UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Master 2 Big Data et fouille de données

Fouille de données

Date : 23/01/2017

Etude et analyse de thèse
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

Présentée et soutenue publiquement par

Till WOHLFARTH

Réalisé par

Nabil Redha BELKHOUS

Numéro d'étudiant : 16705491

Email : nbelkhous@gmail.com

Enseignants

Nedra Mellouli

Gilles Bernard

Table des matières

1 Introduction

Ces dernières années, l'achat de billets d'avions sur internet a considérablement augmenté et beaucoup de sites de vente de voyage en ligne ont fait leur apparition.

LiliGo.com qui est un site de vente de voyages en ligne a voulu se démarquer en proposant une aide à la décision concernant l'achat de voyage. Le but est de proposer à l'utilisateur un service qui le conseillerait au mieux pour acheter un billet d'avion lorsqu'il fait ses recherches.

Ce service sera activé à chaque recherche utilisateur effectuée sur le moteur de recherche du site et pour le billet choisi, on lui prodigue un conseil d'achat, car les prix vont augmenter dans les jours qui suivent, ou un conseil d'attente car les prix des billets vont baisser.

Donc le rôle du service est de prédire l'évolution des prix de billet à un **instant t** et non pas de prédire le meilleur moment d'achat.

Avec ce service, on aura pour objectif de conseiller le client sur l'achat immédiat ou non du billet recherché tout en lui faisant économiser de l'argent.

Pour réaliser ce service, on a pensé à mettre en place plusieurs méthodes de fouille de données qui vont permettre de prédire l'évolution du prix d'un billet.

Le but de ma démarche en choisissant cette thèse est d'apprendre de nouvelles méthodes de fouilles de données.

L'intérêt du projet et la multitude de méthodes de fouilles de données utilisés ont fait que j'ai choisi cette thèse « **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** » pour vous la présenter.

De par ce document je présenterai :

— La collecte et structure de données utilisées
— La représentation de ces données (Modélisation des données)
— Les méthodes de fouilles de données utilisées (Clustering et classification des séries de prix)
— Les résultats obtenus avec les différentes méthodes de fouilles de données
— Les méthodes retenues
— L'évolution des performances du service d'aide à la décision dans le temps.

2 Collecte et structure de données utilisées

2.1 Collecte des données

LiliGo.com dispose d'un moteur de recherche de voyage qui permet de sélectionner les milliers de vols qu'il récupère à partir de site d'agence de voyages (Expedia, GoVoyages, TravelGenio, BudgetAir, etc ...) et des compagnies aériennes (Air France, EasyJet, Qatar airways, etc ...) afin d'offrir à l'utilisateur le prix du billet le moins cher en comparant le prix provenant de différentes sources.

Les données collectées sont issues de plusieurs provenances :

- Il y a les recherches utilisateurs qui permettent d'avoir la ville ou l'aéroport de départ et d'arrivée ainsi que la date de départ et de retour.
- Nous avons aussi les résultats de ces recherches qui nous donnent des informations sur les sites qui vendent le billet, la compagnie aériennes, les horaires de vol et les prix.
- Il y aussi les alertes (recherches automatiques quotidiennes envoyées à son adresse mail) qui peuvent être définies par l'utilisateur. Elles permettent d'obtenir un certain nombre de résultats de recherches échantillonnés (les 15 meilleurs résultats).

Sachant qu'une recherche donne lieu à plusieurs résultats (en moyenne 300) on est arrivé à construire un base de données volumineuse (sur un historique de recherche de 6 ans) qui nous a permis de construire notre base d'apprentissage et de test.

2.2 Structure des données utilisées

Afin d'extraire les informations et structurer l'ensemble de nos données nous avons construit une base de données qui contient trois tables principales :

- vol : qui est caractérisé par : un aéroport de départ (*departureStation*), un aéroport d'arrivée (*arrivalStation*), 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*).

  N-uplet vol : { *id_unique_flight*, aéroport de départ, aéroport d'arrivée, date de départ, date de retour, code transporteur aller, code transporteur retour*}
- série de prix : qui est caractérisée par : un vol, un site vendeur (provider) et un prix (correspondant à un instant t). Chaque ligne de la table correspond à un vol unique, un site marchand, le prix à un instant t qui est la série $P_i(t)$. Sachant qu'un vol peut être vendu à la fois par n agences différentes, on aura donc n séries distinctes.

  On conclut qu'un vol a plusieurs prix selon le revendeur. On obtient donc pour un vol une série de prix.

  N-uplet séries de prix : {*id_flight*, id_unique_flight , provider, prix*} Ainsi pour construire la série de prix pour un vol, il suffira de détecter les id_flight pour chaque vol (id_unique_flight) et extraire le prix correspondant. Voilà à quoi ressemble une série de prix pour un vol
- Attributs : À chaque vol est associé un vecteur d'attributs regroupant toutes les informations possibles sur le billet (résultat de la recherche). Les attributs sont très utiles pour extraire des statistiques sur la base d'apprentissage (ex : destination la plus populaire en Asie, prix min observé depuis Paris vers une autre ville). C'est à partir de ces attributs que la prédiction de l'évolution des prix pourra être faite.

  Les attributs seront divisés en quatre catégories :

  1. Les attributs qui définissent un vol (*day, month, year, DepartureHour, DepartureMinute, TransportCode, DepartureStation, ArrivalStation*)
  2. Attributs dérivés des attribut qui définissent un vol : attributs temporelles (*Seanson, length_of_stay, ...*), attributs géographiques (*departureCity, departureCountry, ...*), attributs liés au trajets (*stops, distance, ...*)
  3. Attributs liés au site marchand (*Provider, Type, ...*)
  4. Attributs contextuels qui évoluent avec les points de la série temporelle (*Volatilty, Volatility_hausse, ...*)

  Ces attributs vont être stockés dans une table avec *id_flight* et *l'instant t* comme clé primaire.

Grâce à l'architecture de la base de données construite, on peut extraire de nombreuses caractéristiques (les séries de prix, leurs attributs et toutes les statistiques dérivées possibles) pour construire le modèle de prédiction.

Pour étudier correctement les comportements des séries de prix, il faut détecter le maximum de comportements différents tout en minimisant la taille volumineuse de la base de données. Nous avons sélectionné les vols qui sont les plus recherchés (ceux dont la date de départ approche). Nous avons constaté que la période où les prix variés le plus était lors du dernier mois avant le départ. On a donc décidé de travailler avec les **28 derniers jours** afin de sélectionner les différents comportements des prix.

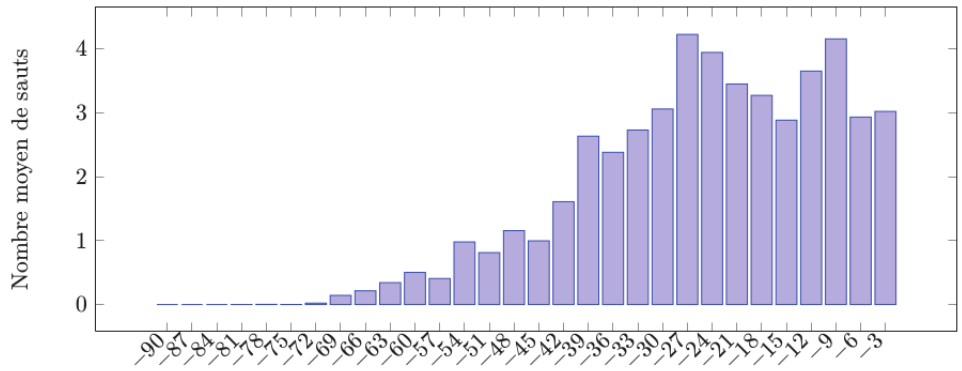

FIGURE 1 – Nombre de saut par rapport au nombre de jours avant le départ (tranches de 3 jours) [1]

3 Représentation des données (modélisation)

Afin de classifier les comportements des prix des vols nous sommes amenés à les comparer.
Faire une classification permettrait d'extraire des groupes de comportements similaires.
À l'état actuel, nous ne sommes pas capables de comparer les séries de prix, il est donc nécessaire de les transformer en une représentation qui permettrait de faire cette comparaison pour les classifier selon leur comportement.

Pour la comparaison entre les séries, on ne doit pas prendre en considération l'ordre de grandeur du prix mais son évolution (hausse et baisse).

Modélisation :

Nous avons vu précédemment que les prix d'un vol constituent une série de prix dans le temps.

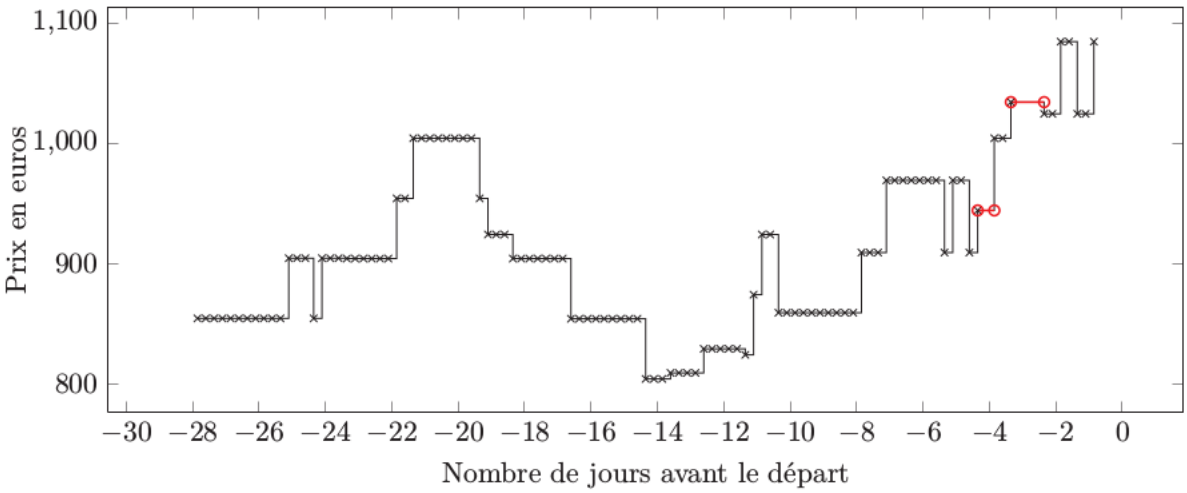

FIGURE 2 – Time Series : Paris-Bangkok départ le 11/01/2013 pour 14 jours par Qatar [1]

Il a été supposé que le nombre d'apparitions de changements de prix suit une distribution de poisson.

Pour faire abstraction de la grandeur des prix et en se basant sur la supposition précédente, on a transformé les séries de prix temporelles en séries de rendements par des processus ponctuels dans le plan

temps-rendement (réalisé avec le framework COX [1]). On corrigera les fluctuations anormales de prix qui sont de faibles variations (qui ajoutent du bruit à la courbe).

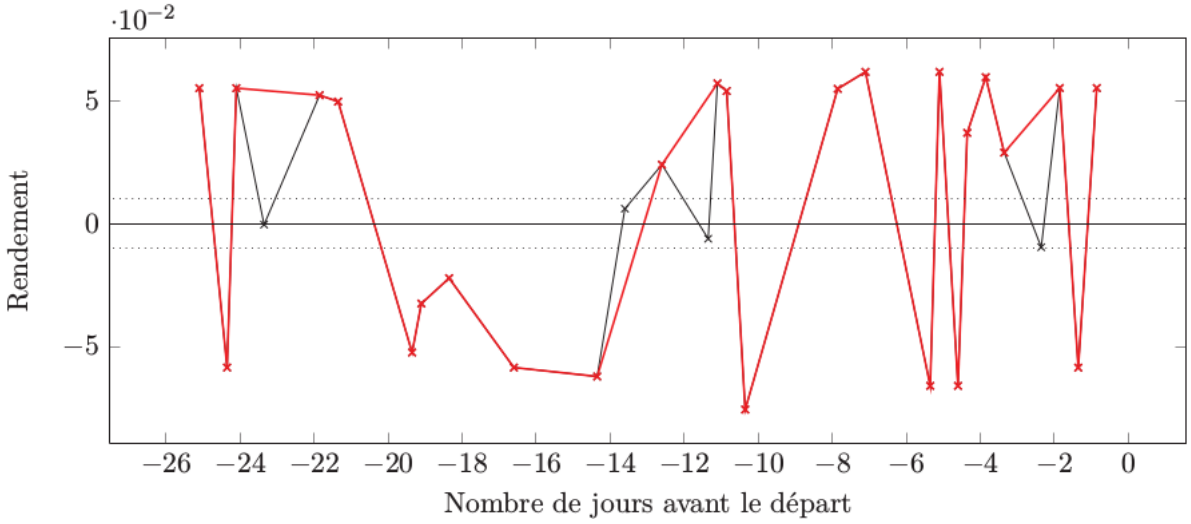

FIGURE 3 – Série de rendements après filtre : Paris-Bangkok départ le 11/01/2013 pour 14 jours par Qatar [1]

À la suite de cela, on a modélisé les séries de rendements par une estimation de l'intensité sous forme de niveau de gris. C'est grâce à la représentation sous format de niveaux de gris qu'on pourra comparer les séries temporelles de prix et les classifier.

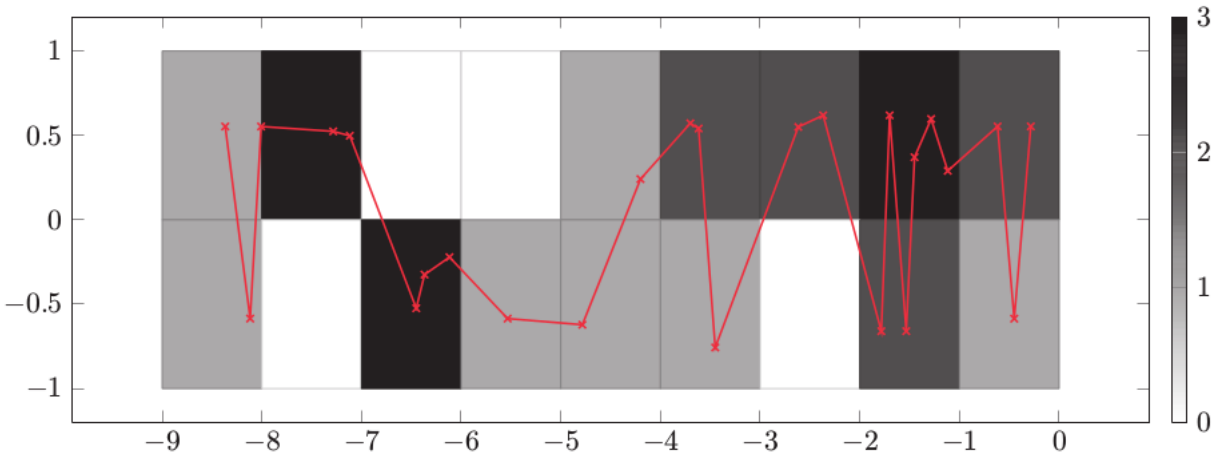

FIGURE 4 – Niveau de gris après filtre : Paris-Bangkok départ le 11/01/2013 pour 14 jours par Qatar [1]

Une information très importante qu'il faut souligner, c'est qu'à partir d'un niveau de gris, on peut reproduire la série de prix en faisant une simulation.

4 Méthodes de fouilles de données utilisées

Afin de pouvoir comparer les séries temporelles des prix, il fallait les modéliser en séries de rendements puis en niveaux de gris.

Maintenant, à partir des niveaux de gris nous allons appliquer un apprentissage non supervisé pour créer des groupes de comportements similaires qui contiennent chacun un comportement type.

4.1 Clustering (Apprentissage supervisé)

Dans le but d'utiliser la méthode de clustering la plus appropriée à notre structure de données nous avons choisi de travailler avec trois méthodes d'apprentissage non supervisé et voir la méthode qui présente la meilleure segmentation.

4.1.1 L'algorithme K-means

K-means est un algorithme **d'apprentissage non supervisé** qui appartient à la famille des segmentations à optimisations itératives en deux étapes : création du modèle et ré-attribution des données [1].

L'algorithme est initialisé avec un **nombre de départs** (nstar : le nombre d'initialisations aléatoires des paramètres internes de l'algorithme où on va choisir la meilleure initialisation pour éviter qu'il converge vers un optimum local) et le **nombre de groupes** (application de métrique comme **l'indice de Calinski and Harabasz** pour obtenir le nombre le plus optimal).

Il choisira aléatoirement le premier centroïde de chaque classe pour optimiser ce choix à chaque itération jusqu'à la stabilisation ou attendre le nombre maximum d'itérations . Son but est de regrouper les niveaux de gris (représentant les séries de prix) qui ont un comportement similaire autour d'un comportement type (centroïde). Il est à savoir qu'on évalue la similarité entre deux séries temporelles avec la **distance euclidienne case à case des niveaux de gris**.

4.1.2 L'algorithme Bagged K-means

Bagged K-menas est une version améliorée de K-means qui permet de **minimiser l'influence de l'initialisation**. Le principe est d'appliquer l'algorithme K-means sur un nombre de sous-ensembles bootsrapés (principe de ré-échantillonnage statistique) de la base d'apprentissage et d'agréger ensuite les résultats de chaque sous-ensemble (Bagging aggregating).

Il permet entre autres d'augmenter les chances d'obtenir le maximum global sans ce soucier de l'influence de l'initialisation [1].

4.1.3 L'algorithme Espérance-Maximisation

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 non observées [1].

EM est souvent utilisé pour la segmentation de données. Il comporte deux étapes :

- Une étape d'évaluation de l'espérance E .
- Une étape de maximisation de la vraisemblance M .

On utilise les paramètres trouvés dans l'étape de maximisation comme entrée de l'étape de l'évaluation de l'espérance à l'itération suivante.

Les données d'entrée sont les mêmes que pour l'algorithme K-means (niveau de gris) et l'initialisation des paramètres (I_k = centroïde et α_k = Probabilité d'appartenance à un groupe) peut se faire de deux manières différentes :

- Attribution aléatoire d'un numéro de groupe à chaque X_i .
- Initialiser les paramètres en appliquant le K-means.

Pour α_k , dans les deux méthodes d'initialisation, nous procédons de la manière suivante :
$\alpha_k = \frac{cardX_i \in y}{n}$ [1].

Espérance-Maximisation modélisera chaque classe par une distribution de probabilité et groupe les niveaux de gris afin de maximiser la vraisemblance aux centroïdes.

Contrairement à K-means, en sortie on n'aura pas d'étiquette Y_i directement, on appliquera alors une fonction la vraisemblance à α_y et I_y pour déterminer à chaque vol i son étiquette.

4.2 Classification (Apprentissage non supervisé)

Après l'étape de segmentation où on a classifié les séries de prix des vols selon leurs comportements et avons déterminé les comportements types de chaque groupe, nous voulons être capables de donner un comportement parmi ceux trouvés dans la base d'apprentissage aux nouveaux résultats de recherches en se basant sur les attributs des vols d'apprentissages. Ceci dans le but de pouvoir prédire le comportement d'un vol.

Pour cela, un apprentissage supervisé basé est appliqué aux données segmentées dans lesquelles chaque vol possède une étiquette et une série d'attributs sur lesquels on créera des règles de classifications pour pouvoir affecter les vols de la base de test à leurs groupes le plus vraisemblable.

Là aussi, on teste plusieurs méthodes de classification afin de trouver celle qui est la plus appropriée à notre structure de données. Nous avons choisi de travailler avec quatre méthodes d'apprentissage supervisé basées sur les arbres de décision, et de voir la méthode qui présente la meilleure classification et qui offre le meilleur modèle de prédiction.

4.2.1 Arbres de décision : CART & C4.5

CART : c'est un algorithme basé sur les arbres de décision qui découpent l'espace des attributs récursivement en maximisant à chaque étape le gain d'information [1]. L'algorithme se déroule en trois étapes : la construction de l'arbre, l'arrêt de la construction et l'élagage.

Parmi ces avantages on peut citer qu'il obtient un arbre binaire facilement interprétable où il suffit de lire l'arbre en remontant depuis les feuilles pour obtenir les règles de classification. Il supporte à la fois les données qualitatives et quantitatives. De plus le temps de calcul sur de grands volumes de données reste faible et sans dégradation des performances. À l'inverse on peut dire en ce qui concerne ses points faibles qu'il sur-apprend les données d'apprentissage et devient mauvais en généralisation.

C4.5 : il reprend le principe de construction d'un arbre de décision de CART avec deux différences[1] :

- CART utilise **l'indice de Gini** comme critère de sélection d'attributs alors que C4.5 utilise **l'entropie croisée**.
- l'arbre généré n'est pas binaire (le nombre de branches est en relation avec les niveaux de variables qualitatives et l'intervalle des variables discrètes).

On peut réduire l'erreur d'un classifieur peu performant en combinant les prédictions de plusieurs classifieurs de mêmes types entraînés sur les mêmes données. La combinaison de plusieurs classifieurs permet de fiabiliser les décisions, élargir l'ensemble des solutions possibles, éviter les optima locaux et traiter des problèmes trop complexes pour un simple classifieur. En définitive, CART et C4.5 permettent de faire une segmentation de l'espace des attributs et d'extraire des règles de prédiction des comportements en se basant sur les attributs.

4.2.2 Adaboost

Adaboost exécute d'une manière itérative CART sur la base d'apprentissage pondérée différemment. La pondération améliore la prédiction des vols 'difficiles' (mal classés) en leur donnant plus de poids [1].

Adaboost est un algorithme d'arbre de décision qui utilise la technique de boosting basée sur la sélection itérative de classifieurs faibles (CART). L'idée est d'améliorer les compétences d'un faible classifieur sachant que le boosting est réalisé sur la base des erreurs de prédictions en apprentissage sur ce dernier. Il a l'avantage de réduire les variances et le biais de prévision, il ne nécessite pas un long pré-traitement et ne nécessite pas un réglage fin des paramètres.

4.2.3 Forêts aléatoires (Random Forest)

C'est un algorithme qui est basé sur la combinaison de classifieurs élémentaires de types arbre de décision et qui va agréger leurs résultats de prédictions pour avoir une prédiction finale. Ils utilisent la technique du bagging vue précédemment.

Dans les Random Forest, la sélection des sous-ensembles de variables et la construction des classifieurs sont faites aléatoirement.

Pour contourner le problème de sur-apprentissage, il faut multiplier le nombre d'arbres. Il suffit de donner en entrée comme paramètres : la taille des sous-ensembles de variables et le nombre d'arbres.

Cette méthode donne de meilleurs résultats que les autres classifieurs car elle est robuste au sur-apprentissage, augmente les performances en généralisation grâce au bagging et ne nécessite pas de validation croisée ou d'une base de tests indépendante pour estimer l'erreur en test.

Le meilleur avantage de cette méthode est le fait de fournir un classement des attributs les plus importants (feature selection) dans la classification ce qui permettra d'alléger le processus de prédiction en travaillant uniquement sur les variables les plus influentes.

5 Résultats expérimentaux

Nous rappelons que le but du service d'aide à la décision n'est pas seulement de prédire les hausses et les baisses d'un billet d'avion à un instant t, mais aussi de procurer un gain d'argent à l'utilisateur.

Afin de mesurer la qualité de la prédiction, on a utilisé principalement entre autres deux méthodes :

- Matrice de confusion : métrique qui permet de mesurer la qualité de classification.
- Courbes ROC (Receiver Operating Characteristic) : mesure l'évaluation d'un classifier binaire en fonction d'un seuil discriminant.

On a défini pour cela les notions : TP = Vrai positif (correcte prédiction d'une baisse), TN = Vrai négatif (correcte prédiction d'une hausse), FP = Faux positif (fausse prédiction d'une baisse), FN = Faux négatif (fausse prédiction d'une hausse) On construit la matrice de confusion avec le nombre de TP, TN, FP et FN.

		Réalité	
		0	1
Prédit	0	TN	FP
	1	FN	TP

FIGURE 5 – Matrice de confusion [1]

Dans notre cas, la matrice obtenue ne sera pas toujours équilibrée car il y a plus de hausse de prix que de baisse de prix (70% des cas sont des hausses de prix) et l'importance d'un FN n'est pas la même qu'un FP car si on prédit une baisse et qu'il y a une hausse on risque de faire perdre de l'argent au client contrairement à la prédiction d'une hausse de prix qui est finalement une baisse. C'est pour cela qu'on doit minimiser le nombre de FP.

Grâce aux matrices de confusions on peut visualiser le pourcentage global de bonnes prédictions au cours du temps. Le taux global de bonne prédiction se calcule comme suit : $f_{global}(t) = \frac{TP+TN}{TP+TN+FP+FN}$ [1] . On peut aussi déterminer la probabilité de détecter une hausse correctement : $f_{spec}(t) = \frac{TN}{FP+TN}$ [1] et calculer aussi la proportion de baisse détectée parmi les vols : $f_{sens}(t) = \frac{TP}{TP+FN}$ [1] .

Pour garantir un bon taux global de prédiction en gardant un équilibre entre les détections de hausse et de baisse il faut déterminer un seuil de prédiction s. Ce seuil de prédiction sera calculé grâce aux courbes ROC (Receiver Operating Characteristic) en calculant l'index de Youden ou la distance diagonale (utilisée dans notre cas).

En utilisant les matrices de confusion, on a évalué les gains et les pertes pour avoir une idée de ces derniers en fonction de la qualité de la prédiction. La matrice de confusion fût calculée comme suit :

$d = fisrtPrice - lastPrice$ avec $fisrtPrice$: prix du billet au moment de la prédiction et $lastPrice$: prix du billet à la fin de la fenêtre de la prédiction.

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

FIGURE 6 – Matrice de confusion des gains et pertes [1]

3 courbes ont été construites : le gain global, le gain moyen par vol et la perte moyenne par vol.

Mais nous avons constaté à travers l'évaluation de ses courbes qu'une bonne prédiction ne veut pas dire un gain sur le prix initial.

Il faudra donc trouver un équilibre entre le taux de bonnes prédictions et l'augmentation de gain.

5.1 Résultats de segmentation

Pour la comparaison des résultats il a été conclu qu'on regarderait les performances des différentes méthodes à -21 jours de la date de départ. Pour cette phase de classifications nous avons utilisé des Random Forest avec 600 arbres.

5.1.1 K-means

Comme vue précédemment, le nombre de départs **nstar** et le nombre de clusters influencent la qualité de la segmentation. C'est pour cela qu'il faut faire des choix optimaux.

Plus le nombre de départs est grand plus il y a de chances d'avoir un meilleur clustering mais cela se répercute sur le temps d'exécution de K-means qui augmente. Après avoir fait plusieurs tests, on constate que l'impact de nstar sur le taux de bonnes prédictions en fonction de la date de départ (-21 jours) se stabilise rapidement. Avec nstar = 10 et nstar = 100 il y a une légère différence et à partir de 100 il n'y a plus d'influence sur le pourcentage de bonnes prédictions.

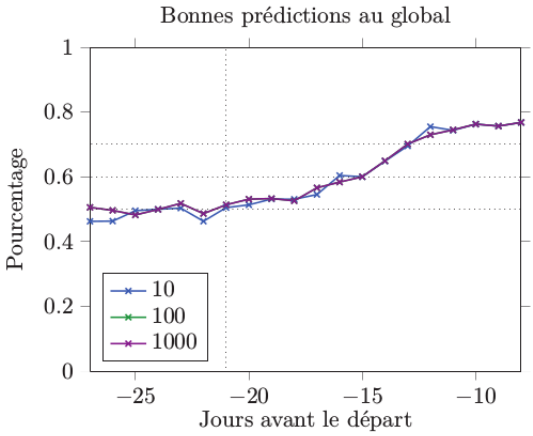

FIGURE 7 – K-Means - Évolution du pourcentage de bonnes prédictions en fonction de la date avant le départ et courbe ROC à -21 jours pour différents nstart [1]

Le deuxième paramètre qui est le nombre de clusters peut-être défini d'une manière optimale avec l'indice de Calinski-Harabasz, RAND, ...

En faisant des tests avec un nombre nstar =100 et les nombres de clusters 5,10,30 et 40 on obtient les résultats suivants :

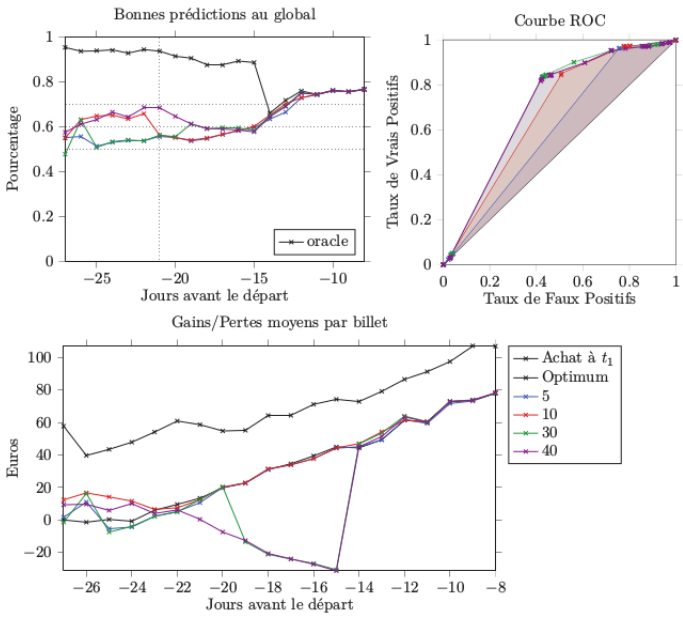

FIGURE 8 – K-Means - Évolution du pourcentage de bonnes prédictions en fonction de la date avant le départ et courbe ROC à -21 jours pour différents nombre de clusters [1]

On peut voir que la prédiction finale avec K-means conserve un taux de bonne prédiction stable à partir de 10 clusters. On remarquera que plus il y a de groupes plus le taux de prédiction est meilleur mais cela n'implique pas un gain comme on peut le voir avec 40 clusters (Meilleurs taux avec presque 70% mais fait perdre 2 euros à l'utilisateur).

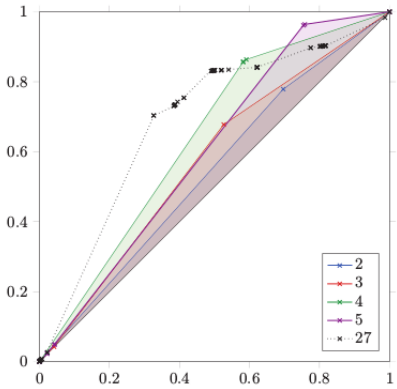

FIGURE 9 – Courbe ROC pour un K-Means à 2, 3, 4 et 5 clusters comparées aux K-Means à 27 groupes [1]

Au vu des résultats observés dans les courbes précédentes, il semble que K-means avec 27 groupes et nstar = 100 donne de bons résultats (taux de prédiction et gain pour l'utilisateur).

5.1.2 Bagged K-means

Comme on a pu le voir précédemment, Bagged K-menas réduit l'influence des initialisations de K-means et favorise la convergence vers un optimum global. Malgré cette réduction d'influence, il donne légèrement de moins bons résultats (taux de prédiction et gain utilisateur) que K-means.

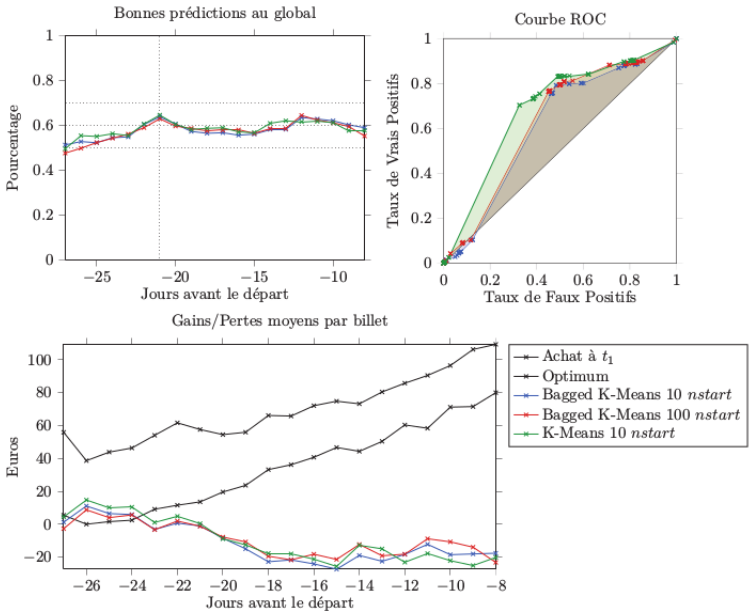

FIGURE 10 – Comparaison des Bagged K-Means avec un K-Means [1]

5.1.3 EM

EM est très sensible à la variation contrairement à K-means. Lorsqu'EM est paramétré d'une manière optimale, il donne de meilleurs résultats (Taux de prédiction et gain utilisateurs) que K-means. Il faut donc tester les différents choix de clusters afin de trouver le meilleur résultat.

Pour ce qui est de l'initialisation de EM, nous avions dit qu'il était possible soit de l'initialiser aléatoirement soit avec les résultats du K-means. On peut voir sur la figure suivante qu'une initialisation avec K-means donne légèrement de meilleurs résultats.

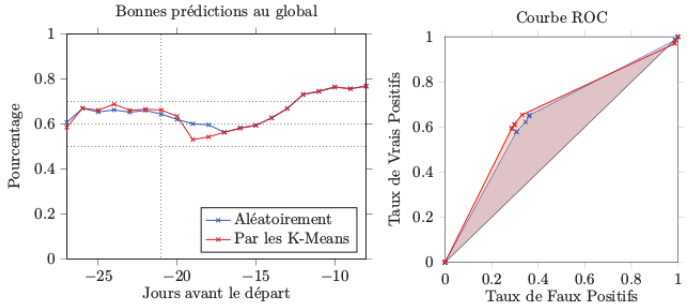

FIGURE 11 – Comparaison des performances de l'algorithme EM en fonction de son type d'initialisation [1]

Comparatif entre les deux méthodes :

Méthode 1	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 €

[1]

On a choisi donc d'initialiser EM avec K-means constatant que EM avec une initialisation de 5 clusters offrait le meilleur résultat.

5.2 Résultats de classification

Là aussi, pour évaluer les performances des différentes méthodes on regarde uniquement le point à -21 jours de la date de départ. On va combiner EM initialisé par K-means (nstar = 100 et nbr clsuters = 5) avec les différentes méthodes de classification pour voir le meilleur couple à utiliser.

5.2.1 Arbres de décision : CART C4.5

CART : il est connu pour le sur-apprentissage en créant un arbre de décision trop profond. mais dans notre cas plus l'arbre est profond, meilleur sont les résultats car lors de l'élagage des clusters n'apparaissent plus. On peut constater que sans élagage (courbe violette) on obtient le meilleur résultat en terme de prédiction globale.

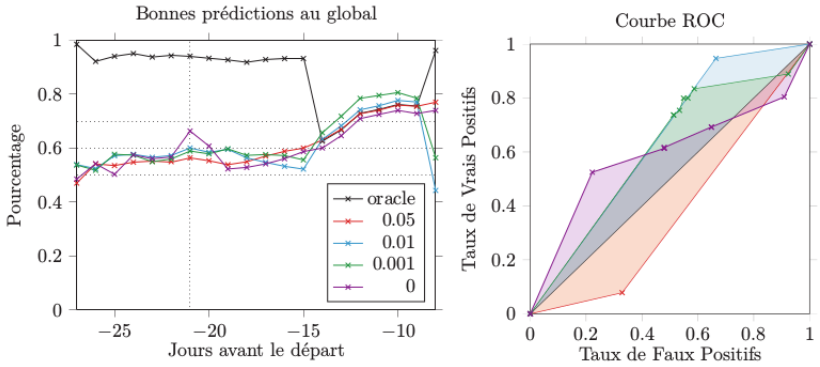

FIGURE 12 – Évolution du pourcentage de bonnes prédictions en fonction de la date avant le départ et courbe ROC à -21 jours pour différents élagages [1]

C4.5 : comme pour CART C4.5 possède un paramètre d'élagage mais contrairement à CART un arbre trop profond va mal généraliser et un arbre trop élagué va trop généraliser. En choisissant le meilleur coefficient d'élagage qui est 0.25 dans ce cas il donne de mauvais résultats qu'on peut voir que les courbes ROC sont proches de la diagonale ce qui veut dire que c'est une prédiction systématique à la hausse [1] .

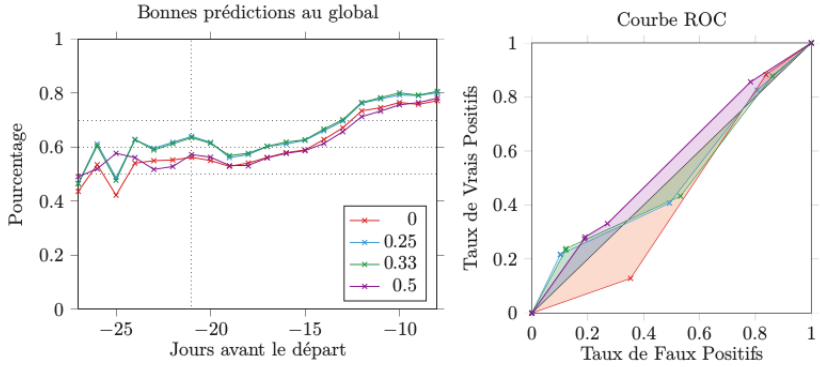

FIGURE 13 – Évolution du pourcentage de bonnes prédictions en fonction de la date avant le départ et courbe ROC à -21 jours pour différents élagages [1]

5.2.2 Forêts aléatoires (Random Forest)

Comme décrits, ils représentent une agrégation de multiples arbres de décision CART appliquée à des sous-ensembles aléatoires de la base d'apprentissage. Les paramètres principaux sont le nombre d'attributs choisis aléatoirement pour chaque nœud (mtry) et le nombre d'arbres. Sachant que ces paramètres influencent la qualité de la prédiction finale, nous avons fait des tests pour faire les choix les plus optimaux. Nous avons choisi de travailler avec mtry = 12 car l'erreur OOB (out-of-bag error est une méthode qui mesure l'erreur de prédiction dans les arbres forêts aléatoires, Adaboost, ...) [1] diminue et devient stable à partir de mtry = 9.

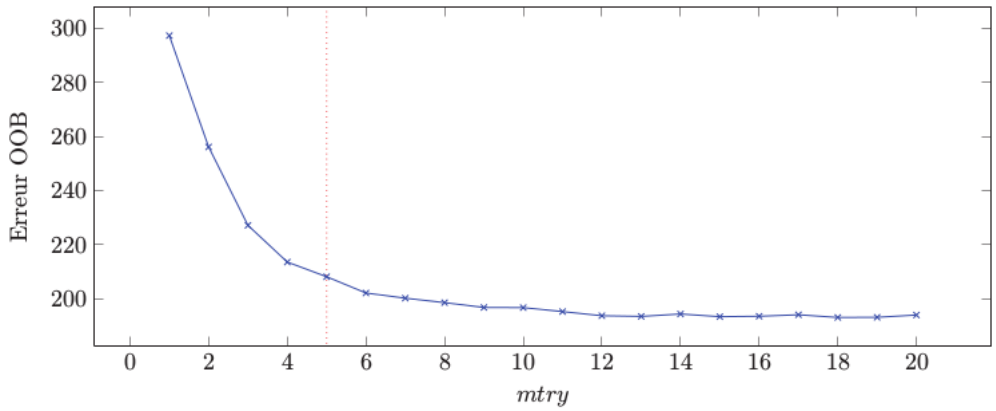

FIGURE 14 – Évolution du taux d'erreur OOB en fonction du paramètre mtry [1]

Pour le nombre d'arbres, on sait que plus il y a d'arbre plus la qualité de la prédiction est meilleure. Nous avons constaté que le taux de bonnes prédictions était au alentour de 120 arbres. L'espace mémoire devenant conséquent, nous avons choisi de travailler avec 100 arbres.

5.2.3 Adaboost

On a mis en compétition les arbres de décisions et Adaboost avec EM comme étape de segmentation avec 5 clusters. Dans un premier temps nous avons utilisé tous les attributs et dans un second temps nous avons utilisé uniquement les 10 attributs les plus discriminants (choisis grâce à la méthode feature selection). Adaboost étant basé sur CART on a choisi de ne pas faire d'élagage là aussi. Les deux méthodes donnant presque les mêmes résultats on a choisi de travailler avec les Random Forest pour éviter le coût de ressource avec Adaboost en liaison avec les arbres non élagués. (grande profondeurs)

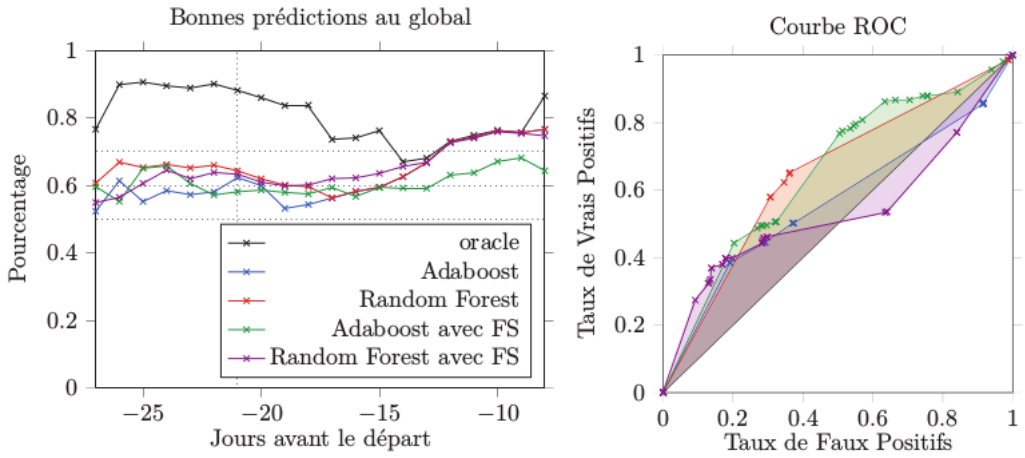

FIGURE 15 – Comparaison entre Adaboost et RandomForest [1]

5.3 Couple de méthodes choisi

En effectuant la classification avec les différentes méthodes sur le résultat de la segmentation réalisé par EM on constate que les résultats sont équivalents en terme de taux de prédiction mais les Random Forest permettent un meilleur gain pour l'utilisateur.

Comparatif entre les méthodes :

Méthode 1	Nb groupes	Taux BP	Economie/t_1
RF	5	66%	4 €
RF	30	64%	1 €
Adaboost	5	63%	0 €
CART	5	62%	0 €
CART	30	62%	-3 €
C4.5	5	66%	-2 €
C4.5	30	61%	-7 €

[1]

Au regard des résultats obtenus, nous avons choisi de travailler avec le couple : EM initialisé avec K-menas (nstar = 100 et nbr clusters = 5) et les forêts aléatoires avec 100 arbres CART.

5.4 Influence de la taille de la base d'apprentissage sur la qualité de la prédiction

On a choisi de travailler avec les 50 derniers pourcents (représentant les deux dernières années) de la base pour des raisons de volumétrie de données et nous pensons que c'est suffisant pour produire une segmentation de qualité. Nous constatons que ce choix donne le meilleur résultat, car les 10 derniers pourcents de la base reflètent les vols les plus anciens, sachant que le modèle s'adapte avec le temps il donnera de meilleurs résultats avec les vols les plus actuels. Et avec 70% on aura plus de nouveaux vols mais qui reflètent une minorité par rapport à l'étendue du modèle sur les vols. C'est pour cela que les 50 derniers pourcents sont un bon compromis entre nouveaux et anciens vols.

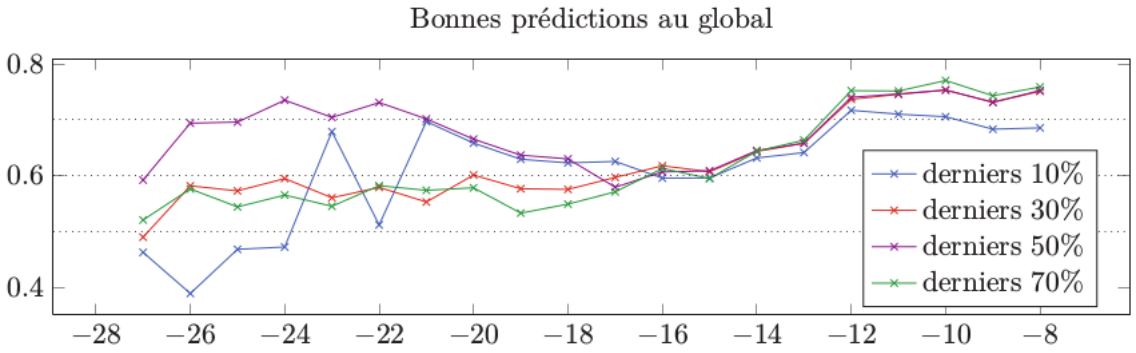

FIGURE 16 – Evolution du taux de bonnes prédictions avec l'augmentation des la base d'apprentissage, pour un couple EM/5 clusters et Random Forest [1]

6 Evolution des performances dans le temps

Après avoir fait des simulations de performances, on constate que le premier et le deuxième mois donnent plus de 63% de taux de prédiction mais cela diminue considérablement au-delà.

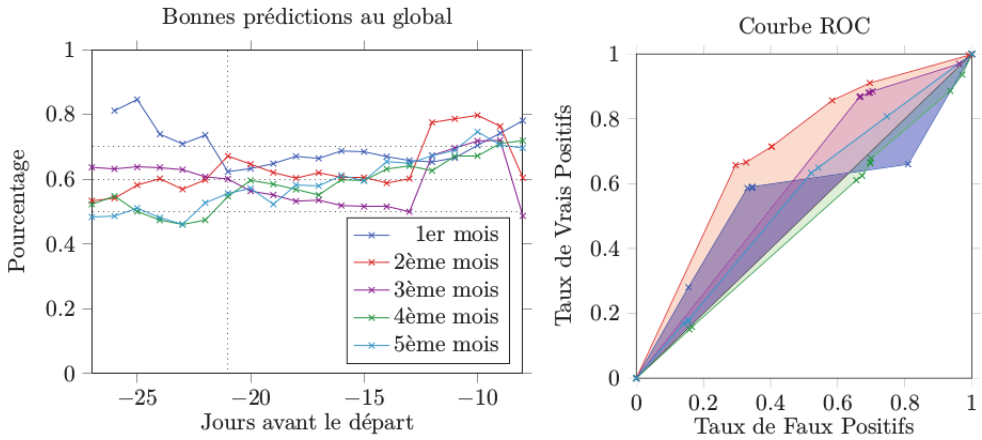

FIGURE 17 – Évolution du taux de bonnes prédictions en fonction du temps [1]

On a conclu qu'il fallait renouveler le modèle en viront tout les 45 jours.

7 Conclusion

Le but de cette thèse était de fournir aux utilisateurs de Liligo un service d'aide à la décision qui les conseillerait d'acheter ou d'attendre en fonction des hausses et des baisses du billet trouvé et leur permettraient ainsi de gagner de l'argent ou à défaut de ne pas en perdre. Pour cela, des méthodes de fouilles de données ont étaient mises en place pour construire le préducteur.

Tout d'abord, il y a eu la phase de collecte de données (base de données des historiques de recherches utilisateurs de Liligo depuis 6 ans), il fallait extraire les informations à savoir les variations des prix de vols. Cela a été réalisé grâce à la mise en place d'une base de données structurée qui permettait d'avoir pour chaque vol une série temporelle de prix. Il fallait s'affranchir de la notion de grandeur des prix et ne garder que leurs comportements (hausses ou baisses). On les a donc transformées en séries de rendements par des processus ponctuels dans le plan temps-rendement puis en niveaux de gris.

Afin de grouper les vols qui avaient le même comportement de prix, on a réalisé une segmentation sur la base d'apprentissage grâce à l'algorithme EM (apprentissage non supervisé).

Maintenant pour prédire l'évolution de prix pour les nouveaux vols, nous avons réalisé un apprentissage supervisé sur les attributs des vols de la base d'apprentissage segmenté. Nous avons utilisé les forêts aléatoires.

En dernier, l'étude des performances de notre service au cours du temps nous a montré qu'il fallait renouveler le modèle de prédiction environ tous les mois.

8 Conclusion personnelle

À travers ce travail, j'ai pu élargir mes connaissances en méthodes de fouilles de données. Cela m'a permis de mieux comprendre les principes et détails des algorithmes de segmentations utilisés et des algorithmes de classification de types arbres de décisions. J'ai pu voir qu'il existe des techniques qui permettent d'apporter des améliorations aux algorithmes de fouilles de données comme la combinaison de classifieurs, le bagging, le boosting, ...

En dernier lieu, j'ai appris à analyser des travaux de recherches et à réfléchir aux méthodes alternatives pour apporter des améliorations. Dans cette optique-là, j'ai pensé qu'utiliser un réseau de neurones SOM à la place de l'algorithme EM initialisé par K-means permettrait peut-être d'apporter une optimisation en terme de temps d'exécution et d'améliorer la qualité de la prédiction.

Références

[1] Till WOHLFARTH. 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, 2013.