification et enfin nous analyserons l'influence des paramètres d'entrée (taille de la base, dimension des pixels,...).

Nous nous focalisons tout d'abord sur notre base de vols échantillonnés toutes les 6 heures sur une période de 28 jours. Cette base d'apprentissage s'arrête au 1er janvier 2013 représentant 444503 vols depuis juin 2011 et la base de test est constituée des points compris entre le 1er janvier 2013 et le 28 février 2013 correspondant à 19848 vols. Pour éviter des temps de calcul trop importants, nous avons réduit la base d'apprentissage de moitié. Nous conservons ainsi un nombre de vols suffisant pour représenter un large ensemble de comportements et nous conservons la dernière moitié pour que ces comportements soient actuels. Nous étudierons dans un second temps notre base étendue à 90 jours, et enfin nous appliquerons nos algorithmes sur une base totalement indépendante.

5.2.1 Segmentation

K-Means

Dans le Chapitre 3, nous avons décrit les différents paramètres de l'algorithme des K-Means : le nombre de départs aléatoires permet de réduire les chances de converger vers un minimum local, et le nombre de groupes représente le nombre de comportements que nous souhaitons modéliser. Ce dernier paramètre doit coller à une certaine réalité mais le nombre de comportements

présents dans notre base de données n'existe pas réellement.

Il est important de différencier l'influence des paramètres sur la qualité de la segmentation et sur la prédiction finale, car ces dernières ne sont pas nécessairement corrélés. Nous afficherons donc pour chaque résultat les deux critères de performance. Nous ajouterons lorsque cela pertinent, l'évolution de la somme des gains et des pertes en fonction du nombre de jours avant la date de départ. En effet, nous devons nous soucier de taux de bonnes prédictions mais aussi du gain global apporté à l'utilisateur. Un bon taux de bonnes prédictions n'implique pas une bonne qualité dans les prédictions. C'est ce que nous verrons dans les graphes suivants.

Un des critères de performance des K-Means est la somme des distances des niveaux de gris $X_i(s,t)$ au centroïde $I(s,t)$ à l'intérieur de chaque partition, décrit dans le chapitre 3 :

$$w_y = \sum_{X_i \in y} \|X_i - I_y\|^2$$

$$W = \sum_{i=y}^{C} w_y$$

C'est le critère qu'optimise les K-Means et nous l'utilisons donc dans un premier temps pour le choix des paramètres des K-Means.

Évolution des performances en fonction du nombre de départs Le nombre de départs aléatoires est essentiel dans le bon déroulement de l'algorithme. Un mauvais point de départ peut faire converger les K-Means vers un minimum local mais pas forcement l'optimal. Il est donc nécessaire de multiplier le nombre de départs et faire dérouler l'algorithme pour au final choisir le meilleur point de départ. Sur la Figure 5.3, nous observons la variance de W sur plusieurs itérations en fonction du nombre de départs initiaux.

On constate une diminution de la variance de W avec le nombre de départs en même temps qu'une diminution de la valeur moyenne de W c'est-à-dire de la variance intra-centroïde, représentée par les droites horizontales. La probabilité de converger vers l'optimal global augmente donc avec le nombre de départs aléatoires.

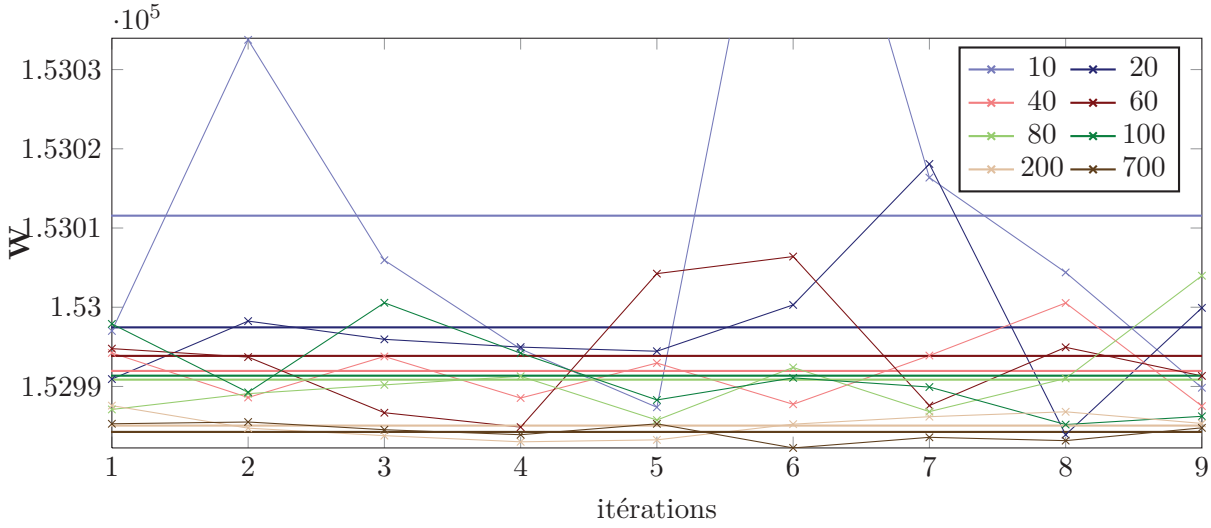


FIGURE 5.3 – Différentes itérations avec un nombre de départs différent

Cependant le temps de calcul et les ressources utilisées par les K-Means sont un frein au choix d'un grand nombre de départs. Sur la Figure 5.4, nous constatons que le temps de calcul est quasiment linéairement corrélé au nombre de départs, impliquant un compromis dans le choix de la valeur de *nstart*.

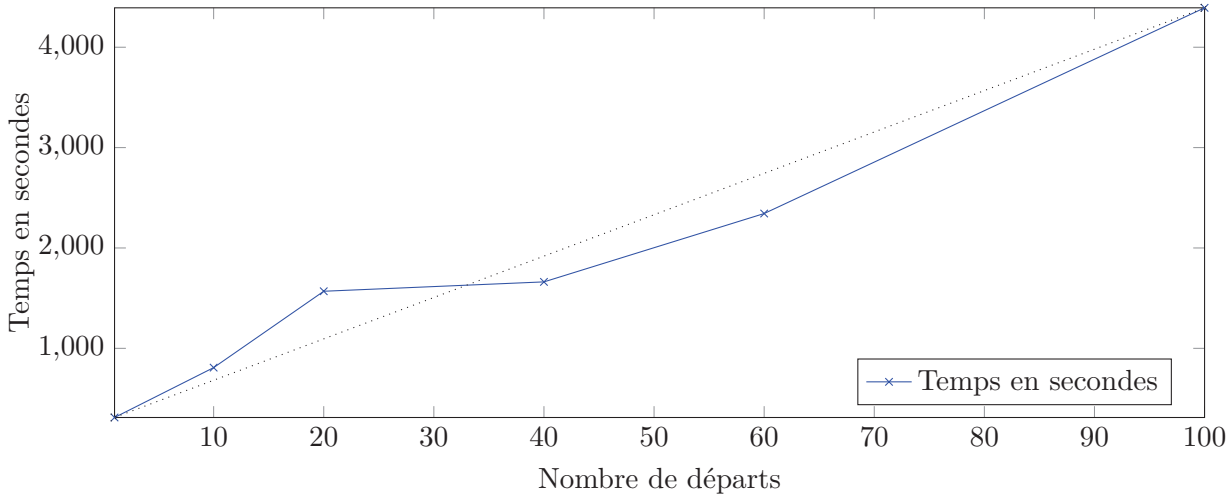


FIGURE 5.4 – Évolution du temps d'exécution des K-Means sur la moitié de la base d'apprentissage pour différents *nstart* (30 clusters)

En ce qui concerne la prédiction finale, l'influence du nombre de départs est minime et s'estompe à partir d'un certain seuil. La Figure 5.5 nous montre l'influence du paramètre *nstart* sur l'évolution du pourcentage de bonnes prédictions en fonction de la date avant le départ et sur les courbes ROC associées à $t = -21$. Une différence minime est visible entre 10 départs et

100 départs, alors qu'aucune différence n'est observée entre 100 et 1000. Nous décidons donc de poser par défaut les nombre de départs à 100 pour les K-Means.

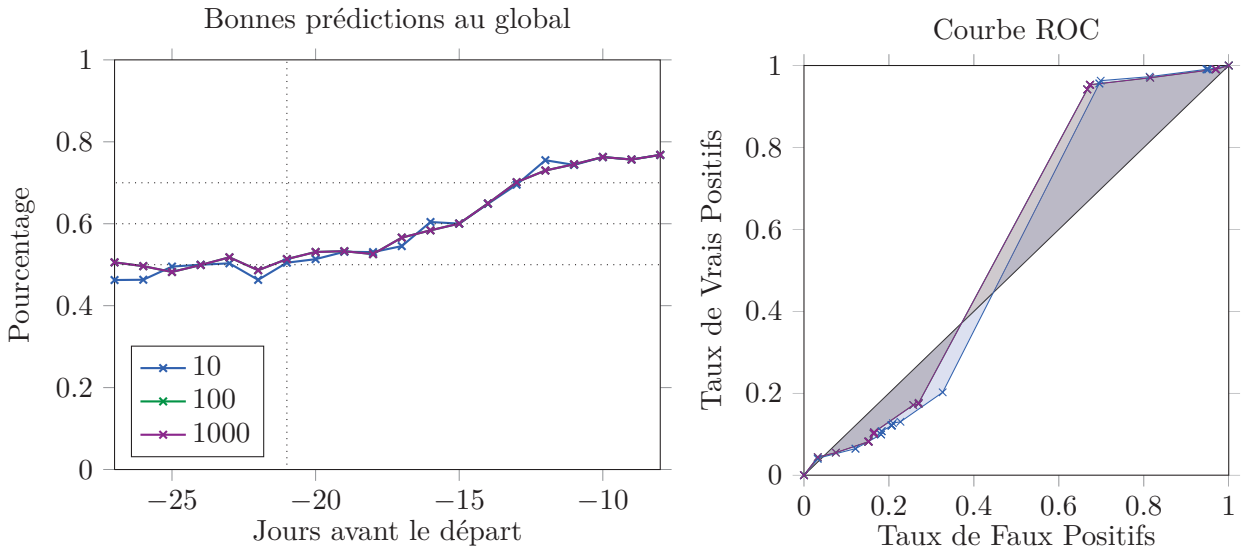


FIGURE 5.5 – K-Means - Évolution du pourcentage de bonnes prédictions en fonction de la date avant le départ et courbe ROC à -21 jours pour différents *nstart*

Evolution des performances en fonction du nombre de groupes Le nombre de clusters correspond à notre estimation du nombre de comportements existants dans la base d'apprentissage. Nous observons sur la Figure 5.13 les évolutions de W (somme des distances intra-centroïdes et critère qu'optimise les K-Means) et de l'indice de Calinski-Harabasz (métrique décrite dans le chapitre 3). Ces deux indicateurs de qualité de segmentation sont strictement décroissants et nous donnent peu d'informations sur le meilleur nombre de groupes à choisir. L'évolution de l'indice de Rand (aussi décrit dans le chapitre 3) donnant les mêmes résultats, nous n'avons pas ajouté sa trajectoire à la Figure 5.13.

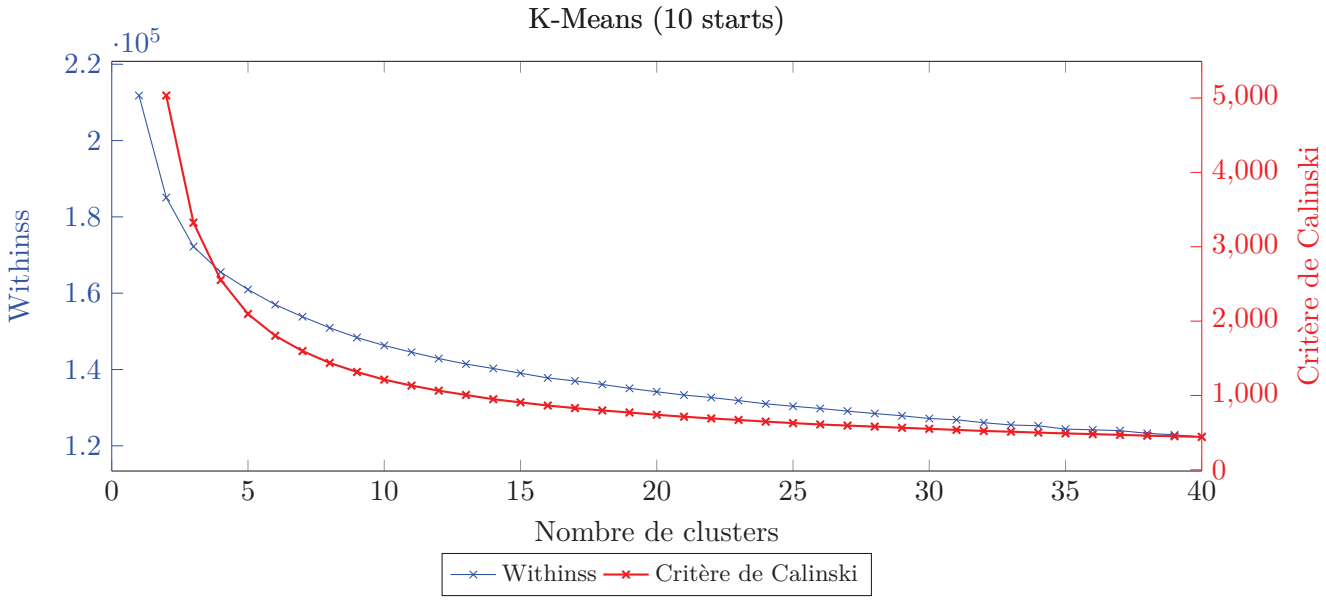


FIGURE 5.6 – Évolution de la somme des distances aux centroïdes et du critère de Calinski par rapport au nombre de clusters

En ce qui concerne les prédictions finales, les K-Means semblent conserver un taux de bonnes prédictions stable à partir de 10 groupes avec des courbes ROC à -21 jours similaires (Figure 5.7). Les différences observées sur l'évolution du taux de bonnes prédictions et sur la somme des gains et des pertes sont dues au choix du seuil différenciant la prédiction d'une hausse de la prédiction d'une baisse. L'ajout de groupes va uniquement créer des groupes très proches multipliant le nombre de points très proches sur la courbe ROC sans multiplier ses inflexions.

Il est à noter que le taux de bonnes prédictions peut-être ici trompeur car, toujours sur la Figure 5.7, le choix de 40 groupes affiche le meilleur taux de bonnes prédictions alors même qu'il fait perdre de l'argent à l'utilisateur. En effet, l'augmentation des prédictions se fait en réalité sur la détection des baisses au détriment des hausses. La manière dont nous avons défini l'événement "wait", correspondant aussi bien à une baisse qu'à un prix stable et observant que les variations à la hausse ont souvent un rendement plus important, l'augmentation des bonnes prédictions à la baisse peut faire augmenter le prix moyen par billet acheté.

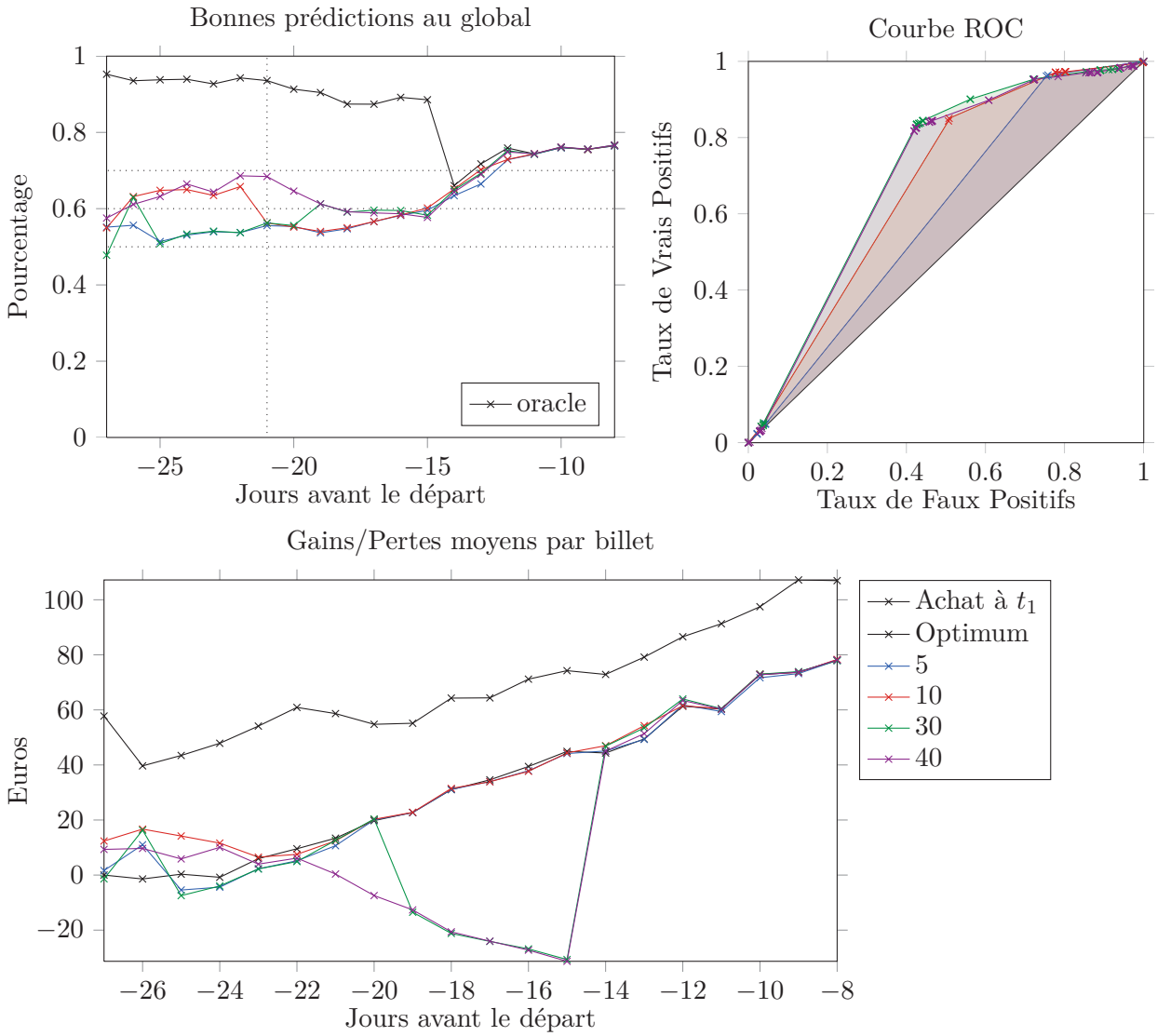


FIGURE 5.7 – K-Means - Évolution du pourcentage de bonnes prédictions en fonction de la date avant le départ et courbe ROC à -21 jours pour différents nombre de clusters

Enfin observons l'expertise de l'algorithme GAP décrit dans le chapitre 3 sur la Figure 5.8. Cet algorithme est coûteux en temps de calcul et en ressource, c'est pourquoi nous l'avons appliqué aux 5 derniers pourcents de la base d'apprentissage, pour un nombre de groupes allant de 1 à 50. Différentes métriques existent alors pour choisir le meilleur nombre de cluster :

1. La valeur maximale de GAP_k. Ici **50**.
2. La première occurence d'un maximum local. Ici **27**.
3. Le plus petit k pour lequel $f(k) \geq f(k+1) - s_{k+1}$ (Tibshitani et al. dans [48]). Ici **27**.

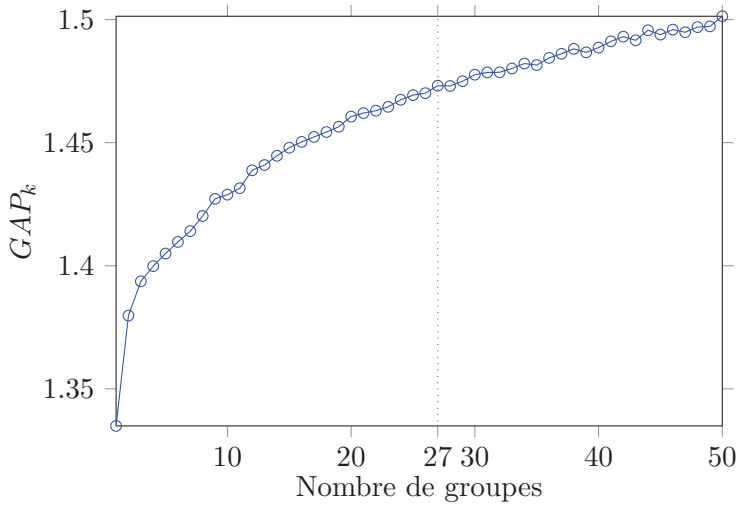


FIGURE 5.8 – Evolution de l'indice GAP en fonction du nombre de groupes (K-Means)

Les K-Means appliqués aux 10 derniers pourcents de la base d'apprentissage, pour 27 groupes donnent effectivement de bons résultats au vue de la courbe ROC mais tendent à prédire correctement les baisses au détriment des hausses donnant ainsi un faible taux de bonnes prédictions global et des pertes pour l'utilisateur. Nous appliquons ce choix aux algorithmes "EM" et "Bagged K-Means" sur la Figure 5.9. Les trois premiers cadrans représentent respectivement l'évolution du taux global de bonnes prédictions, celui des bonnes prédictions à la baisse et celui de bonnes prédictions à la hausse en fonction du nombre de jours avant la date de départ. Puis nous comparons le gain pour l'utilisateur pour chaque algorithme par rapport au gain global à l'achat immédiat et au choix.

Nous constatons tout d'abord que l'algorithme des K-Means et celui des Bagged K-Means ont des performances similaires et un taux de prédiction global quasiment égal (respectivement 59% et 58%). L'EM quant à lui, ne suit pas la même trajectoire mais atteint 60% de bonnes prédictions au global malgré une aire sous la courbe ROC à 21 jours bien moins importante. Cela s'explique par la répartition des bonnes prédictions à la hausse et à la baisse. L'EM détecte mieux les hausses qui représentent des variations de prix plus importantes et sont en plus grande quantité. C'est pour cela qu'en gain utilisateur, ce dernier est très proche en prix d'achat d'un achat immédiat, quand les deux autres algorithmes font progressivement perdre de l'argent à l'utilisateur.

L'expertise par la méthode de GAP ne s'applique donc pas à tous les algorithmes et ne fournit pas non plus la valeur optimale à choisir. Elle donne en revanche une idée de l'ordre de grandeur du nombre de groupes minimisant W.

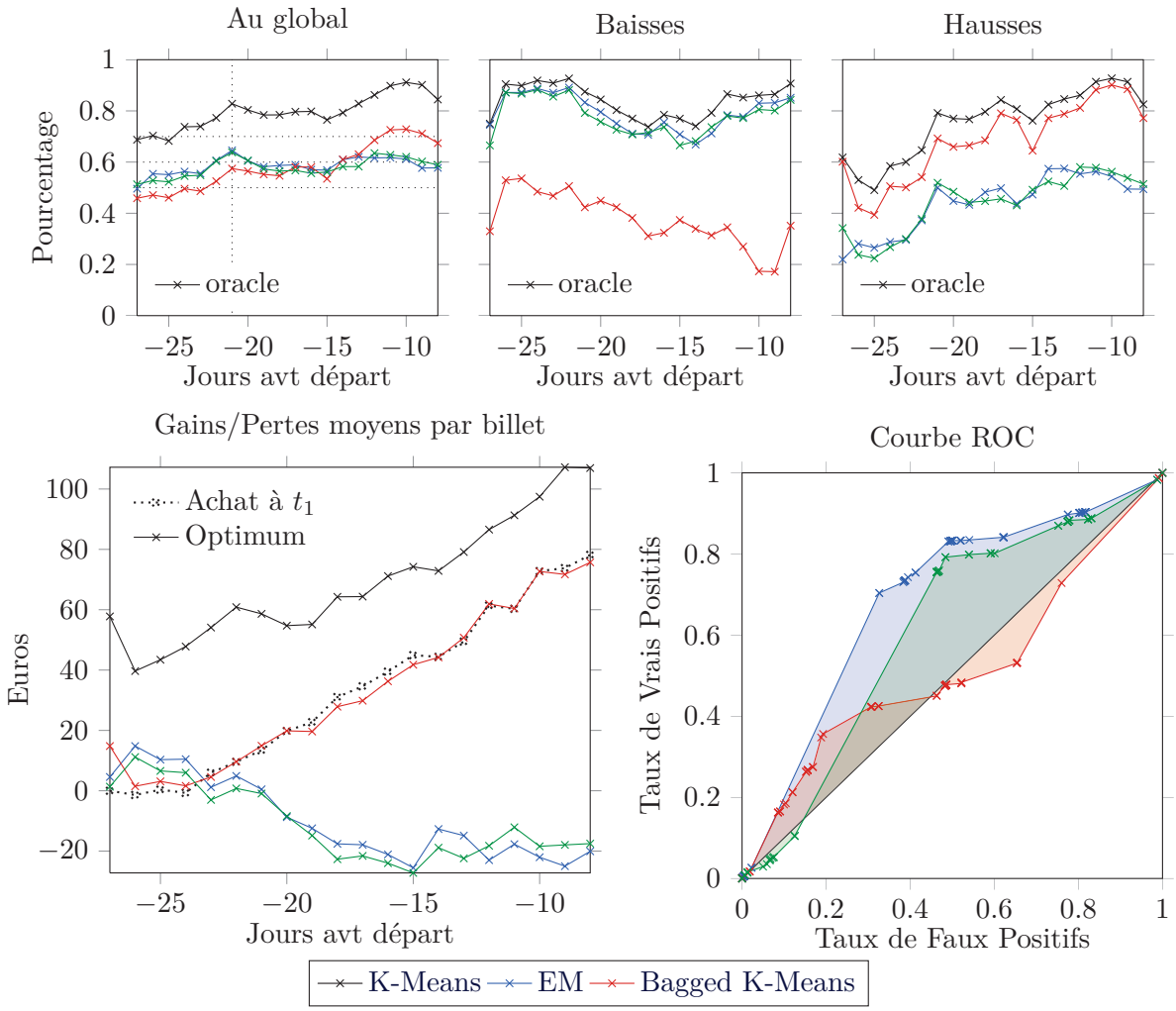
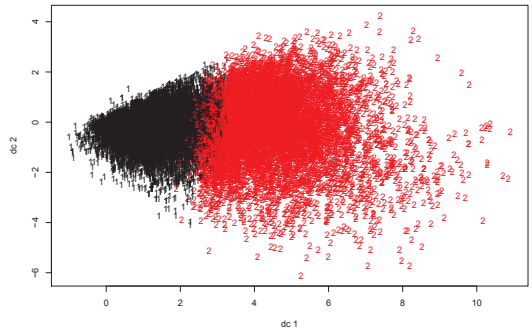


FIGURE 5.9 – K-Means, EM et Bagged K-Means avec 27 groupes sur les 10 derniers pourcents de la base.

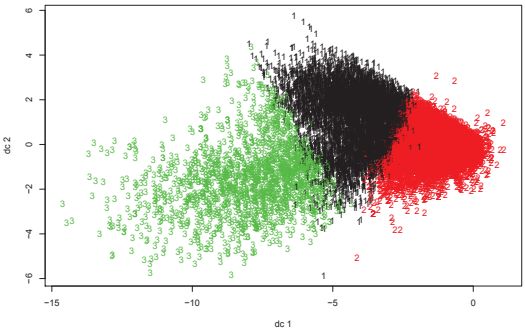
Une visualisation des nuages de points générés par la projection sur l'axe de deux variables issues de l'analyse en composante principale (Chapitre 3) sur la Figure 5.10, nous indique (i) qu'il ne semble pas y avoir de séparation évidente en groupes de comportements et (ii) qu'à partir de 4 clusters, le nuage de points n'est plus divisé en groupes distincts. Dans les faits la courbe ROC semble meilleure pour 4 clusters, mais l'ensemble de ces dernières restent moins bonnes que celle à 27 clusters (Figure 5.11).

Les Bagged K-Means

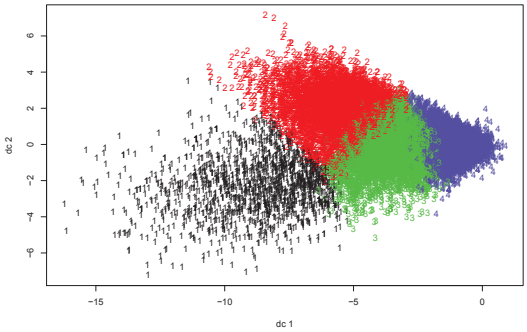
L'algorithme des Bagged K-Means tente de réduire l'influence des l'initialisation des K-Means et de favoriser la convergence vers un optimum global en appliquant l'algorithme des K-Means sur des sous-ensembles tirés aléatoirement avec remise et en agrégeant les résultats de chacune



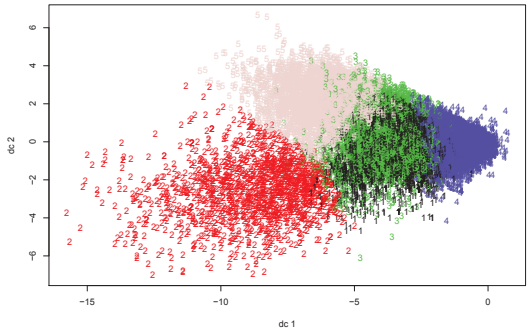
2 groupes



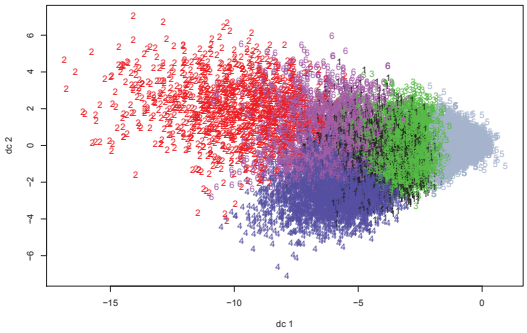
3 groupes



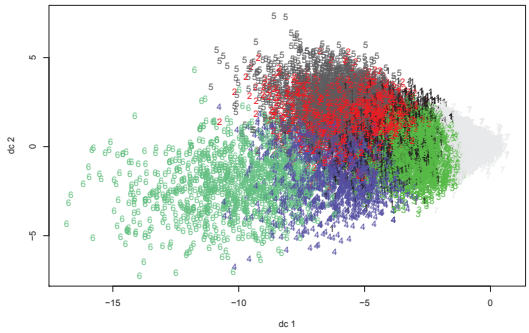
4 groupes



5 groupes



6 groupes



7 groupes

FIGURE 5.10 – Évolutions des nuages de points en fonction du nombre de groupes pour un K-Means sur les 10 derniers pourcents de la base d'apprentissage

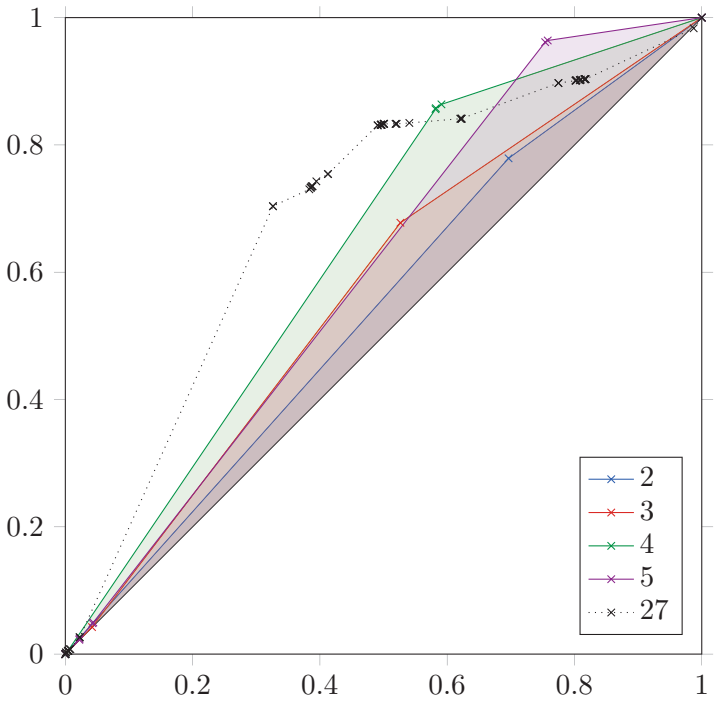


FIGURE 5.11 – Courbe ROC pour un K-Means à 2, 3, 4 et 5 clusters comparées aux K-Means à 27 groupes

de ces segmentations. Il est possible de modifier le nombre d'initialisations des K-Means, mais comme le montre la Figure 5.12, et partant du principe que les Bagged K-Means suppriment l'influence de ce paramètre, le nombre d'initialisations n'influe pas sur les performances. Malgré tout l'algorithme peine à donner de bons résultats et tend à mieux prédire les baisses faisant ainsi perdre de l'argent à l'utilisateur.

Nous ne retenons donc pas cet algorithme et préférons conserver les K-Means pour une question de performance mais aussi de rapidité de calcul.

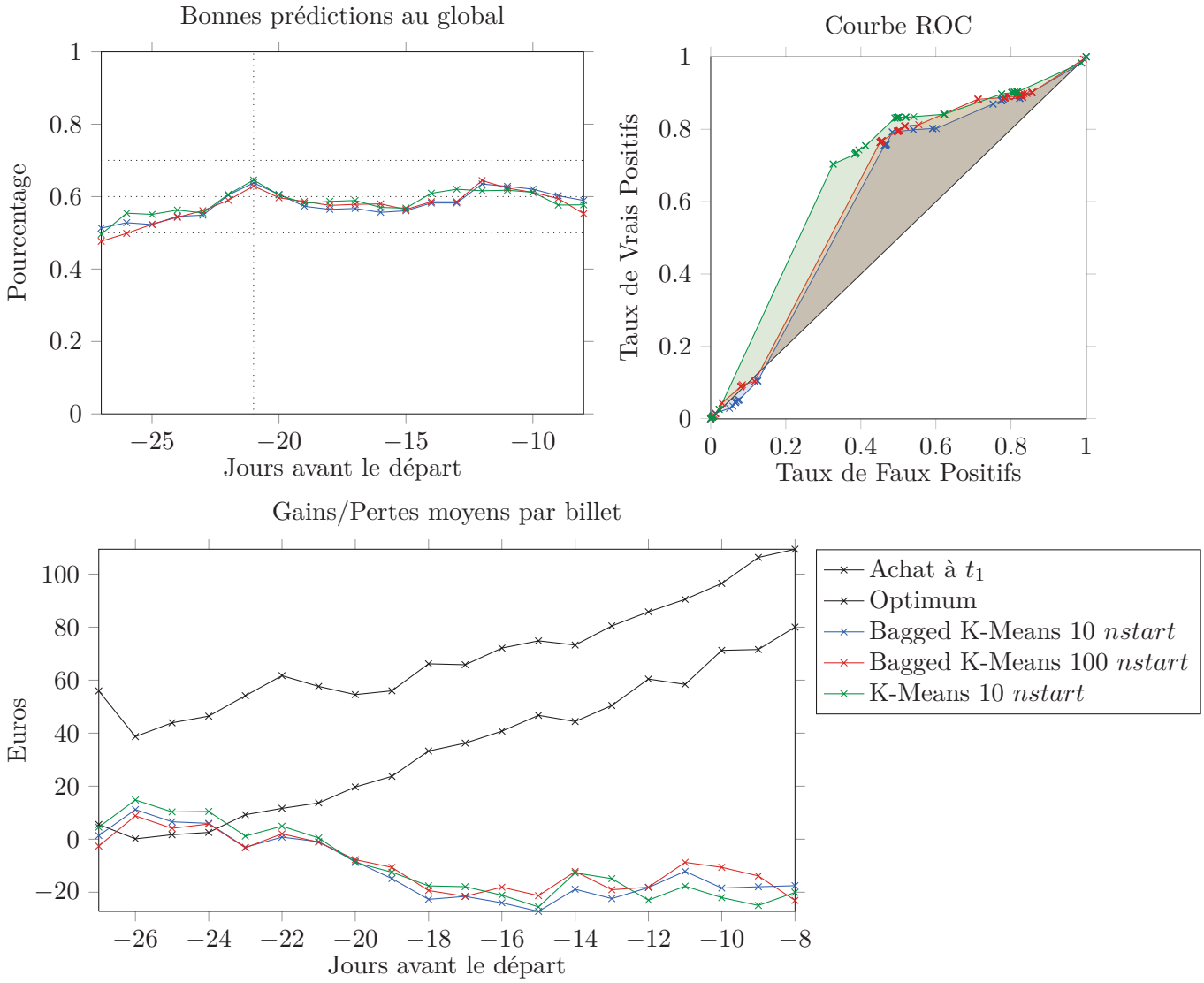


FIGURE 5.12 – Comparaison des Bagged K-Means avec un K-Means

Espérance-Maximisation

Pour l'algorithme d'Espérance-Maximisation, les problématiques sont les mêmes que pour les K-Means : convergence locale, nombre de départs, nombre de clusters, etc. Nous utilisons donc les mêmes métriques afin d'évaluer le paramétrage idéal. Cependant le critère à optimiser n'étant pas le même pour les deux algorithmes (W pour les K-Means et la vraisemblance globale pour l'EM), nous allons principalement comparer leur qualité de prédiction.

Comme pour les K-Means, l'évolution du critère à optimiser (ici la vraisemblance globale) s'améliore avec l'augmentation du nombre de groupes, et l'évolution de W est aussi strictement décroissante (Figure 5.13).

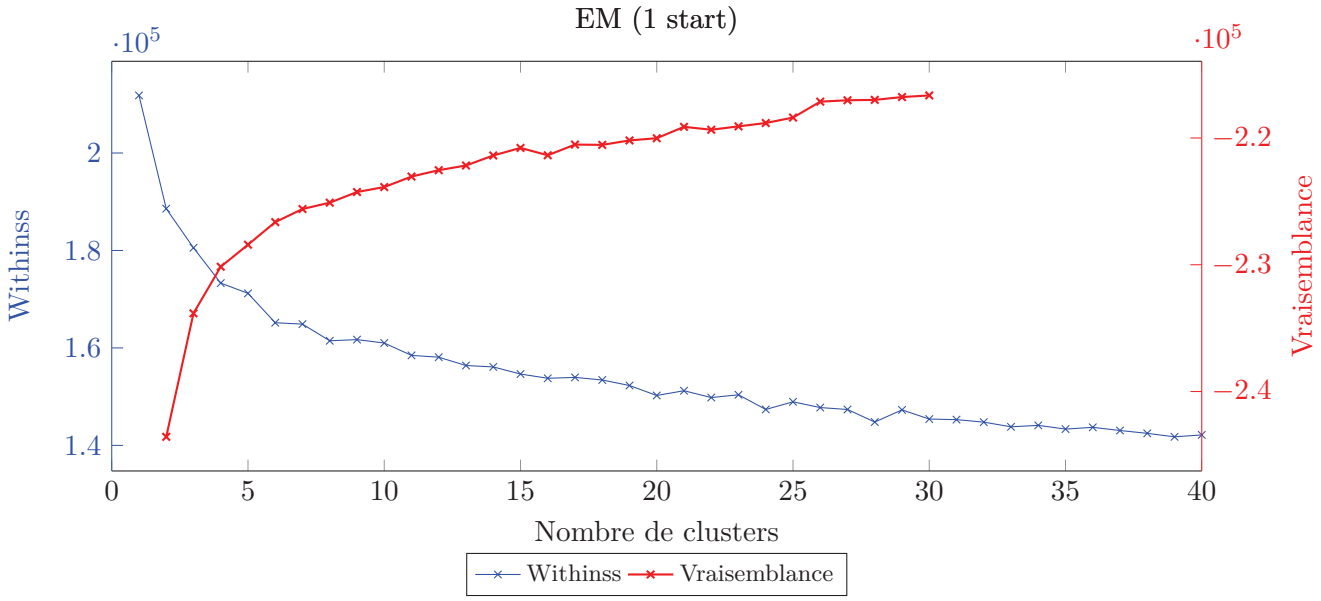


FIGURE 5.13 – Évolution de la somme des distances aux centroïdes W par rapport au nombre de clusters ainsi que de la vraisemblance globale

En revanche, à la différence des K-Means, l'algorithme Espérance-Maximisation est très sensible aux variations de ses paramètres : lorsque le bon nombre de groupes est choisi, les performances en bonnes prédictions face aux K-Means sont meilleures et le gain pour l'utilisateur est significatif, alors que pour un nombre de clusters différent les résultats sont radicalement diminués (Figure 5.14). Il est donc nécessaire de tester différents choix et de sélectionner le nombre de groupes qui donne les meilleurs résultats. Dans notre cas 5 et 30 groupes semblent se démarquer.

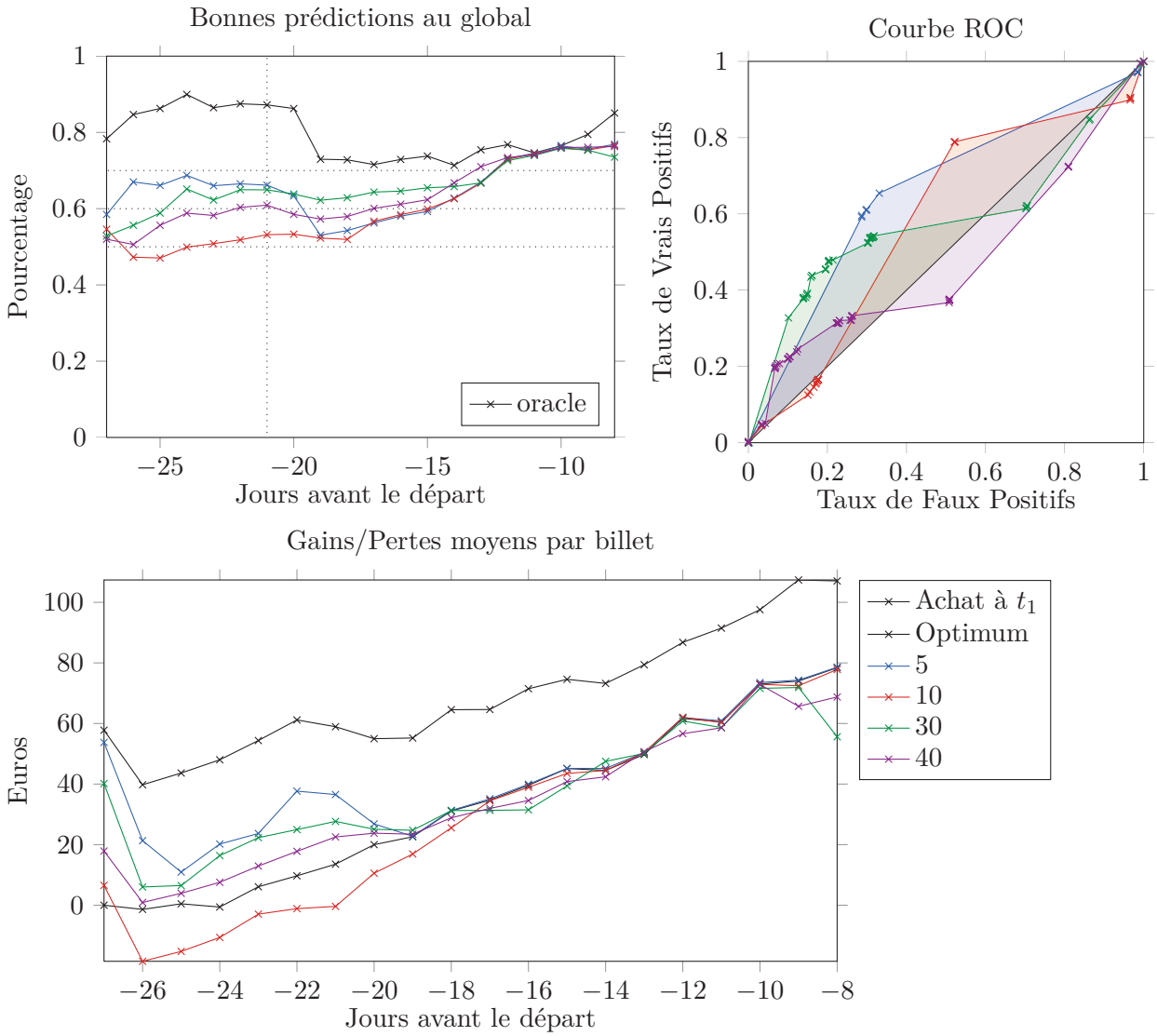


FIGURE 5.14 – Evolutions des performances de l'algorithme EM en fonction du nombre de clusters

Dans la section 3.3.3, nous avons décrit les différentes manières d'initialiser l'algorithme d'Espérance-Maximisation. La première est une initialisation aléatoire des paramètres et la seconde consiste à utiliser les résultats d'une première segmentation par les K-Means. La Figure 5.15 montre qu'il n'y a pas de différences majeures entre les deux méthodes. Le léger avantage de l'initialisation par les K-Means nous fait adopter cette méthode.

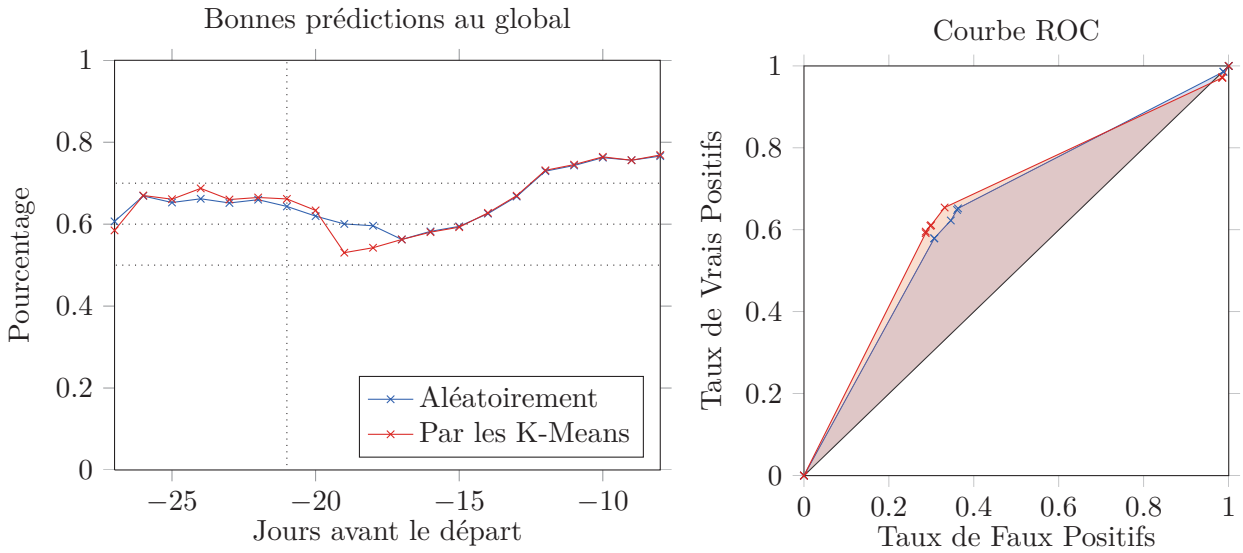


FIGURE 5.15 – Comparaison des performances de l'algorithme EM en fonction de son type d'initialisation

Comparatif

Des trois algorithmes de segmentation, l'EM est celui qui a le meilleur taux de bonnes prédictions et dont la qualité des prédictions permet de faire économiser de l'argent à l'utilisateur. Le tableau 5.3 propose un récapitulatif des performances finales des différentes configurations de la segmentation. Nous avons choisi systématiquement la même étape de classification et celle la plus stable afin de ne pas biaiser les résultats : en l'occurrence, il s'agit des forêts aléatoires avec 600 arbres.

L'achat systématique à l'instant de la prédiction t_1 fait économiser en moyenne 39€ par billet alors qu'acheter au meilleur moment fait économiser en moyenne 56€ par billet. L'économie maximale par rapport à l'achat immédiat est donc de 17€. Nous avons représenté sur la quatrième colonne du tableau 5.3 l'écart de gain à l'achat immédiat.

Méthode	Nb groupes	Taux BP	Economie/t_1
K-Means	5	62%	-1 €
EM	5	66%	4 €
K-Means	10	64%	1 €
EM	10	62%	-3 €
K-Means	30	62%	-7 €
EM	30	64%	1 €
K-Means	40	66%	-18 €
EM	40	64%	1 €

TABLE 5.3 – Comparatif

La première constatation est que les taux de bonnes prédictions sont très similaires, mais que les gains pour l'utilisateur peuvent varier de la faible hausse à forte baisse. Les performances sont plutôt faibles mais nous verrons dans la section suivante que l'algorithme de classification joue un rôle important. De cette section, nous retenons donc deux configurations qui sont l'EM avec 5 clusters et avec 30 clusters.

5.2.2 Classification

Nous allons à présent appliquer nos algorithmes d'apprentissage supervisé sur nos deux meilleures configurations de segmentation. Ainsi nous sélectionnerons les couples de méthodes optimaux à la résolution de notre problème.

CART

CART est connu pour sa tendance à sur-apprendre en créant un arbre de décision trop profond. Le paramètre important est donc le taux de pureté supplémentaire minimum pour valider la séparation d'un nœud. Cependant, en observant la Figure 5.16, nous constatons que plus l'arbre est profond, meilleurs sont les résultats. En effet, lorsque l'arbre est élagué, la plupart des groupes ne sont plus représentés limitant ainsi la répartition des vols de test sur deux ou trois clusters uniquement. Dans notre cas, il est préférable de représenter le plus de comportement possible, c'est pourquoi nous conservons l'ensemble de l'arbre créé par l'algorithme de CART au terme de l'étape de construction.

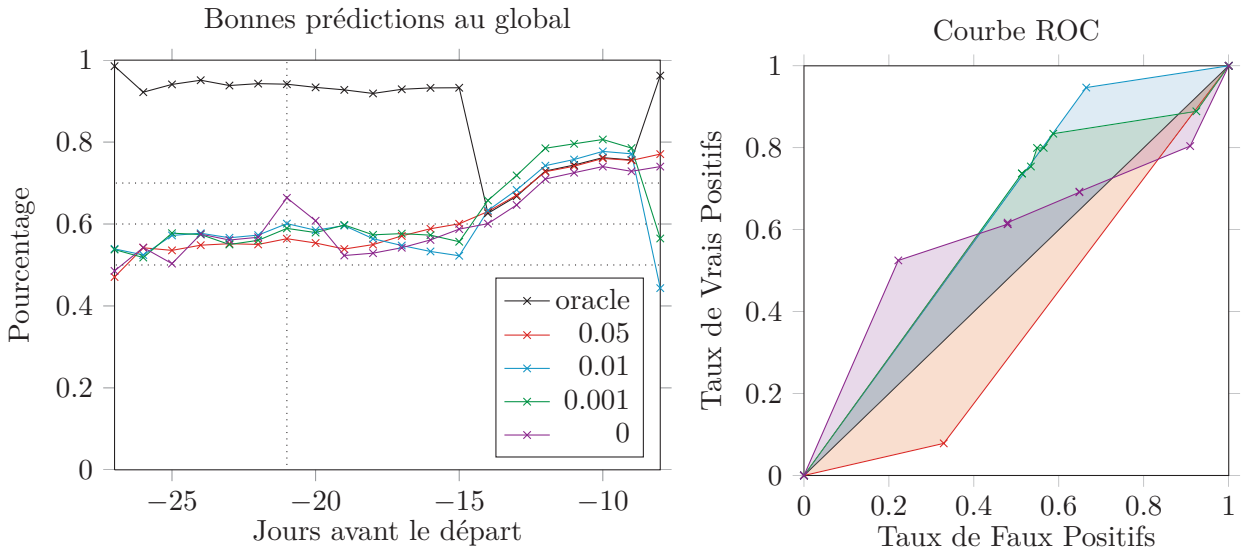


FIGURE 5.16 – Évolution du pourcentage de bonnes prédictions en fonction de la date avant le départ et courbe ROC à -21 jours pour différents élagages

Malgré tout, les performances sont plutôt faibles, et le taux de bonnes prédictions dépasse rarement les 60%. Nous tentons alors d'utiliser un autre algorithme d'arbre de décision : le C5.0

(amélioration technique du C4.5) auquel il est souvent comparé.

C5.0

Comme pour CART, l'algorithme C5.0 possède un paramètre réglant l'élagage de l'arbre créé, mais contrairement aux résultats constatés avec CART, un arbre trop profond va mal généraliser et un arbre trop élagué va trop généraliser. Sur la Figure 5.17, nous avons étudié différentes valeurs du "Confidence Factor" compris entre 0 et 1 où une diminution du facteur correspond à un élagage plus important. Nous constantons alors que la valeur par défaut qui est 0.25, donne les meilleurs résultats.

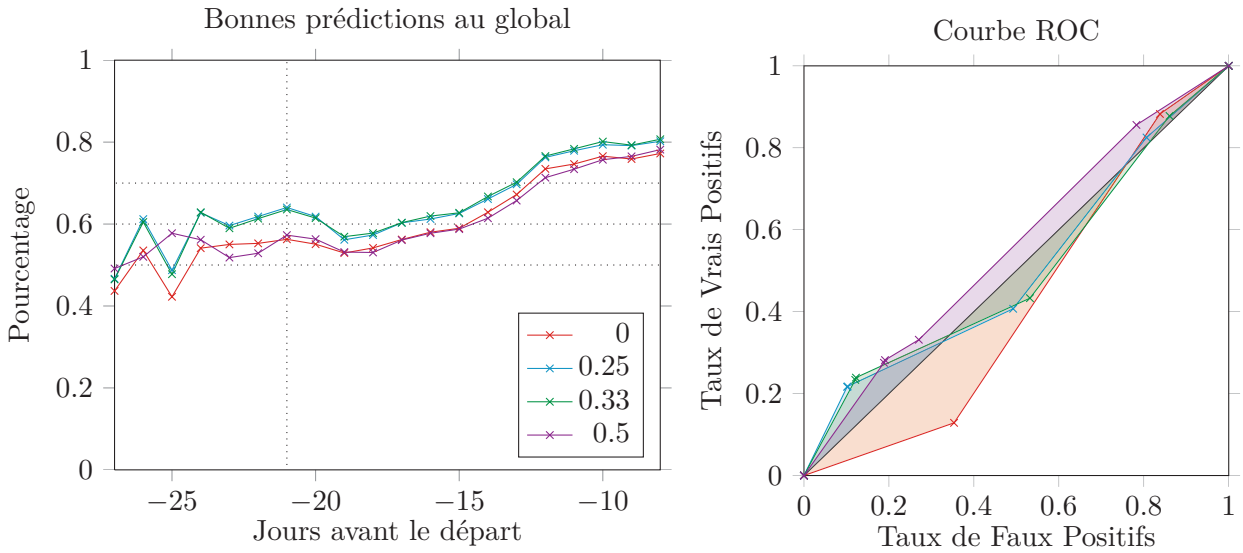


FIGURE 5.17 – Évolution du pourcentage de bonnes prédictions en fonction de la date avant le départ et courbe ROC à -21 jours pour différents élagages

Les résultats sont cependant peu satisfaisants : les courbes ROC sont proches de la diagonale ce qui se traduit la plupart du temps par une prédiction systématique à la hausse. L'interêt du service est alors très limité, c'est pourquoi nous allons nous tourner vers les algorithmes plus complexes tels qu'Adaboost et les forêts aléatoires.

Forêts Aléatoires

Comme expliqué dans le chapitre précédent, les forêts aléatoires sont une agrégation de multiples arbres de décision CART appliqués à des sous-ensembles aléatoires de la base d'apprentissage. Les deux paramètres principaux sont le nombre d'attributs choisis aléatoirement à chaque nœud et le nombre d'arbres construits. Nous allons donc étudier en détails l'influence de ces deux paramètres sur la qualité de la classification puis sur la prédiction finale.

Nombre d'attributs choisis aléatoirement à chaque nœud : *mtry* Nous rappelons que le paramètre K fixe le nombre de caractéristiques sélectionnées aléatoirement à chaque nœud au cours de la procédure d'induction d'un arbre. Sa valeur est donc choisie dans l'intervalle $[1 \dots M]$, où M représente la dimension de l'espace de description. Ce nombre contrôle la quantité d'aléatoire introduite dans le processus de sélection de caractéristiques, de telle sorte que plus la valeur de K est petite et plus on introduit d'aléatoire. Dans le cas où K = 1 par exemple, la caractéristique de chaque règle de partitionnement de l'arbre est choisie entièrement aléatoirement à partir des caractéristiques disponibles. À l'inverse, lorsque K = M , l'aléatoire n'intervient pas dans la sélection de la règle de partitionnement, et chaque arbre est alors construit à l'aide d'une procédure d'induction classique. Dans ce cas particulier, l'aléatoire est introduit à l'aide de la méthode de Bagging uniquement.

Par défaut il est choisi comme étant $\sqrt{p}$. Concernant la valeur de K, Breiman affirme que les performances n'y semblent pas sensibles dans la mesure où les écarts moyens des taux d'erreur obtenus avec $K = 1$ et $K = log_2(M) + 1$ ne dépassent pas les 1%. Comme dans [4], nous montrons avec des expérimentations que l'influence de ce paramètre sur les performances des forêts n'est pas négligeable. Le test est simple : lancer l'apprentissage de plusieurs forêts pour des valeurs de K allant de 1 et 20, et calculer pour chacune d'elles l'erreur out-of-bag. Sur la Figure 5.18, nous avons représenté l'évolution de cette erreur avec le nombre K et avons placé un axe pointillé rouge à la valeur théorique $\sqrt{p}$. Ce paramètre n'étant pas critique du point de vue de la consommation de ressources, nous avons décidé de fixer la valeur de *mtry* à 12.

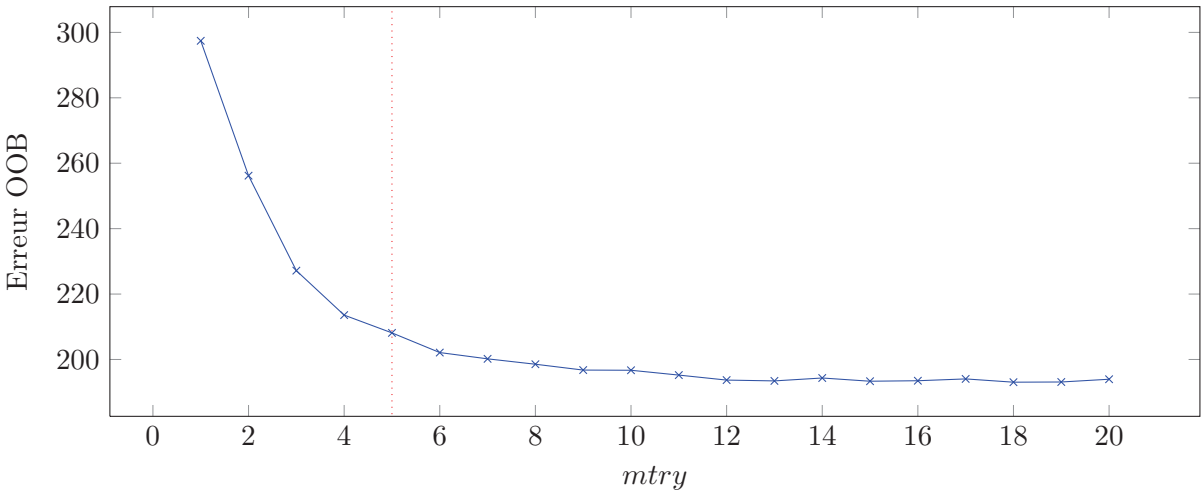


FIGURE 5.18 – Évolution du taux d'erreur OOB en fonction du paramètre *mtry*

Nombre d'arbres : *ntree* Breiman énonce que le taux d'erreurs en généralisation d'une forêt aléatoire converge nécessairement vers une valeur limite, quand le nombre d'arbres qui la composent augmente. Une interprétation de ce résultat est qu'il n'est pas nécessaire, quand on construit une forêt aléatoire, d'ajouter des arbres de décision pour obtenir de meilleures

performances. Au-delà d'une certaine quantité d'arbres aléatoires, aucun gain de performance significatif ne sera réalisé en en ajoutant d'autres. C'est ce que nous montre la Figure 5.19.

Nous voulons par ailleurs que la classification soit stable, c'est-à-dire que l'attribution d'un groupe aux vols d'apprentissage soit toujours la même. La part d'aléatoire des random forest, nous oblige donc à augmenter le nombre d'arbres et nous observons, toujours sur la Figure 5.19, que le pourcentage de différence entre deux classification par des forêts aléatoires identiques, évolue exactement de la même manière que l'erreur OOB.

Il faut donc savoir quel est le nombre limite d'arbres au-delà duquel il n'est plus utile d'en ajouter d'autres. Une limite vient naturellement lors des expérimentations dû à un problème de mémoire et de temps de calcul. La place mémoire prise par la création du modèle et sa sauvegarde en fichier, bien que linéairement croissante, nous oblige à limiter le nombre d'arbres à 100 .

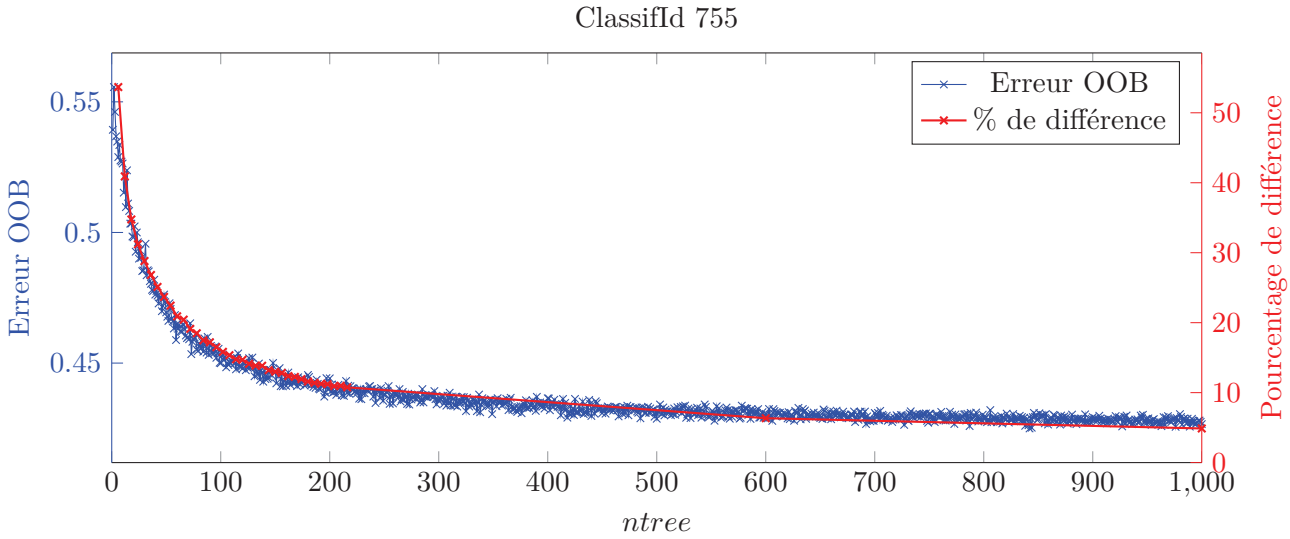


FIGURE 5.19 – Évolution du taux d'erreurs en fonction du nombre d'arbres *ntree* et du pourcentage de différence entre deux forêts aléatoires pour le même nombre d'arbre *ntree* ($mtry = 12$)

Sur la Figure 5.20, nous visualisons l'évolution des performances de forêts aléatoires en fonction du nombre d'arbres construits. Si l'on se soucie du taux de bonnes prédictions les variations deviennent minimes à partir de 120 arbres c'est pourquoi nous conservons notre choix de 100 arbres.

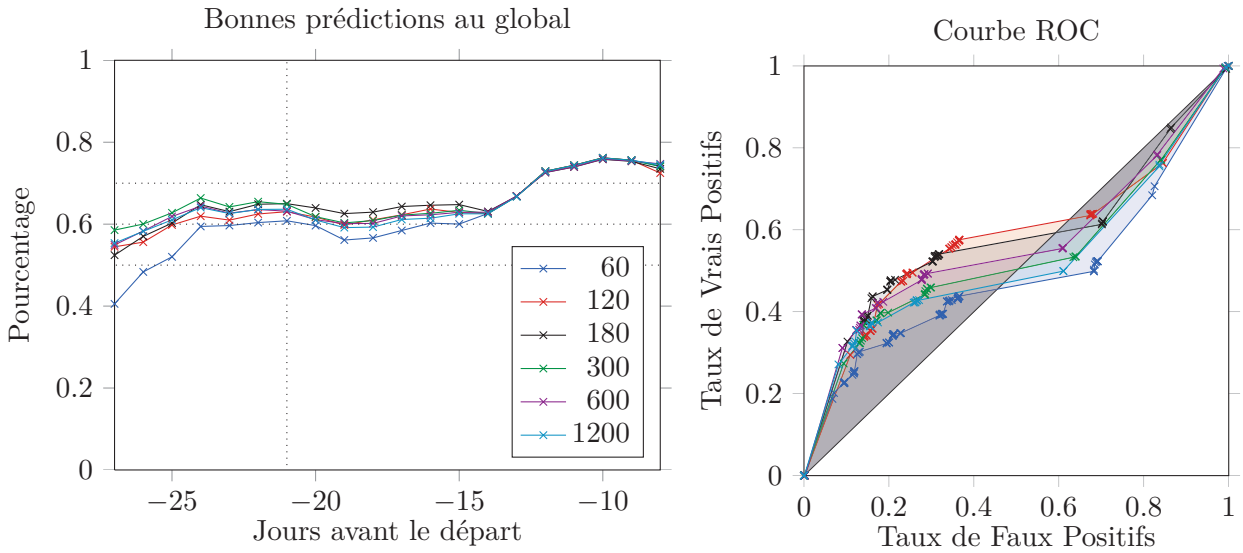


FIGURE 5.20 – Évolution du pourcentage de bonnes prédictions en fonction de la date avant le départ et courbe ROC à -21 jours pour différents $nbTree$

Adaboost

L'algorithme des forêts aléatoires est très souvent comparé à l'Adaboost : Breiman a tout d'abord comparé les performances de ses random forest avec celui-ci qui était alors l'algorithme le plus efficace dans le cadre de l'induction de forêts de décision. Par ailleurs il est intéressant de mettre en compétition ces deux méthodes car elles s'opposent dans leur idée principale : le Boosting s'appuie sur un principe déterministe pour la création de la diversité dans les ensembles alors que les forêts aléatoires par définition le font via le principe de randomisation. Dans de nombreux articles, ces deux algorithmes présentent des résultats très similaires, mais dans notre cas, l'Adaboost semble moins performant. Sur la Figure 5.21, nous mettons en compétition les forêts aléatoires et Adaboost pour une même segmentation par l'EM avec 5 clusters. Dans le premier cas nous utilisons l'ensemble des attributs à notre disposition alors que dans le second, le nombre d'attributs pour la classification a été réduit en sélectionnant uniquement les 10 meilleurs, comme expliqué dans la section 4.3.3 du chapitre précédent (Feature Selection, FS). Cette dernière optimisation qui améliore les résultats des deux algorithmes, permet à l'Adaboost d'avoir des performances similaires à celles des forêts aléatoires même si ces derniers conservent toujours un avantage.

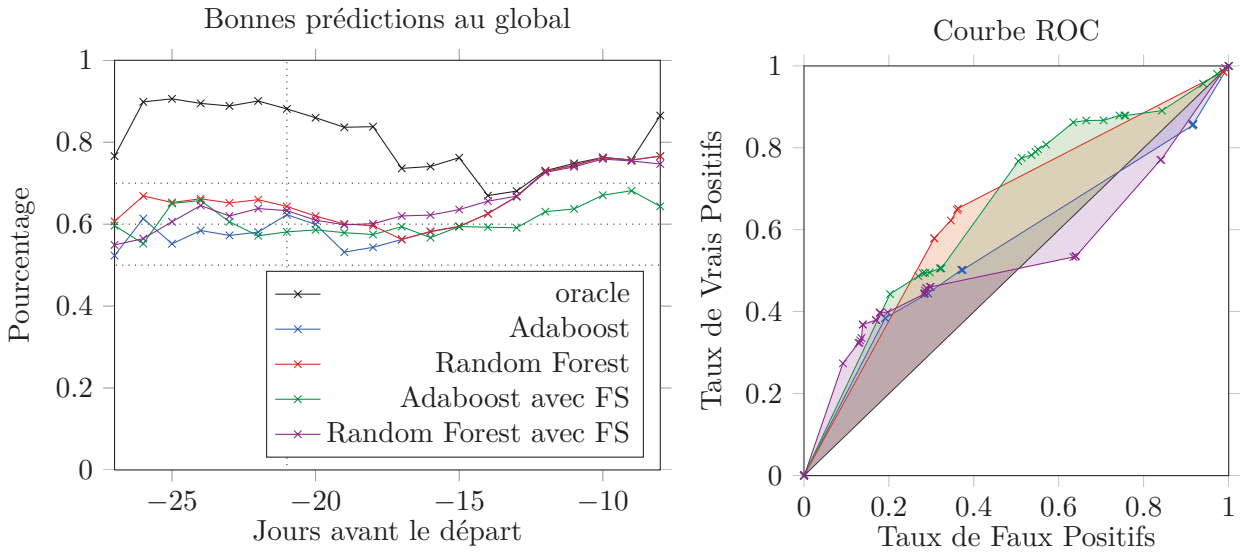


FIGURE 5.21 – Comparaison Adaboost RandomForest

L'Adaboost étant basé sur l'algorithme CART, nous avons choisis d'utiliser la configuration optimale de ce dernier, c'est-à-dire la suppression de l'étape d'élagage. Les arbres générés sont donc de profondeur maximal malgré le conseil d'utilisation qui préconise un fort élagage.

L'implémentation de l'algorithme est très coûteuse en ressource et demande énormément de temps d'exécution sans qu'une optimisation par la parallélisation des processus soit possible. Pour l'efficacité et la rapidité des random forest, nous mettons donc de coté l'utilisation d'Adaboost.

EMlogit

Comme expliqué dans le chapitre 4, l'EMlogit est l'application simultanée de la segmentation et de la classification. Il faut donc, comme pour les précédents algorithmes de segmentation, choisir un nombre de groupes et un nombre d'initialisations. L'Emlogit étant une extension de l'EM décrit dans le chapitre 3, nous reprenons la conclusion des expérimentations sur l'initialisation pour ne conserver qu'une unique initialisation par les K-Means.

La transformation des attributs qualitatifs en autant d'attributs que leur nombre de niveaux conduit à la construction d'une matrice très volumineuse demandant beaucoup de ressources, c'est pourquoi nous devons faire un choix dans les variables utilisées et filtrer leurs niveaux. Nous choisissons tout d'abord les variables les plus pertinentes d'après une classification par les forêts aléatoires (voir section suivante). Puis pour les attributs qualitatifs avec un grand nombre de niveaux, nous sélectionnons les valeurs les plus fréquentes pour en faire des attributs binaires et nous regroupons les autres en un seul et même attribut.

Les attributs conservés sont : la ville d'arrivée (*arrivalStation*), le pays d'arrivée (*arrivalCountry*), le site marchand (*provider*), le type de vol (*haul* : court/moyen/long courrier), billet

vendu par la compagnie opérant le vol (*directSeller*), la saison (*season*), le mois du billet aller (*month*) et le mois du billet retour (*monthRet*).

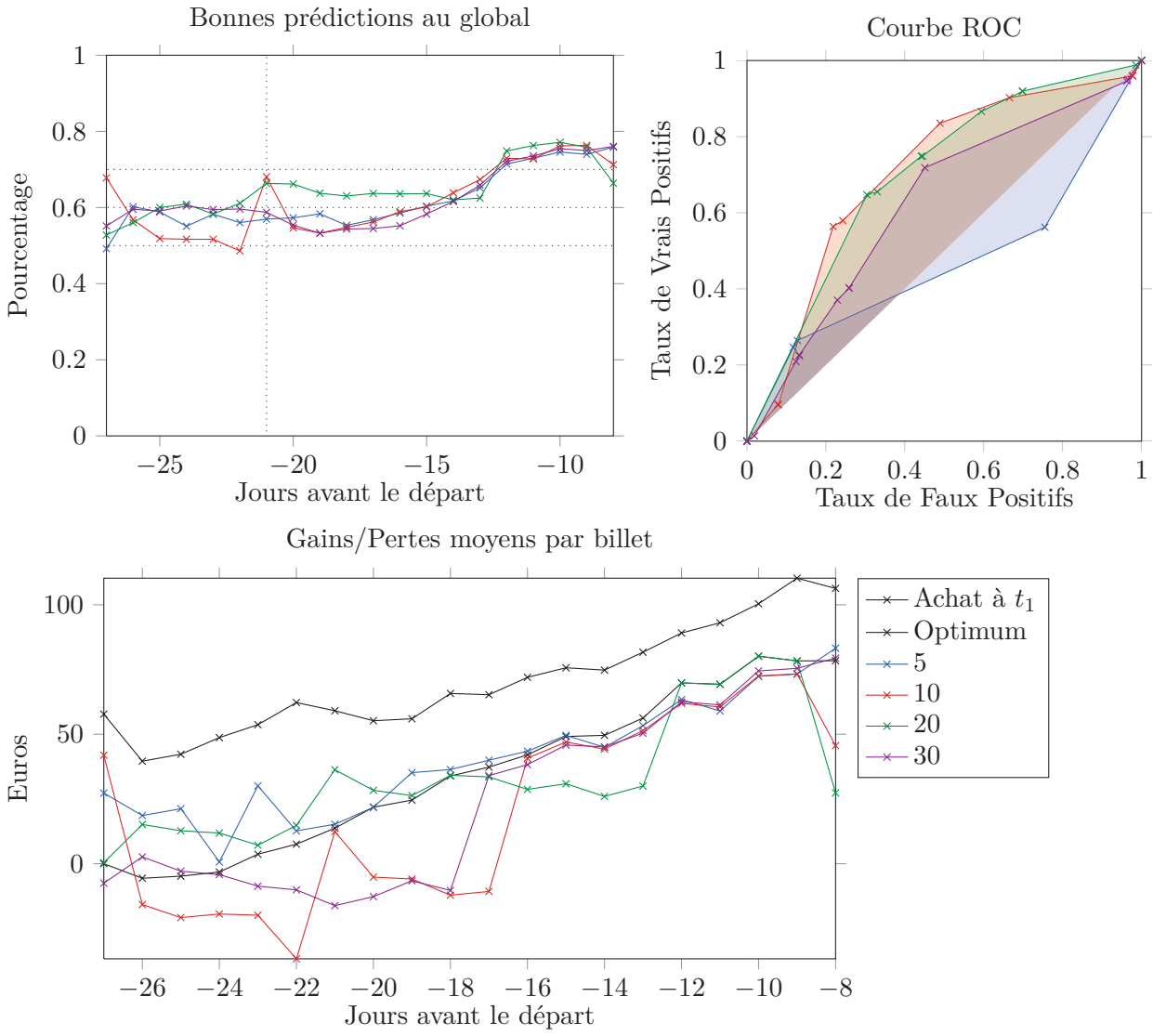


FIGURE 5.22 – Évolutions des performances de l'EMlogit en fonction du nombre de groupes

Sur la Figure 5.22, nous comparons les performances de l'EMlogit en fonction du nombre de clusters. Nous rappelons que la courbe ROC reflète une performance ponctuelle à -21 jours et qu'à la vue de l'évolution du taux de bonnes prédictions, la performance de la segmentation en 10 groupes est uniquement ponctuelle. Cependant pour la segmentation en 5 et 20 groupes les résultats sont prometteurs. Nous nous en servirons donc pour nos futurs comparatifs.

Comparatif

Le tableau 5.4 résume les résultats précédemment présentés. Toutes les expériences ont été réalisées avec la même étape de segmentation EM sauf l'EMlogit qui est couple la segmentation et la classification[1].

Méthode	Nb groupes	Taux BP	Economie/t_1
EMlogit	5	63%	2 €
EMlogit	20	66%	-2 €
RF	5	66%	4 €
RF	30	64%	1 €
Adaboost	5	63%	0 €
CART	5	62%	0 €
CART	30	62%	-3 €
C5.0	5	66%	-2 €
C5.0	30	61%	-7 €

Table 5.4 – Comparatif

Encore une fois, nous observons une grande similitude dans les taux de bonnes prédictions mais pas nécessairement dans le gain final par rapport à l'achat immédiat systématique. Même si l'EMlogit, les Random Forest et l'algorithme C5.0 arrivent à 66%, seules les forêts aléatoires parviennent à équilibrer le taux de bonnes prédictions et le gain utilisateur faisant ainsi économiser 4€ par billets en moyenne. Cependant nous restons encore loin des 17€ économisables.

Nous tentons maintenant d'améliorer les performances des différents algorithmes en sélectionnant intelligemment nos attributs.

Après sélection des variables

Comme expliqué dans la section 4.3.3 du chapitre précédent, après une première classification par les forêts aléatoires, il est possible de classer les attributs par leur ordre d'importance. En utilisant uniquement les 10 premiers attributs, nous pensons pourvoir améliorer les performances des algorithmes de classification et par la même occasion réduire les ressources utilisées lors des calculs. Nous avons appliqué les deux approches décrites dans la section 4.3.3 et nous comparons le classement obtenu sur la Figure 5.23.

[1] Nous précisons que nous n'avons pas pu tester l'Adaboost avec 30 clusters car l'algorithme demande trop de ressources.

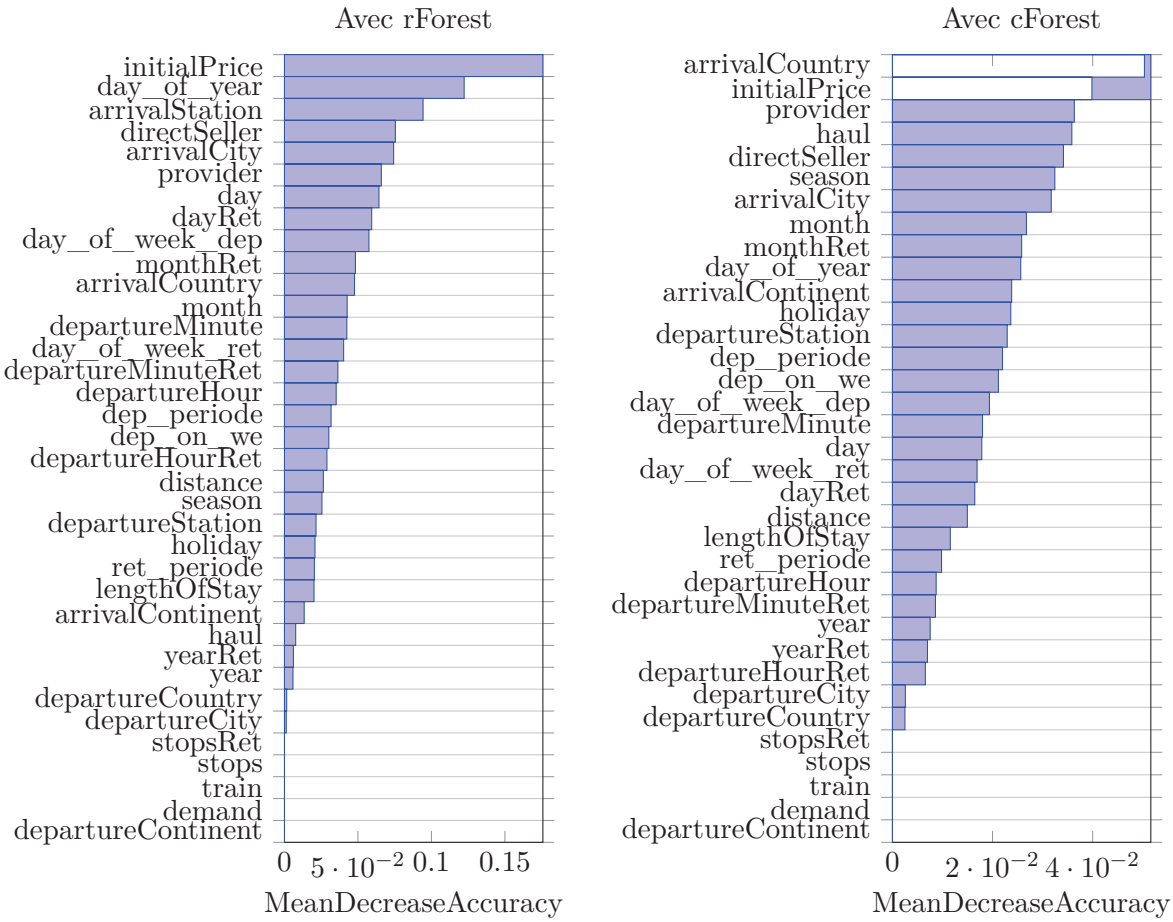


FIGURE 5.23 – Ordonnancement de l'importance des variables par l'algorithme R rForest à droite et l'algorithmes R cForest à gauche

Nous retrouvons dans les deux classements le prix à -28 jours (*initialPrice*), le jour de l'année (*day_of_year*), la ville d'arrivée (*arrivalCity*), le fait que le vendeur soit la compagnie qui opère le vol (*directSeller*), le site marchand (*provider*), et le mois de retour (*monthRet*). Lorsque l'on applique les deux méthodes au même algorithme, les résultats sont quasiment les mêmes avec un léger avantage à la version rForest (Figure 5.24), laissant supposer que les 6 variables précédemment citées sont les plus pertinentes.

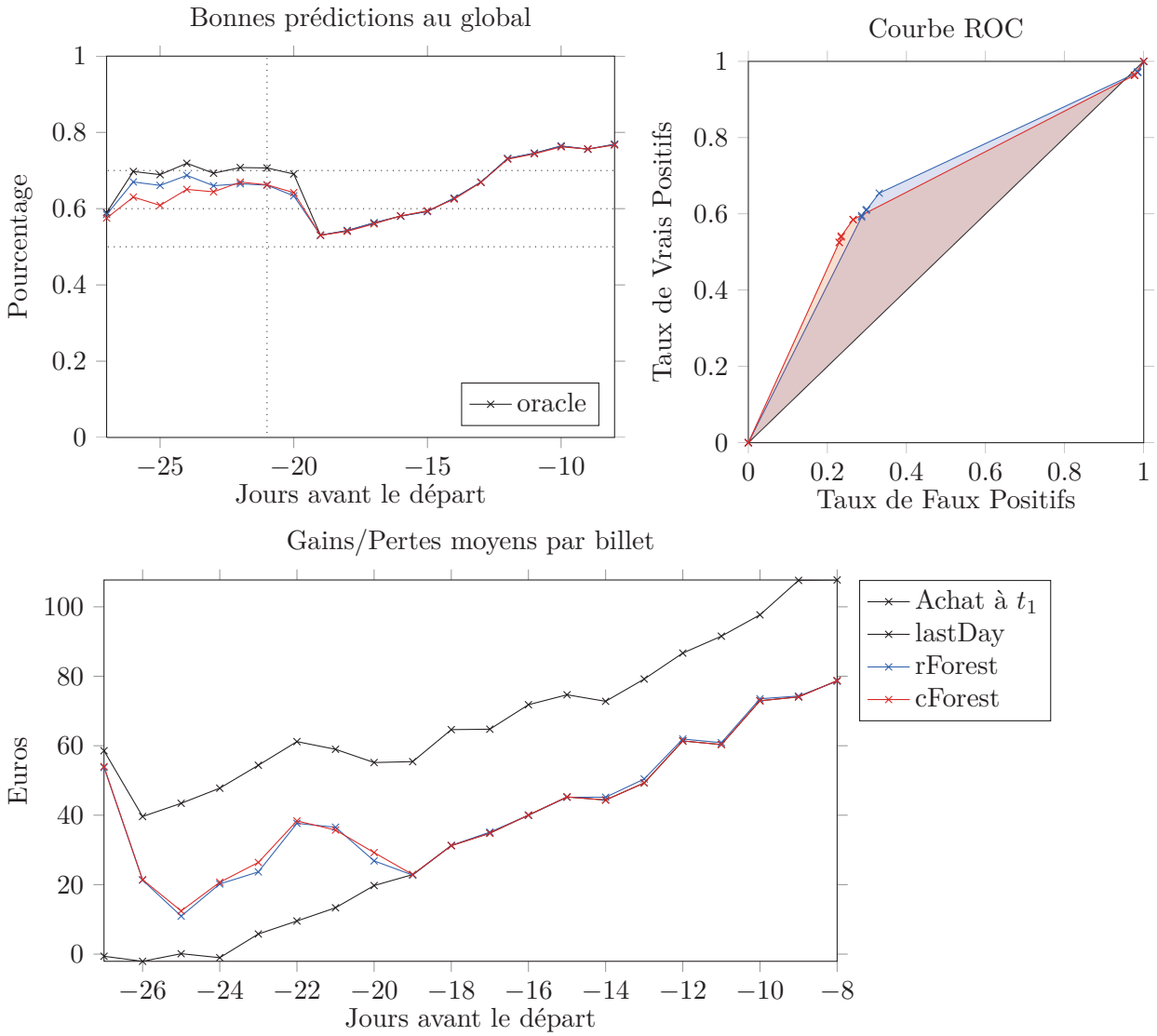


FIGURE 5.24 – Différence de performances entre la sélection de variables par rForest et cForest

Nous utilisons donc la méthode rForest sur les algorithmes de classification précédemment sélectionnés et nous obtenons :

Méthode	Nb groupes	Taux BP	Economie/t_1
RF	5	66 $\rightarrow$ 66%	4 $\rightarrow$ 4 €
RF	30	64 $\rightarrow$ 64%	1 $\rightarrow$ 1 €
Adaboost	5	63 $\rightarrow$ 65%	0 $\rightarrow$ 1 €
CART	5	62 $\rightarrow$ 63%	0 $\rightarrow$ 1 €
CART	30	62 $\rightarrow$ 64%	-3 $\rightarrow$ -2 €
C5.0	5	66 $\rightarrow$ 71%	-2 $\rightarrow$ 0 €
C5.0	30	61 $\rightarrow$ 67%	-7 $\rightarrow$ 5 €

TABLE 5.5 – Comparatif

Les effets sont visibles sur quasiment tous les algorithmes et en particulier sur l'algorithme C5.0. Sur la Figure 5.25, nous visualisons l'ensemble des performances de chaque algorithme et nous remarquons que l'algorithme C5.0 après sélection des variables donne de très bon résultats. Les arbres de décision étant connus pour sur-apprendre, nous appliquons alors les mêmes algorithmes sur une base indépendante, de vols partant en janvier 2013 (Figure 5.26).

La première chose que nous pouvons noter sur les courbes ROC est l'augmentation des bonnes prédictions à la baisse au détriment des bonnes prédictions à la hausse : les courbes ont alors tendance à se rapprocher du point $(1, 1)$. Comme expliqué précédemment, c'est un comportement que nous souhaitons limiter car, comme le montre le résultat sur l'évolution du gain pour l'utilisateur, l'erreur à la hausse fait chuter drastiquement les économies du voyageur.

Nous observons cependant que l'EMlogit réussi à généraliser ses prédictions en conservant une courbe ROC identique et à faire économiser encore un peu d'argent, le problème restant que les prédictions sont très souvent à la hausse.

Il est aussi intéressant de noter que toutes les approches semblent avoir des difficultés à prédire correctement après 18 jours avant la date de départ. Quelque soit le seuil appliqué les performances sont soit égales soit inférieures à l'achat immédiat. Cela peut s'expliquer par l'approche imminente de la date de départ et de l'ajustement quotidien beaucoup plus dépendant de la demande durant les jours précédents. Les Figures 1.11 et 1.13 du premier chapitre nous montrent bien que les augmentations de prix sont beaucoup plus importantes et que la meilleure stratégie est souvent d'acheter à t_1. Chaque prédiction à la baisse peut alors faire perdre beaucoup d'argent.

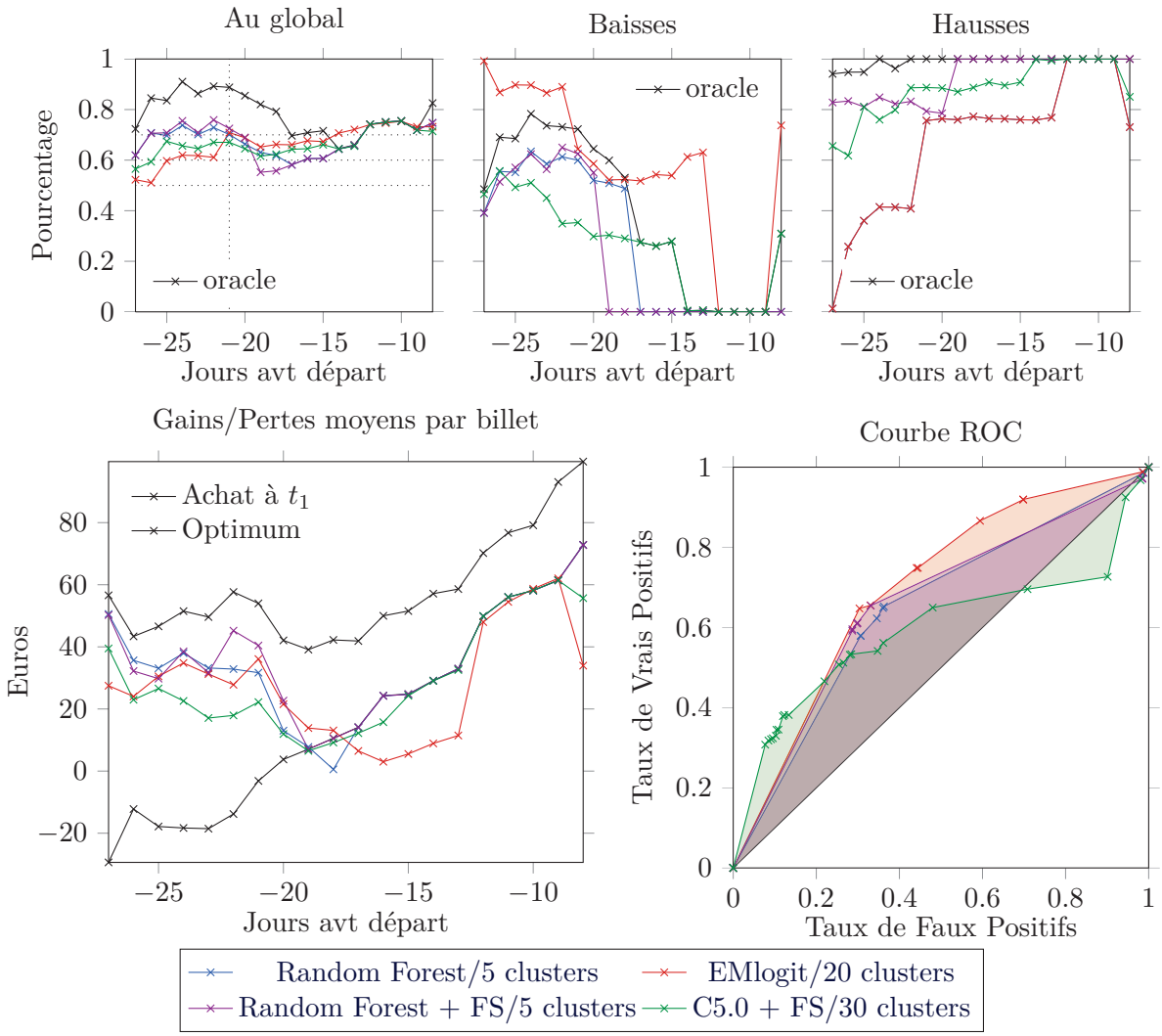


FIGURE 5.25 – Comparatifs de la validation croisée des 4 algorithmes sélectionnés

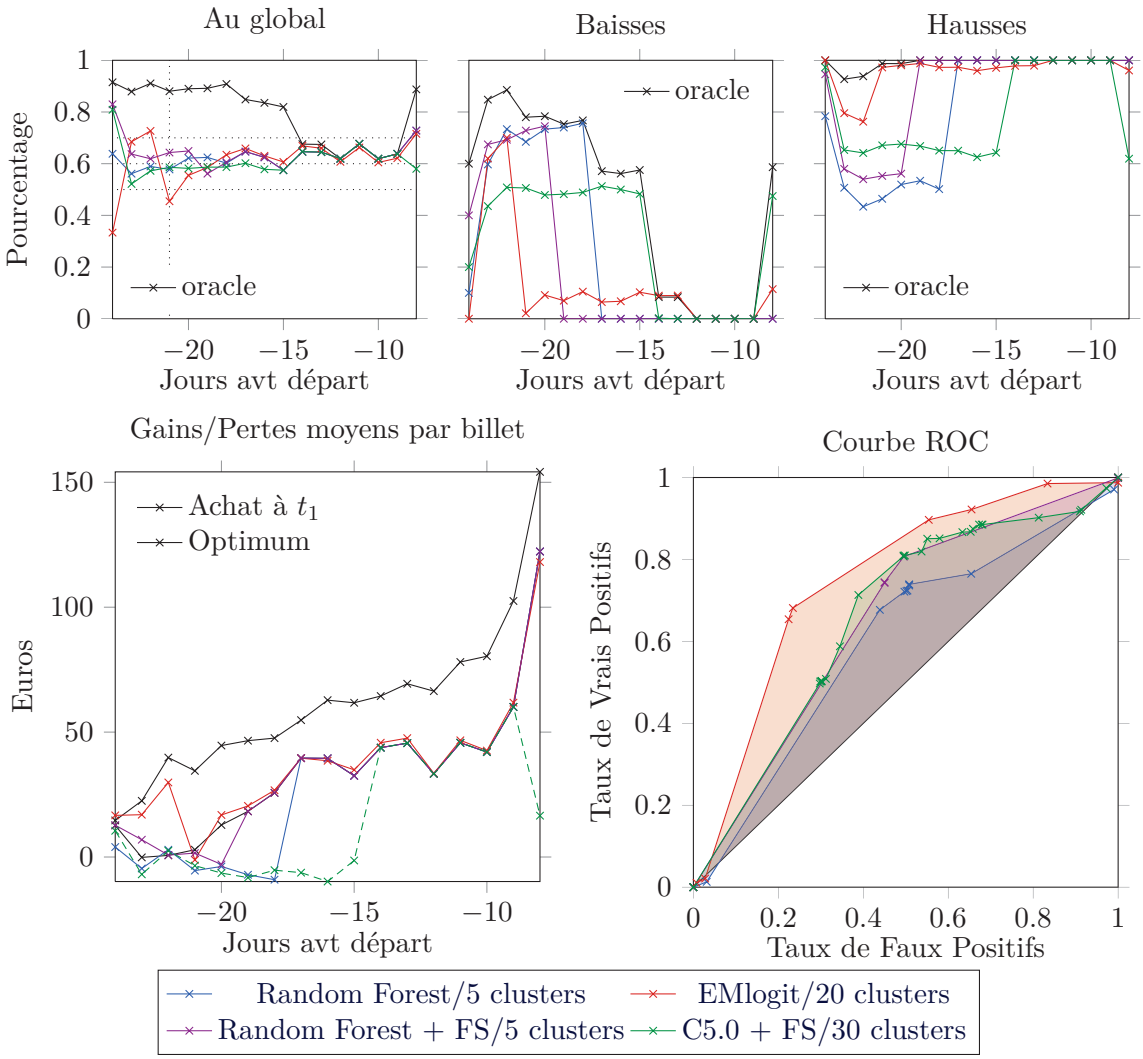


FIGURE 5.26 – Comparatifs des 4 algorithmes sélectionnés sur une base indépendante

5.2.3 Influence de paramètres extérieurs

La taille de la base d'entraînement

Lors de notre phase de segmentation, la topologie des comportements de notre base d'apprentissage a une grande influence sur les prédictions finales. Elle doit refléter au maximum l'ensemble des trajectoires possibles afin d'affiner l'attribution d'un comportement à un vol de test. Nous avons choisi de sélectionner la moitié de notre base de données pour des questions volumétriques mais aussi parce que nous pensons que sur les deux années qu'elle représente, l'échantillon de vols est suffisant pour produire une segmentation de qualité. L'ajout de vols plus anciens n'apporte donc pas d'information supplémentaire et même peut s'avérer contre-productif en insérant des comportements qui n'existent plus. Ainsi sur la Figure 5.27, nous

observons de très mauvaises performances avec les 10 derniers pourcents, une augmentation des performances lorsque nous passons des 30 derniers pourcents de la base d'apprentissage aux 50 derniers, puis une diminution du taux de bonnes prédictions lorsque nous rajoutons encore 20 pourcents supplémentaires.

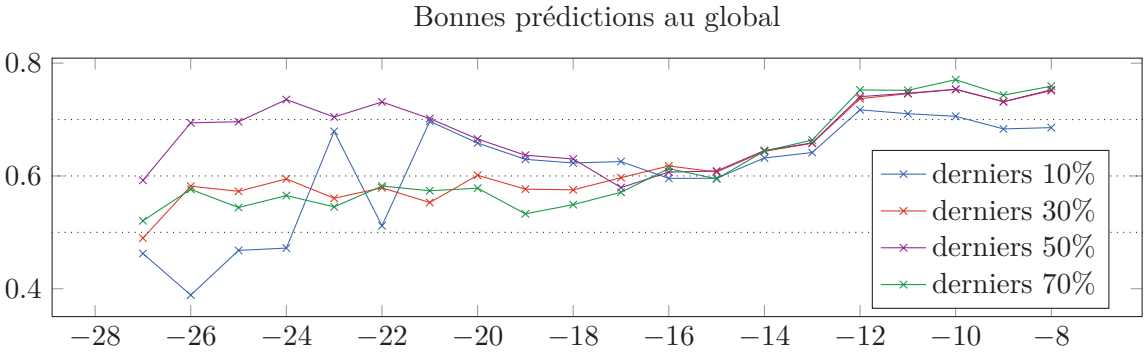


FIGURE 5.27 – Evolution du taux de bonnes prédictions avec l'augmentation des la base d'apprentissage, pour un couple EM/5 clusters et Random Forest. 1 représente une division binaire, 0.1 tous les 10% et 0.05 tous les 5%.

La configuration des niveaux de gris

De la même manière nous avons trouvé un équilibre dans la représentation de nos niveaux de gris en choisissant une segmentation horizontale par palier de 10% et une subdivision temporelle de 3 jours. La Figure 5.28 nous montre qu'une subdivision binaire des rendements, où seul le signe de la variation est pris en compte, ne permet pas de fournir une prédiction de qualité. Avec l'augmentation de la précision verticale, les performances tendent vers un optimal vite atteint à partir de la division en segments de 10%. L'augmentation de la complexité des calculs pour un gain en performance minime, nous incite donc à conserver une subdivision de 10%.

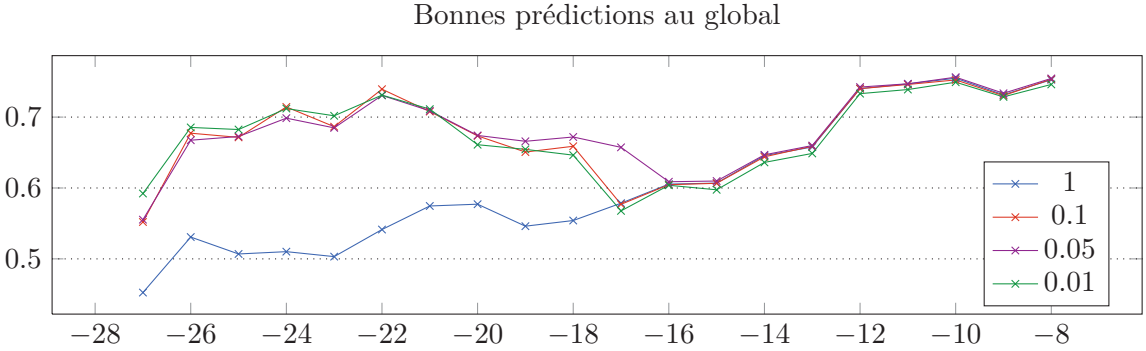


FIGURE 5.28 – Évolution du taux de bonnes prédictions avec la modification de la subdivision verticale des niveaux de gris, pour un couple EM/5 clusters et Random Forest avec sélection de variables

5.3 Autres approches

5.3.1 Prédiction directe

La prédiction directe, présentée dans le chapitre 4, a pour principe d'utiliser l'algorithme d'apprentissage en vue de prédire directement la variation à l'instant t φ_t et non pas le numéro de groupe des vols. Cela implique qu'il faut créer autant de modèle que de t et donc la multiplication des temps de calcul (par 28 dans notre cas). Sur la Figure 5.29, nous observons les performances de l'EMlogit et des forêts aléatoires en prédiction directe.

Nous pensions qu'en réduisant la complexité du problème, nous améliorerions les performances de prédiction mais nous constatons que les résultats sont équivalents voire moins bons qu'une prédiction par les modèles. En observant la courbe ROC, nous constatons que celle-ci s'apparente à une courbe ROC d'une approche comportementale avec des points d'inflexions tout le long de la trajectoire. Il est donc important de noter que l'étape de segmentation et la généralisation des prédictions par les simulations n'est pas la cause

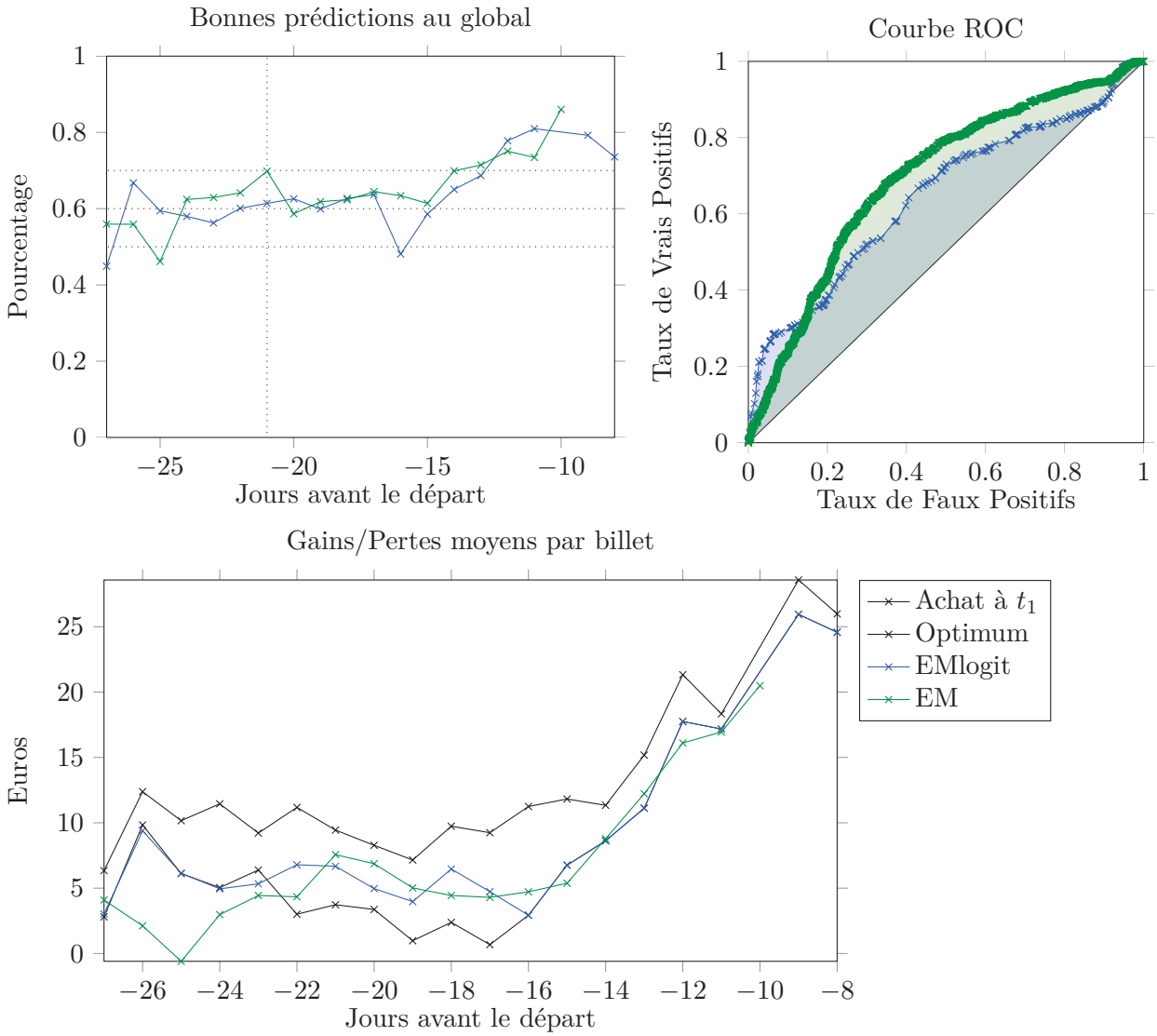


FIGURE 5.29 – Évolutions des performances de la prédiction directe par l'EM et l'EMlogit

5.3.2 Ajout des premières variations

La prédiction par les premiers points consiste à utiliser les paramètres de l'EM (α_y,$I_y(s,t)$) pour calculer la vraisemblance d'appartenance à un cluster des vols de test, grâce à leur niveau de gris partiel. C'est une prédiction qui dépend donc du nombre de jours avant le départ impliquant la création d'autant de modèles que de jours.

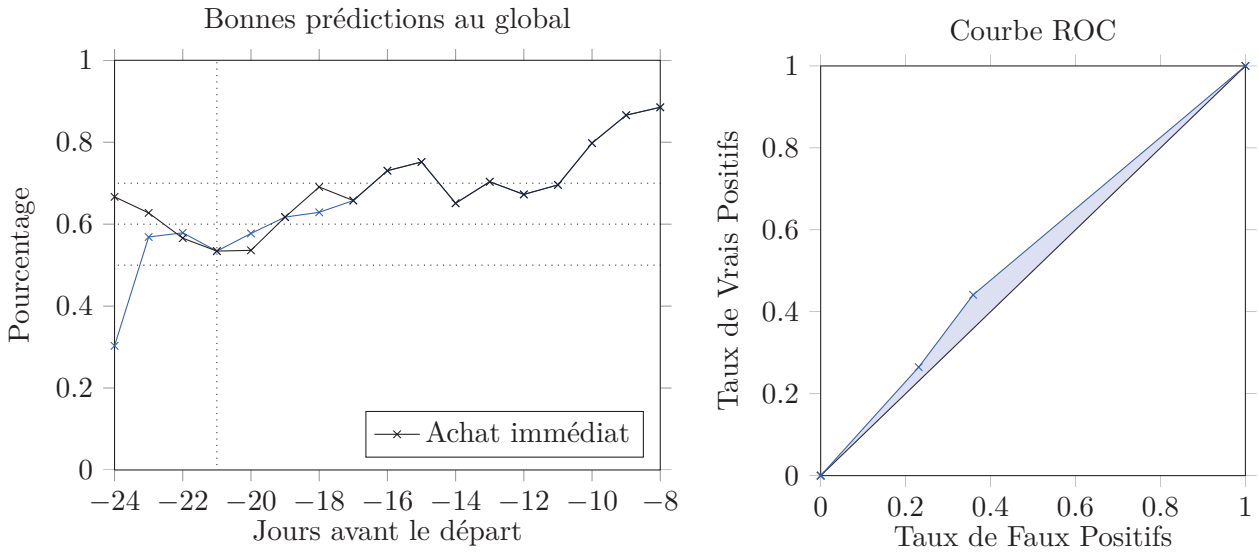


FIGURE 5.30 – Évolution du pourcentage de bonnes prédictions en fonction de la date avant le départ et courbe ROC à -21 jours pour une prédiction uniquement par les premiers points

L'information contenue dans les attributs n'est donc pas utilisée et l'unique utilisation des premières variations tend à prédire systématiquement à la hausse (Figure 5.30).

Il est alors possible, comme expliqué dans la section 4.6.2, d'ajouter cette information dans la procédure d'optimisation de l'EMlogit. Cependant la Figure 5.31 nous montre que les performances sont diminuées comparé à l'utilisation simple des attributs.

Dans les deux cas, nous n'observons pas d'augmentation des performances avec le temps laissant supposer que l'information contenu dans les premiers sauts est moins pertinente que les caractéristiques du vol en question. Il est important de noter que la période étudiée est une période cruciale pour les compagnies aériennes qui doivent finir de remplir leurs vols et pour les voyageurs partant à la dernière minute qui achètent plus rapidement. L'évolution du "bid-price" (décrit dans la section 1.2) est alors plus erratique ne permettant pas aux premiers points récoltés de bien identifier le groupe le plus vraisemblable. Cependant sur les périodes plus stationnaires de début de vente des billets, il est possible que l'information des premières variations de prix soit pertinente.

Rapellons enfin que cette méthode nécessite autant de modèles que de jours avant la date de départ et donc la prédiction d'un cluster différent pour chaque jour. Ce manque d'unité dans la prédiction peut aussi être une raison de la dégradation des performances.

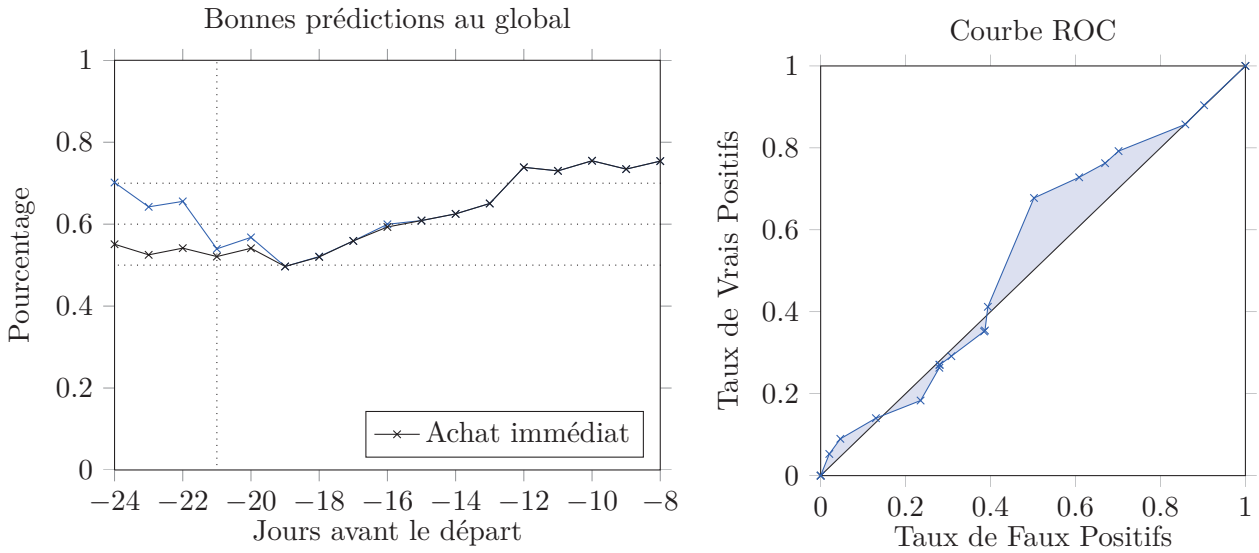


FIGURE 5.31 – Évolution du pourcentage de bonnes prédictions en fonction de la date avant le départ et courbe ROC à -21 jours pour une prédiction uniquement par les premiers points

5.4 Extensions

5.4.1 Évolution des performances dans le temps

Les précédentes expérimentations ont été effectuées sur une base de test dont les points sont compris entre janvier 2013 et février 2013, soit deux mois après le dernier prix collecté de la base d'apprentissage. Il est alors important de connaître les limites de notre modèle en terme de longévité. Pendant combien de temps notre moteur de prédiction reste-t-il performant ? Cette information nous permettrait de programmer le renouvellement systématique de notre modèle.

La Figure 5.32 nous montre que les performances des deux premiers mois sont au-dessus de 63% mais qu'à partir du 3ème mois les taux de bonnes prédictions ainsi que les courbes ROC sont fortement diminués. Nous verrons dans le chapitre suivant que nous utiliserons un système de contrôle des performances quotidien afin d'évaluer le moment à partir duquel il est nécessaire de renouveler notre modèle.

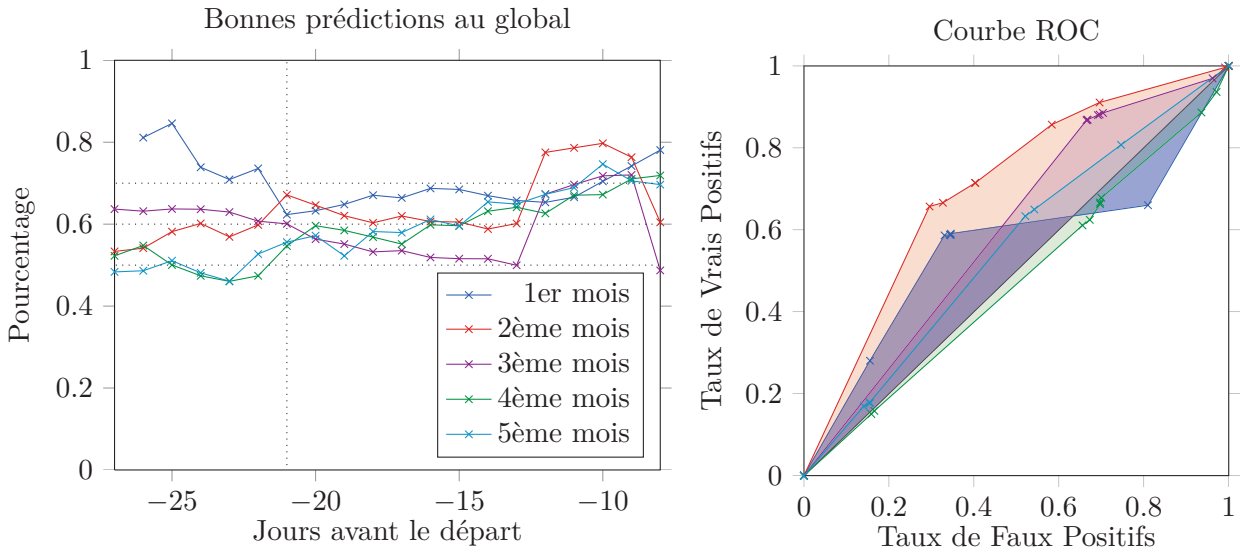


FIGURE 5.32 – Évolution des performances de l'EMlogit pour 20 clusters sur les 5 premiers mois

5.4.2 Base étendue à 90 jours

Nous voulons enfin savoir si les méthodes utilisées sur notre base échantillonnée toutes les 6 heures pendant 28 jours s'appliquent à une base de vols à 90 jours. Celle-ci est composée de l'ensemble des recherches utilisateurs possédant au moins 60 jours distincts de collecte sur un ensemble de 90 jours. Nous avons appliqué quelques-uns de nos algorithmes sur cette base afin d'observer leur capacité d'adaptation à de nouvelles données. Sur la Figure 5.33, nous visualisons les résultats après une segmentation par l'EM avec 5 clusters et une classification par les forêts aléatoires. L'aplication d'autres combinaisons d'algorithmes donne des résultats similaires c'est pourquoi nous ne présentons ici qu'une seule configuration.

Nous constatons que notre approche n'est pas adaptée à une base étendue de vols de nature différente et nécessite donc des modifications. Il est tout d'abord possible de changer les paramètres de construction des niveaux de gris en augmentant la taille des pixels. Cependant la période est trop longue pour que le rapprochement de vols similaires se fasse aisément. Il serait alors judicieux de la découper en trois périodes de trente jours : dans le premier chapitre nous avons observé qu'il existait plusieurs périodes sur les 90 jours observés ayant chacune un comportement bien distinct. Il faudrait alors découper les niveaux de gris de nos vols en trois parties de 30 jours et effectuer l'ensemble du processus de création d'un modèle sur chacune de ces parties afin de réduire les différences entre les comportements. Nous pensons d'ailleurs que les performances sur les deux parties les plus éloignées de la date de départ peuvent donner de bien meilleurs résultats : comme nous l'avons constaté sur les Figures 5.25 et 5.26, les taux de bonnes prédictions tendent à diminuer vers 20 jours avant le départ dû aux variations plus aléatoires lorsque la date de départ approche. Il est donc loisible de penser que les comportements sont plus prévisibles et plus stables depuis la date de première emission du billet jusqu'à un mois avant le départ.

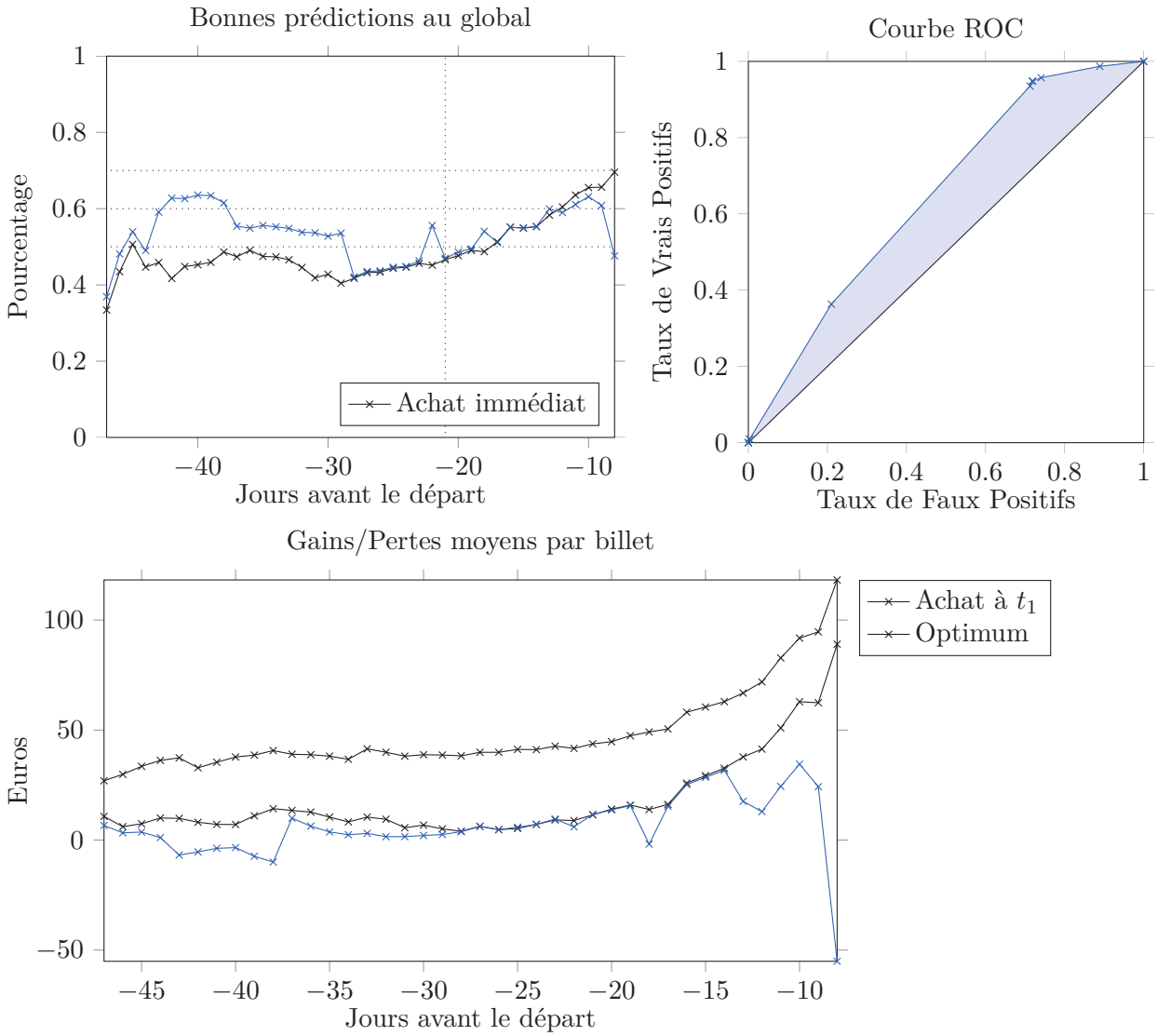


FIGURE 5.33 – Évolutions des performances sur la base des vols à 90 jours. EM 5 clusters

5.5 Conclusion et perspectives

Dans ce chapitre nous avons mis en compétition tous nos algorithmes et leurs paramètres respectifs afin de sélectionner les configurations optimales. Nous avons examiné l'évolution du taux global de bonnes prédictions tout en vérifiant le détail à la hausse et à la baisse afin de sélectionner la configuration minimisant les mauvaises prédictions à la baisse. Nous observons en parallèle le gain apporté à l'utilisateur et la différence de gain par rapport à un achat immédiat systématique. Nous souhaitons en effet que notre service permette à l'utilisateur d'économiser de l'argent mais surtout qu'il lui soit plus profitable de suivre nos prédictions que d'acheter au

jour de la demande de prédiction. L'augmentation du pourcentage de bonnes prédictions par l'amélioration de la détection des baisses au détriment des hausses, peut entraîner une diminution des économies faites par l'utilisateur. nous devons donc trouver un équilibre entre le maintien d'un bon taux de bonnes prédictions et le gain en terme d'économie pour l'utilisateur .

La première étape de comparaison consiste à appliquer nos différentes approches sur une base de validation croisée. Nous avons mis en lumière les défauts et les qualités des différents algorithmes et nous avons retenu quatre configurations que nous avons ensuite testées sur une base indépendante constituée de vols non consistants sur un panel élargi de destinations.

Nous retenons de l'ensemble de nos expérimentations qu'aucune approche ne se dégage réellement malgré quelques configurations donnant en validation croisée de meilleurs résultats. Nous possédons une base de données constituée à deux tiers de sauts à la hausse et nous voulons détecter en priorité ces dernières pour éviter de faire perdre de l'argent à l'utilisateur. Paradoxalement, la plupart de nos approches tendent à prédire plus souvent à la baisse en validation croisée et encore plus sur la base des vols indépendants. Il nous a donc fallu choisir des algorithmes moins performants mais plus conservateurs.

Nous pensons qu'il serait possible de gommer cet effet en équilibrant les valeurs de φ dans la base d'apprentissage tout en conservant un panel représentatif des comportements ou en modifiant le calcul de φ_t : prédiction à 3 jours, baisse de prix significative dans les 15 jours, etc.

Le problème posé est complexe et les critères d'optimisation multiples (gains utilisateur, taux de bonnes prédiction, minimisation des faux négatifs), c'est pourquoi nous pensons qu'une solution serait, comme expliqué par les vainqueurs de la compétition KDD 2009 [35], d'agréger les prédictions de l'ensemble de nos classifieurs. En effet, comme le montrent les courbes des oracles sur les Figures 5.25 ou 5.26, le choix optimal du classifieur à suivre pour chaque prédiction donne de très bons résultats. Cette agrégation se ferait alors au fur et à mesure de l'évolution de la base d'apprentissage.

Deux approches complémentaires ont été étudiées : l'utilisation des premiers points dans la prédiction et la prédiction directe. Nous avons montré que l'apport des premières variations de prix diminuait les performances car celles-ci reflétaient mal le comportement global du vol. De plus, l'attribution d'un comportement type différent à chaque jour avant la date du jour (un modèle par jour) reduit la cohérence dans les prédictions.

La prédiction directe quant à elle ne nous a pas donné les résultats escomptés. En simplifiant la complexité du problème (classification binaire) et en réduisant les sources d'erreurs (segmentation, simulations), nous pensions augmenter drastiquement les performances. En observant les courbes ROC, nous constatons que l'on a seulement "étalé" les points d'inflexion mais que les performances restent identiques, voire diminuent. Ce constat valide d'autant plus notre approche de prédiction des comportements et nous évite ainsi de multiplier les modèles.

Nous allons maintenant détailler l'architecture de notre service en décrivant les différentes briques applicatives permettant de stocker l'ensemble des recherches utilisateur, de fournir une prédiction à chaque recherche et de renouveler le modèle au moment voulu.

Chapitre 6

Implémentation

Contents

Introduction

Dans les chapitres précédents, nous avons décrit l'ensemble de la partie théorique du projet et les résultats observés sur des bases statiques extraites de celle mise à jour en temps réel. Le service, destiné à être mis en ligne, doit répondre à des exigences de mise en production qui influencent le choix de nos paramètres. Les critères pris en compte sont la rapidité de la génération du modèle impliquant l'utilisation d'outils adaptés à l'optimisation logicielle, le partage de ce modèle pour qu'il puisse être interrogeable à chaque requête utilisateur et par n'importe quelle technologie et enfin le contrôle de la qualité des prédictions passant par le stockage et l'analyse régulière de celles-ci. Nous devons gérer une quantité importante de données, les conserver et les exploiter pour enfin les présenter clairement à l'internaute.

Dans ce dernier chapitre, nous décrivons l'implémentation des modules de notre service et la manière dont les différentes étapes s'articulent. L'architecture globale peut se découper en quatre parties qui sont : (i) la création du modèle de prédiction en R, incluant la transformation des données, la segmentation de l'ensemble des vols d'entraînement et l'apprentissage des comportements aboutissant à la construction d'un modèle exportable en PMML. (ii) La création d'un module de comparaison des performances des modèles afin de sélectionner le meilleur ensemble de paramètres. (iii) L'implémentation d'un service web qui charge en mémoire le modèle précédemment créé ainsi que les simulations, reçoit les résultats des recherches utilisateurs et renvoie pour chacun une prédiction. (iv) Le stockage des prédictions et l'évaluation à la volée de la qualité du moteur de prédiction afin d'enclencher le renouvellement du modèle lorsque les performances sont trop faibles.

Nous décrivons les problématiques rencontrées et les solutions que nous avons envisagées afin de créer un service clair, rapide et de qualité. Nous axons notre modélisation sur l'indépendance des composants applicatifs afin de rendre chaque sous-ensemble évolutif sans modifier l'ensemble de l'architecture : nous pouvons ainsi par exemple passer l'historique des prix en *NoSQL*, ou migrer la construction du modèle sous *Apache Mahout*.

6.1 Collecte des données

Nous rappelons que nos données sont entièrement issues des recherches utilisateurs. Chacune de ces recherches déclenche un enchaînement d'actions impactant l'ensemble des briques applicatives du service de prédiction :

1. A la suite d'une recherche utilisateur, nous stockons les données récoltées dans une base de log que nous avons adaptée afin de conserver l'ensemble des résultats.

2. Ces données sont ensuite formatées pour respecter la structure de la base de données du service de génération des modèles décrite dans le premier chapitre et sur la Figure 6.1.

3. Chaque ligne de la page de résultat possède un bouton "Détails" faisant une demande de prédiction. Cette demande va interroger le service XML du module de prédiction qui va renvoyer l'ensemble des informations nécessaires à la présentation du module d'aide à la décision : la probabilité d'évolution du billet d'avion sélectionné, le taux de confiance et

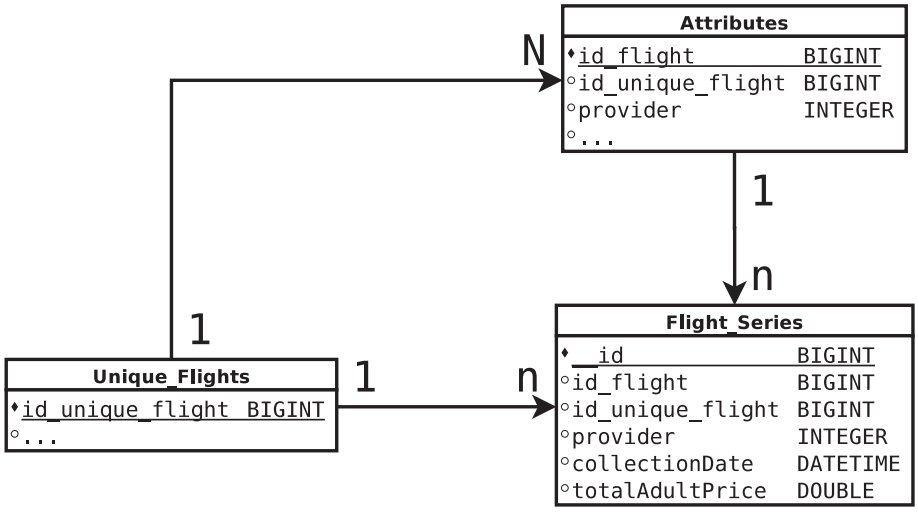


FIGURE 6.1 – Structure de la base de données sur service de génération du modèle

la fourchette de prix du futur tarif. Une base de données des vols encore actifs (dont la date de départ n'est pas encore passée) est interrogée et retourne les données nécessaires à l'affichage des variations de prix passées de la série temporelle. Les prédictions sont enfin stockées en base.

4. Afin d'évaluer les performances de notre service de prédiction, nous vérifions à chaque recherche si une prédiction a été faite 7 jours avant et le cas échéant, nous vérifions et stockons la qualité du conseil. Avec le système d'alerte qui effectue quotidiennement la même recherche et conserve les 5 meilleures offres, nous sommes capable d'évaluer plus de 30% de nos prédictions. En ajoutant les recherches utilisateurs, nous constituons un ensemble représentatif suffisant pour que l'évaluation de nos prédictions reflète la qualité globale de celles-ci.

5. Lorsque le taux de bonnes prédictions baisse en dessous d'un seuil donné, nous lançons la génération d'un nouveau modèle sur la base de données actualisée de l'historique des vols. Celle-ci transmettra alors au service XML du module de prédiction un fichier PMML représentant le modèle de classification et les simulations de tous les centroïdes à tous les temps t.

La Figure 6.2 résume l'ensemble du cycle de vie d'un modèle et reprend les quatre composants représentés sur le diagramme de séquence de la Figure 6.3 : le moteur de recherche recevant les appels utilisateurs et les enregistrant en base de données, le service de web effectuant les demandes de prédictions et le générateur de modèle utilisant l'historique des vols. Les zones de couleurs représentent l'ensemble des parties créées ou modifiées pour le service de prédiction.

Un des points clef du service est le renouvellement du modèle et la détection d'une diminution des performances. Cependant le processus de création du modèle étant coûteux en temps et en

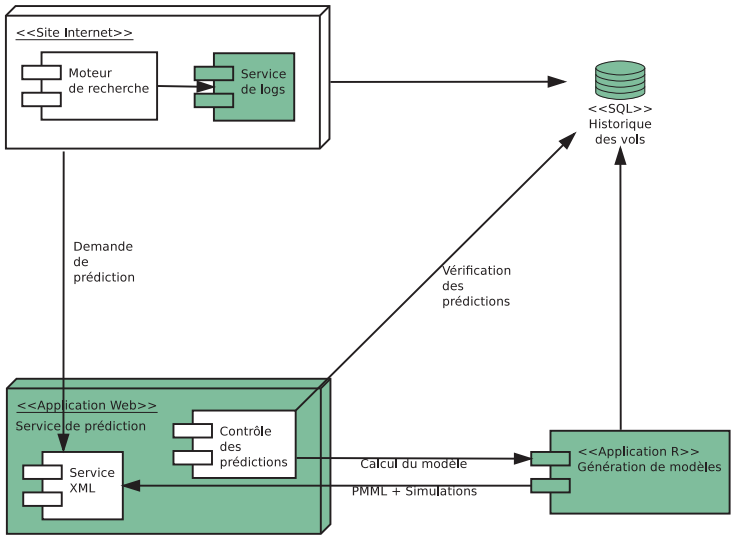


FIGURE 6.2 – Diagramme de déploiement : Composants applicatifs

ressource, il est important de limiter son exécution. Comme expliqué dans le chapitre précédent, il est important de différencier le pourcentage global de bonnes prédictions et l'économie moyenne faite à l'utilisateur. Dans le faits, nous privilégierons la somme économisée par l'utilisateur en observant l'économie faite grâce aux prédictions mais aussi l'écart de cette économie par rapport à celle faite lors de l'achat immédiat systématique. Notre service n'a de sens que s'il fait économiser plus d'argent que la solution "conservatrice". Si le gain réalisé grâce aux bonnes prédictions passe sous la barre du gain à l'achat immédiat, le modèle doit être mis à jour.

6.2 Construction du modèle

La construction du modèle est décrite dans les Chapitres 2, 3 et 4. Les différentes étapes sont synthétisées sur la Figure 6.4 et détaillées ci dessous :

1. Tout d'abord nous stockons à la volée les données de recherche dans les tables précédemment décrites : *Unique_Flights*, *Flight_Series* et *Attributes*.
2. Après avoir extrait les vols consistants, nous créons une base d'apprentissage des niveaux de gris M_{train} grâce à la transformation des séries temporelles. Cette transformation se fait suivant les paramètres b_t et b_r décrivant les dimensions des fenêtres de l'image pixélisée.
3. Nous segmentons ensuite la base des niveaux de gris $X_i(s,t)$ en utilisant les algorithmes décrits dans le Chapitre 3. Les centroïdes, les paramètres des modèles et l'attribution des clusters aux vols de la base d'apprentissage sont enregistrés en base pour pouvoir les utiliser ultérieurement.
4. Nous générons alors les φ_t pour tous les clusters et tous les t possibles grâce aux simulations créées à partir des centroïdes précédemment calculés.

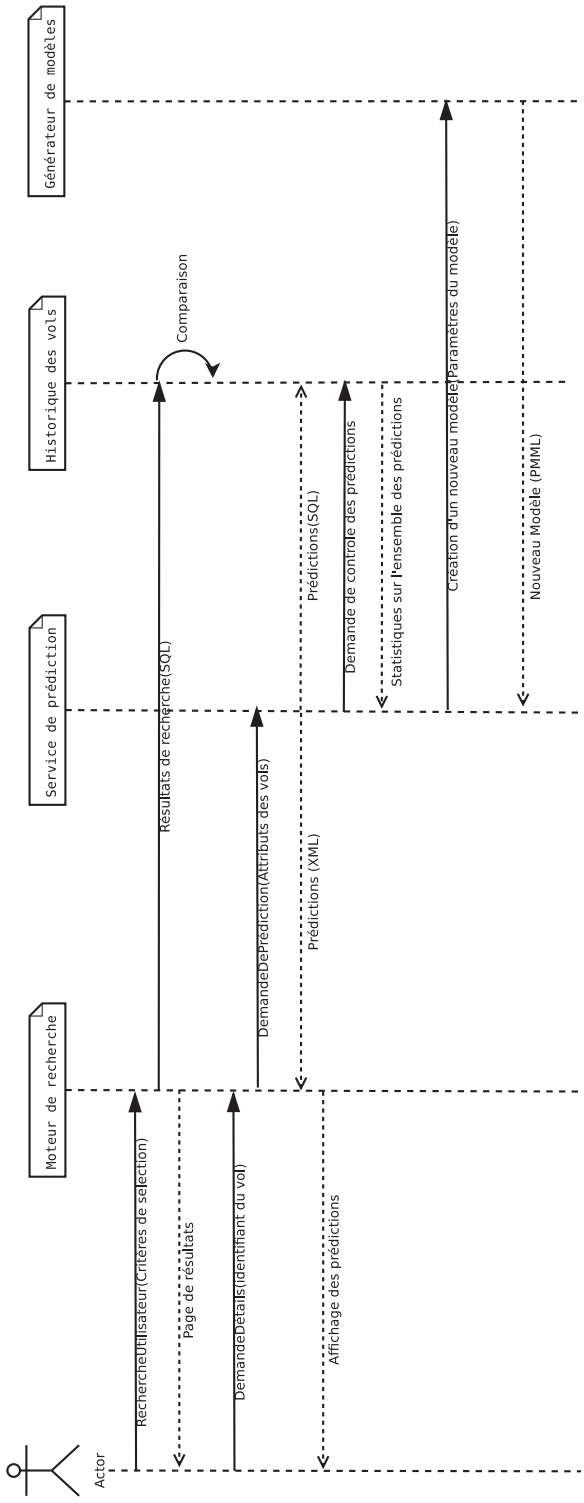


FIGURE 6.3 – Diagramme de séquence

5. Enfin nous créons le modèle de classification en appliquant les algorithmes d'apprentissage des comportements à partir des attributs V_i stockés dans la base N_{train}. Nous exportons ce dernier sous le format PMML pour qu'il puisse être ensuite interrogé par le service de prédiction.

Chacune de ces étapes est indépendante, c'est-à-dire qu'elle possède son paramétrage propre et peut être exécutée dans différentes configurations sans avoir à ré-exécuter les étapes précédentes. L'avantage est de pouvoir tester facilement différents paramètres et ainsi identifier plus facilement les meilleures configurations.

Nous allons maintenant détailler les difficultés rencontrées lors de la création d'un modèle et les solutions que nous avons mises en œuvre.

6.2.1 Étapes menant à la création du modèle

La création d'un modèle est la génération d'un objet interrogeable par le service de prédiction du site liligo.com. Cet objet est capable de fournir pour chaque ligne de la page des résultats, une prédiction du groupe le plus probable, au vue des paramètres du vol. Les simulations de chaque cluster, préalablement effectuées, nous fournissent les φ_t pour tous les t et tous les clusters. Le tableau des φ_t est alors mis en mémoire et associé à la prédiction de groupe pour afficher au final une prédiction de comportement.

Si le taux de bonnes prédictions vient à diminuer, il est important de pouvoir re-générer ce modèle dans des délais raisonnables en gardant la même qualité de prédiction. Nous avons donc utilisé le logiciel *R*, équivalent libre de *Matlab*, dont l'importante communauté permet d'avoir accès à de nombreux algorithmes d'intelligence artificiel et à des optimisations de code permettant la manipulation d'importants volumes de données en des temps raisonnables.

Optimisation du code

La problématique de la création rapide du modèle de prédiction provient du volume de données à traiter : la manipulation de grande matrice utilise une grande quantité de ressources, à la fois en mémoire et en nombre de cœurs, interdisant ainsi certaines opérations tout en demandant un temps de calcul considérable. Par exemple, lors de la segmentation, les niveaux de gris sont "aplatis" pour avoir un vecteur pour chaque vol dont chaque valeur représente l'intensité d'une case. La matrice M_{train} dont chaque ligne correspond à un vol et chaque colonne à une case de l'image pixélisée, peut alors obtenir une taille de 10000×198.

Nous avons donc utilisé différents packages de *R* pour pouvoir manipuler ces matrices et appliquer des algorithmes de segmentation dans des temps raisonnables :

1. La librairie *bigmemory* permet la création, le stockage et l'accès à de très grandes matrices, celles-ci étant allouées à la mémoire partagée. Le partage de mémoire est un moyen de partager des données entre différents processus en permettant à plusieurs processus d'accéder à une même zone de la mémoire vive.

2. Une librairie connexe (*biganalytics*), implémente une version de l'algorithme de clustering K-Means nommée Big-K-Means, utilisant cette implémentation des matrices, évitant ainsi

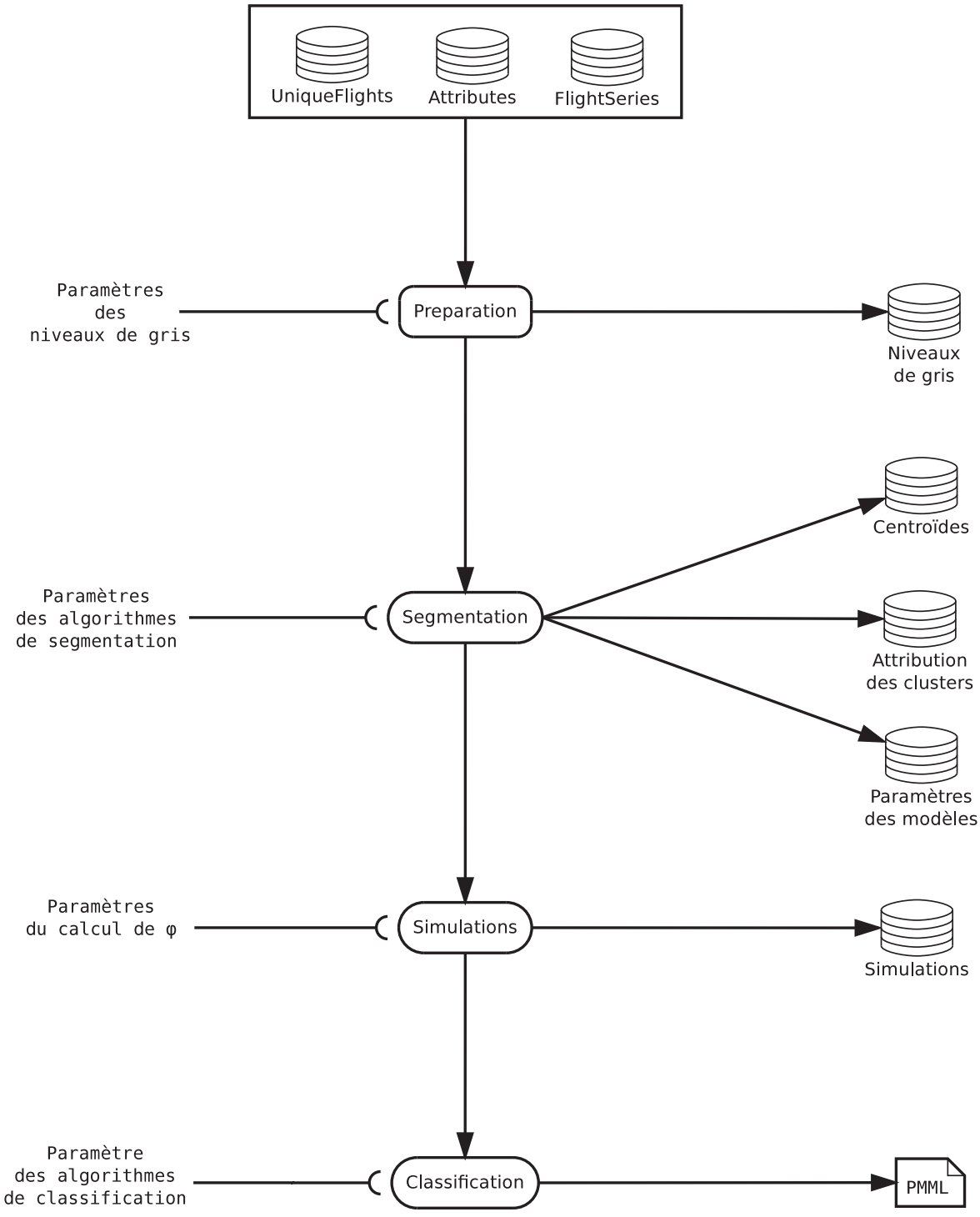


FIGURE 6.4 – Création d'un modèle

la surcharge de données en mémoire et autorisant la distribution des différentes initialisations sur l'ensemble des cœurs disponibles. Cet algorithme ne nécessite pas de mémoire supplémentaire autre que les données tandis que l'algorithme usuel fait au moins deux copies supplémentaires de la matrice de données. Un choix de plusieurs départs aléatoires s'il n'est pas exécuté en parallèle utilise aussi beaucoup de ressources et de temps et nécessité de stocker autant de vecteurs d'adhésions aux clusters que de départs.

3. Cette parallélisation des processus se fait grâce au paquet *foreach* capable d'exploiter en parallèle l'ensemble des cœurs d'un ordinateur. Il est possible de coupler *foreach* avec de nombreux autres paquets de *R* comme celui des forêt aléatoires, permettant de paralléliser la création des arbres et de combiner par la suite les différentes forêts entre elles.

Export des arbres de décisions en un format standard

R n'offrant pas les garanties de stabilité requises à une mise en production, la possibilité pour une application grand public de déployer son service sur une plateforme de Cloud Computing est restreinte. C'est pourquoi nous avons dû trouver un format de fichier vers lequel exporter notre modèle de classification afin d'être utilisé par nos serveurs lors d'une demande de prédiction. Ce format devait être neutre et standard.

La création d'un fichier au format neutre, indépendant de tout langage de programmation, a plusieurs autres avantages. Tout d'abord elle permet de conserver une indépendance entre les composants applicatifs : il n'y a pas besoin de faire dialoguer *R* et l'interface web pour effectuer les prédictions. Ensuite, si l'on souhaite changer la technologie derrière la génération du modèle (par exemple, utiliser l'API *JAVA Weka* ou le framework *Apache Mahout* qui implémente des algorithmes de fouille de données distribués sur la plateforme *Apache Hadoop*), la partie cliente du service n'a pas a être modifiée. Enfin la création de fichiers interrogeables, nous permet de multiplier le nombre de modèles pour affiner la prédiction en fonction du nombre jours avant la date de départ. En effet, nous avons constaté des différences de performance entre le modèle créé sur les vols de 90 jours et celui sur les vols de 28 jours. Deux modèles sont alors créés et interrogés en fonction de la date de demande de prédiction. Par ailleurs, chaque modèle pourra avoir ses propres paramètres d'interprétation de la probabilité $\mathbb{P}(\varphi = 1)$, ajoutant encore de la flexibilité à notre service.

Predictive Model Markup Language ou *PMML* est un langage de marquage basé sur *XML* conçu pour définir des modèles de données et visant à rendre inter-opérables les systèmes de data-mining. Nous utilisons ce langage pour exporter l'arbre de décision généré par l'algorithme CART ou les forêts d'arbres décisionnels. Nous importons par la suite cet arbre de décision dans un module web qui sera interrogé à chaque recherche utilisateur.

L'inconvénient d'un format issu de l'XML, est la taille du fichier dû à la verbosité du langage, et donc au temps de création du fichier. La parallélisation n'est pas possible et le temps de calcul dépasse souvent 3 heures. Ce temps est par ailleurs indexé sur le nombre d'arbres créés si l'on a choisi les forêts aléatoire, rendant le choix de ce paramètre délicat lors de la création du modèle.

Enfin tous les algorithmes ne sont pas encore pris en charge par *PMML*, comme Adaboost, ou par l'implémentation *PMML* d'*R* comme l'algorithme C4.5.

6.2.2 Segmentation des étapes de calcul du modèle

Notre approche a nécessité de nombreux tests et de multiples combinaisons de paramètres, nécessitant d'effectuer les mêmes étapes de calcul pour toutes les combinaisons. Pour éviter d'effectuer toutes les étapes coûteuses en temps (création des niveaux de gris, clustering etc.) lors du changement d'un paramètre sur le nombre d'arbre dans les random forest par exemple, nous avons décorrélé ces étapes et stocké tous les résultats intermédiaires dans des tables, que l'on interroge par la suite. Cela nous permet également de conserver des statistiques sur tous les algorithmes testés.

Si l'on reprend la Figure 6.4, nous constatons clairement cette séparation ainsi que les différentes données conservées en base afin d'être réutilisées à l'étape suivante :

1. **Création des niveaux de gris** : Pour chaque vol nous stockons en base une version compressée du niveau de gris dans différentes configurations : $b_r = \{1, 0.1, 0.05, 0.01\}$ et $b_t = \{1 \times 24, 3 \times 24, 5 \times 24\}$.

2. **Clustering** : Selon la configuration souhaitée, nous chargeons les niveaux de gris appropriés sous la forme d'une matrice. Une fois la segmentation effectuée, nous stockons les statistiques de cette dernière, les centroïdes et l'étiquette E_i associée à chaque vol.

3. **Simulations** : Le choix du critère de prédiction n'intervient qu'au dernier moment, car l'apprentissage se fait sur le comportement global et non sur φ. Comme nous effectuons les simulations à l'avance pour pouvoir les interroger à la volée, nous devons statuer de la prédiction à faire en amont. Nous effectuons alors pour chaque t possible 1000 simulations, et calculons un version simple et une version par cadran de φ_t que nous stockons ensuite en base. Celle-ci sera alors chargée en mémoire sur les machines de production en vue d'être associées à la prédiction d'un identifiant de cluster et de fournir une prédiction d'évolution de prix. Cette méthode évite de devoir effectuer les simulations à la volée et garantit une reproductibilité des prédictions.

4. **Classification** : Nous chargeons la matrice N_{train} contenant l'ensemble des attributs des vols, sous la forme d'une matrice à laquelle nous ajoutons une colonne représentant l'étiquette du vol créée à l'étape précédente. Une modèle de classification est alors créé et exporté au format R (pour une utilisation locale) et PMML (pour être utilisée en production).

Identification unique des configurations

La multiplication des combinaisons possibles d'algorithmes et de leurs paramètres, a nécessité la création d'un système d'identifiants uniques imbriqués pour chaque étape, correspondant à un jeu de paramètres. La Figure 6.5 nous montre les différents identifiants existants et la Figure 6.6 est une visualisation simplifiée de la hiérarchie ainsi créée.

Nous avons dans pour la première étape de transformation l'identifiant correspondant à la configuration des niveaux de gris. Puis l'identifiant de la longueur des séries qui va définir avec

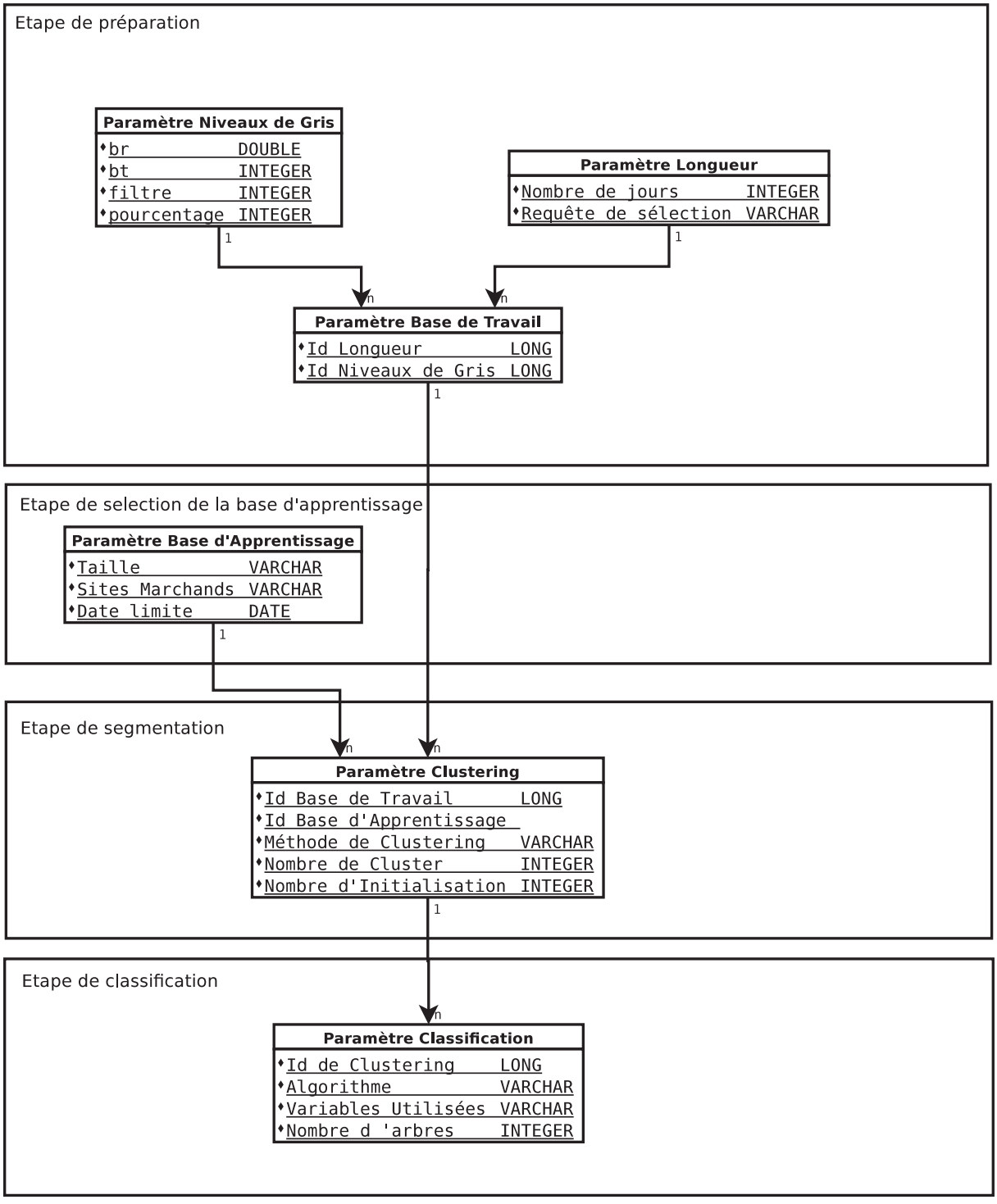


FIGURE 6.5 – SchemaDB

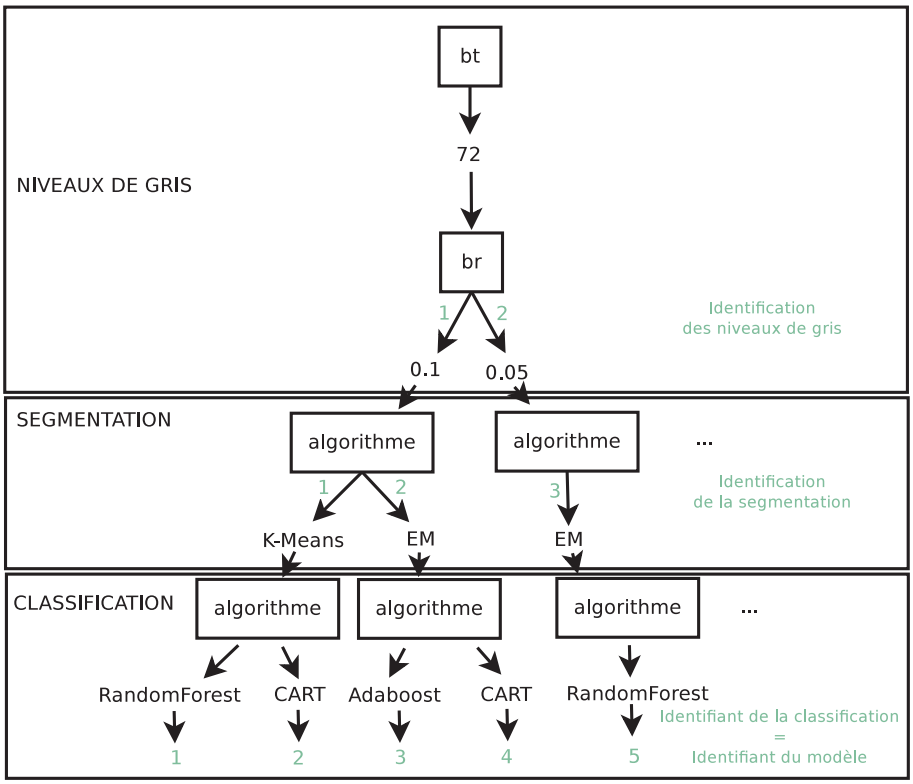


FIGURE 6.6 – Identifiants

la "requête de sélection" la liste des vols consistants. Enfin un table regroupant ces deux identifiant définit l'ensemble des niveaux de gris de la base d'apprentissage M_{train} et une dernière sélectionne dans cette base le nombre de vols étudié et filtre sur les sites marchands si besoin.

Vient ensuite la configuration de la segmentation qui comprend l'algorithme utilisé et ses paramètres et enfin la configuration de la classification de manière identique.

6.3 Interface de comparaison des modèles

Pour permettre la comparaison rapide de nos modèles, nous avons créé une interface permettant d'afficher simultanément les performances des différents modèles selon plusieurs critères (Figure 6.7). Il est possible d'afficher les performances d'autant de modèles que l'on souhaite ainsi que de modifier à la volée certains critères, comme le seuil de séparation des vols à la hausse et à la baisse ou bien la taille et la topologie de la base de test (Figure 6.8).

1. La première étape consiste à sélectionner les paramètres du modèle que l'on souhaite afficher : paramètres des niveaux de gris (b_t, b_r, etc.), les paramètres de la segmentation

(type d'algorithme de segmentation, nombre de groupes, etc.) et les paramètres de la classification (algorithme de classification, variables utilisées, etc.).

2. Il est ensuite possible de sélectionner une base de test spécifique (base des vols consistants, base des vols des recherches utilisateurs ou autre) et pour chacune d'elle filtrer les prédictions par destination ou par site marchand. Nous avons ajouter cette possibilité pour identifier les destinations qui diminuent les performances du service de prédiction ou les sites marchands qui donnent de moins bons résultats. Nous pouvons enfin définir le seuil à partir duquel $\mathbb{P}(\varphi_t = 1)$ est considéré comme une baisse de prix et lancer l'affichage des résultats

3. Tout d'abord pour chaque attribut qualitatif de V_i nous affichons un camembert représentant la répartition des différents niveaux dans l'ensemble des vols bien prédits et celui des vols mal prédits. Là encore, nous voulons pouvoir détecter les situations où la prédiction est moins bonne.

4. Ensuite nous avons trois volets décrivant l'évolution de la bonne prédiction avec le nombre de jours avant la date de départ. Tout à gauche, nous affichons le taux de bonnes prédictions global, au centre celui des vols à la baisse et a droite celui des vols à la hausse. Diviser la prédiction globale en deux permet de vérifier que le prédicteur ne prédit pas tout à la hausse, conservant ainsi une bonne prédiction globale tout en étant complètement inutile.

5. Puis nous affichons les courbes ROC à l'instant T_{-21}. A chaque inflexion de la courbe, nous affichons la valeur du seuil associé ainsi que la distance à la diagonale

6. Nous présentons les statistiques pécuniaires qui sont : la sommes de prix à t, à $t + 7j$, la somme du plus bas prix des deux, et celle du plus élevé. La somme des prix si l'utilisateur suit la prédiction, la différence avec le premier prix et la différence avec l'optimal.

7. Enfin nous affichons sur trois nouveaux volets l'évolution de la somme des gains et des pertes en fonction du nombre de jours avant la date de départ, la somme des gains et la somme des pertes. Dans le premier cadran nous afficherons le gain total si l'utilisateur choisit systématiquement d'acheter à l'instant t et le gain total si l'utilisateur prenant toujours la meilleure solution. Cela permet de situer les performances de notre service du point de vue de l'économie de l'internaute et d'affirmer qu'une aide à la décision permet d'économiser de l'argent face à l'achat immédiat.

6.4 Implémentation du service web

Le processus de prédiction ne doit pas ralentir l'expérience utilisateur et doit donc être décorrélé du retour des résultats sur la page des offres. Il doit donc être interrogé comme un service web. Les informations présentées doivent être claires et permettre à l'utilisateur de comprendre l'aide que l'on lui fournit. La prédiction brute n'est pas suffisante : il est important de donner un indice de confiance et de présenter si possible l'historique des prix sur ce trajet. Ainsi l'internaute sera à même de prendre une décision avec ces outils supplémentaires.

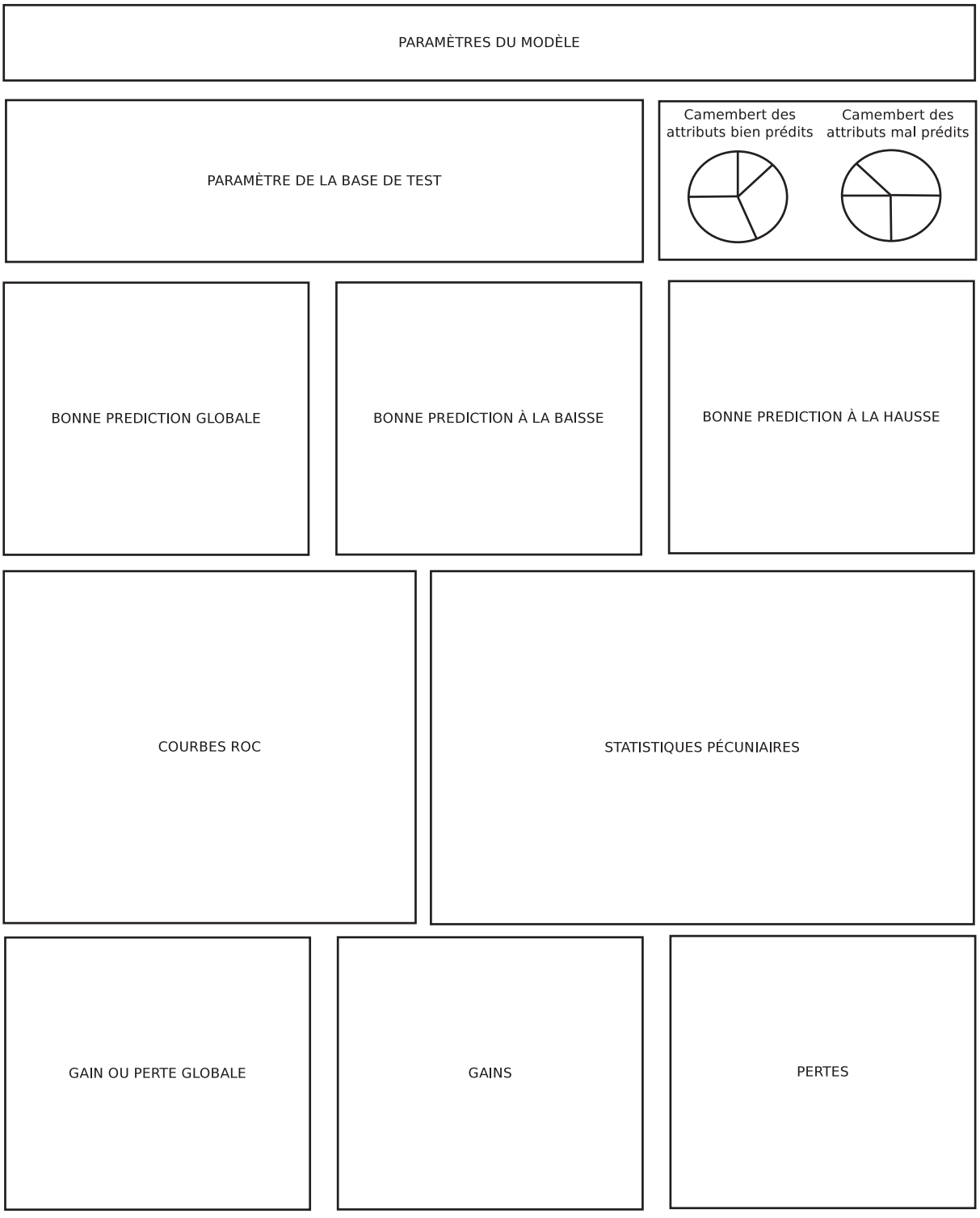


FIGURE 6.7 – Interface de test

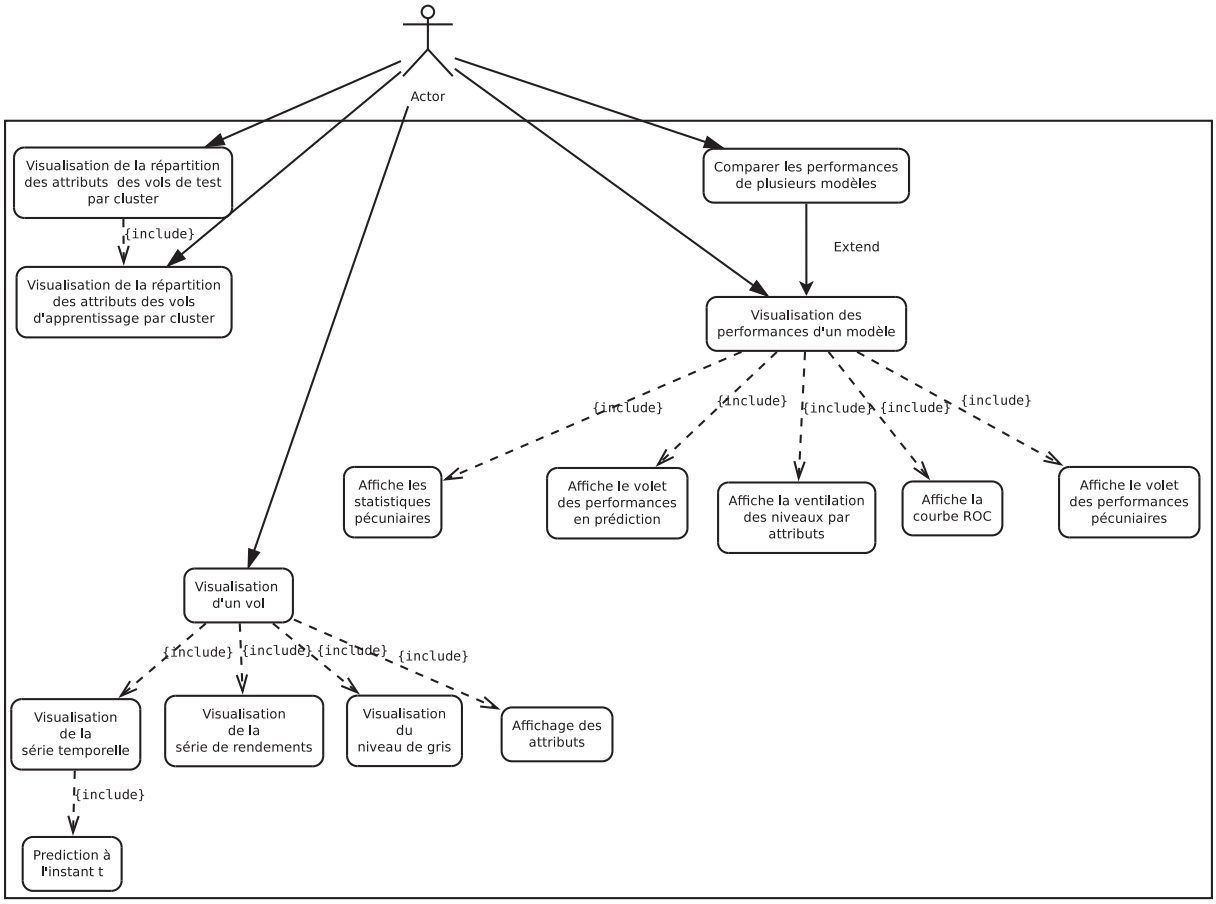


FIGURE 6.8 – Use Case de l'interface de test

6.4.1 Rapidité

Les simulations sont pré-calculées pour toutes les situations possibles : nous calculons φ_t pour les t possible et pour tous les clusters du modèle. Ainsi il est possible de mettre ce tableau $nbCluster \times 90$ (90 étant le nombre de jours maximum que nous gérons actuellement) en mémoire et, une fois la prédiction du groupe faite, extraire le $\mathbb{P}(\varphi_t = 1)$ associé. Les attributs supplémentaires tels que la saison ou le type de site marchand sont calculés aisément à la volée.

6.4.2 Clarté

Pour aider l'utilisateur à prendre sa décision, nous ajoutons au conseil d'achat, la courbe de prix $p_i(t)$ sous la forme d'un graphe et l'évolution présumé de celle-ci. L'évolution prédite correspond à variations du cadran dont la probabilité est la plus importante. Ainsi nous utilisons le pourcentage d'évolution au prix actuel du billet auquel nous ajoutons une variabilité pour enfin afficher une fourchette d'évolution du prix.

6.4.3 Reproductibilité

Enfin il est essentiel de reproduire les prédictions à chaque recherche identique pour ne pas égarer l'utilisateur. Les prédictions étant déterministes, il nous faut maintenant associer une prédiction fixe à chaque cluster et chaque t. Pour cela, ainsi que pour une question de performance, nous avons effectué les simulations de séries à partir des centroïdes du modèle interrogé, et conservé les prédictions issues de ces simulations en mémoire.

6.5 Conclusion et perspectives

Nous avons construit un ensemble de briques applicatives décoréllées permettant la modification des technologie et des algorithmes utilisés sans la refonte complète de l'architecture. Un premier composant, le moteur de recherche, effectue les recherches de vols, log les résultats et lorsque l'utilisateur le souhaite fait une demande de prédiction au second composant. Celui-ci, le service de prédiction, est une application web composé d'un service XML capable de donner des prédictions en fonction d'attributs et de contrôler ses mêmes prédictions 7 jours plus tard. Le fichier XML est généré par le troisième composant, l'application *R*, qui aura utilisé les données historiques stockées dans le dernier composant, la base *SQL*.

L'application *R* a elle aussi était conçu pour permettre une flexibilité et une rapidité d'exécution nécessaire à la génération du fichier PMML. La création de ce fichier se fait par étapes successives, chacune décorélée de la précédente et identifiée par une clef unique dépendante des paramètres de l'étape courante et des étapes précédentes. Ainsi il est possible de combiner plusieurs approches sans réitérer les calculs déjà effectués.

Enfin une interface de test a été implémentée afin de comparer les performances de nos différentes approches. Comme évoqué dans le chapitre précédent certaines problématiques se pose quant au choix de la bonne métrique de comparaison. Selon le seuil choisi les résultats peuvent être très différents et l'équilibre entre le taux de bonnes prédictions et le gain moyen par billets est souvent délicat à trouver. Il serait intéressant d'étudier d'autres mesure de performance afin

de réduire le temps passé à comparer les différents algorithmes.

Nous souhaitons à l'avenir nous migrer la génération des modèles sur des framework open-source tel qu'*Apache Hadoop* et son framework d'algorithme de fouille de données *Apache Mahout* tout comme l'adaptation du modèle au fil de l'évolution des données de la base d'apprentissage. L'augmentation constante de la base de données nécessite par ailleurs l'optimisation de la structure de notre application, comme par l'exemple le passage au *NoSQL* afin d'exploiter plus facilement les informations des premiéres évolutions de prix.

Bibliographie

[1] Wesam Barbakh and Colin Fyfe. Online clustering algorithms. International Journal of Neural Systems, 18(03) :185–194, 2008.

[2] Peter P Belobaba. Survey paper—airline yield management an overview of seat inventory control. Transportation Science, 21(2) :63–73, 1987.

[3] Jürgen Beringer and Eyke Hüllermeier. Online clustering of parallel data streams. Data & Knowledge Engineering, 58(2) :180–204, 2006.

[4] Simon Bernard. Forêts Aléatoires : De l'Analyse des Mécanismes de Fonctionnement à la Construction Dynamique. These, Université de Rouen, December 2009.

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth and Brooks, Monterey, CA, 1984.

[6] Leo Breiman. Bagging predictors. Machine learning, 24(2) :123–140, 1996.

[7] Leo Breiman. Bagging predictors. Mach. Learn., 24(2) :123–140, August 1996.

[8] Leo Breiman. Bias, variance, and arcing classifiers. 1996.

[9] Leo Breiman. Random Forests. Machine Learning, 45(1) :5–32, October 2001.

[10] Jan K. Brueckner. International airfares in the age of alliances : The effects of codesharing and antitrust immunity. The Review of Economics and Statistics, 85(1) :105–118, 2003.

[11] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. Communications in Statistics-theory and Methods, 3(1) :1–27, 1974.

[12] William W. Cohen. Fast effective rule induction. In In Proceedings of the Twelfth International Conference on Machine Learning, pages 115–123. Morgan Kaufmann, 1995.

[13] D.R. Cox and H.D. Miller. The theory of stochastic processes. Wiley publications in statistics. Wiley, 1965.

[14] D. J. Daley and D. Vere-Jones. An introduction to the theory of point processes. Vol. I. Probability and its Applications (New York). Springer-Verlag, New York, second edition, 2003. Elementary theory and methods.

[15] S. Daudel and G. Vialle. Yield management : applications to air transport and other service industries. Paris : les Presses de l'Institut du transport aérien, 1994.

[16] Thomas G. Dietterich. Ensemble methods in machine learning. In Proceedings of the First International Workshop on Multiple Classifier Systems, MCS '00, pages 1–15, London, UK, UK, 2000. Springer-Verlag.

[17] B. Efron and R. J. Tibshirani. An Introduction to the Bootstrap. Chapman & Hall, New York, 1993.

[18] Oren Etzioni, Rattapoom Tuchinda, Craig A. Knoblock, and Alexander Yates. To buy or not to buy : mining airfare data to minimize ticket purchase price. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '03, pages 119–128, New York, NY, USA, 2003. ACM.

[19] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In Computational learning theory, pages 23–37. Springer, 1995.

[20] Tak-chung Fu, Fu-lai Chung, Vincent Ng, and Robert Luk. Pattern discovery from stock time series using self-organizing maps. In Workshop Notes of KDD2001 Workshop on Temporal Data Mining, pages 26–29. Citeseer, 2001.

[21] Robin Genuer. Forêts aléatoires : aspects théoriques, sélection de variables et applications. PhD thesis, Université Paris Sud-Paris XI, 2010.

[22] Greg Hamerly and Charles Elkan. Alternatives to the k-means algorithm that find better clusterings. In Proceedings of the eleventh international conference on Information and knowledge management, CIKM '02, pages 600–607, New York, NY, USA, 2002. ACM.

[23] T. Hastie, R. Tibshirani, and J. H. Friedman. The Elements of Statistical Learning. Springer, corrected edition, July 2003.

[24] Gareth R Horton and Great Britain. Forecasts of CO2 emissions from civil aircraft for IPCC. DTI, 2006.

[25] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. Unbiased recursive partitioning : A conditional inference framework. JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS, 15(3) :651–674, 2006.

[26] Paul Jaccard. The distribution of the flora in the alpine zone. 1. New Phytologist, 11(2) :37–50, 1912.

[27] Annaka Kalton, Pat Langley, Kiri Wagstaff, and Jungsoon Yoo. Generalized clustering, supervised learning, and data assignment. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pages 299–304. ACM, 2001.

[28] Leonard Kaufman and Peter J Rousseeuw. Finding Groups in Data : An Introduction to Cluster Analysis. Wiley, 2005.

[29] Jacob Kogan. Introduction to Clustering Large and High-Dimensional Data. Cambridge University Press, New York, NY, USA, 2007.

[30] Miroslav Kubat. Reinforcement learning by ag barto and rs sutton, mit press, cambridge, ma 1998. Knowl. Eng. Rev., 14(4) :383–385, December 1999.

[31] Anthony Owen Lee. Airline reservations forecasting : Probabilistic and statistical models of the booking process. Technical report, Cambridge, Mass. : Flight Transportation Laboratory, Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology,[1990], 1990.

[32] Friedrich Leisch and Kurt Hornik. Stabilization of k-means with bagged clustering. In Proceedings of the 1999 Joint Statistical Meetings, Statistical Computing Section, pages 174–179, 1999.

[33] T. Warren Liao. Clustering of time series data - a survey. Pattern Recognition, 38(11) :1857–1874, 2005.

[34] Marina Meilă and David Heckerman. An experimental comparison of several clustering and initialization methods. In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, pages 386–395. Morgan Kaufmann Publishers Inc., 1998.

[35] Hugh Miller, Sandy Clarke, Stephen Lane, Andrew Lonie, David Lazaridis, Slave Petrovski, and Owen Jones. Predicting customer behaviour : The university of melbourne's kdd cup report. Journal of Machine Learning Research - Proceedings Track, 7 :45–55, 2009.

[36] Glenn W Milligan and Martha C Cooper. An examination of procedures for determining the number of clusters in a data set. Psychometrika, 50(2) :159–179, 1985.

[37] Marc Möller and Makoto Watanabe. Advance purchase discounts versus clearance sales*. The Economic Journal, 120(547) :1125–1148, 2010.

[38] T. K. Moon. The expectation-maximization algorithm. IEEE Signal Processing Magazine, 13(6) :47–60, November 1996.

[39] E. Poutier and P. Legohérel. Revenue Management - Optimisation des ventes dans les Services : Optimisation des ventes dans les Services. Marketing - Communication. Dunod, 2011.

[40] J. Ross Quinlan. Induction of decision trees. Machine learning, 1(1) :81–106, 1986.

[41] John Ross Quinlan. C4. 5 : programs for machine learning, volume 1. Morgan kaufmann, 1993.

[42] William M Rand. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association, 66(336) :846–850, 1971.

[43] Sidney Resnick. Adventures in stochastic processes. Birkhäuser Boston Inc., Boston, MA, 1992.

[44] CT Shaw and GP King. Using cluster analysis to classify time series. Physica D : Nonlinear Phenomena, 58(1) :288–298, 1992.

[45] B. Smith, J. Leimkuhler, R. Darrow, and Samuels. Yield management at American Airlines. Interfaces, 22(1) :8–31, 1992.

[46] Barry C Smith, John F Leimkuhler, and Ross M Darrow. Yield management at american airlines. Interfaces, 22(1) :8–31, 1992.

[47] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures : Illustrations, sources and a solution. BMC Bioinformatics, 8(25), 2007.

[48] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a dataset via the gap statistic. 63 :411–423, 2000.

[49] Laura Tolosi and Thomas Lengauer. Classification with correlated features : unreliability of feature ranking and solutions. Bioinformatics, 27(14) :1986–1994, 2011.

[50] Michail Vlachos, Jessica Lin, Eamonn Keogh, and Dimitrios Gunopulos. A wavelet-based anytime algorithm for k-means clustering of time series. In In Proc. Workshop on Clustering High Dimensionality Data and Its Applications. Citeseer, 2003.

[51] Lawrence R Weatherford and Samuel E Bodily. A taxonomy and research overview of perishable-asset revenue management : yield management, overbooking, and pricing. Operations Research, 40(5) :831–844, 1992.

[52] Yimin Xiong and Dit-Yan Yeung. Mixtures of arma models for model-based time series clustering. In Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on, pages 717–720. IEEE, 2002.

[53] Mingjin Yan. Methods of Determining the Number of Clusters in a Data Set and a New Clustering Criterion. PhD thesis, Virginia Polytechnic Institute and State University, 2005.

[54] W. J. Youden. Index for rating diagnostic tests. Cancer, 3(1) :32–35, 1950.

[55] Dell Zhang and Yisheng Dong. Semantic, hierarchical, online clustering of web search results. In Advanced Web Technologies and Applications, pages 69–78. Springer, 2004.

Méthodes de fouilles de données pour la prédiction de l'évolution du prix d'un billet et application au conseil à l'achat en ligne

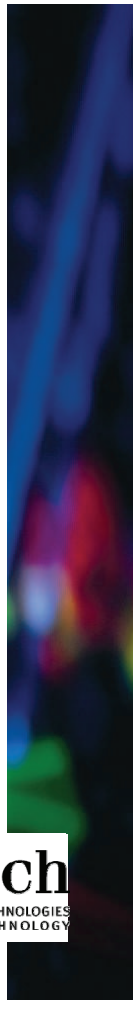
Till WOHLFARTH

RESUME : Nous nous intéressons au problème de la prédiction de l'occurrence d'une baisse de prix pour fournir un conseil à l'achat immédiat ou reporté d'un voyage sur un site web de comparaison des prix. La méthodologie proposée repose sur l'apprentissage statistique d'un modèle d'évolution du prix à partir de l'information conjointe d'attributs du voyage considéré et d'observations passées du prix et de la "popularité" celui-ci. L'originalité principale consiste à représenter l'évolution des prix par le processus ponctuel inhomogène des sauts de celui-ci. A partir d'une base de données constituée par liligo.com, nous mettons en oeuvre une méthode d'apprentissage d'un modèle d'évolution des prix. Ce modèle permet de fournir un prédicteur de l'occurrence d'une baisse du prix sur une période future donnée et donc de prodiguer un conseil d'achat ou d'attente au client.

MOTS-CLEFS : Fouille de données, Prédiction, Apprentissage, Segmentation

ABSTRACT : The goal of this paper is to consider the design of decision-making tools in the context of varying travel prices from the customer's perspective. Based on vast streams of heterogeneous historical data collected through the internet, we describe here two approaches to forecasting travel price changes at a given horizon, taking as input variables a list of descriptive characteristics of the flight, together with possible features of the past evolution of the related price series. Though heterogeneous in many respects (e.g. sampling, scale), the collection of historical prices series is here represented in a unified manner, by marked point processes (MPP). State-of-the-art supervised learning algorithms, possibly combined with a preliminary clustering stage, grouping flights whose related price series exhibit similar behavior, can be next used in order to help the customer to decide when to purchase her/his ticket.

KEY-WORDS : Data Mining, Prediction, Machine Learning, Clustering



TELECOM ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY