Recherche et développement d'une plateforme de prédiction et de recommandation pour le secteur de l'hôtellerie last-minute

Xavier Daull

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

PROVENCE-ALPES-COTES D'AZUR

MEMOIRE

présenté en vue d'obtenir : le DIPLOME D'INGENIEUR CNAM

SPECIALITE : Informatique
OPTION : Systèmes d'information

par : Xavier DAULL

« Recherche et développement d'une plateforme de prédiction et de recommandation pour le secteur de l'hôtellerie last-minute »

Soutenu le 9 juillet 2015

JURY

PRESIDENT : Yves LALOUM

MEMBRES :

Monsieur Bastien PESCE	Professeur au CNAM
Monsieur Noel QUESSADA	Professeur au CNAM
Monsieur Sébastien HOUZE	Directeur technique de Rezzza (VeryLastRoom)
Monsieur Nicolas SALIN	Directeur général de Rezzza (VeryLastRoom)

Résumé

Ce document décrit les travaux réalisés dans le cadre du projet R&D « Irma ». L'objet de ce projet est de fournir à la société Rezzza des outils de prédiction et de recommandation, dédiés au secteur de la réservation d'hôtel à la dernière minute, et de découvrir de nouvelles opportunités pour l'entreprise en exploitant les techniques de machine learning. Les stratégies déduites de ces travaux comprennent l'optimisation des prix de réservation, la disponibilité des offres de chambres sur le marché, le marketing ciblé en anticipant le comportement du client, l'identification de la meilleure stratégie de prix que devrait appliquer l'hôtelier…

Summary

This document describes the work carried out within the R & D project "Irma". The purpose of this project is to provide Rezzza company with prediction and recommendation tools dedicated to the last minute hotel booking business and to discover new opportunities for the company using machine learning techniques. It should provide real-time forecast information in order to optimize the company's offer. The strategies derived from this work include optimizing reservation price, room offers availability on the market, targeted marketing by anticipating customer behavior, the best pricing strategy that would apply for each hotel...

Remerciements

Je tiens à remercier tout particulièrement Bastien Pesce, enseignant au CNAM et responsable de la filière Systèmes d'Information à Aix en Provence. Sa patience, ses critiques, ses conseils et ses encouragements ont été d'une grande aide pour réaliser ce mémoire.

Sébastien Houzé, directeur technique de la société Rezzza SAS (VeryLastRoom), qui m'a proposé ce projet passionnant. Il a toujours été de bon conseil, prompte à challenger mes réflexions ou résultats. Merci!

Je remercie bien sûr toute l'équipe du CNAM, toujours très accueillante, disponible et qui m'a accompagnée toute au long de ces années.

Ma femme, mes enfants, mes parents et Rodolphe qui ont été une aide inestimable pour arriver au bout de ce long voyage.

Sommaire

Liste des abréviations

API : Application Programming Interface, ou interface de programmation applicative
CEP : Complex Event Processing ou Traitement d'événements complexes
ETL : Extract Transform Load ou Extraire transformer charger
FN : False Negative ou Faux négatif
FP : False Positive ou Faux positif
MAE : Mean Absolute Error ou erreur moyenne absolue
ML : Machine learning ou apprentissage automatisé
MOOC : Massive Online Open Courses
RMSE : Relative Mean Squared Error
ROC : Receiver Operating Characteristic ou fonction d'efficacité du récepteur
TN : True Negative ou Vrai négatif
TP : True Positive ou Vrai positif
VN : voir TN
VP : voir TP

Glossaire

Attribut (d'un exemple) : une des variables d'entrée composant un exemple d'apprentissage.

Batch (ou traitement par lot) : le traitement par lots (ou batch processing) est un enchaînement automatique d'une suite de commandes (processus) sur un ordinateur sans intervention d'un opérateur.

Big data : volumes massifs de données variées difficiles à travailler avec des outils classiques de bases de données ou de gestion de l'information.

Cloud (computing) : exploitation de la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau, généralement Internet. Ces serveurs sont généralement loués à la demande.

Estimation par analogie (analogique) : estimation basée sur la comparaison, suivant différents paramètres, du projet ou de la tâche à estimer aux projets et tâches passés.

Estimation par jugement expert : estimation d'un ou plusieurs experts qui appliqueront le plus souvent la méthode analogique de manière informelle.

Exemples (en machine learning) : exemple de données utilisé pour l'apprentissage en machine learning composé de variables ou attributs d'entrée (et d'une valeur de sortie dans le cas d'un apprentissage supervisé).

GitHub : service web de gestion de versions décentralisé, basé sur le projet open source Git, et offrant des options avancées de collaborations et d'interactions sociales.

Intelligence artificielle : discipline scientifique qui recherche des méthodes de création ou simulation de l'intelligence.

Machine learning ou apprentissage automatique : discipline scientifique concernée par le développement, l'analyse et l'implémentation de méthodes automatisables du processus d'apprentissage par une machine (au sens large) pour la réalisation de tâches définies.

Méthode PMP : méthode de gestion de projets, provenant de la certification PMP, se basant sur le guide du corpus des connaissances en gestion de projet PMBOK (Project Management Body of Knowledge).

Scrum : méthode agile dédiée à la gestion de projets. Scrum est issu des méthodes incrémentales facilitant la maîtrise d'une production planifiée.

Sprint (Scrum) : la méthode Scrum s'appuie sur le découpage d'un projet en boîtes de temps, nommés « sprints ». Les sprints peuvent durer entre quelques heures et un mois. Chaque sprint commence par une estimation suivie d'une planification opérationnelle. Le sprint se termine par une démonstration de ce qui a été achevé.

Stream processing : traitement de flux de données au fur et à mesure de leur disponibilité (au fil de l'eau).

Temps réel : dans le cadre de ce document, le terme "temps réel" n'est pas utilisé comme une contrainte "temps réel" forte à proprement parler, mais pour indiquer que ce sont

des prédictions ou recommandations délivrées à tout moment et mises à jour en permanence. Différents termes y sont associés: apprentissage online, apprentissage en flux. L'utilisation la plus classique du machine learning étant en batch ou apprentissage offline.

Travis CI : service d'intégration continu utilisé pour tester automatiquement les projets hébergés sur GitHub.

Introduction

Rezzza, (connue sous l'appellation commerciale VeryLastRoom) est une start-up basée à Marseille spécialisée dans la réservation de chambres d'hôtel à la dernière minute sur smartphone. J'y ai mené une mission de recherche & développement de 10 mois, dont l'objet était la réalisation d'un service de prédiction et de recommandation dédié au domaine de la réservation d'hôtel à la dernière minute. Il fournira en permanence des informations d'anticipation des offres et des demandes du marché de la chambre d'hôtel d'une part, et de recommandation de produits (chambres) et de stratégie optimale de prix d'autre part.

L'équipe technique était au départ composée de 6 personnes, de sous-traitants spécialisés, et a depuis régulièrement recruté de nouvelles personnes. Suite à une importante levée de fond réalisée en février 2014, l'entreprise amorçait une nouvelle phase de développement avec des challenges techniques et commerciaux majeurs. C'est dans ce cadre que m'a été proposé ce projet de R&D. Il doit contribuer à terme à une importante avancée technique et commerciale face à la concurrence.

Après 9 années passées à diriger une équipe spécialisée en solutions de géolocalisation pour le secteur pétrolier, j'ai rencontré Sébastien Houzé (mon actuel directeur technique) qui m'a proposé cette mission de nature très différente de mes objectifs passés. Sébastien Houzé est reconnu pour sa compétence en développement logiciel et architecture web. La mission proposée au sein de cette équipe constitue un environnement motivant, formateur, et une opportunité parfaite pour réaliser mon mémoire d'ingénieur.

J'étais autonome dans la gestion du projet. En termes de ressources humaines, j'étais le principal développeur et architecte de la solution à réaliser. J'avais à ma disposition un budget pour contracter de courtes missions de sous-traitance ainsi que la possibilité de recruter un stagiaire.

Mes objectifs initiaux ont été clairement définis par une charte de projet (objectifs, budget, risques, planification, livrables…). Je l'ai développée, rédigée et fait valider par le comité de pilotage, suite à des entretiens et brainstormings avec des spécialistes techniques, commerciaux et cadres dirigeants de Rezzza, et des recherches sur les thèmes à aborder

« Machine learning » et « big data » ?

Une partie de ce mémoire présentera le domaine du « machine learning », aussi appelé « apprentissage automatique » ou « apprentissage artificiel ». J'utiliserai souvent le nom anglais car c'est son appellation la plus commune en entreprise. Pour le lecteur qui ne connaîtrait pas ce domaine, c'est une discipline de l'intelligence artificielle qui a connu un essor considérable ces dernières années. Cette discipline a pour vocation de créer des algorithmes apprenant, par des exemples de données, à réaliser une tâche de manière optimale. *Ces tâches peuvent être : la prédiction météorologique en apprenant de la météo des dernières années, la reconnaissance de caractère en apprenant des exemples de*

caractères écrits scannés et leur valeur, la recommandation de produit en apprenant des intérêts passés des différents clients, la création automatique de groupes de clients homogènes (segmentation) en apprenant de l'historique des ventes d'une société et des fiches clients. Le regain d'intérêt pour le « machine learning » provient de l'augmentation considérable de données disponibles via Internet qui a donné naissance au phénomène « big data ». Ce phénomène est l'utilisation de ces données massives provenant de sources différentes et du « machine learning » pour créer des applications nouvelles de prédiction, de recommandation, d'optimisation... Les avantages stratégiques que peuvent apporter ces applications en font aujourd'hui un axe majeur d'innovation.

I. Problème à résoudre et bases du projet « Irma »

A. Justification du projet

1. Introduction

Rezzza, spécialiste de la vente de chambres à la dernière minute sur smartphone, développe un système d'enchères à prix dégressif en temps réel. Dans ce créneau, l'entreprise essaie d'offrir les prix les plus intéressants du marché tout en maximisant sa rentabilité.

Les attentes initiales du projet pour l'entreprise n'étant pas clairement définies, une série d'interviews a été conduite auprès des personnes impliquées dans ce projet, tout en leur présentant les possibilités que peuvent offrir des outils de prédiction et de recommandation. Les interviews ont été entrecoupées de petits tests et prototypes présentés aux différents intervenants pour vérifier leurs attentes comme les réelles possibilités dans le domaine de la R&D de ce projet. L'objectif de cette phase de pré-étude est de dégager un consensus sur les problématiques prioritaires à étudier dans le cadre du projet et de valider la justification de celui-ci. Le critère majeur d'évaluation des priorités du projet est le retour sur investissement attendu.

La phase de pré-étude a dégagé, les axes de travail ci-dessous :

- prédictions des prix, changements et disponibilités (chambres, offres).
- recommandations à objectifs (stratégies de prix ou de produit).
- marketing par anticipation.

Plus tard dans le projet, la question de l'optimisation dynamique de l'architecture de collecte d'information est venue s'ajouter aux axes de travail ci dessus.

Les problématiques à résoudre liées à ces axes d'étude sont décrites ci dessous.

2. Problématique de prédictions de prix et de disponibilités

L'équipe commerciale de Rezzza assure sous forme de services pour de nombreux clients la gestion permanente des offres des hôteliers (tarifs et quantités). C'est un travail complexe nécessitant de nombreuses informations et analyses où l'ajustement des prix face à la concurrence est crucial. Il faut pouvoir maximiser le nombre de réservations par le client final. D'importants investissements ont déjà été réalisés pour faciliter le travail de l'équipe. Un système de découverte et collecte automatique des prix de la concurrence via des robots autonomes a notamment été créé (projet BeeSpy). Ce travail est en cours d'amélioration avec notamment la collecte des disponibilités de chambre (projet BeeSpy2). Il est limité par la capacité en ressources à collecter un maximum d'informations dans un temps donné, pour un coût raisonnable, et, n'offre pas encore de valeur ajoutée à ces données.

Le projet Irma répondra, en premier lieu, aux problématiques de prédiction suivantes :

- informer sur “quand” et “comment” les prix de la concurrence vont changer.
- concentrer la collecte des prix par les robots aux seuls hôtels dont l’information est susceptible de changer. Cela réduira ainsi le coût de cette infrastructure tout en améliorant son efficacité.
- prévoir la disponibilité des chambres sans risque de surbooking pour que l’équipe commerciale n’hésite pas à proposer des offres. Cela maximisera le nombre d’offres de chambres disponibles (ou d'en avoir pour un hôtel ou un secteur quand la concurrence n'en a pas).

3. Maximisation des transactions par la recommandation

Sur un autre plan, Rezzza souhaite rendre l'utilisation des applications de ventes optimales, que ce soit pour les hôteliers ou pour les clients finaux, afin de maximiser les transactions :

- pour les hôteliers cela consisterait à leur suggérer des offres de prix optimales en fonction de la demande future des clients, et, de leurs objectifs de rentabilité.
- pour les clients finaux, cela consisterait à leur suggérer les offres les plus en adéquation à leurs goûts, contextes ou autres critères de choix.

Ces problématiques de recommandations optimales constituent ainsi un deuxième volet du projet Irma.

4. Marketing par anticipation (optionnel dans le projet Irma)

Dans une dernière partie, en exploitant les résultats des premier et deuxième volets, il sera possible de prévoir les préférences et besoins futurs des clients et d’offrir un marketing anticipé, mieux ciblé et plus pertinent pour les clients.

5. Optimisation de l'architecture

La question de l’optimisation de l’architecture de BeeSpy2 n'avait pas été formulée au départ mais s'est révélé être une opportunité majeure pour améliorer ce futur service. En évaluant le coût d’architecture nécessaire pour les différents objectifs de R&D (coût élevé), nous nous sommes aperçus que nous pourrions largement diminuer le coût associé à l’exploitation du futur projet BeeSpy2 (voir ci dessus) tout en améliorant ses performances et offrir des bases techniques solides pour la suite.

Le projet BeeSpy2 destiné à collecter tous les prix de la concurrence a un important coût de fonctionnement. Il est destiné à “connaître constamment les prix de la concurrence afin de garantir d’avoir des prix toujours les plus bas”. La majorité des collectes par les robots ne servent à rien car les prix n’ont pas changé par rapport au dernier relevé. Les prix ayant fluctués sont relevés après un délai que nous souhaiterions réduire, d’où un retard dans l’alignement des prix face à la concurrence. Les changements de prix les plus importants sont ceux que nous voudrions détecter en priorité. La réduction du coût d’architecture et l’amélioration de la performance du service peuvent se faire en optimisant la planification des

robots, afin de prioriser le rendement maximal qui s'exprime par "probabilité de variation" x "% de variation".

6. Validation de la nécessité de la R&D

Pour répondre aux problématiques citées, il n'existe à ce jour pas de solutions dédiées à notre domaine d'activité (réservation à la dernière minute de chambres d'hôtel). Les différents éléments techniques composant une telle solution existent mais un ensemble de recherches, prototypes et développements spécifiques seront nécessaires pour réaliser une solution répondant aux objectifs et contraintes du projet. Ainsi, après une étude de faisabilité technique et de pertinence pour l'entreprise, il a été décidé d'engager un véritable processus de R&D sur les problématiques exposées et les opportunités associées, ainsi que sur le socle mathématique et informatique nécessaire à sa faisabilité.

B. Objectifs techniques du projet

A partir des entretiens initiaux que j'ai mené pour éclaircir la justification du projet et les différents besoins, une première analyse a dégagé les objectifs prioritaires ci-dessous (validés par le comité de pilotage composé du directeur commercial et du directeur technique) :

1. 1er objectif : architecture et 1ère prédiction

Le premier objectif du projet est de fournir le socle technologique pour toutes les futures applications de prédiction et recommandation. Pour illustrer le fonctionnement du socle technologique du projet Irma il faudra livrer la 1ère prédiction nommée "price change trigger", qui fournit l'information nécessaire pour alerter "par anticipation" sur la nécessité de l'ajustement des offres de certains hôtels (hausse ou baisse), face à la concurrence. Cette information sera aussi utilisée pour optimiser la planification des robots de collecte des prix et de la disponibilité des chambres. L'exploitation de cette prédiction par les autres applications (information à l'utilisateur sous la forme adéquate: alerte, filtre, information visuelle), n'est pas incluse dans le projet bien que réalisée en parallèle par une autre équipe.

2. 2ème objectif : autres prédictions et recommandations

Au delà de la fourniture de la première prédiction il faudra résoudre les problèmes de prédiction suivants et mettre en œuvre les services associés

Prédiction 2 "Overbooking risk & room availability": estimer le nombre maximal de chambres disponibles pour un risque de surbooking <1%.

Prédiction 3 "Autoyield recommandation" : recommander de manière optimale la stratégie de tarification par anticipation selon différents critères et la prise en compte de la pression concurrentielle. Le backoffice pourra ainsi automatiser la définition du prix d'un hôtel, de suggérer un tarif plus adapté et d'alerter d'une probable erreur de tarification.

Prédiction 4 “Customer booking probability” : prédire la probabilité que le client ait un besoin de réservation à l’instant T. Pour cela, il sera nécessaire dans un autre projet de collecter les informations d’utilisation de l’application (ouverture, temps d’utilisation, lieu…) afin de ne pas avoir seulement la liste des réservations pour créer le modèle. Un exemple d’application serait d’envoyer des push marketing quand le client n’a pas encore réservé.

Prédiction 5 “Customer recommended offers" : proposer des recommandations d’offres personnalisées pour les clients en fonction de leurs habitudes de réservation (type hôtel, prix, lieux…) et ses offres consultées.

3. Optimisation de l’architecture d’exploitation

Les systèmes de prédiction demandent beaucoup de ressources de calcul, de mémoire vive et de stockage[1]. Or, les ressources financières de VeryLastRoom sont une contrainte forte, et les compétences techniques en machine learning y sont aussi limitées. En début de projet, il est difficile de savoir quelles seront les solutions techniques et l’architecture finale. Sont donc prévues 2 ou 3 phases :

- une première phase de prototypage libre en termes d’architecture.
- une phase d’optimisation des performances (vitesse, calcul, mémoire, coût) afin d’avoir une solution plus adaptée aux capacités financières de l’entreprise.
- optionnellement, une phase de rationalisation de la solution pour réduire la diversité des composants et en intégrant une configuration plus classique dans le domaine (normes, architectures, logiciels, “meilleurs pratiques”). L’architecture sera ainsi plus robuste car appuyée quasi uniquement sur des produits éprouvés, plus facile à maintenir et à faire évoluer par l’équipe technique de l’entreprise.

C. Enjeux et difficutés techniques

Le succès d’un tel projet dépend grandement de l’évaluation et de la maîtrise des enjeux techniques du projet. Ce point doit être abordé au plus tôt dans la phase de préparation, et tout au long du projet, pour anticiper les risques et besoins associés. Pour comprendre et maîtriser ces enjeux techniques, une stratégie de formation a été développée dès le début du projet (voir II.A).

1. Prédiction en temps réel

Un premier volet important des enjeux techniques est la prédiction en temps réel. Il faut prévoir le prix, les risques de changement et la disponibilité futurs de chaque hôtel individuellement, sans connaître à priori leur stratégie de tarification.

Ainsi, les différents problèmes détaillés ci-dessous se sont posés :

- ✓ maîtriser les contraintes de ressources et de temps réel

[1] Habituellement les solutions de machine learning et de Big Data utilisent Hadoop ou solution équivalente pour leur architecture. Or ces solutions supposent une architecture lourde qui représente malgré tout un coût important et une gestion complexe.

- ✓ individualiser la modélisation des stratégies de tarifications
- ✓ créer et gérer le flux d'apprentissage, détecter le changement ou de l'évolution de la stratégie de tarification
- ✓ améliorer des algorithmes par l'analyse mathématiques

a. Ressources et temps réel

La plupart des éléments composant de telles solutions de prédiction ou de traitement flux de données volumineux nécessitent des ressources massives (nombreux serveurs en cluster, CPU, RAM, stockage…) et sont rarement prévus pour des applications temps réel gérant de nombreux modèles diversifiés. La recherche dans ce domaine en est à ses balbutiements pour tenter de rationaliser algorithmes et architectures associés (GraphChi[2] - Melon University), et faciliter le traitement d'importants flux de données en apprentissage "online" (projet SAMOA [3] - Waikato university / Yahoo) mais ne répondent pas aux problématiques simultanément ou sont limités en algorithmes d'apprentissage. Durant tout le processus R&D, les solutions seront conçues dans l'optique de satisfaire simultanément ces contraintes.

b. Individualisation de la modélisation des stratégies de tarifications

Des modèles spécifiques seront créés dynamiquement pour chaque hôtel avec des modèles alternatifs. Ainsi chaque stratégie est modélisée mais en prenant compte des données contextuelles et historiques, les corrélations avec des hôtels similaires. Un modèle n'est utile que s'il est performant. Il faut donc pouvoir évaluer et choisir automatiquement des modèles optimaux dans un temps limité.

c. Flux d'apprentissage

Pour construire les différents modèles de chaque hôtel, il faut récupérer en permanence les informations nécessaires et des évènements, puis les enrichir avec des données temporelles, contextuelles, les coupler avec des sources similaires et les lier à des données futures. La conception de ces flux doit être automatique et rapide tant dans sa collecte que dans sa génération d'informations enrichies. Un moteur de base de données (SGBD) ne peut effectuer ce type de traitement dans le temps nécessaire. C'est la problématique fréquemment rencontrée dans les problèmes de *big data* et la nécessité de les traiter en temps réel limite les possibilités. J'ai étudié différentes solutions (ex. Esper, StreamSQL..) mais celles ci n'étaient pas satisfaisantes en termes de fonctionnalités, de prix et de contraintes d'exploitation. J'ai donc opté pour la conception d'une solution dédiée pour agréger un flux temps réel dédiée à l'apprentissage pour la prédiction et la recommandation.

[2] GraphChi permet de remplacer une batterie d'ordinateurs en cluster par un simple Mac pour analyser des volumes de données très impressionnants, comme par exemple l'ensemble des tweets de 2010 à 2012 en moins d'une heure au lieu de 400 minutes avec 1000 ordinateurs, projet auquel j'ai contribué à l'amélioration de la dernière version de l'algorithme.

[3] Le projet SAMOA offre une architecture compatible avec des flux de données distribuées (tels que Storm, S4….) et un très grand nombre d'algorithmes d'apprentissage quasi temps réel dérivé des fameux projets Weka et MOA (Université de Waikato, NZ) avec lesquels j'ai eu des échanges.

Elle est largement optimisée en termes de ressources et temps nécessaires pour la préparation des données pour l'apprentissage, la prédiction ou la recommandation.

d. Changement ou évolution de la stratégie de tarification (Concept Drift)

Un hôtel peut changer de stratégie, intentionnellement ou non, le modèle détecté comme optimal peut ne plus l'être car il ne prenait pas en compte des données non existantes auparavant. Il faut donc pouvoir le détecter au plus tôt et basculer sur d'autres modèles existant ou en construire de nouveaux.

e. Amélioration par l'analyse mathématique

Les modèles utilisés pour la prédiction diffèrent totalement suivant la nature des données traitées. Il faut étudier aussi bien les types de données, leur distribution et l'impact de leurs combinaisons. Cela ne suffit pas. Il faut aussi étudier les modèles, probabilistes ou non, les plus adaptés et comment ceux ci pourront être améliorés. Une première approche empirique a été de tester de nombreux algorithmes de machine learning, pour en trouver les modèles mathématiques associés qui seraient les plus pertinents pour notre secteur. Cette approche validé une liste restreinte de modèles mathématiques à étudier. Je les ai étudiés pour, soit les optimiser ou/et les combiner ou concevoir des nouveaux algorithmes, en vue de la correction de données rares ou déséquilibrées, de la corrélation des flux, ou de l'apprentissage automatique à objectif.

2. Systèmes de recommandation

Un deuxième volet de l'étude est les systèmes de recommandation. Il existe quelques librairies et services (open source ou non) mais, à l'exception des projets de recherche, ceux-ci ne prennent pas directement en compte le contexte (ex. localisation) ou le font mais de manière très limitée. De plus, les systèmes de recommandation non triviaux doivent être conçus spécifiquement pour le domaine de la réservation de dernière minute.

Ils doivent notamment prendre en compte :

- ✓ les stratégies de recommandation en fonction des seuils de données disponibles et des cas d'utilisation
- ✓ la traduction des données collectées en données d'évaluation
- ✓ la combinaison de données implicites et explicites
- ✓ la pondération en fonction des objectifs de recommandation

Il faudra combiner différents systèmes de recommandation pour répondre au besoin du domaine de la réservation de dernière minute, et concevoir un moteur dédié capable d'exploiter les différents modèles et algorithmes développés suivant nos contraintes d'exploitation.

3. Optimisation multi-objectifs

Le dernier volet de l'étude est les problèmes d'optimisation posés par les différents objectifs du projet de R&D : suggestion de choix optimum de stratégie de prix pour les

hôteliers selon ses préférences, rechercher des modèles de prédiction ou recommandation optimaux sous contraintes. J'ai ainsi étudié les différentes approches mathématiques suivantes : optimisation bayésienne, processus de décision Markovien, méta heuristiques, théorie des jeux, optimisation graduée...

II. Méthodes, outils et processus

Pour gérer ce projet, je me suis principalement inspiré de deux méthodes différentes (détaillée plus bas) : PMP pour la gestion du projet dans son ensemble, Scrum pour la gestion des mini-projets issus du projet global avec un volet analyse plus développé. J'apporterai quelques précisions sur la gestion des ressources externes qui ont joué une part importante dans la bonne réalisation du projet.

Je présenterai ensuite les méthodes d'analyses et les différents outils utilisés pour l'exécution du projet.

J'aborderai enfin un volet crucial pour la réussite d'un projet R&D : la gestion de la connaissance et le processus de recherche de solutions.

A. Gestion « Macro » : Initiation, planification initiale, exécution & maîtrise, clôture

Pour la gestion globale du projet, j'ai utilisé la méthodologie PMP simplifiée. Elle couvre les différents phases d'un projet (initiation, planification, exécution vs maîtrise, clôture) sur ses différents aspects (intégration, définition et périmètre, coûts, délais, qualité, ressources humaines, risques, achats, communication, personnes impliquées dans le projet).

Cette gestion de projet est lourde si elle est appliquée en totalité et serait surtout disproportionnée pour les temps alloués aux différents volets de ce projet dont je suis à la fois : gestionnaire, développeur principal et responsable de la R&D. Par ailleurs, PMP regroupe de très bonnes pratiques en matière de gestion de projet. J'ai choisi, pour ce projet, ce qui me semblait essentiel dans cette méthode :

- la phase d'initiation du projet : découvrir et rencontrer les acteurs du projet, détecter les opportunités et établir un consensus sur la charte de projet (objectifs et limites, rôles, risques, budget…).
- la première phase de planification : exigences, découpage des tâches par WBS, ordonnancement, planification Gantt des tâches et les méthodes d'estimation, planification de la gestion globale du projet.
- les processus allégés de gestion :
    - de la communication
    - des risques
    - de la qualité
    - du contrôle du cadre du projet
    - et des ressources.

La "charte de projet" et le "plan de gestion de projet" détaillés ci-dessous illustrent la gestion du cadre global du projet.

1. Charte de projet : définition du projet et consensus

La charte de projet clarifie et valide les caractères essentiels du projet auprès des différents acteurs. Elle marque le début du projet accepté par tous. Ce document couvrait les points suivants :

- Historique du projet
- Business Case (justification du projet)
- Objectifs du projet
- Produits à livrer
- Validation des livrables
- Frontières du projet
- Mesure du succès du projet
- Facteurs critiques de succès du projet
- Principaux risques à maîtriser
- Stakeholders (acteurs du projet) : attentes & obligations respectives
- Ressources
- Milestones (étapes clés)
- Estimation du Budget (et Modèle de coût)
- Structure de gouvernance, Project Manager, Responsabilités
- Sponsor et Signatures du comité de pilotage

La charte, validée par la direction et développée avec les différents acteurs du projet, a réduit les zones d'ombre, permis de s'accorder sur les responsabilités de chacun et de donner une visibilité sur les risques, la planification, le coût, les résultats attendus.

a. Livrables et frontières du projet

Les livrables ont été définis comme suit mais peuvent être modifiés selon le processus de gestion des modifications :

- Plateforme avec base de données agrégées et enrichies automatiquement
- API Web Service de prédiction
- Fabrication automatique des modèles
- Méthode semi-automatique de création de nouveaux modèles et prédiction
- Documentation de création de modèles et prédiction
- Architecture distribuée adaptée au machine learning et flux big data
- Prédiction 1 "Price change trigger"
- Prédiction 2 "Overbooking risk & room availability"
- Prédiction 3 "Autoyield recommandation"
- Prédiction 4 "Customer booking probability"
- Prédiction 5 "Customer recommended offers"

Le projet inclut une première version du service de prédictions et de recommandations mais pas son exploitation dans les applications tels que le développement côté back-office (ex. alertes, affichage des prédictions....) et le fournisseur de données en amont (BeeSpy2).

b. Mesures de succès du projet

Pour évaluer le succès du projet, des premiers indicateurs ont été définis (ci-dessous). Ils restaient modifiables mais donnaient un cadre initial que l'on peut ainsi consulter tout au long du projet pour vérifier que l'on ne dévie pas des objectifs réels[4] .

1. Fiabilité des prédictions
2. Facilité de création de nouvelles prédictions
3. Après mise en production des produits issus du projet :
    a. Satisfaction des utilisateurs du backoffice (prédictions 1, 2, 3)
    b. Amélioration d'indicateurs clés de performances (KPI) de l'entreprise :
        i. Prédiction 4 "Customer booking probability" : nombre de réservations liées aux actions marketing associées en temps réel.
        ii. "Customer recommended offers" : évolution du panier moyen des clients, des avis clients sur leurs séjours à l'hôtel, et leur fréquence d'achat.
        iii. Évolution du taux d'overbooking (actuellement très faible donc risque qu'il augmente faiblement). Le succès tiendra également compte de la stabilité du taux.

c. Principaux risques à maîtriser

Il est essentiel d'identifier au plus tôt et le mieux possible les risques potentiels pour pouvoir les réduire au minimum et en permanence, prévoir des solutions de réponse rapide.

Un registre des risques majeurs a été créé couvrant les points suivants :

- Impossibilité d'atteindre l'objectif de "fiabilité" de prédiction
- Impossibilité d'atteindre l'objectif de délai de prédiction
- Retard dû à des recherches ou formations complémentaires nécessaires ("Manque de compétence, manque de connaissances nécessaires dans la technologie pour résoudre le problème et correctement prédire")
- Pertes côté serveur : architecture ou base de données
- Pertes côté équipe : code, données projet, ordinateurs
- Exigences mal définies
- Exigence de projet manquante
- Ressources humaines manquantes
- Mauvaise estimation des tâches
- Risque de régression du code
- Manque de fiabilité de l'architecture

Voir aussi "facteurs critiques de succès du projet"

[4] Il faudra éventuellement mettre à jour les indicateurs en fonction de l'évolution des objectifs

Pour chaque risque ont été définis : un responsable, les causes probables, les conséquences probables, les solutions de réduction de risque (dont une réponse si incident).

d. Facteurs critiques de succès du projet

Pour prévenir les risques majeurs du projet, il faut s'accorder et constamment communiquer avec les acteurs du projet sur des éléments critiques sur lesquels chacun doit garder son attention dès le début et tout au long du processus.

Les facteurs critiques sur lesquels nous nous sommes accordés avec le comité de pilotage sont :

- ✓ la disponibilité des ressources hommes, matériel et données (BeeSpy2, base de données de VeryLastRoom, services tiers).
- ✓ un accord solide sur les objectifs, une implication commune dans le projet notamment pour maîtriser les changements (alerter au plus tôt de la nécessité de changement tout en limitant les changements).
- ✓ un feedback rapide et fréquent du comité de pilotage sur la pertinence des prédictions (fiabilité, adéquation avec l'objectif métier recherché).
- ✓ la prédictibilité de l'objet recherché (ex. certains hôteliers pourraient agir de manière quasi illogique, cela rend alors difficiles certaines prédictions).

e. Acteurs et responsabilités

(Voir aussi "Stakeholders et attentes/obligations respectives" et "Structure de gouvernance du projet" de la charte du projet.)

J'ai travaillé avec les différents acteurs du projet des services techniques, commerciaux et direction pour définir les besoins et devoirs des acteurs du projet. Chaque acteur voyait ainsi son rôle clarifié et répertorié dans un annuaire des responsabilités disponible pour tous.

f. Ressources humaines

Les ressources humaines ont été définies comme suit :

- Ressources assignées :
  - Xavier : Project Manager et Développeur principal du projet.
- Ressource à assigner :
  - 15% Sébastien Houzé : Gestion VM Plateforme Irma, Décisions et réflexions techniques.
  - 10% d'un développeur VLR interne: support développement sur l'API Web Service Symfony2.
  - 1 ou 2 stagiaires après la livraison de la première prédiction seront nécessaires pour la préparation des nouvelles prédictions, la recherche et les nombreux tests nécessaires pour la phase scaling & automatisation.
- (hors projet mais à garder en tête: intégration côté back-office).

Cela restait flexible mais alertait au plus tôt sur les besoins futures et l'implication sur la structure.

g. Planification des étapes clés

A partir de l'hypothèse d'avoir toutes les ressources nécessaires, les étapes clés ont été estimées comme suit :

- Lundi 12 janvier 2014 : validation charte (conditions du démarrage projet).
- Lundi 20 janvier 2014 : "kick-off" du projet.
- Plateforme et 1ère prédiction (20 janvier au 2 avril)
    - Une réunion en comité restreint devrait être ajoutée avant le 12 mars 2014.
- Réunion résultat intermédiaire : Vendredi 12 mars 2014
    - Réunion de validation finale et kick-off 2ème prédiction : Mercredi 17 avril 2014
- 2ème prédiction : du 22 avril au 5 juin 2014
    - Réunion de validation finale et kick-off phase suivante : 5 juin2014.
- Scaling et automatisation amélioration performances : du 5 juin au 15 juillet 2014.
    - Réunion validation finale et kick-off phase suivante : 15 juillet 2014.
- 3ème prédiction : du 16 juillet au 6 août 2014.
    - Réunion de validation finale et kick-off phase suivante : 6 août 2014.
- 4ème prédiction: 6 août juin au 29 septembre 2014.
    - Réunion de validation finale et kick-off phase suivante: 29 septembre 2014.

Nous disposions ainsi d'un horizon du projet et une anticipation des besoins et résultats.

h. Estimation du Budget

L'estimation du budget a été faite depuis un premier Gantt / WBS sans correction des écarts possibles et incluait :

- ✓ Coût des différentes ressources : salaire et coûts administratifs, sous-traitance.
- ✓ Coût de l'architecture.
- ✓ Développement de la connaissance : livres, évènements (ex. conférence), consulter un spécialiste machine learning (pour ne pas bloquer sur certains problèmes avancés).
- ✓ Pas de licences spécifiques à acquérir.
- ✓ Réunions comité de pilotage.

2. Plan de gestion de projet : processus et méthodes

Le plan de gestion de projet est un document essentiel qui a été établi en début de projet. Il indiquait la méthode globale de gestion et couvrait les points ci-dessous. Il peut paraître lourd mais il est un effort nécessaire de réflexion sur l'insertion de ce projet dans l'entreprise avec des méthodes et processus adaptés pour les différentes facettes de celui-ci. Nous n'avions pas toujours toutes les réponses au départ et le plan de gestion a aussi facilité la découverte de lacunes et de risques non identifiés dans le projet.

- ✓ Cadre et définition du projet : Pourquoi ce projet ? Quels sont les objectifs ? Les livrables ? Les limites ?
- ✓ Etapes clés : J'ai simplement repris le planning approximatif des étapes clés définies dans la charte de projet. Cela rappelle les contraintes d'échéance de réalisation dans lesquels s'intègrent les différents processus et méthodes.
- ✓ Gestion des modifications et processus associé : J'y ai défini une procédure minimale et pragmatique à suivre afin de s'assurer qu'un changement en cours de projet est bien évalué et accepté par les décideurs ?
- ✓ Gestion de la communication : Les problèmes de communication dans un projet peuvent être fatals. Incompréhensions, tensions, erreurs qui auraient pu être facilement évitées, manquer le réel objectif, perte de temps et lenteurs de décision… Organiser correctement la communication est une des clés de la réussite d'un projet, donc c'est aussi très rentable. Nous avons donc défini un cadre, des règles simples et des responsabilités dans cette communication. Chaque règle défini : type de communication (ex. informel, rapport x…), description, fréquence, format (email, en personne...), participants / distribution, livrable, responsable.
- ✓ "Coûts et contrôle des coûts" et "Gestion des achats" : Après avoir fait état des coûts prévisionnels et de la méthode d'estimation utilisée, j'ai défini comment suivre et maîtriser les coûts pour anticiper tout dérapage et réagir. Pour la gestion des achats, j'ai simplement défini un budget et un seuil minimal d'achat sans autorisation ainsi que la procédure d'autorisation. Définir ce processus au départ fluidifie la suite du projet pour éviter des complications inutiles pour des achats qui peuvent être cruciaux.
- ✓ Stratégie de contrôle du cadre du projet : Comment le cadre a été défini (charte, Gantt, WBS, qualité, risques) ? Comment sera-t-il contrôlé pendant la réalisation ? Comment sera-t-il validé au final ?
- ✓ Plan de gestion du calendrier : Comment le calendrier a été défini ? Comment est-il contrôlé et mis à jour ?
- ✓ Plan de gestion de la qualité : Comment la qualité a été définie et sera contrôlée ? Il souligne l'importance d'éviter autant la sous-qualité que la sur-qualité qui est une fréquente source de retard dans les projets. Si la sur-qualité peut sembler nécessaire c'est que le niveau d'exigence de qualité a mal été défini.
- ✓ Stratégie de gestion de la connaissance : Sachant l'importance de la gestion de la connaissance, comment sera-t-elle gérée ? Pour plus de détail voir le chapitre II.A.
- ✓ Registre des Risques : Un document annexe spécial a été défini avec les risques répertoriés avec leurs sources potentielles, leurs conséquences, la réduction et le contrôle choisi ainsi que la personne responsable.
- ✓ Plan de gestion des risques : Comment les risques ont été identifiés, réduits et approuvés à priori ? Comment seront-ils contrôlés et mis à jour tout au long du projet ?
- ✓ Plan de gestion du personnel et Calendrier des ressources : De quelles ressources humaines aurons-nous besoins ? Comment gérer leur disponibilité et prévenir des risques associés ?
- ✓ Normes de qualité du projet : Des standards en termes d'objectif minimum de performance, de satisfaction, de documentation codage, de testing. Cela facilite

l'évaluation permanente et, en fin de projet, la validation de l'atteinte des objectifs en termes de qualité.

- ✓ Acceptation par le demandeur du projet (client ou sponsor) : Dernier point qui n'est pas un détail, l'acceptation des méthodes et processus par le sponsor établie un cadre précis d'exécution du projet.

B. Gestion « Micro » : Planning itératif, inspiration Scrum, TDD

La gestion du projet global par la méthode PMP assure un cadre solide mais trop rigide pour être appliqué aux nombreux petits projets qui en découlent. Ainsi pour tous les projets issus du projet global, que ce soit pour la R&D ou le développement classique, j'ai opté pour une approche plus flexible et pragmatique : un planning itératif, une méthodologie similaire à Scrum, la mise en parallèle de sprint (tâches), le développement ou prototypage orienté test.

1. Planning itératif

Le planning Gantt initial a laissé place à un planning itératif sous forme de carte mentale (réalisée avec Freemind) enrichie avec l'avancée de mes connaissances et du développement. Basé sur une copie du planning initial, il évoluait en permanence vers plus de détails et quelques modifications avec la meilleure compréhension de la technologie, et de ses utilisations par les différentes personnes impliquées dans le projet.

2. Méthodologie similaire à Scrum

La méthodologie utilisée est comparable à Scrum :

- le "backlog" correspondait à la liste des recherches, des prototypes, des développements à effectuer. Il est matérialisé par la carte mentale du planning itératif complétée des demandes et besoins à étudier, annotations de recherche et développement.
- le "sprint backlog" correspondait à la liste des tâches avec objectifs choisies parmi le backlog. Elles sont développées depuis le backlog puis insérées et gérées dans Waffle (un Kanban en ligne synchronisé avec des milestones sous Github).
- un "sprint" constituait chaque tâche avec objectif qui était lancée par moi-même ou l'équipe externe.
- le "Daily Scrum" (réunion quotidienne) était la courte réunion quotidienne en début de journée, souvent autour d'un café, pour faire la planification de la journée après revue de l'avancée.
- la rétrospective personnelle du sprint était la réunion de débriefing en fin de tâche avec éventuelle correction sur la suite du projet.
- démo et revues de sprint, à certaines étapes, étaient les réunions avec les autres membres de l'entreprise pour faire état des résultats et produits intermédiaires.

3. Paralléliser les sprints pour éviter les blocages

Que ce soit les activités de R&D ou les développements pour les livrables, ceux ci évoluent en permanence dans un cadre incertain de réussite. J'ai donc décidé de paralléliser les différentes tâches (sprint) de deux façons.

Ces sprint allaient entre quelques jours et 3 semaines maximum. Chaque intervenant avait en général 2 sprints en parallèle pour éviter de ne pas rester bloqué sur une tâche et donc être sur de toujours avancer sur le projet global malgré les aléas. C'est également un bon facteur de motivation d'avoir une certaine alternance d'activités à effectuer suffisamment compatibles pour ne pas perdre en concentration sur les sujets pointus abordés.

Pour certains prototypes et développements, j'ai lancé un "sprint" similaire sur 2 technologies différentes en parallèle qui étaient réalisées par 2 personnes différentes pour minimiser le risque d'échec et de retard dans le planning.

4. Développement orienté test ou TDD

Que ce soit pour le développement classique ou pour certains développements de prototypes, nous nous sommes imposés de créer les tests en amont sur les résultats à obtenir ou fonctionnalités à mettre au point. Cela aide à développer une bonne vision du résultat seul ou si possible en équipe, et à se concentrer sur l'objectif pendant le développement et avoir un retour rapide sur l'avancement.

C'est aussi un outil très efficace pour de la sous-traitance, nous le verrons dans le chapitre suivant sur la sous-traitance.

5. Les normes de qualité et de communication

Les normes de qualité (documentation, codage, performance…) et de communication sont celles du projet global défini dans le "'plan de gestion du projet".

C. Ressources externes : recrutement et gestion

A part ponctuellement, je n'avais pas de ressources dédiées en interne au début du projet. Réaliser un tel projet quasiment seul relevait de l'impossible sauf à réduire grandement les exigences. Ayant peu de moyens alloués, j'ai ainsi fait appel à deux types de ressources externes pour des besoins cibles : stagiaire et sous-traitant.

1. Stage (et recrutement) – processus et utilisation de challenge

Un stagiaire mal géré peut souvent être une charge importante plus qu'une aide pour réaliser un projet. Il faut ainsi bien adapter le périmètre d'un stage pour qu'il soit réalisable par un junior dans temps très limité tout en restant motivant pour ce dernier et utile pour l'entreprise. L'autre difficulté est que le recrutement d'un stagiaire peut prendre beaucoup de temps pour ne pas atteindre le résultat escompté. Après avoir contacté des universités ayant des formations en machine learning et appelé quelques stagiaires potentiels peu convaincants, j'ai donc conçu un scénario avec le directeur technique afin de maîtriser ce risque.

Nous avons défini un sujet de stage répondant à certains de nos besoins réalisable dans le temps d'un stage et pour les compétences d'un junior. Sur cette base, nous avons créé

des problèmes de machine learning à résoudre en ligne sur la base des sujets à aborder pendant le stage. Nous avions clairement indiqué que ces problèmes à résoudre correspondaient à la difficulté et aux sujets à aborder pendant le stage. L'évaluation est très rapide car ils doivent d'abord atteindre un score et si ce score est atteint nous évaluons la qualité de la démarche. Ainsi nous pouvions évaluer l'intérêt pour le sujet, la compétence à atteindre un objectif similaire avec la qualité attendue. La création de ce test m'a pris une journée mais a rendu possible la sélection et l'évaluation rapide des très nombreux stagiaires ayant postulé. 32% de ce qui ont passé le test ont atteint le score minimum attendu. De ce panel, quatre personnes ont été sélectionnées sur la base de qualité et la performance de la réponse au test ainsi que leur CV pour une courte interview téléphonique. Au final, 2 ont été rencontrés et le final a été une très bonne recrue dans le temps imparti. Soit un total de 2 jours/homme en incluant tous les temps directs et indirects dédiés à ce processus pour ce recrutement.

J'ai dédié 2 jours à la formation de ce stagiaire et lui ai donné un programme de 5 jours d'autoformation. Il alors été principalement affecté au volet recommandation du projet. Il était intégré dans la gestion de projet PMP et Scrum décrite plus haut notamment. Au début de chaque "sprint" (tâche), nous définissions les tests en amont ensemble de telle sorte que je pouvais facilement suivre son avancée de manière quantitative avant de passer plus de temps dans l'analyse qualitative de son travail.

La bonne qualité du travail du stagiaire nous a poussé à convertir ce stage en CDD (puis en prestation externe) avec un partage de son temps en entreprise entre ce projet et d'autres projets nécessitant ses bonnes compétences en Java.

2. Sous-traitance distante via Upwork (ex-oDesk)

Afin de maîtriser les délais du projet, j'ai fait appel à de la sous-traitance sur certaines tâches parallélisables du projet. De nombreuses tâches ne sont pas facilement externalisables mais celles qui le sont, avec une bonne sélection de prestataire et une gestion adéquate de la sous-traitance, atteindront des niveaux de qualité élevés pour des prix compétitifs.

a. Identification et sélection du besoin à externaliser

Ces tâches doivent être compatibles avec l'externalisation :

- ✓ facile à cadrer à distance et dans un temps limité
- ✓ qualité facilement maîtrisable à distance dans un processus semi-automatisé
- ✓ ne correspond pas à une compétence stratégique à internaliser
- ✓ pas de risque réel de fraude sur nos données ou connaissances
- ✓ l'instabilité possible du prestataire impactera faiblement le projet principal

Parmi ces tâches externalisables, j'ai choisi celles sur lesquelles la rentabilité d'une telle externalisation serait la plus forte. J'ai donc sous traité le développement de la plateforme CXF Java, du web service Java Thrift avec gestion de la mémoire en cluster pour la libraire

Weka et MOA. Il existe de très bons développeurs Java spécialisés dans certains pays et les développements Java sur des modules spécifiques sont chronophages de part la courbe d'apprentissage parfois importante.

b. Préparation de la recherche de prestataire

Sur les documents de spécifications existants, j'ai enlevé ou remplacé ce qui relevait du secret de l'entreprise et ajouté une introduction claire au projet ainsi qu'une ébauche de la liste des tests qui seraient utilisés pour le contrôle du projet. A ce stade, je limite toujours le temps investi dans cette phase de préparation pour éviter d'avoir perdu trop de temps si je m'aperçois rapidement que ne pourrais trouver le bon prestataire ou que je devrais reconfigurer le projet.

c. Recherche et sélection de prestataire

Les difficultés dans ce processus d'externalisation après la bonne identification et description du besoin est de trouver le bon prestataire. J'ai comparé sur plusieurs plateformes (Elance, Upwork…) là où il y avait le meilleur vivier de personnes compétentes sur les technologies requises. Je postais alors ma demande sur 2 plateformes avec les documents associés en faisant attention à masquer l'identité de l'entreprise dans ce processus publique de recherche. En fonction des premières réponses de prestataires avec estimation, je sélectionnais une plateforme sur laquelle j'allais approfondir mes échanges.

J'investis alors plus de temps dans l'amélioration du dossier de sous-traitance en agrégeant les différentes questions et demandes de précisions des différents prestataires et une amélioration des tests de fonctionnalités définis en amont. Ces tests sont très utiles pour les 2 parties dans la suite du projet notamment pour automatiser en partie le contrôle de l'avancement du sous-traitant avec un retour clair sur les points bloquants. Il complète le contrat de sous-traitance avec un processus claire et non équivoque.

A la lecture du dossier de sous-traitance finalisé, les sous-traitants ajustaient leur offre. Je leur demandais aussi leur compréhension du projet pour éviter tout problème de communication et jugeais par la même notre facilité de communication.

Je choisissais ainsi selon les facteurs : offre, expérience, prix, facilité de communication et compréhension du projet.

Je privilégie le contrat au forfait pour des développements ou prototypes définis et à l'heure (avec maximum fixé) pour des conseils ou finalisation de projet avec des multiples petites corrections ou améliorations.

Pour des projets où j'avais des doutes sur la facilité à sous-traiter ou sur le sous-traitant, je faisais une étude préalable ou une première étape de sélection décidant de la suite du projet.

J'ai au final travaillé longtemps avec un très bon développeur et chef de projet Java en Russie sur plusieurs contrats. Après le premier contrat, je l'ai impliqué dans les cahiers des charges et il a apporté de bonnes solutions. Les autres prestataires ont été très ponctuels.

d. Gestion de la sous-traitance

La sous-traitance s'intègre dans les mêmes méthodologies PMP et Scrum du projet mais avec des outils de gestion propre :

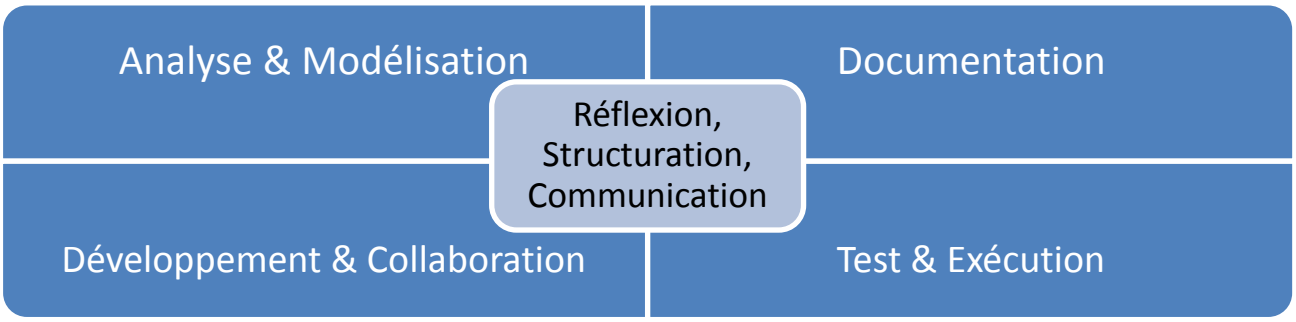
- Upwork : pour la gestion des termes du contrat, les échanges sur les modifications pour que cela soit bien archivés sur la plateforme.
- Github : pour partager la gestion du code, des branches de développement, les commentaires de code ou fonctionnalités, les validations et invalidations du développement.
- Travis CI : intégré avec Github, il lance automatiquement tous les tests existant et retourne les résultats au sous-traitant comme au donneur d'ordre. Il interdit au prestataire la demande de validation d'un développement sur Github si les codes ne passent pas.
- Google doc : pour la collaboration en ligne sur le cahier des charges (notes, questions…)
- Skype : pour les éventuelles conférences téléphoniques ou session de partage d'écran pour faciliter certaines explications.

Dès que tous les tests passaient sur Travis CI, après d'éventuelles négociations de modification du cahier des charges et des tests, le développement final était approuvé sur Github et le contrat ou l'étape (milestone) validée sur Upwork pour autoriser le paiement selon les termes du contrat.

A noter que ces projets de sous-traitance exigent aussi un management et une communication adéquats car un sous-traitant sur ce type de plateforme pourrait facilement décider d'abandonner un projet ou le mettre en attente, les possibilités de réaction dans ce cas sont limitées. Si le projet est critique, il faut soit éviter, soit doubler la sous-traitance chez un autre prestataire. Ces différents projets de sous-traitance ont par ailleurs toujours été de bonnes expériences interpersonnelles.

D. Les outils de conception et l'environnement

Pour la réalisation du projet, différents outils ont été utilisés en fonction des besoins de modélisation, de documentation, de développement et de collaboration, de test et d'exécution. Sans oublier les outils de brainstorming et de prise de notes structurées, de planification, de suivi, de communication qui ont structuré la réflexion comme l'exécution du projet.



1. Analyse & Modélisation

La majorité des analyses ont été réalisés sous forme de cartes mentales en utilisant Freemind. C'est un outil rapide pour développer et structurer son analyse ainsi qu'un bon outil de communication visuelle.

En fonction des besoins j'ai fait appel à plusieurs types de diagramme de modélisation UML.

Les « cas d'utilisation » modélisent les interactions entre le système, les utilisateurs et autres systèmes externes. Ils aident surtout à la visualisation des exigences et besoins. Les besoins pouvaient être complétés par des petits scénarios écrit en syntaxe Gherkhin (ex. « Scénario xxx : sachant que .. et … et .. quand … alors .. »). C'est un bon outil visuel de communication compréhensible même par les personnes non techniques.

Les « diagrammes de séquence » modélisent les processus métier et les échanges de données. Il en a été créé pour chaque service à plusieurs niveaux d'échange.

Le « diagramme d'architecture » a été utilisé pour modéliser et communiquer visuellement sur l'ensemble de l'architecture, son fonctionnement, ses services.

Quelques « diagrammes de classe » ont été utilisés pour modéliser les classes, types, interfaces et relations entre eux. J'en ai créé uniquement au début pour la classe la plus complexe et pour la classe d'un service dont le développement a été sous-traité. Cela nécessite une constante mise à jour pour rester utile et sachant le temps disponible et les nombreux prototypes à réaliser, ce n'était pas pertinent. J'ai opté pour une documentation claire et des tests illustratifs.

2. Documentation

La norme de l'entreprise est simple mais efficace :

- fichiers markdown (md) : à la racine de chaque projet, produit, librairie, un fichier markdown présente les éléments essentiels du module (description / résumé, options, exemples, librairies). Cette documentation est facilement mise à jour car elle est inclue dans le répertoire versionné de ce qu'elle documente et s'affiche automatiquement quand on accède au projet, produit, librairie dans Github.
- schémas ou graphiques : chaque élément d'analyse ou de documentation supplémentaire est placé dans un répertoire doc à la racine du répertoire versionné de chaque projet, produit, librairie.

3. Système d'exploitation pour le développement et la production

Le système d'exploitation utilisé est Linux : Ubuntu pour les postes de développement, Debian pour l'environnement serveur virtualisé (12 GB de RAM, 8 cœurs). Il a été choisi notamment car l'entreprise ne souhaitait pas avoir à acquérir de licences côté serveur et que de nombreux projets de machine learning offrent des facilités pour le développement sous Ubuntu.

4. Développement et collaboration

a. IDE (Développement)

Eclipse et IntelliJ sont des IDE de développement multiplateformes en Java. Les deux ont été utilisés en fonction du projet open source sur lequel nous travaillons car ils étaient souvent fournis avec une pré-configuration. Tous les projets ont finalement été migrés, si nécessaire, pour un développement sous IntelliJ qui a été notre préférence en termes d'ergonomie et de fiabilité. Dans le cadre de ce projet, ils sont utilisés pour les modules de recommandation basés sur Lenskit, pour les modules de machine learning utilisant Weka et MOA ainsi que pour la passerelle SOAP / CXF / ADAMS.

Pour le C++, VIM est une IDE simple et classique en mode terminal mais efficace. Elle a été utilisée pour la contribution au projet open source de machine learning Vowpal Wabbit et des compilations C++ pour Python.

Pycharm est un IDE pour Python basé sur Java. Dans le cadre de ce projet, il est utilisé pour le développement du module de WSIrma, streamer, Poller et certains modules métiers de recommandation.

b. Gestionnaires de dépendances

Les gestionnaires de dépendances ont été utilisés suivant les langages : Maven ou Gradle pour Java, PIP ou PyPI pour Python, NPM pour Node.JS,

c. Gestionnaire collaboratif de version

Github est un gestionnaire de version décentralisé. Dans le cadre de ce projet, il a été utilisé pour stocker les différents projets, faciliter le travail collaboratif, définir des règles de développement, automatiser les tests avec une intégration avec Travis CI.

5. Tests automatiques, pré-production et production

Travis est un environnement de test pour l'intégration continue. Dans le cadre de ce projet, il a servi pour les tests fonctionnels et unitaires. Ces tests étaient créés avec les librairies JUnit pour Java, unittest pour Python, l'extension des tests existant en Perl pour le C++ de Vowpal Wabbit.

Pour la pré-production et la production, l'environnement de serveur virtuel dédié au projet a été utilisé (décrit dans « Système d'exploitation »). Quelques tests comparatifs ont aussi été effectués dans des environnements « cloud » sur Heroku et Amazone.

6. Réflexion, structuration, communication

Afin de gérer les différents besoins de réflexion, prise de note, structuration, et de communication, j'ai utilisé :

- ✓ Freemind pour les brainstorming, feuilles de route, organisation et log des expériences.
- ✓ Planner (équivalant Ubuntu à MS Project sur Windows) pour la planification et les WBS.
- ✓ GitHub pour archiver avec le code tous les documents d'analyse, de modélisation, de documentation.
- ✓ L'intégration Waffle avec Github pour les objectifs, sous-objectifs, fonctionnalités, tâches, bugs sous forme de kanban dans Waffle et de tags colorés dans GitHub.
- ✓ Google Docs pour les divers documents ad hoc (charte de projet, spécifications…).
- ✓ Slack, Gmail entreprise, Skype pour les communications écrites ou vocales.

E. Nécessité de formation et de recherches pour le projet

1. Gestion de la connaissance, une priorité

Pour un tel projet, la gestion de la connaissance a été une priorité majeure et le prérequis aux différentes étapes de la recherche de solutions.

Vu les ressources très limitées, je n'ai pas mis en place un outil lourd de gestion de la connaissance mais :

- Une carte mentale (mindmap) principale avec feuille de route par thèmes et objectifs avec des arbres de connaissance.
- Des répertoires de ressources: sites et liens web, livres, articles scientifiques et thèses, cartes mentales thématiques, algorithmes et codes spécifiques.
- Des processus d'apprentissage et de veille : suivi de MOOC et cours en ligne, cours de mathématiques avec professeur d'université, abonnements RSS, participation à des newsgroups et communautés, autoformations, mini-projets, suivi de conférence.
- J'ai développé un petit réseau avec quelques experts en machine learning à travers les projets open source auxquels j'ai contribué, des contacts directs avec des spécialistes, mes rencontres à la conférence PAPIs.io à Barcelone.

J'ai ainsi partagé et développé cette connaissance au fur et à mesure autour de ces questions :

- Quel est le besoin de formation actuel et futur ? Le processus d'identification et de sélection définit ce qu'il faut apprendre en évitant le sur apprentissage. Ce processus sera permanent dans notre cas.
- Comment acquérir ces connaissances efficacement (méthodes, outils et supports, évaluation, structuration, motivation...) ?
- Comment optimiser l'acquisition des connaissances face aux contraintes du projet ? L'acquisition de connaissance peut être longue et sans fin, tout comme le défaut de connaissance peut faire perdre beaucoup de temps voire être fatal au projet. Acquérir ces connaissances au bon moment et dans un temps limité est crucial. Pour que cette gestion de la connaissance soit pertinente, elle doit se développer en l'optimisant face aux différentes contraintes du projet : *périmètre et objectifs, délais, risques, budget, qualité, ressources*.
- Comment en faire bénéficier l'entreprise ? Ces connaissances et leur gestion sont un investissement stratégique pour l'entreprise. Cette question est externe aux objectifs techniques cités mais nécessaire dans une démarche R&D qui vise notamment à ouvrir des pistes de développement futur pour saisir de nouvelles opportunités. J'ai donc, par exemple, régulièrement pris du temps pour diffuser cette connaissance nouvelle pour l'entreprise en machine learning et data mining auprès de l'équipe technique comme commerciale. L'objectif était de faire mieux comprendre ce domaine, ses contraintes, son potentiel et ainsi générer de nouvelles opportunités.

Évaluer le besoin de formation nécessite un diagnostic entre ce qui doit être acquis pour atteindre les objectifs et les connaissances actuelles. La principale difficulté de ce processus d'identification et de sélection du besoin de formation venait de la nature des objectifs du projet :

- des objectifs flous au départ : l'entreprise savait qu'elle pouvait gagner de l'argent en investissant ce domaine, elle avait des pistes mais pas d'objectifs concrets. Il m'a donc fallu préciser ces objectifs tout en développant ma connaissance pour que ces objectifs soient les plus pertinents possibles dans l'état de l'art actuel.
- une fois les objectifs définis, ils ont tout de même évolué (voir III.A). Le besoin de formation évolue avec les objectifs et l'amélioration des connaissances, il faut donc faire attention à constamment capitaliser sur le travail déjà effectué tout en optimisant les besoins à venir.

2. Diagnostique de départ et thèmes de connaissance

J'ai donc fait un diagnostic de départ sur les différents thèmes de connaissances nécessaires pour la réalisation des objectifs. Le but étant d'en déduire les besoins initiaux et pistes futures. Ce diagnostic a évolué au fur et à mesure du projet.

Les axes de compétences et connaissances nécessaires pour ce projet ont été les suivants :

a. Gestion de projet

Ayant géré de nombreux projets informatiques (analyse, développement, déploiement, administration) avec d'importantes similarités (forte orientation données, R&D, e-commerce), mon besoin de formation sur ce point était plus faible. A noter que j'étais nouvel arrivant et au départ seul sur ce projet, cela a constitué les principaux challenges et besoins d'adaptations de mes connaissances sur ce point.

b. Analyse et modélisation : méthodes et outils

J'ai une bonne expérience et régulière sur ce point à l'exception de quelques révisions nécessaires sur la modélisation la plus proche de l'implémentation car mon expérience des dernières années est la gestion de projet plutôt que le développement lui-même.

c. Gestion du développement

Scrum, TDD (développement orienté test) et Travis CI, gestion de version avec Github

J'avais une bonne connaissance des méthodes mais les outils étaient neufs pour moi et impliqués des processus nouveaux, le besoin de formation était donc fort en début de projet pour être rapidement efficace.

Pour me former, j'ai donc :

- ✓ suivi des formations interactives sur codeschool.com pour GitHub.
- ✓ combiné mes lectures web avec des mini-projets et tests pour TravisCI, Scrum, GitHub.
- ✓ repris un MOOC (BerkeleyX CS 169.2x) que j'avais suivi sur le développement orienté test.
- ✓ posé des questions sur Stackoverflow.com dès que j'allais être bloqué. Je basculais alors sur un autre « sprint » (mini projet) en attendant la réponse.
- ✓ beaucoup appris avec les membres de l'équipe technique.

d. Intégration et développement

Technologies abordées : Python, Java, C/C++, Thrift, Node.JS, MySQL / Elasticsearch, Storm / Spark, SOAP / XML / JSON / REST, IntelliJ / Eclipse, Ant / Maven / Gradle, RabbitMQ / ZeroMQ, Java Infinispan / Java CXF

Je devais apprendre le Python et de nombreux outils mais mon besoin majeur était surtout de rafraîchir mes connaissances pour retrouver une bonne efficacité en développement objet sachant que je supervisais des projets mais codais rarement depuis quelques années, à l'exception de prototypes en Java et C embarqué.

Pour les connaissances ponctuelles mais non stratégiques à acquérir ou posant un problème de timing, j'ai fait appel à de la sous-traitance ou délégué si possible à une personne de l'équipe technique.

Les principaux supports de formation utilisés pour ce point : formations interactives sur codeschool.com et équivalents, ressources web que je valide avec des mini-projets, stackoverflow.com sur lequel je pose souvent des questions, documentations, mini-projets.

e. Connaissances « métier »

(Réservation hôtel dernière minute (tarification, stratégies), yield management, e-commerce, secteurs similaires.)

Le besoin sur ce point fut majeur et prioritaire pour atteindre un résultat pertinent pour l'entreprise.

Le principal vecteur de formation furent les discussions avec les personnes des différents services de l'entreprise combinées avec la lecture d'un ensemble de ressources (papiers scientifiques, présentations sur slideshare.com) sur les différentes thématiques du métier de l'entreprise ou de métiers dont les problématiques sont similaires (ex. yield management et prédiction pour les billets d'avion).

f. Données et machine learning (voir aussi la partie III du document)

La principale difficulté (élément stratégique de connaissance du projet) était lié au traitement de données délivrant différents services de prédiction, d'optimisation et de recommandation.

Cela couvre notamment :

- l'analyse des données
- le stockage des données
- la gestion de flux de données et agrégation
- la sélection des algorithmes de machine learning, leur évaluation, leur optimisation
- la préparation des données pour le machine learning : sélection des caractéristiques et extraction, optimisation et automatisation
- des connaissances mathématiques en statistiques et probabilités
- les librairies majeurs de machine learning et statistiques (R, Mahoot, Scikit, Weka, Vowpal Wabbit, Torch, Caffé, Theano, Orange, MOA / SAMOA, Panda, Numpy/Scipy)
- le machine learning en ligne (flux, temps réel) : algorithmes en ligne (online), out-of-core, gestion du concept drift (glissement ou changement du modèle de donnée), évaluation temporelle (ex. backtesting)
- le machine learning pour les systèmes de recommandation
- le machine learning pour l'optimisation
- les architectures de service pour le machine learning et les gros volumes de données (ex. architectures lambda)

Après une première expérience de création d'une plateforme B2B e-commerce et logistique utilisant de gros volumes de données, j'ai créé et géré pendant 10 ans le développement de solutions télématiques pour le secteur pétrolier. J'ai ainsi acquis une bonne expérience en conception de systèmes d'information, développement de plateformes orientées données, la gestion de base de données volumineuses et critiques, le développement et l'administration de solutions télématiques (flux d'événements, statistiques, communication, API).

A part un court projet, le volet machine learning était réellement nouveau pour moi tout en étant un prolongement de mon expérience. Mon expérience en machine learning au début du projet était donc limitée :

- j'avais développé pour un client pétrolier un service utilisant un classifieur linéaire pour générer des alertes sur les risques pour des flottes de véhicules et conducteurs en exploitant ses données télématiques (conduite, géolocalisation, planification).
- suite à cela, j'ai suivi avec passion et réussi le MOOC de l'université de Stanford sur le machine learning composé de cours et de challenges nécessitant programmation et mathématiques sur : l'apprentissage supervisé et non supervisé, les fonctions de coût et la descente de gradient, la régression logistique, les réseaux de neurones, les SVM, le clustering, la réduction de dimensionnalité, la scalabilité des projets de machine learning.
- de nombreux domaines du machine learning et des systèmes de recommandation m'étaient inconnus.
- je n'avais pas d'expérience concrète dans l'utilisation de la majorité des modèles majeurs utilisés en machine learning (réseaux de neurones, SVM, ensembles…) et systèmes de recommandation.
- je n'avais aucune expérience ni connaissance sur le machine learning en flux et en ligne, ni sur l'architecture nécessaire et encore moins sur son optimisation.
- j'avais oublié de nombreuses notions et notations mathématiques nécessaires pour comprendre les articles de recherche véritable source de qualité sur ce domaine nouveau ou en perpétuel évolution.

3. Sources et vecteurs de connaissance en machine learning

Pour me former j'ai combiné différentes sources et vecteurs de formation afin d'acquérir les connaissances nécessaires tout en effectuant une veille permanente sur ce domaine en pleine évolution :

- Cours de Mathématique avec un professeur d'université spécialisé en probabilités et statistiques.
- Suivi de nombreux MOOC anglophones en machine learning et big data.
- Lecture de différents livres de référence conseillés sur les forums spécialisés.
- Abonnements à des flux RSS pour la veille dans le domaine et pour développer une culture plus long terme.
- Lecture de thèses et articles sur le machine learning dont ceux appliqué aux métiers similaires à celui de l'entreprise.

- Réalisation systématique de mini projets sur les librairies de machine learning pour les comprendre et les comparer.
- Amélioration des connaissances via les sites de concours de machine learning.
- Conférence : suivi de la première conférence mondiale dédiée aux API de machine learning.
- Sites de Questions / Réponses anglophones spécialisés en machine learning et statistiques.
- Développement d'un réseau et implication dans les forums de différentes librairies[5].
- Twitter (personnes et journaux de référence).
- Aide de spécialistes.

Une liste détaillée des ressources utilisée est disponible en fin de document.

F. Processus de recherche de solutions et d'opportunités (R&D)

Le processus de recherche a été un processus itératif fortement couplé avec les phases de développement.

C'est un petit projet (1 à 3 personnes) avec des cycles courts (2 à 6 semaines par itération). Le choix a été fait, en plus des prototypes intermédiaires, de réunir les premiers produits dans une architecture simple et éphémère pour valider la voie engagée et évoluer, dans un deuxième temps, vers une architecture consolidée.

Ce projet intègre de la R&D, pour définir les objectifs et détecter des opportunités tout en progressant dans la connaissance, et la réalisation des différents objectifs issus de cette R&D.

Le processus de recherches s'intégrait ainsi tout au long du projet dans les différentes étapes des cycles allant des besoins à leurs solutions. Plusieurs cycles étaient constamment lancés en parallèle pour ne pas bloquer l'avancé globale du projet sachant que la nature propre de la R&D fait qu'un cycle peut ne pas aboutir ou être fortement retardé.

Les étapes d'un cycle et l'implication du processus de recherche peuvent se résumer comme suit :

1. Pré-étude (1ère analyse)

Sur la base d'un besoin, un ensemble de recherches sont faites pour :

[5] Je suis aujourd'hui le modérateur principal du forum de Vowpal Wabbit (librairie majeure de machine learning en temps réel)

- ✓ définir les différents champs de recherche à approfondir pouvant répondre à ce besoin.
- ✓ identifier les challenges.
- ✓ limiter les risques.
- ✓ estimer la charge de recherche et développement.

Une feuille de route défini ainsi les différentes étapes possibles (prototypes) et exigences (formation, recherche, prototype existant et autres ressources) pour atteindre l'objectif.

2. Planning

Une première planification est réalisée en s'appuyant sur les évaluations des différents acteurs, la connaissance acquise, notamment dans la phase de pré-étude, et éventuellement l'aide d'experts extérieurs.

3. Analyse

Une analyse est réalisée avec une éventuelle modélisation d'un ou plusieurs prototypes. Cette analyse est souvent approfondie dans la phase suivante.

4. Cycles « Formation / Recherches / Prototypes »

Les pistes de chaque option envisagée sont ainsi approfondies jusqu'aux prototypes. Ainsi, un cycle de formation, recherches, et de prototypage sont nécessaires pour arriver à l'objectif en une ou plusieurs itérations. Les recherches effectuées dans cette phase amènent fréquemment à corriger l'objectif voir à relancer une itération depuis la phase de pré-étude.

Sur la base des résultats des prototypes un choix est opéré, souvent basé sur l'adéquation avec l'objectif initial, la complexité, la performance, les opportunités comparées.

5. Développement

Les produits retenus issus du prototypage sont programmés pour le développement (voir partie III). Vu que l'équipe projet était très restreinte cela a souvent consisté à enchaîner directement sur le refactoring du code des prototypes, documentation, consolidation des tests (unitaires, intégration et non-régression).

6. Amélioration ou élimination

a. Retours et Progression de la connaissance

Les retours du produit comme l'évolution de la connaissance, acquise tout au long de la démarche R&D, ont plusieurs fois conduit à lancer des phases d'amélioration et dans certains cas à abandonner le produit développé à la faveur d'un successeur ou pas.

b. Evolution de l'architecture

La R&D s'est faite par de multiples itérations avec, dès le début, le choix d'imaginer des solutions avec une première architecture simplifiée pour réussir ou échouer vite dans cette

première phase. Une deuxième phase a consisté à consolider l'architecture. Les connaissances actuelles en architecture dans ce domaine ne résument pas cette évolution à une simple migration, cette deuxième phase faisait donc partie de la R&D qui impliquait pour certains produits une nouvelle étude.

III. Le « Machine learning » : Connaissances nécessaires

Je vais développer dans ce chapitre les principaux axes de connaissance ci-dessous sur lesquels j'ai travaillé pour structurer mes recherches et nécessaires pour mieux comprendre le projet et ses enjeux techniques :

- présentation et domaines d'application du machine learning
- les différents types d'apprentissage
- la préparation des données
- les modèles générés par les algorithmes d'apprentissage et leur évaluation
- les architectures typiques du machine learning et du big data
- une proposition de cycle de vie efficace pour un projet de machine learning

Ces sujets sont été à chaque fois approfondis et référencés (voir « Gestion de la connaissance, une priorité »).

A. Présentation et domaines d'application

Le machine learning (ML), ou apprentissage automatisé, est un champ de l'intelligence artificielle. Il peut être défini comme couvrant l'étude, la conception et le développement d'algorithmes donnant la possibilité à des machines d'apprendre sans avoir été explicitement programmées (définition d'Arthur Samuel en 1959). Au lieu d'écrire un programme à la main, un algorithme de ML va analyser de nombreux cas d'exemple pour produire un programme ou modèle pour effectuer la tâche illustrée par les exemples. Un tel algorithme peut combiner un très grand nombre de données et règles différentes, et être mis à jour en lui fournissant de nouveaux exemples. Si les exemples fournis sont représentatifs du processus à modéliser, l'algorithme fonctionnera aussi bien sur de nouveaux cas d'exemple. La forte augmentation des capacités de calcul et de données disponibles rendent le ML préférable au développement manuel de tel traitement et permet la création nouveaux programmes auparavant irréalisables sachant la complexité et le temps de développement qu'il aurait fallu.

Une tâche de machine learning peut ainsi apprendre à : classifier, prédire, recommander, optimiser, détecter des motifs ou anomalies, filtrer… Les tâches de machine learning peuvent évidemment être combinées avec d'autres algorithmes ML ou non pour produire des algorithmes plus avancés.

Les applications sont nombreuses :

- prédire la météo, le prix d'une action, une prédisposition génétique, la page la plus pertinente pour une question (moteurs de recherche), la probabilité de survenue d'un virus, d'un incident actuel ou future.
- prédire la performance : d'une pub ou campagne marketing (ex. Adwords, Criteo), d'une page web, d'un canal de distribution, des implantations de magasins, d'un produit pour un profile de client.
- recommander par la prédiction : des produits les plus complémentaires ou qui seront les plus appréciés par un client (Amazon, Netflix), la stratégie la plus adéquate

(gestion de portfolio en trading), l'action de maintenance la plus adaptée (ex. gestion automatisée des datacenters comme pour Office 365).

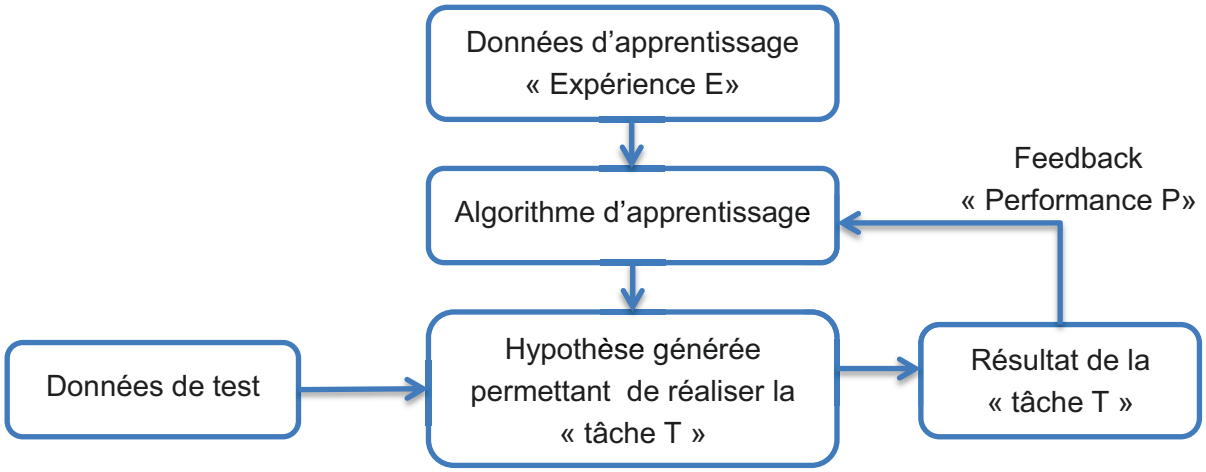
- faire de la reconnaissance : vocale, faciale, de mouvement (ex. Kinect de Microsoft), d'écriture (ex. les codes postaux sont identifiés dans les centres de tri depuis très longtemps par des réseaux de neurones).
- l'exploration de données ou data mining.
- classer ou identifier :
    - des éléments dans une image ou vidéo pour la classer, la commenter, l'indexer, réagir (ex. système de sécurité embarqués dans les voitures).
    - les sujets (ex. Google News) et les sentiments dans des textes (ex. analyses Twitter en temps réel pour trading et marketing) ou même le niveau de langue.
    - les prospects susceptibles de devenir clients (meilleures opportunités) ou de redevenir client (winback).
    - les clients susceptibles de partir (churn prediction) ou d'être intéressés par une offre.
    - les réclamations clients pour un processus adapté (ex. réponse automatique ou manuelle).
- découvrir et identifier les meilleures stratégies : de jeu (ex. Google DeepMind pour les jeux Atari), de stabilisation d'un robot ou d'un aéronef.
- détecter : des fraudes potentielles, des anomalies réseaux, une anomalie de capteurs (ex. alertes préventives centrales nucléaires), des objets ou comportements suspects (ex. aéroports), des erreurs d'orthographe ou de codage.
- optimiser l'exploitation par la prédiction de l'utilisation des ressources (ex. réseau de vélos et voitures, réseau électrique…).
- la traduction automatique.

B. Fonctionnement de l'apprentissage automatisé

1. Objectif et caractéristiques du problème d'apprentissage

La définition de l'objectif de l'apprentissage automatisé par Tom M. Mitchell, directeur du département de Machine Learning à la Carnegie Mellon University, clarifie le problème à maximiser :

- Étant donné : l'expérience E, une classe de tâches T, une mesure de performance P.
- On dit qu'une machine apprend si : sa performance sur une tâche de T mesurée par P augmente avec l'expérience E.



Dans notre cas :

- L'expérience sera nos données d'apprentissage souvent appelées exemples.
- L'hypothèse générée par l'algorithme d'apprentissage sera appelée « Modèle »
- Les mesures de performances sont couramment : la précision, le taux d'erreur, la variance entre le résultat donné par l'algorithme et le résultat attendu.

Un problème d'apprentissage peut ainsi se caractériser par :

- Un type d'apprentissage définissant la façon d'interagir avec l'environnement.
- Une sortie dont on va mesurer l'erreur généralement sous forme d'une « fonction de coût » à minimiser.
- Un « modèle » et ses paramètres.
- Un algorithme pour créer et adapter le modèle en utilisant les exemples d'apprentissage issus de l'environnement, de façon à optimiser la fonction de coût.

2. Les principaux types d'apprentissage

Pour s'orienter et structurer la résolution du problème, il est essentiel d'identifier le type d'apprentissage. Les types classiques d'apprentissage sont les suivants :

a. L'apprentissage supervisé

En analysant une base d'exemples contenant chacun des données d'entrée "avec" un résultat cible en sortie, une fonction de prédiction sera produite généralisant la règle d'association entre les données d'entrée et de sortie. Ainsi la fonction générée aura pour paramètre des données d'entrée similaires aux exemples d'apprentissage ou nouvelles et retournera le résultat inféré sur la base de la règle d'association produite.

b. L'apprentissage non-supervisé (ex. clustering)

En analysant une base d'exemples contenant chacun des données d'entrée "sans" un résultat cible en sortie, une fonction sera produite classant en groupes homogènes ces données selon une règle d'association généralisant ce classement. Ainsi la fonction générée aura pour paramètre des données d'entrée similaires aux exemples d'apprentissage ou nouvelles et retournera le résultat inféré sur la base de la règle d'association produite. La

différence majeure avec l'apprentissage supervisé est qu'il n'y a ici pas de résultat à priori, le but est de découvrir le meilleur classement des données fournies et des structures invisibles.

c. L'apprentissage semi-supervisé

L'apprentissage semi-supervisé est une technique d'apprentissage supervisé exploitant l'apprentissage non-supervisé pour analyser les données d'exemple qui n'ont pas de données de sortie. En apprentissage supervisé, avoir des données d'exemple avec la sortie souhaitée peut être difficile ou onéreux. Ainsi, il est courant d'avoir qu'une fraction des données disponibles avec la valeur de sortie et la majorité des données sans.

Un exemple serait une banque de données de millions de photos que nous voudrions classer : nous pouvons classer 5% des photos manuellement et analyser automatiquement le reste des photos pour découvrir les traits majeurs distinctifs de ces photos.

Le processus d'apprentissage non-supervisé, en ajoutant aux données d'apprentissage des informations de classement et de structure découvertes sur un volume de données bien plus important, aidera l'apprentissage supervisé à produire une règle d'association qui se généralisera beaucoup mieux sur l'ensemble des données.

d. L'apprentissage par renforcement (ex. q-learning)

L'apprentissage par renforcement apprend un comportement décisionnel optimal à partir de récompenses ponctuelles. L'algorithme optimisera le comportement pour obtenir la récompense quantitative maximale au cours du temps. Il est courant de l'illustrer par un agent autonome plongé dans un environnement qui doit apprendre de ses décisions permanentes en ayant ponctuellement des récompenses positives ou négatives, il doit alors comprendre et apprendre de son expérience passée pour optimiser sa stratégie.

3. Autres caractéristiques du problème d'apprentissage

En complément de celles définies par le type d'apprentissage, d'autres caractéristiques majeures facilitent la compréhension et la formalisation le problème :

a. Sortie : classification ou régression (ou autre) ?

Le résultat à produire en sortie de l'algorithme est classiquement une classification ou une régression. La classification a pour but de catégoriser les données d'entrée fournies dans des classes simples (ex. vrai / faux) ou multiples (ex. les éléments d'une scène), la régression doit prédire des valeurs numériques de sortie (ex. probabilité, score, valeur). Il est aussi possible de produire une structure, une séquence, des règles d'association.

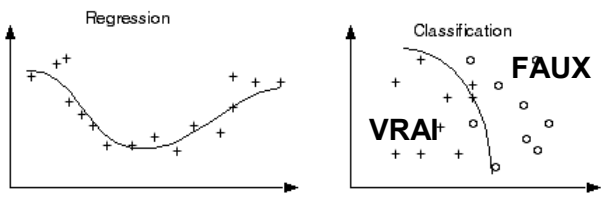


Figure 1 : classification vs régression

b. Apprentissage “hors-ligne” vs “en-ligne”

L’apprentissage hors-ligne est la méthode la plus classique où l’on apprend dans un premier temps avec les exemples disponibles pour créer un modèle statique puis on utilise ce modèle pour prédire. L’apprentissage en-ligne met à jour le modèle au fur et à mesure que de nouvelles données d’apprentissage arrivent. L’apprentissage hors-ligne atteint en général de meilleures performances pour un même jeu d’exemples donné. L’apprentissage en-ligne est utilisé quand il y a un grand nombre de données d’apprentissage, car elles peuvent être traitées au fil de l’eau sans nécessité de stockage, et qu’avoir des données récentes dans le modèle peut significativement améliorer le modèle.

c. Et les systèmes de recommandation ?

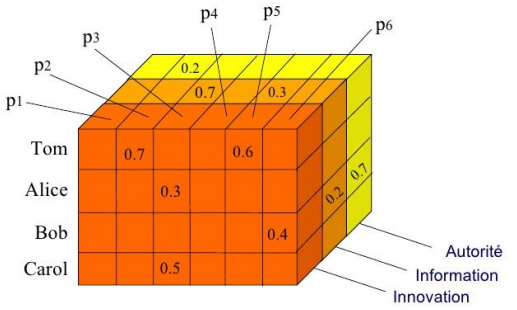


Figure 2 : matrice de recommandation multiple

Les systèmes de recommandation visent à prévoir le score ou la préférence d’un utilisateur pour un produit ou plus généralement le score évaluant la qualité d’association entre des entités. Ils apprennent depuis des scores existants, explicites (ex. note d’un utilisateur) ou implicites (ex. temps sur une page, click...), pour prédire les autres scores entre les entités afin de recommander les meilleures associations. C’est donc en général un problème d’apprentissage supervisé avec des techniques propres aux systèmes de recommandation (ex. filtrage collaboratif “item-item” ou “user-user”).

d. Optimisation combinatoire : parcours de graphe, heuristique

Les problèmes de machine learning sont souvent couplés à des problèmes d’optimisation combinatoire comme la recherche du chemin le plus court, la tournée optimale du voyageur de commerce.

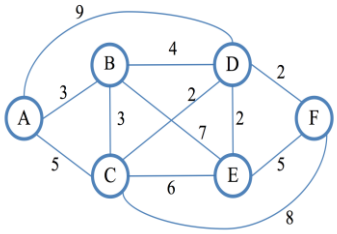


Figure 3 : exemple de formalisation de graphe pour recherche du chemin le plus court

On peut résoudre ces problèmes avec les algorithmes classiques liés aux parcours de graphe suivant le problème : recherche de chemin le plus court (Dijkstra, A*, DFS, BFS), recherche des flots maximum (Ford-Fulkerson, BFS), calcul d’un arbre recouvrant (Prim), trouver la séquence la plus probable (Viterbi).

Pour résoudre des problèmes où les méthodes classiques ne fonctionnent pas ou sont trop longues à converger, de nombreuses heuristiques, recherchent la solution par différentes règles de recherche itératives :

- Les algorithmes génétiques,
- Les algorithmes utilisant l’intelligence globale et l’auto-organisation : « colonies de fourmis », cartes de Kohonen (non supervisé), Optimisation par Essaim Particulaire (OEP)
- Le renforcement apprenant de manière différé sur l’expérience (Q-learning, Sarsa)
- Les algorithmes de parcours de l’espace de recherche et spécialisation (ex. convergence par tâtonnement) : “recuit simulé”, “recherche tabou”.

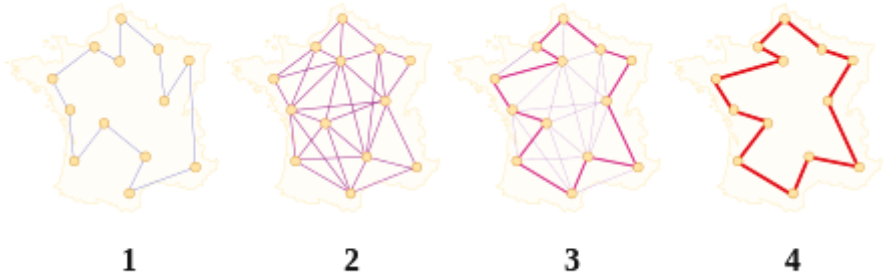


Figure 4 : Illustration de l'algorithme d'optimisation « colonies de fourmis » qui explore dans de multiples zones locales et converge vers une solution optimale par intelligence collective

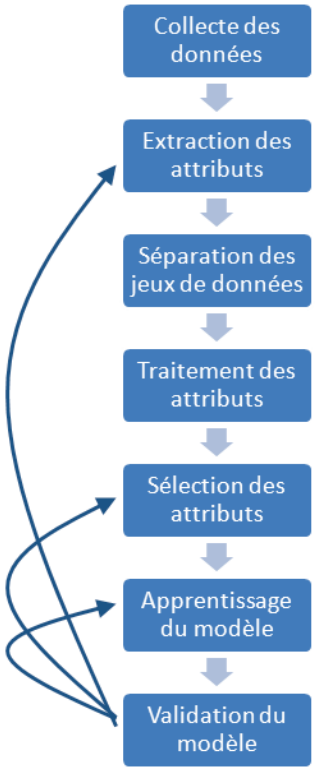


Figure 5 : sélection des attributs à différentes phases

4. Choix et préparation des données d'apprentissage (attributs ou variables explicatives)

Les données d'entrée des exemples d'apprentissage (hors valeur de sortie) ou de prédiction sont individuellement appelés attributs. La sélection et l'amélioration des attributs interviennent continuellement dans le processus de création et d'amélioration d'un traitement de machine learning. Fournir des données pertinentes est évidemment primordial mais leur qualité l'est tout autant :

a. Malédiction de la dimension : réduction et sélection

Le nombre d'attributs (valeurs) fournis à l'algorithme d'apprentissage va grandement influer sur la complexité, le temps de calcul, notre propre travail de préparation des données. Il faut ainsi essayer de réduire le nombre d'attributs pour pouvoir se concentrer sur les plus intéressants. Heureusement il existe différents algorithmes aidant à la sélection des attributs les plus pertinents comme l'analyse en composante principal (PCA) et d'autres.

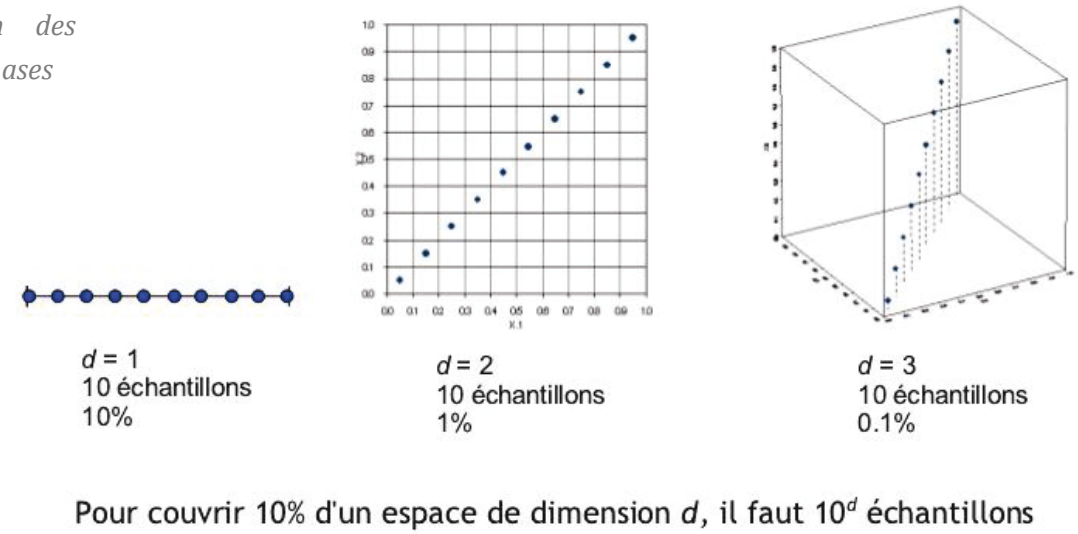


Figure 6 : Illustration de la malédiction de la dimension montrant l'accroissement rapide de la complexité par l'ajout de dimensions (ajout d'attributs dans les exemples d'apprentissage)

b. Attributs manquants ou "sparse data", erronés, bruités, redondants ou dépendants

Il est rare d'avoir des données parfaites. Pour chaque cas des solutions existent :

- ✓ attributs manquants ou clairsemés (sparse) : remplacer par 0, la moyenne, valeur probable par analyse statistique des autres exemples. Il existe de nombreuses techniques d'imputation et si de nombreuses données sont manquantes par nature (sparse data), il faudra opter pour un algorithme d'apprentissage qui accepte ce type de données.
- ✓ attributs erronés ou bruités : l'analyse de la distribution des données est une bonne méthode pour découvrir les valeurs erronées ou bruitées à supprimer ou corriger. Si des données sont bruitées et qu'elles ne peuvent pas être supprimée ou corrigée, il est important de choisir un algorithme d'apprentissage qui soit robuste face au bruit.
- ✓ attributs redondants et dépendants : il est important d'éviter d'avoir des attributs qui soient redondants ou dépendants notamment pour les algorithmes qui supposent que les données sont statistiquement indépendantes. Si besoin, l'analyse de corrélation des attributs rend possible la découverte automatique d'attributs trop fortement corrélés.

c. Normalisation

De nombreux algorithmes sont plus performants suivant la plage des données, il est ainsi assez courant d'avoir à les ré échantillonner sur [0, 1], [-1, 1], ou autre plage et de les centrer sur 0.

d. Discrétisation

Des données numériques réelles peuvent être transformées en valeur discrètes selon différents stratégies (ex. plage de distribution) pour être gérés comme des attributs catégoriques, cela peut conduire à une importante amélioration de la performance.

e. Amélioration ou création de données calculées

La majorité des algorithmes d'apprentissage ne peuvent comprendre qu'une date correspond à un jour de la semaine, qu'utiliser le log d'une valeur améliorera sa capacité à faire des liens de distribution, qu'il est préférable d'utiliser le produit de 2 valeurs que les 2 valeurs indépendantes (ex. surface = données 1 x données 2). Créer de ces données calculées sur des données pourtant existantes peut largement améliorer la performance de prédiction en aidant l'algorithme à faire de nouveaux liens pertinents entre les données d'entrée et la sortie.

f. Données temporelles et d'état

La plus part des algorithmes d'apprentissage supposent les exemples comme indépendants des uns des autres et ne peuvent prendre en compte un quelconque lien. Si

des exemples ont un lien temporel ou un autre lien de séquence, il peut être crucial de créer des nouveaux attributs caractérisant l'évolution temporelle ou séquentielle : moyenne pondérée, delta, variable à état sur différents échelles de temps...

5. Les familles d'algorithmes

Il n'existe pas encore d'algorithme magique répondant à tous les besoins et de manière optimal. Il faut ainsi avoir une bonne connaissance des différentes familles d'algorithme pour s'orienter dans les algorithmes à tester, les paramétrer correctement, les combiner (voir section "super modèles"), transformer les données si besoin et mieux comprendre les résultats pour gagner du temps.

a. Résolution ou approximation de systèmes linéaires simple ou multiple

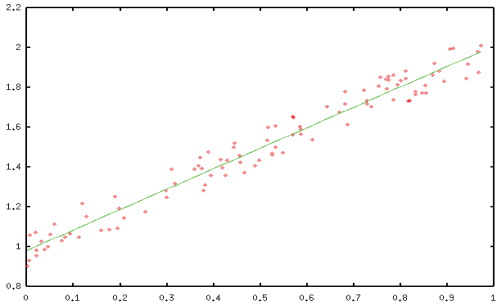


Figure 7 : simple droite de régression linéaire

De nombreuses méthodes mathématiques classiques rendent possible la résolution ou l'approximation de systèmes linéaires et non-linéaires, ou la classification des données. Les algorithmes correspondant ont ainsi été largement implémentés et optimisés pour différents cas d'utilisation. Les fonctions de régressions sont très utilisées car fournissent rapidement un résultat approximatif avec une implémentation simple mais pour des problèmes complexes ont tendance à produire des modèles qui se généralisent mal (surapprentissage).

b. Arbres de décision

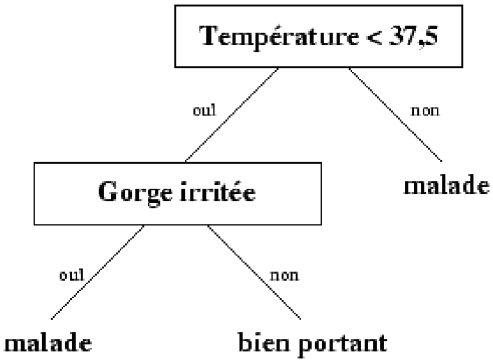


Figure 8 : arbre de décision

Ces algorithmes construisent des arbres de décision liant les données d'entrée jusqu'à une sortie plus ou moins optimale. Ils sont majoritairement dédiés aux problèmes de classification mais peuvent être adaptés pour de la régression. Un objectif majeur des algorithmes ML étant leur capacité de généralisation, le choix se fera par les critères de segmentation utilisés, les méthodes d'élagages implémentées pour éviter le surapprentissage, et la manière de gérer les données manquantes dans le parcours de l'arbre.

c. Réseaux de neurones artificiels et "Deep learning"

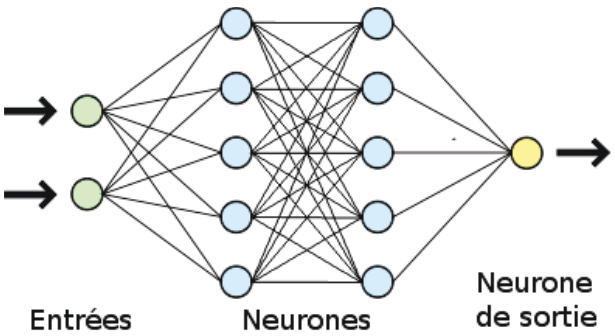


Figure 9 : réseau de neurones artificiels

Inspiré du fonctionnement des neurones biologiques, les réseaux de neurones artificiels offrent une très grande flexibilité mais nécessitent beaucoup de calcul et de données d'exemple pour l'apprentissage ainsi qu'un paramétrage parfois complexe. Ils bénéficient d'un important regain d'intérêt depuis quelques années, à travers le "Deep Learning", grâce à l'augmentation des

capacités de calcul, les nombreuses données disponibles et la découverte de méthodes d'apprentissage raccourcissant le temps de convergence et des structures complexes très performantes.

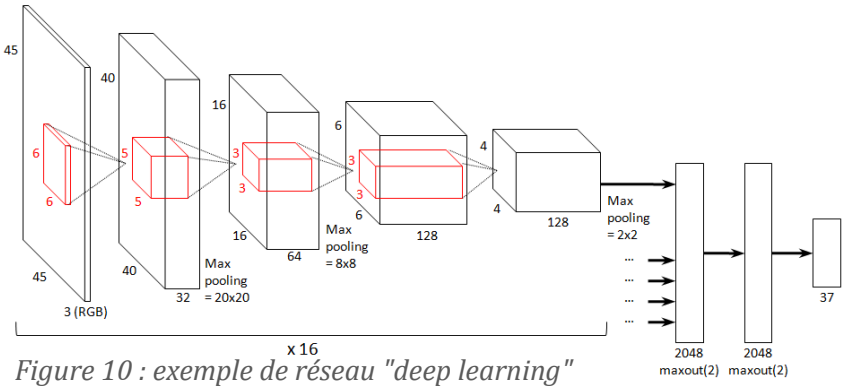


Figure 10 : exemple de réseau "deep learning"

Le "Deep Learning" utilise des réseaux de neurones plus profonds (plusieurs étages de réseau de neurones), empile et connecte différents réseaux de neurones spécialisés (ex. segmentation d'une image et découverte de caractéristiques) et améliore les prédictions par la découverte de concepts intermédiaires pour l'apprentissage final. Il automatise ainsi une partie d'une des tâches les plus complexes dans un projet de machine learning : la sélection et l'enrichissement des données d'entrée permettant à l'algorithme d'extraire différentes informations et concepts clés pour améliorer la performance d'apprentissage. Le temps d'apprentissage et de convergence pour cette famille d'algorithme reste très élevé en comparaison des autres familles d'algorithmes.

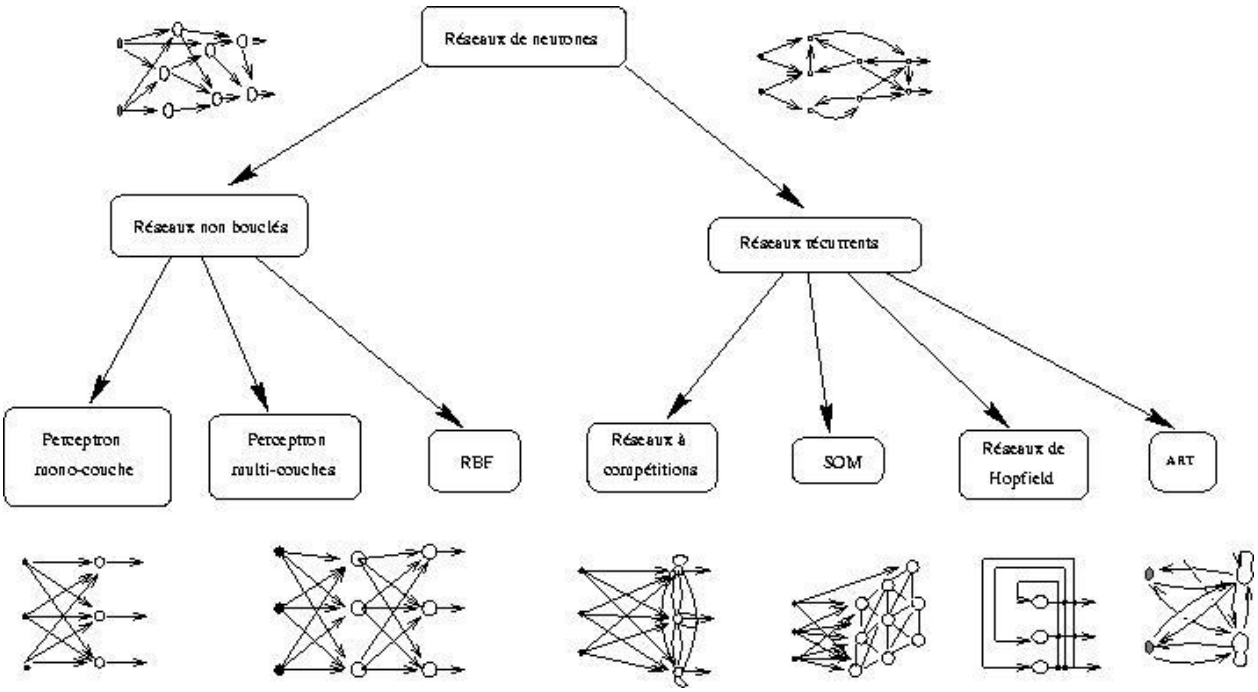


Figure 11 : les principaux types de réseaux de neurone

d. Machine à vecteurs de support (SVM) : noyaux et marges maximales

Les machines à vecteurs de support sont une généralisation des classifieurs linéaires. Elles rendent notamment possible la classification de données non linéairement séparables en utilisant un "noyau" (fonction de transformation) et de classifier sur N dimensions (les hyperplans). Elles maximisent la marge, ou distance moyenne, entre les données et les

frontières de séparation de telle sorte qu'elles trouvent les hyperplans optimaux séparant ces données pour une bonne généralisation. Les SVM supportent des données de grande

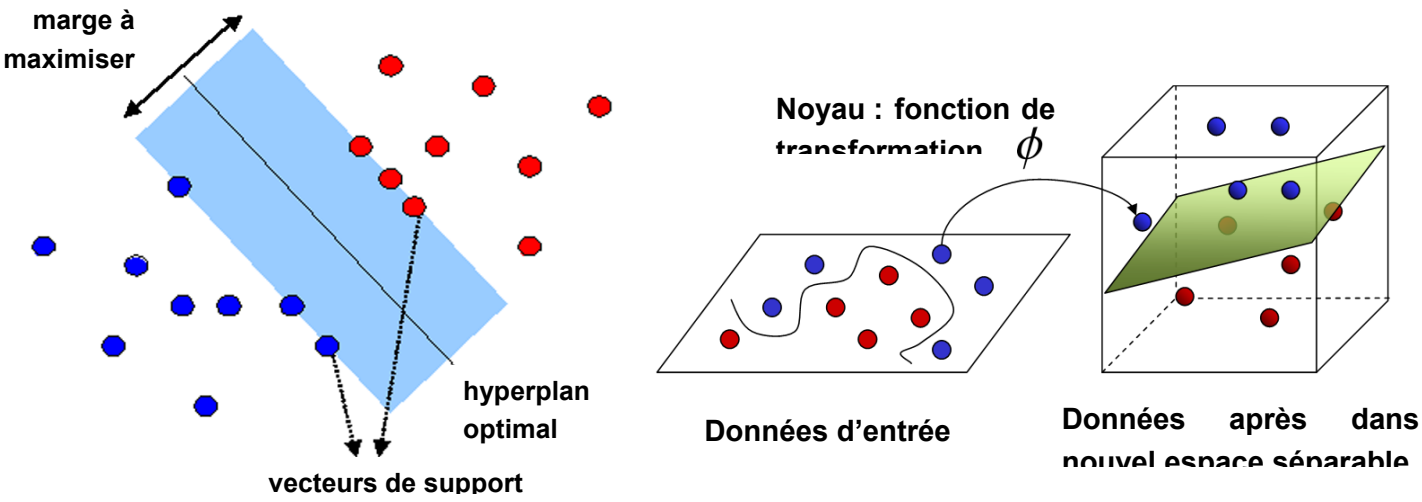


Figure 12 : illustration des concepts des algorithmes SVM

dimension en entrée (nombreuses variables), nécessitent peu de paramètres, leurs garanties théoriques et leurs résultats pratiques sont bons. C'est donc une famille d'algorithme performante et polyvalente dont la vitesse d'apprentissage est le seul bémol mais qui reste plus rapide que les réseaux de neurone.

e. Méthodes probabilistes et graphes

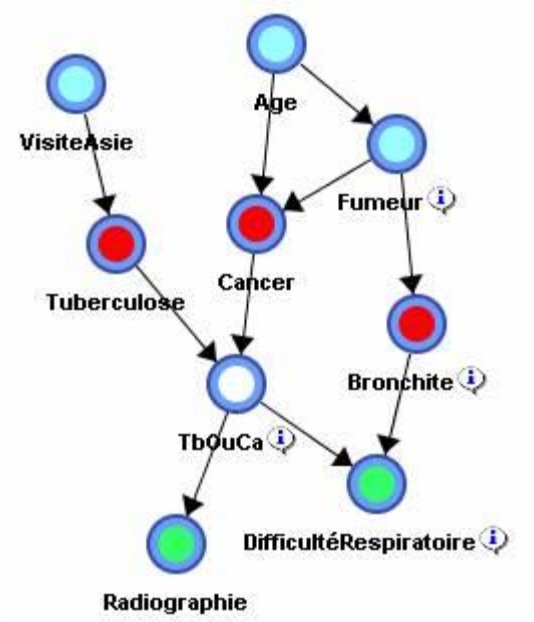


Figure 13 : exemple de réseau Bayésien

Basées sur les probabilités et hypothèses de distribution des données d'entrée et de sortie, pouvant aussi intégrer les liens entre données, la prédiction par l'utilisation de modèles probabilistes exploitent majoritairement les règles de Bayes et les différentes analyses de distribution. C'est un pilier du machine learning. Cela va du modèle très simple "bayésien naïf" sur les probabilité issues des statistiques des valeurs d'entrée par rapport à la donnée de sortie jusqu'aux différentes modèles dérivés des réseaux bayésiens et des propriétés de Markov et de Gauss intégrant les graphes de probabilités, les notions de séquences et de temporalité. C'est une famille trop vaste pour donner un jugement globale : à part les algorithmes "bayésiens naïfs" qui sont simples à mettre en œuvre et efficaces s'il y a des dépendances de distribution, beaucoup nécessitent une bonne compréhension pour les exploiter efficacement mais sont la base des systèmes les plus aboutis en intelligence artificielle et en recherche d'information (ex. algorithme PageRank de Google).

f. Par analogie (ex. kNN)

Ces algorithmes utilisent un modèle de prédiction par analogies sur les exemples d'apprentissage.

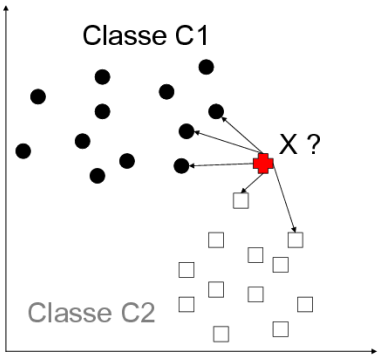


Figure 14 : k plus proche voisin

Le plus connu de cette famille est le k plus proches voisins qui stock l'ensemble des exemples d'apprentissage et au moment d'une prédiction recherche les k exemples les plus similaires pour décider par moyenne ou vote de la prédiction optimale.

g. Générateur de règles

Des algorithmes de génération de règles (DTNB, M5, Jrip, DecisionTable, NNGE, PART, OneR...) génèrent des règles optimales pour la classification ou la régression des données. Beaucoup s'appuient sur un algorithme d'arbre de décision ou naïve bayésien pour en déduire les règles. L'intérêt est de produire un modèle facilement compréhensible, portable. A noter, que OneR qui génère une seule règle basique et ZeroR qui ne retourne que la valeur la plus fréquente constatée pendant l'apprentissage sont très utiles comme référence de comparaison dans un processus de sélection de modèle : un modèle devrait avoir au minimum une performance supérieure à ZeroR et OneR.

- SI p(gène A) >= 0.32 ALORS tissu = bronches
- OU SI p(gène B) >= 0.9 AND p(gène A) < 0.1 ALORS tissu = cerveau
- OU SI p(gène C) >= 0.15 ALORS tissue = rein

Figure 15 : exemples de règles simples générées par JRip

h. Agrégation de modèles ou méthodes d'ensemble - "super modèles"

Combiner des modèles parfois individuellement moyennement performants autorise la construction de "super modèles" plus performants et plus stables. On peut voir cela comme la création d'un comité d'experts avec différentes stratégies pour la prise de décision collégiale. Les méthodes d'ensemble les plus connues sont :

Bagging (ou agrégation boostrap) : consiste à ré-échantillonner au hasard avec doublons les exemples d'apprentissage et faire générer à l'algorithme voulu un modèle pour chaque sous-échantillon. On obtient ainsi un ensemble de modèles dont les différentes prédictions doivent être moyennées si c'est une régression ou choisies par vote si c'est une classification.

- Avantages : performant, rapide et facilement parallélisable.
- Inconvénients : perte de compréhension du modèle créé (commun à la majorité des méthodes ensemblistes)

Boosting : plusieurs modèles apprennent en parallèle et en séquence pour chaque exemple. L'apprentissage d'un modèle est boosté, en appliquant un poids plus fort sur l'exemple à apprendre, à chaque fois que le précédent modèle a prédit avec erreur après

apprentissage. Cela créé ainsi un ensemble de modèles chacun spécialisés sur certaines erreurs.

- Avantages : garantie théorique d'amélioration, généralement plus performant que bagging si pas de bruit. Une version améliorée de l'algorithme appelée "Gradient Boosting Machine" (GBM ou GBRT) est un des algorithmes les plus performants et polyvalents actuellement.
- Inconvénients : sensible au bruit, algorithme séquentiel, possible sur-apprentissage (non prouvé).

Stacking : consiste à appliquer un algorithme de machine learning à des modèles générés par un autre algorithme de machine learning. On va donc prédire quels seront les meilleurs modèles à utiliser suivant les données en entrée.

- Avantages : très performant (fameux vainqueur du Netflix prize) et polyvalent par la possibilité de combiner des algorithmes très différents.
- Inconvénients : temps d'apprentissage, temps de prédiction et taille du modèle final.

Forêts aléatoires (Random Forest) : ce modèle très performant nécessitant la création d'un nombre important de modèles à combiner (ex. 500) ajoute à la technique de bagging le fait d'échantillonner au hasard aussi les variables d'entrée.

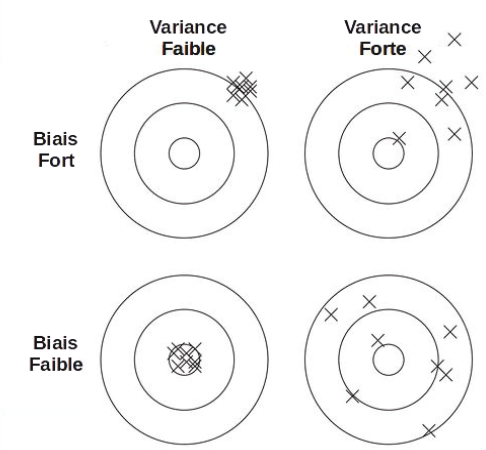
- Avantages : parmi les algorithmes les plus performants et polyvalents, forte élimination du biais et de la variance.
- Inconvénients : temps d'apprentissage et taille du modèle final.

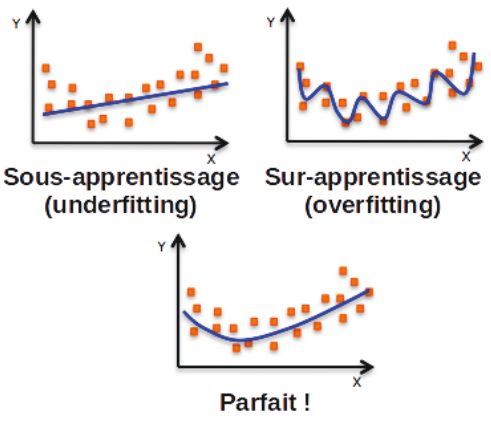
6. Choix d'algorithme

Chaque famille d'algorithme a des avantages et inconvénients liés au type de problème, à sa complexité, sa fiabilité, au temps et ressources nécessaires. De même chaque algorithme a souvent différents paramètres appelés 'hyper-paramètres" qui ont une grande influence sur la performance de l'algorithme pour les données fournies. Il faut ainsi pouvoir définir de bons outils de comparaison, comprendre les résultats pour corriger les données d'entrée ou l'algorithme, concevoir le modèle ou la combinaison des modèles finaux.

a. Biais, Variance, Généralisation

Un enjeu majeur d'un bon processus d'apprentissage est sa capacité à bien se généraliser et non à produire des règles qui ne seraient valables que pour les exemples





d'apprentissage donnés. Pour cela, il faut trouver un bon compromis entre biais et variance :

- ✓ Biais : le biais est la différence entre l'espérance des données réelles et celle qui ont été prédites. Si il est de 0, le modèle est sans biais. Les moyennes correspondent entre prédiction et réalité mais les données ont peut-être systématiquement un écart important et se retrouvent finalement que sur la moyenne, c'est typique du sous-apprentissage (underfitting). Il faut donc aussi mesurer la variance.
- ✓ Variance : la variance mesure la dispersion entre les données réelles et celle qui ont été prédites, c'est la moyenne de l'écart au carré. A vouloir trop réduire la variance, on peut arriver au surapprentissage.
- ✓ Généralisation vs surapprentissage : l'apprentissage sur des exemples à pour but de pouvoir prédire sur d'autres données, c'est la généralisation. Un modèle qui a trop appris sur les données d'exemple sans dégager les tendances générales créé un modèle proche des données d'exemple mais sans capacité de généralisation.

b. Fonction de coût, descente de gradient et régularisation

Les algorithmes d'apprentissage optimisent leur modèle en minimisant l'erreur qui est mesurée classiquement par une fonction de coût et un algorithme de minimisation comme la descente de gradient.

Dans ce processus d'optimisation automatique, l'algorithme peut tendre vers un modèle non généralisable dû à un surapprentissage (overfitting).

Pour éviter cela, on utilisera une méthode de régularisation pénalisant la fonction de coût pour éviter les valeurs extrêmes afin de favoriser une fonction plus générale.

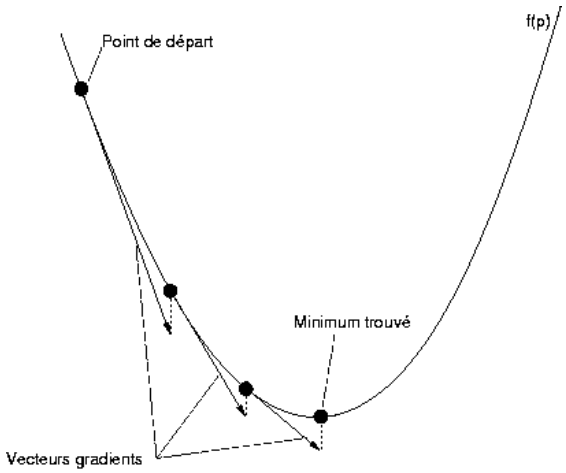


Figure 16 : illustration du processus de descente de gradient pour la recherche du minimum local d'une fonction

c. Critères de comparaison

Pour comparer la performance des algorithmes il faut choisir un critère correspondant à l'objectif souhaité.

Imaginons que nous souhaitions évaluer la performance d'une prédiction d'un évènement qui à 2% de chance de se produire. En prédisant systématiquement "faux" l'évènement n'arrivera pas, on devrait tendre vers un beau taux de succès de 98% mais quelle utilité ? Dans ce cas il serait plus intéressant d'avoir un autre critère d'évaluation comme : "le rappel

de la classe faux'", le coefficient de confiance Kappa, l'aire AUC en-dessous de la courbe ROC.

i. Classification

Pour les classements binaires de type vrai / faux on différencie les :

- ✓ VP (TP) : vrai positif (true positive), c'est vrai on ne s'est pas trompé.
- ✓ FP : faux positif (false positive), c'est une fausse alarme !

	Malade	**Non malade**
Test +	Vrai Positif (VP / TP)	Faux Positif (FP)
Test -	Faux Négatif (FN)	Vrai Négatif (VN / TN)

- ✓ VN (TN) : vrai négatif (true negative), c'est faux on ne s'est pas trompé.
- ✓ FN : faux négatif (false negative), c'est pas faux !

Une classe est une des possibilités de classification pour une prédiction. Pour une classification binaire, les classes sont "vrai" et "faux", mais cela pourrait être un classement multiple comme : "nul", "moyen", "bien", "super".

Les mesures élémentaires les plus utilisées sont la précision, la sensibilité et la spécificité que j'ai illustrées ci-dessous dans le cas d'un test de maladie :

Nom de la mesure	**Formule**	**Interprétation**
Précision (Precision)	TP / (TP + FP)	Probabilité qu'une personne soit malade si le test est positif. Un test peut tricher et obtenir 100% en retournant juste une fois positif en choisissant la prédiction la plus facile.
Rappel (Recall) / Sensibilité / Taux de vrais positifs	TP / (TP + FN)	Probabilité que le test soit positif chez les malades. Un test peut tricher et obtenir 100% en retournant toujours «positif».
Spécificité / Taux de vrais négatifs	TN / (FP + TN)	Probabilité que le test soit négatif chez les non malades. Un test peut tricher et obtenir 100% en retournant toujours «négatif».

On remarque que l'on doit faire une combinaison de critères d'évaluation pour qu'une évaluation soit représentative de la performance de prédiction sur les différents cas possibles. Il existe donc des mesures plus équilibrées sur les différents cas possibles comme :

- La courbe ROC (Receiver Operating Characteristic) : est une courbe de mesure de la performance d'un classifieur binaire facilitant la visualisation de son spectre d'action en affichant les taux de vrais positifs en fonction du taux de faux positifs.

- AUC : c'est l'aire totale en-dessous de la courbe ROC, plus elle est élevée, plus le classifieur est globalement performant.
- Le F1-Score : score global de performance d'un classifieur intégrant la précision et le rappel.
- Le coefficient de Kappa : score d'accord réciproque de classement entre différents observateurs. Ce score est un bon indicateur de la qualité globale sur les différentes possibilités de classement.

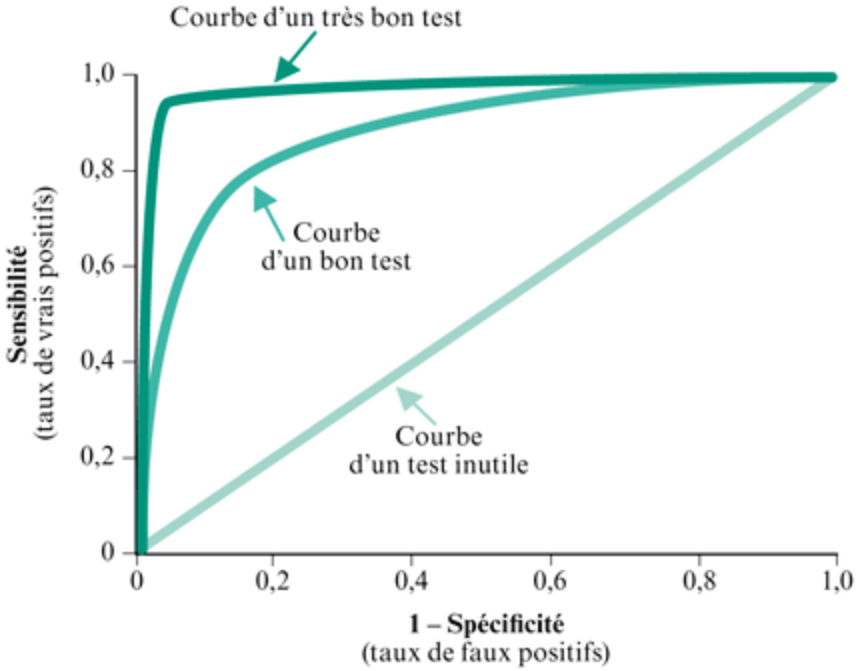


Figure 17: exemple de courbe ROC

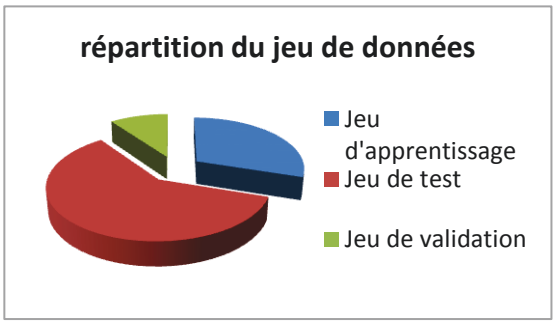
ii. Régression

Les mesures les plus utilisées en régression sont :

- RMSE (root mean squared error) : l'erreur quadratique moyenne est la mesure la plus utilisée pour des problèmes de régression.
- MAE (mean absolute error) : la moyenne de l'erreur.

d. Méthodes de test : jeux de données, validation croisée, backstesting

i. Jeux de données



Les données d'exemple sont généralement réparties en un jeu d'apprentissage (training set), un jeu de test (test set). Il est intéressant de garder un jeu final de validation neutre (validation set) pour contrôler que l'apprentissage n'a pas été baisé pendant le processus de recherche tendant à améliorer le résultat du test.

ii. Validation croisée (Cross-Validation)

Une technique, appelée validation croisée (CV, cross-validation), découpe automatiquement et test successivement différents jeux d'apprentissage et jeux de test. C'est un bon moyen pour valider la capacité de généralisation d'un algorithme d'apprentissage. On divise le jeu de données en k échantillons, un des échantillons est utilisé pour l'apprentissage, le reste pour le test. On répète l'opération avec un autre échantillon jusqu'à avoir testé l'apprentissage avec chacun des échantillons. Les scores finaux sont les moyennes des scores de chacun des k tests (voir illustration ci-dessous).

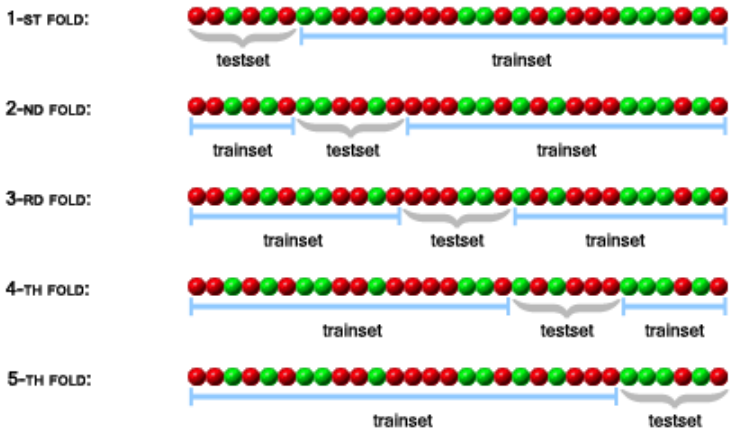


Figure 18 : exemple de k-cross fold validation (validation croisée) avec k=5. Le jeu de données est séparé également 5 fois différemment, sans réutiliser les mêmes données d'apprentissage, et testé à chaque fois. Le résultat moyen des 5 tests est le résultat final

iii. Données temporelles et backtesting

Pour des données temporelles, pour ne pas biaiser l'évaluation, il est important de faire attention à ce que le jeu de test soit composé uniquement d'exemples postérieurs aux exemples du jeu d'apprentissage. La validation croisée n'est donc pas possible dans ce cas.

Une technique très utilisée en trading algorithmique est de simuler le processus dans l'ordre historique pour voir l'évolution d'un algorithme ou du système complet, c'est le backtesting.

e. Recherche des hyper-paramètres optimaux : grid, random, bayesian optimization,

Chaque algorithme à des paramètres appelés hyper-paramètres qu'il faut adapter à chaque nouveau jeu de données. Avec un peu d'expérience, on peut s'orienter plus rapidement dans la définition de bons hyper-paramètres initiaux mais il y a souvent de nombreuses combinaisons à tester. Heureusement, il existe des algorithmes pouvant tester automatiquement ces combinaisons.

- Gridsearch testera itérativement, avec des intervalles définis, toutes les combinaisons des différents hyper-paramètres.
- Randomsearch effectue ces mêmes tests mais de manière aléatoire, cette solution est généralement plus rapide pour trouver des combinaisons performantes sans pour autant être optimale.
- L'optimisation bayésienne commence par une approche aléatoire puis s'oriente vers les combinaisons dont les probabilités sont les plus élevées pour converger vers la solution optimale.

C. Architecture « Machine learning » et « Big data »

Les algorithmes de machine learning peuvent être exploités dans tout type d'architecture allant des systèmes embarqués, pour des robots par exemple, aux architectures de données en cluster utilisés par des entreprises comme Amazon, Deezer, Netflix pour, par exemple, faire de la recommandation de produit. Dans le cadre de ce projet, nous nous intéressons aux architectures de données, qui sont aujourd'hui au cœur du phénomène « big data »qui est le traitement rapide de volumes massifs et diverses de données pour offrir des services nouveaux par l'utilisation du machine learning. Je présente donc ci-dessous les composants et architectures les plus classiques du big data.

1. Stockage et distribution des données massives

Dès que les données ne peuvent pas être gardées en mémoire et transmises via le réseau sans stockage, les données exploitées vont devoir utiliser un média pour être stockée même très temporairement. Il faut ainsi opter pour un ou plusieurs médias de stockage qui dans la plus part des cas sont soit un système de fichier distribué qui peut être partagé sur plusieurs machine (ex. HDFS ou SAN), soit une base de donnée rapide (ex. MongoDB, Hbase, Cassandra, Redis). A noter que si les données peuvent être découpées en nombreux courts messages pour travailler rapidement au fil de l'eau, on pourra utiliser un système de file de message (ex. ZeroMQ, RabbitMQ).

2. Sources de données : collecte, agrégation et transformation

Les algorithmes de machine learning ont besoin de données d'entraînement ainsi que des données à fournir pour la prédiction ou autre tâche de l'algorithme. L'architecture doit donc intégrer les mécanismes de récupération des différentes données brutes, de transformation, avec probable parallélisation de ces tâches si elles sont trop gourmandes en ressources, et de mise à disposition de ces données enrichies pour l'algorithme de machine learning.

a. Collecte des données

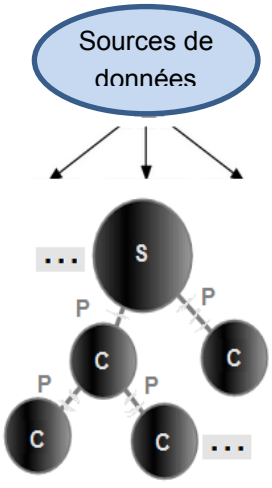
Les sources de données brutes peuvent être très diverses : bases de données d'entreprise, logs des serveurs, données de capteurs, évènement de suivi (tracking) des utilisateurs sur les sites web, services web tiers (ex. météo, données métier…), les « web scrapers » parcourant internet à la recherche de données cible... Ces données sont récupérées via des scripts ou services divers et mis à disposition de l'architecture sous forme de fichier, files d'attentes de message, base de données rapide, requête sur un service dédié. Quelques outils open-source dédiés à la centralisation ces collectes de données sont : Flume, Kafka, Logstash, Fluentd. Ces outils s'intègrent facilement avec du code dédié et de nombreux services et formats existant pour récupérer les données à la source.

b. Agrégation, Transformation – « ETL » vs « MapReduce » vs « CEP » vs « flux distribués »

Les données extraites de différentes sources et mises à disposition sont alors agrégées et transformées (voir II.B.4 – ex. agrégation, enrichissement) pour être mises à disposition de l'algorithme d'apprentissage sous la forme souhaitée (fichier, file d'attente de message…). Cette transformation peut se faire en temps réel, avec une solution CEP ou de flux distribués au fil de l'eau (voir plus bas), ou de manière plus classique en traitement programmé de données par lot via un ETL ou une architecture MapReduce (voir chapitre suivant). Quelques outils ETL et CEP open-source connus sont : Talend (ETL), Esper (CEP), StreamingSQL (CEP). Les outils Logstash et autres systèmes de collecte vus plus haut peuvent réaliser des transformations mais restent très limités.

Type de solution	Traitement par lots ou au fil de l'eau	Traitement parallélisé
ETL	Lots	Non (mais possible)
MapReduce	Lots	Oui
CEP	Fil de l'eau	Non (mais possible)
Flux distribués	Fil de l'eau	Oui

Flux de données distribué au fil de l'eau : ces tâches d'agrégation et de transformation pouvant être massives, elles sont aujourd'hui couramment parallélisées sous forme d'architecture distribuée. Ce travail d'agrégation et transformation est en quelque sorte découpé en petites tâches simples distribués sur de nombreux ordinateurs organisés en grappe (cluster). Dès que des données arrivent, ces ordinateurs réalisent collectivement les traitements nécessaires d'agrégation et de transformation, et envoient au fur et à mesure les données produites vers l'algorithme d'apprentissage. Les outils open-source les plus connus sont : Storm, Samza, Spark.



	Storm	Spark	Samza
(P) Type de données « primitives » utilisées	Tuple	« DStream »	Message
(S) Traitement des données source	« Spouts »	« Receivers »	« Consumers »
(C) Traitement des calculs et transformations	« Bolts »	« Transform operations » et « Window operations »	« Tasks » (tâches)

Figure 19 : comparaison des points communs entre les architectures open-source de traitement distribué de flux

3. Architecture classique - Hadoop avec MapReduce

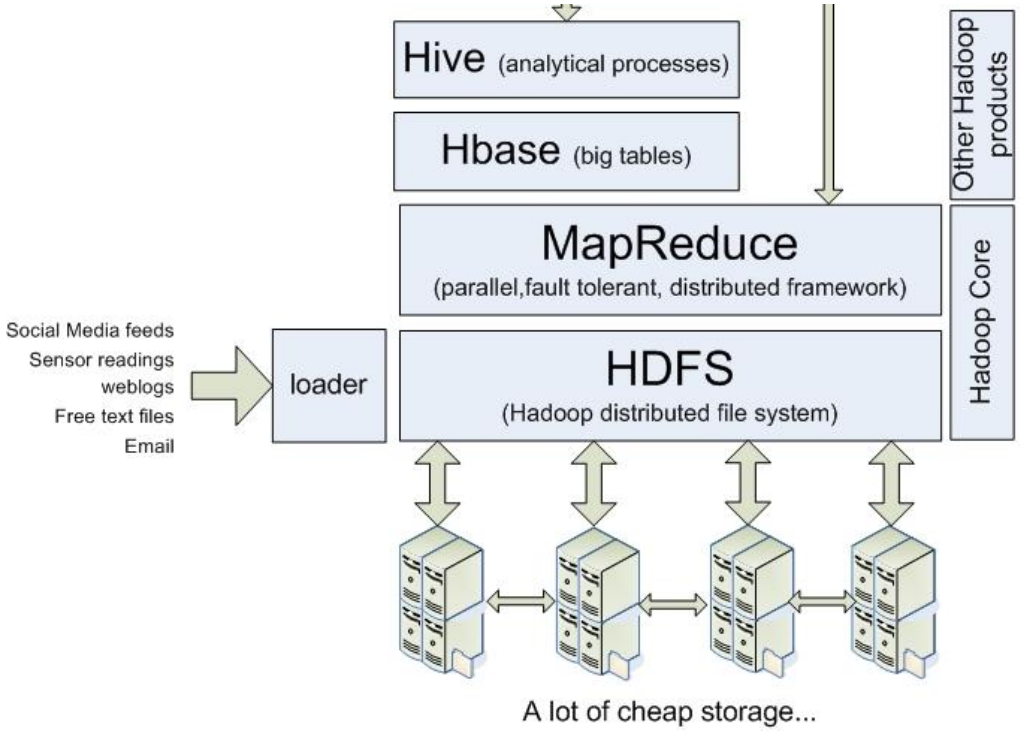


Figure 20 : exemple architecture Hadoop / MapReduce

L'architecture de loin la plus utilisée pour traiter des données en masse (« big data ») est le framework Hadoop. Elle est dédiée au traitement massif de données par la parallélisation des traitements en utilisant le modèle « MapReduce ». Le traitement de données est fait sur des architectures en grappe (cluster) à moindre coût. En effet, Hadoop a été développé sur la base du travail de l'entreprise Google qui pour réduire ses coûts d'infrastructure avait initié le concept de traitement des données en lots distribués sur de multiples ordinateurs à faible coût au lieu d'utiliser des serveurs puissants très couteux. Le framework Hadoop est classiquement composé de :

- ✓ Un système de fichier distribué HDFS.
- ✓ YARN est dédié à la gestion des ressources en cluster, c'est la couche permettant la distribution des traitements avec l'algorithme « MapReduce ».
- ✓ HBase est une base de données distribuée s'appuyant sur le système de fichier HDFS.
- ✓ Différents composants s'appuient sur YARN : Hive pour les requêtes SQL, Mahoot pour le machine learning, Pig pour la programmation en script des traitements…
- ✓ Zookeeper coordonne les traitements.
- ✓ De nombreuses distributions Hadoop (ex. Cloudera CDH, Hortonworks, MapR) offrent des modules facilitant la gestion de l'architecture ou offre des connecteurs de données.

4. Architecture orienté flux et Architecture « lambda »

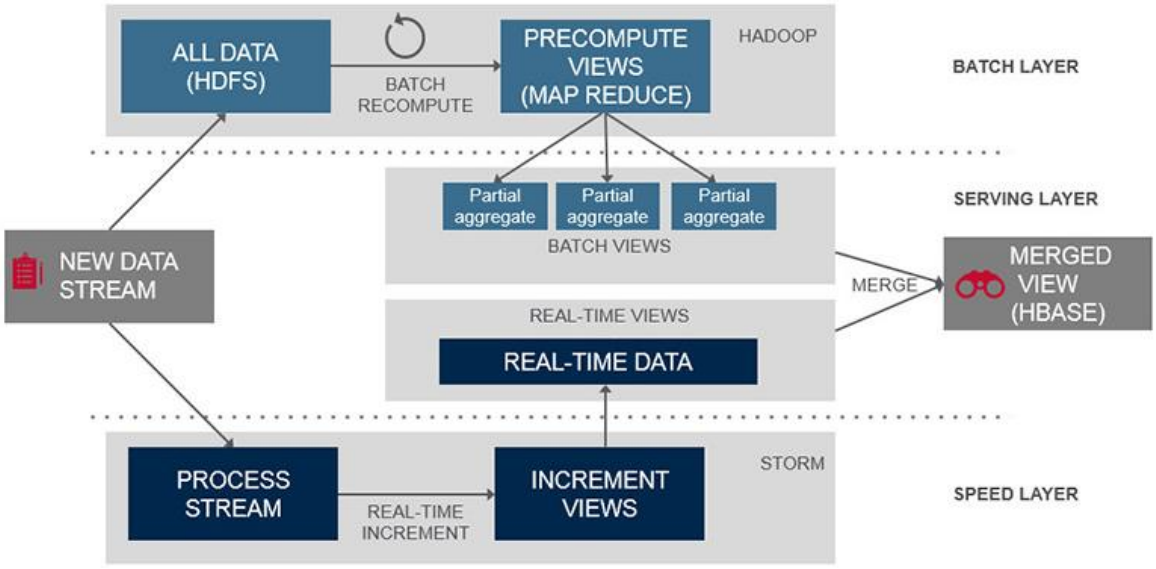


Figure 21 : exemple d'architecture lambda

Le framework Hadoop avec MapReduce n'est pas adapté aux traitements rapides au fil de l'eau car tous les traitements sont réalisés par lot avec une latence importante. Hors de nombreuses applications nécessitent du traitement de données voir de l'apprentissage au fil de l'eau avec une mise à jour constante du modèle tout en offrant des réponses quasi temps-réel. Cela rend possible l'intégration constante des données les plus récentes. Une architecture hybride bénéficiant de la puissance d'Hadoop avec MapReduce pour les traitements longs (ex. apprentissage sur de gros volumes de données historiques) et du traitement en flux distribué au fil de l'eau (ex. mise à jour sur flux de données instantanées) a été développée, c'est l'architecture Lambda. Elle rend possible d'un côté la création des modèles entraînés sur des données massives avec des traitements lourds sur des lots et, de l'autre, agrège et transforme en quasi temps-réel et met constamment à jour certains models.

Il est intéressant de noter la progression très rapide du projet open-source Apache Spark qui intègre la majorité des composants pour réaliser ces deux types de traitement et rend possible ma réalisation de cette architecture dans un environnement unifié Hadoop.

D. Cycle de vie d'un projet de « Machine learning »

Un projet de machine learning peut se décomposer en plusieurs phases :

Figure 22 : chronologie du cycle de vie d'un projet machine learning

1. Comprendre le métier et bien poser le problème

Comme tout projet, le premier objectif du responsable de projet est de bien comprendre ce que le client veut atteindre : comment et avec quel retour sur investissement attendu. Le client peut avoir de nombreux objectifs et contraintes concurrentes qui doivent être bien équilibrés. L'objectif est de découvrir des facteurs importants au début du projet qui peuvent influencer le résultat final pour ne pas produire les bonnes réponses aux mauvaises questions.

Il est important d'expliquer au plus tôt aux acteurs du projet les capacités et limites du machine learning. Cela aidera à réfléchir ensemble à comment décomposer et simplifier au maximum la complexité du problème. Le but est d'avoir un problème bien posé, atteignable, et dans la phase suivante, des données de qualité pour résoudre ce problème.

Cette phase initiale, similaire en de nombreux points à un projet traditionnel, peut s'organiser comme suit :

1. Comprendre le problème et déterminer les critères pertinents pour mesurer le succès du projet
2. Identifier le type de problème à résoudre et essayer de le simplifier avec le client puis par décomposition ou réduction (transformer en un problème équivalant).
3. Identifier les personnes clés du projet
4. Obtenir les spécifications, les besoins, les priorités et identifier les risques
5. Quelle est la précision souhaitée de la solution ? Avons-nous besoin de toutes les données ?
6. Devons nous le réaliser en interne, utiliser une solution externe, sous-traiter ?
7. Comparer et benchmarker les options pour choisir

2. Identifier les données

Un algorithme basique utilisant des données de qualité sera généralement beaucoup plus performant qu'un algorithme généralement très performant mais avec des données moins pertinentes ou de moins bonne qualité. Cette phase peut s'organiser comme suit :

1. Obtenir toutes les données disponibles et vérifier les données de l'échantillon (qualité, compréhension)
2. Explorer les données, les analyser statistiquement, les tester et en faire un dictionnaire
3. Évaluer la qualité des données sur le jeu complet, les valeurs disponibles et les erreurs, les champs partiellement redondants (ex. composition d'un ou plusieurs)
4. Évaluer le besoin pour ces données :
    a. Quel champ devons-nous capturer ?
    b. Combien de données historiques avons-nous besoin ? Avec quelle granularité ?
    c. Avons-nous besoin de données temps réel ? A une certaine fréquence ?
    d. Comment stocker ces données ?
    e. Avons-nous besoin d'un prototype ?

    f. Ces données seront-elle toujours disponibles ?
5. Un travail complémentaire pour améliorer ces données ou en obtenir des sources complémentaires est-il nécessaire ?
6. Quels outils ou langage faudra-t-il pour exploiter ces données (logiciel, batch ou flux, API...)

3. Préparer les données

Cette phase de préparation des données intervient une première fois avant la phase de modélisation mais pourra être affinée plusieurs fois pendant la modélisation. Elle sera ensuite automatisée. La préparation des données est estimée par de nombreux experts comme étant **souvent la phase le plus longue et la plus critique d'un projet de machine learning**. Cette phase peut s'organiser comme suit :

1. Sélectionner les données retenues : toute donnée non pertinente ou redondante doit être éliminée pour éviter d'augmenter la complexité du problème et le temps de calcul inutilement.
2. Nettoyer les données : il y a par exemple couramment des données erronées ou des cas d'exception inutiles comme exemple à utiliser pour l'apprentissage.
3. Enrichir les données : des données mieux construites sur des données existantes permettront souvent de grandement améliorer la performance du ML. Cela peut se faire avec des données induites (ex. extraire les jours de la semaine de la date), données calculées (ex. surface = longueur x largeur), données temporelles (ex. tendances ou moyennes ou état sur différentes périodes de temps).
4. Intégrer les données des différentes sources : la plus part des algorithmes ML apprennent avec des exemples qui doivent contenir toutes les données agrégées à chaque exemple sous la forme suivante : "toutes les données attendues + résultat correspondant"
5. Standardisation des données (moyenne, plage, catégorisation, imputation, normalisation, binarisation) : différentes options de standardisation des données peuvent être requises pour améliorer la performance de l'algorithme ou pour être compatible. Par exemple, beaucoup d'algorithmes ML sont plus performants si la moyenne des données est centrée sur zéro ou que les données sont comprises entre 0 et 1. Certaines données peuvent être manquantes dans de nombreux exemples, il faut prévoir une stratégie adaptée (ex. imputation). Chaque algorithme accepte aussi un certains nombre de format, il est donc souvent nécessaire de reformater les données au préalable.
6. Créer différents jeux de données séparés pour : l'apprentissage, le test, la validation. Il est courant d'utiliser le système de validation croisée pour les jeux d'apprentissage et de test (voir chapitre suivant) mais il existe aussi d'autres stratégies.
7. (Automatiser)

4. Sélectionner le(s) modèle(s)

Il existe de nombreuses familles d'algorithmes, variantes d'algorithmes et chacun peut exiger de nombreux paramètres. Il faudra donc :

1. présélectionner un jeu d'algorithmes à tester correspondant au type de problème et données disponibles.
2. définir des métriques de comparaison pertinents suivant l'objectif.
3. concevoir une méthode d'apprentissage et de test avec le jeu de données, avec éventuelle validation.
4. sur la base des métriques et tests définis, lancer un processus de comparaison et sélection qui peut être manuel ou automatique. Il est courant de commencer par des tests manuels avec différentes visualisation graphiques pour comprendre l'évolution du modèle puis de lancer un processus automatique de recherche d'algorithmes et de paramètres optimaux.
5. analyser et éventuellement créer des modèles plus avancés comme la combinaison de modèles ou des scénarios de modèles suivant les données disponibles (ex. les systèmes de recommandation envisagent des modèles différents au début quand il y a peut de données (cold start) et plusieurs modèles suivant les données disponibles).

5. Architecture, prototype, implémentation

Le modèle choisi comme les données requises seront gérés à travers une architecture et un service à implémenter après avoir validé un ou plusieurs prototypes. L'analyse amont devrait, entre autre, répondre aux questions suivantes :

- comment concevoir une architecture simple, évolutive, robuste, réutilisable ?
- puis-je opter pour une architecture classique en lot (batch) ? ou une architecture en flux (stream) est nécessaire ?
- à quelle fréquence dois-je fournir ses données et mettre à jour les modèles ?
- quelle API ou système dois-je concevoir pour communiquer avec les applications clientes ?
- comment vais-je contrôler l'évolution de la performance de mes modèles, de mon service ?

6. Déploiement et Maintenance

Cette phase peut s'organiser comme suit :

1. Validation du modèle en pré-production et de l'implémentation avec différents tests de charge.
2. Mise en production.
3. Documentation finale, formation et transfert de compétence
4. Cycle récurrent :
    a. Contrôler régulièrement l'évolution de la performance des modèles avec des alertes automatiques et des visualisations graphiques (approche la plus simple pour comprendre un modèle et sa performance)
    b. Mettre à jour.
    c. Vérifier l'opportunité de migration vers une nouvelle plate-forme

IV. Solutions et Réalisation

A. Thématiques du programme R&D

Pour répondre aux différentes problématiques du projet R&D « Irma », j'ai ainsi peu à peu développé mon programme R&D autour des thématiques ci-dessous :

1. « Opportunités du machine learning dans le secteur de réservation d'hôtel à la dernière minute »

Afin de découvrir et valider les premières opportunités à développer, j'ai lancé :

- ✓ une enquête et un travail collaboratif internes sur le sujet
- ✓ la collecte et l'analyse d'études de cas dans le même domaine ou des domaines similaires (ex. yield management pour le transport aérien)
- ✓ la création et l'étude des résultats de mini prototypes, m'ont permis de mieux cerner ce qui était réalisable et pertinent.

2. « Données et sources utiles à la prédiction et la recommandation dans le secteur hôtelier, e-commerce et secteurs associés »

Afin de découvrir, améliorer et créer les sources de données utiles aux différentes prédictions, j'ai recherché dans différents domaines et applications similaires les données pertinentes et des techniques d'enrichissement utilisées.

3. Spécificités des prédictions et recommandations recherchées

J'ai lancé des mini projets de recherche sur ces thèmes associés aux objectifs de prédiction et de recommandation du projet.

- ✓ Prédictions de prix et prédictions associées
- ✓ Optimisation dynamique d'une architecture pour son résultat : évaluation, priorisation, sélection
- ✓ Prédictions de la disponibilité de chambre et risque d'overbooking
- ✓ Recommandations d'offre et marketing par anticipation
- ✓ Recommandations de tarification, configuration

4. Formation « machine learning, flux et architecture »

Un travail constant a été la formation et la recherche sur les architectures de machine learning notamment sur :

- ✓ La gestion de flux de données : sourcing, agrégation, enrichissement, sélection... dans les contraintes limitées de temps, mémoire, précision.
- ✓ L'évaluation et la sélection de modèles : les critères d'évaluation, les possibilités d'automatisation, la simulation par le backtesting.
- ✓ Les techniques de prédictions
- ✓ Les techniques de recommandations, la gestion des différentes phases en fonction des données disponibles, la recommandation en contexte.
- ✓ Le scaling et les possibilités de standardisation de l'architecture.

Les travaux réalisés sur ces thématiques de recherche m'ont ainsi mené aux solutions finales présentées ci-dessous.

B. Solutions retenues

Rappelons que l'objectif du projet est de développer de nouvelles opportunités pour l'entreprise en s'appuyant sur le machine learning. Les prédictions retenues ont déjà été présentées dans la partie I de ce document mais je préciserai comment elles ont été mises en place pour mieux comprendre les solutions d'architecture retenues.

1. Une architecture orientée service

Il n'existait pas de solution intégrée répondants aux différentes contraintes du projet notamment à l'exigence de rapidité de traitement malgré la limitation en termes de ressources matériels (un seul serveur virtuel au départ) et à la variété des algorithmes et prototypes nécessaires sachant les besoins.

J'ai donc opté pour une architecture orientée services autorisant une forte liberté dans les composants, le développement et le prototypage des différentes briques et leur évolution via des services fortement découplés. Elle est aussi ouverte à l'ajout rapide de fonctions à l'architecture sous forme de micro-services (service dédié à une fonction).

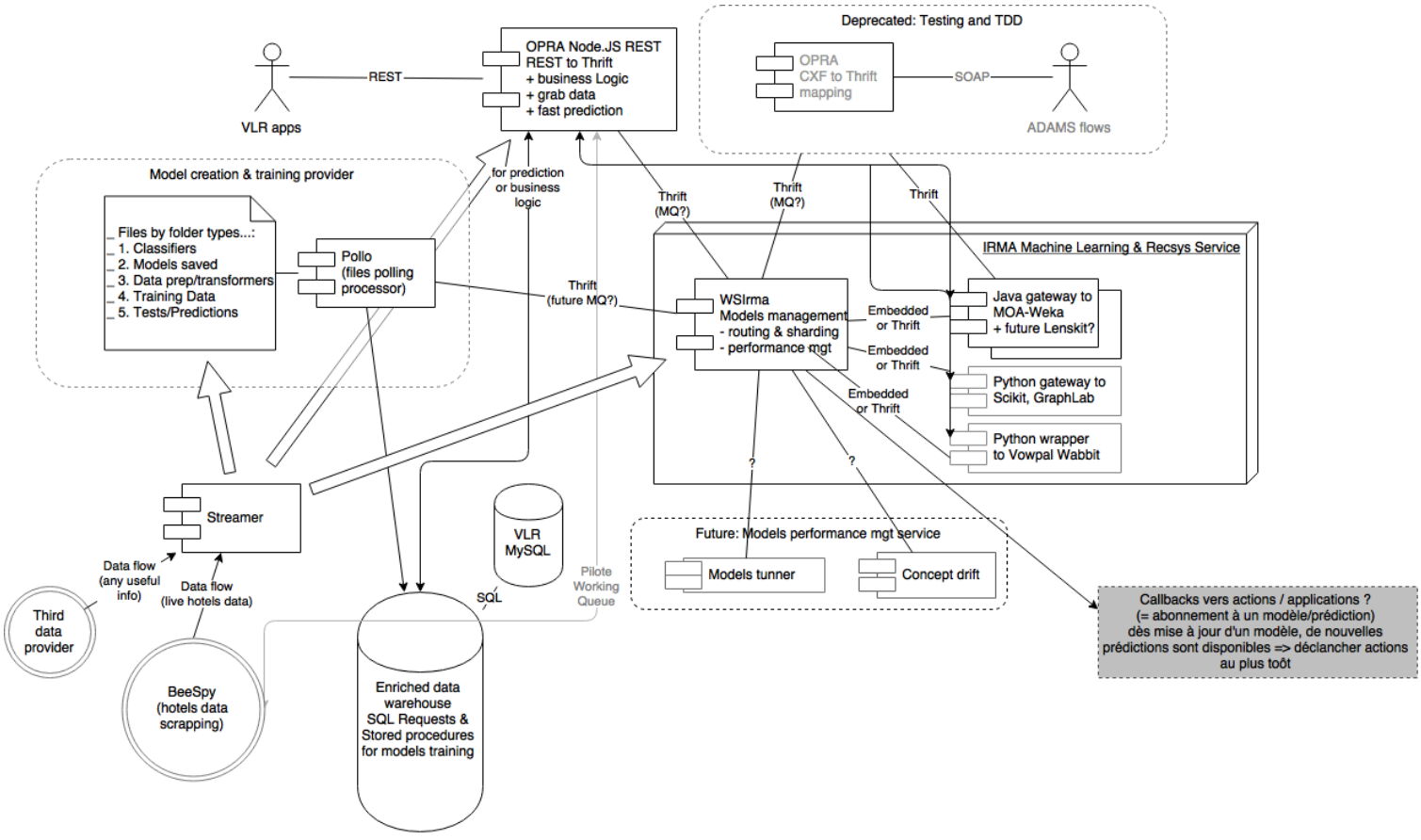


Figure 23 : architecture du projet "Irma"

Les modules applicatifs utilisent différents langages (Java, Python, C++, Javascript pour Node.js) et la couche de communication est Apache Thrift (voir détail plus bas). Cette conception peut s'affranchir du système d'exploitation mais elle a été fortement optimisée pour Linux[6].

a. WSIrma - module central, API publique, gestion intelligente des modèles en mémoire

Le service WSIrma est la pièce majeure de l'architecture qui propose les services de prédictions et recommandations en s'appuyant sur d'autres services de machine learning et de fournitures de données ainsi qu'une gestion intelligente des modèles.

Les services proposés via son API sont : prédire, recommander, optimiser, entraîner un modèle, mettre à jour un modèle, chercher le meilleur modèle, charger un modèle, sauvegarder un modèle, simuler une stratégie dans le temps (backtesting).

WSIrma expose des méthodes de prédictions et de recommandations de manière unifiée provenant de multiples librairies et services dans le but de combiner le meilleur de chaque librairie de manière simple.

WSIrma expose aussi une méthode de sélection automatique de modèle optimal suivant des critères : algorithmes et librairies autorisées, temps, mémoire.

WSIrma supporte différents format de données (CSV/TSV, JSON, ARFF, VW) et convertit automatiquement, si nécessaire, les données au format de l'engin de prédiction ou de recommandation utilisé.

Elle propose aussi quelques méthodes métiers grâce à la possibilité de charger des modules externes incluant du code Python exposant une logique métier. Exemple : top 20 des prix à contrôler par les robots de collecte car les plus susceptibles d'avoir une variation de prix.

Un point fort de WSIrma est sa gestion optimale des nombreux modèles à mettre en mémoire pour faire de la prédiction ou de l'apprentissage. Les modules métiers qui sont embarqués dans WSIrma utilisent cette gestion. Dans le cadre de ce projet, nous pouvons nous retrouver à gérer plus de 5000 modèles prenant en moyenne 100Mb en mémoire, soit 500Gb à stocker en mémoire si nous les chargions tous en même temps. WSIrma peut faire du stockage en mémoire sous forme de cluster mais n'en a majoritairement pas besoin car il apprend et anticipe les besoins de chargement et de déchargement de modèles pour tenir dans une limite fixée (typiquement 10 GB).

[6] Optimisation Linux : utilisation du système de notification lié au système de fichier, compilation optimisée pour le calcul et la parallélisation.

b. Couche de communication Thrift multi-langages et les extensions REST, SOAP, CXF, CSV, JSON

La couche de communication entre les différents services est Thrift mais le service WSIrma peut être interrogé via les autres services de l'architecture en REST, SOAP, CXF ou simplement en déposant des fichiers CSV, JSON, TXT dans les répertoires du module Poller.

Cette couche de communication a été choisie après un processus de comparaison et de réalisation de mini prototypes comparatifs avec RPC, CORBA, WebService / SOAP, Thrift, Protocol Buffer. Sachant le volume de données à échanger et la fréquence la rapidité et la latence était très importante. La structuration des données et la gestion du versioning était aussi un critère majeur.

Protocole	Comparaison Thrift
Sockets	Rapide mais pas orienté objet, aucun contrôle sur la structure de donnée
JSON-RPC	Pas de génération du code client et serveur, optimisation à faire dans chaque langage
CORBA	Bonne structuration (similaire à Thrift avec IDL) mais développement plus long et latence comparée importante
DCOM	Propriétaire, non applicable
RMI, EJB	Développement plus long, latence
WebService / SOAP	SOAP est trop lourd et plus long a développer
REST	Nécessité d'une interface HTTP et latence comparée importante
ZeroMQ	Rapide mais trop peu de contrôle sur les données
Protocol Buffer, Avro, Message Pack	Rapide et structuré mais supporte moins de langages, rapidité de développement équivalente, Thrift: supporte plus de langages et implémentation plus élégante

Figure 24 : tableau de comparaison des protocoles de communication VS Thrift

Thrift rend possible une communication rapide, structurée, parallélisée et des interfaces natives dans de nombreux langages qui peuvent être développés rapidement et simplement.

c. Choix des langages

Les modules composant l'architecture sont en plusieurs langages mais ils respectent des choix simples :

- ✓ Les services développés exploitant une libraire spécifique sont dans le langage natif de la librairie (ex. Java pour Weka et MOA, JS pour Node.JS, C++ pour amélioration

Vowpal) et la compatibilité avec les autres services est assurée via la couche communication Thrift

- ✓ Pour les modules d'intégration ou non liés à une libraire comme WSIrma et Streamer, il a fallu choisir un langage bien adapté au machine learning et au prototypage rapide mais aussi facile à intégrer dans notre architecture : Python. Ses concurrents principaux étaient R, C/C++, Java, Lua. J'ai choisi le langage qui offrait le plus de compromis :
  - Lua : personne ne pratique le Lua dans la région, cela restreindrait drastiquement les capacités de recrutement de collaborateurs.
  - R : le langage open source du data mining est naturellement très utilisé en machine learning. Ses défauts comparés à Python sont : il nécessite un autre langage pour la gestion de l'architecture ou tout autre besoin qui ne soit pas du machine learning, R est en « relative » perte de vitesse comparé à Python.
  - Java est contraignant quand on veut faire du prototypage rapide à cause du temps de compilation et de la structure de code imposée.
  - C / C++ : moins de développeur dans ce langage, risque d'instabilité avec des développeurs non experts, très performant mais ce n'était pas une priorité majeure.
  - Python est très utilisé en machine learning et en progrès constant et est facile à apprendre tout en obligeant une certaine structure aux novices. Le code Python est assez performant, stable, fait de l'allocation dynamique de mémoire, et peut être largement optimisé de manière ciblé en portant des fonctions ou modules en C/C++. Après des tests comparatifs, il a été retenu.

2. Les modules métiers : prédictions, recommandations

Les modules métiers communiquent avec WSIrma en Thrift ou peuvent être embarqués dans WSIrma pour bénéficier de sa gestion intelligente de modèles.

a. Machine learning avec Weka (accessible aux non spécialistes) et MOA (spécialisation flux)

Ce module permet à WSIrma d'exposer via son API les librairies Weka et MOA de l'université de Waikato.

- La librairie Weka de machine learning offre une librairie Java et un environnement accessible à tous pour tester et créer rapidement des modèles depuis une très grande variété d'algorithmes. C'est ainsi permettre aux non spécialistes de l'entreprise de créer ou modifier des modèles, collaborer, sans connaissance du domaine. Le machine learning offre de nombreuses possibilités, un frein est souvent la difficulté d'accès à ce domaine aux non spécialistes.
- La librairie MOA, en partie basée sur Weka, est dédiée aux algorithmes d'apprentissage en flux. C'est une librairie Java et un environnement visuel intégré facilitant la compréhension, les tests et la validation des hypothèses et algorithmes sur des flux de données dans le temps.

b. Machine learning avec Scikit (popularité dans le domaine)

Ce module permet à WSIrma d'exposer via son API les méthodes et algorithmes de la librairie Scikit soutenue par l'INRIA. Cette librairie Python est en passe d'être la librairie la plus utilisée dans le domaine du machine learning. Elle est simple, très complète pour l'apprentissage hors-ligne, s'intègre avec de nombreux autres projets Python. La feuille de route stricte et la forte communauté contribuant au projet ont rapidement mené à un produit fiable, des algorithmes optimisés, des options de calculs distribués, des tutoriels sur de nombreux cas d'utilisation simplifiant l'apprentissage et la découverte par ses utilisateurs. Cette librairie semble incontournable pour faire du prototypage rapide.

c. Machine learning avec Vowpal Wabbit (performance)

Ce module permet à WSIrma d'exposer via son API les méthodes et algorithmes de la librairie Vowpal Wabbit. Cette librairie en C++ est vue par beaucoup de spécialistes comme la plus rapide du secteur pour l'apprentissage de gros volumes de données, notamment en flux, avec de nombreux attributs. Elle est principalement utilisée via un module précompilé accessible en ligne de commande pour différente plateformes (Linux, Windows). Elle rend possible sur WSIrma l'apprentissage et la prédiction en ligne sur des flux massifs de données sur une simple machine[7].

d. Recommandations basées sur Lenskit (spécialisation recsys)

Ce module permet à WSIrma d'exposer, via son API, différents systèmes de recommandations basés sur la librairie Lenskit. Cette librairie Java offre des fonctionnalités très poussées pour les algorithmes de recommandation. Les systèmes de recommandation connaissent souvent plusieurs phases dépendant des données disponibles[8] et combinent souvent plusieurs algorithmes notamment pour tenir compte de multiples facteurs dans la recommandation. Dans notre cas, tenir compte du lieu et du temps nécessitait un développement complémentaire spécifique qui a été inclus dans ce module.

e. Recherche automatique de modèle optimal (algorithmes et hyper-paramètres)

Ce module est dédié à la recherche automatique de modèle optimal par le choix de l'algorithme et des hyper-paramètres les plus performants parmi un champ plus ou moins restreint de possibilités. Il exploite les méthodes « Grid Search » et « Randomized Search » de Scikit-learn et reprend son API car pratique et déjà intégré avec différentes librairies.

[7] Vowpal Wabbit est différent de GraphChi (présenté dans un chapitre précédent) qui est lui dédié à l'apprentissage hors-ligne et est orienté graphes.

[8] Les systèmes de recommandation peuvent observer plusieurs phases suivant le volume de données d'apprentissage disponible : "cold start" quand il y a encore peu de données possibles, assez de données mais pas de couverture globale, beaucoup de données (le modèle sera peut-être alors modifié pour maintenir sa rapidité de traitement)

Un prototype prometteur, mais non encore finalisé, a été réalisé pour supporter l'optimisation Bayésienne (exploration et convergence accélérée par l'utilisation des probabilités pour orienter la recherche) en utilisant Hyperopt avec des fonctions d'arrêt prématuré quand l'apprentissage en court a une convergence trop lente ou quand les étapes d'amélioration ne progressent plus assez rapidement (early stopping).

Le futur « model tuner », présenté plus bas, reprendra cet outil mais pour gérer de manière permanente la recherche d'optimisation possible des modèles existant dans l'architecture.

f. Backtesting pour recherche de stratégie

Ce module simule et évalue dans le temps des stratégies exploitant des résultats de machine learning (prédiction, recommandation, optimisation). En fournissant des données d'historique et en choisissant une stratégie (règles et coût d'un succès ou d'une erreur) : il joue cette stratégie sur cette période historique et retourne le résultat de cette stratégie. Il peut notamment chercher automatiquement parmi un ensemble de choix de stratégies et d'options, pour identifier la meilleure stratégie. Un résultat peut être une combinaison de critères en fonction des priorités (performance, stabilité, risque…).

g. Optimisation et code métier

Quelques méthodes d'optimisation et du code métier sont proposés pour répondre aux différents objectifs du projet. C'est un module externe qui se charge dans WSIrma pour offrir ces nouvelles méthodes. Il peut s'agir de code métier combinant l'utilisation des différents modules comme Scikit et Vowpal Wabbit ou par exemple un algorithme de bandit manchot ou de Q-Learning (apprentissage de politique optimale sur actions multiples, et résultat différé dans le cas du Q-Learning).

h. Future : modules Concept-drift et Model tuner

i. Concept drift detector

Ce module détectera automatiquement le changement de modèle de donnée et pourrait décider de basculer vers un autre modèle, modifier les poids dans un modèle ensembliste, reconstruire des modèles en parallèle, ou augmenter fortement les poids des nouvelles données dans l'apprentissage. Les données pour lesquels nous créons des modèles de prédiction, recommandation, optimisation peuvent changer radicalement dans le temps (progressivement ou brusquement). Nous avons opté pour des tests fréquents et des algorithmes qui embarquent plus ou moins l'évolution vers des données changeantes et l'oublie progressif de données passées mais un module dédié garantie une réactivité forte et pertinente quelque soit le modèle. J'ai déjà les bases théoriques, choisi et testé les outils à utiliser pour réaliser un tel module. Il reste maintenant à réaliser un prototype mais avoir des données de meilleur qualité (notamment un retour d'exploitation prolongée de cette plateforme) que celles actuellement disponibles est nécessaire.

ii. Model tuner

Ce module évaluera en permanence les modèles les plus intéressants à optimiser et fournir alors de nouveaux modèles entraînés si plus performants. Les modèles initiaux sont déjà des modèles plus ou moins optimisés selon plusieurs paramètres mais le contrôle de l'exploitation peut montrer qu'investir plus de temps et de ressources dans l'optimisation de certains modèles peut être très rentable. Ce module le fera automatiquement en fonction des ressources inutilisées pour en permanence tirer le meilleur parti de l'infrastructure et des objectifs à atteindre. La demande d'optimisation de modèle pourra aussi être déclenchée à la demande du concept-drift detector.

3. Les connecteurs ou fournisseurs de WSIrma

a. Poller : exploiter les services de l'architecture avec de simples fichiers

Ce module permet d'exploiter le service WSIrma sous forme de fichier. Un fichier déposé sera immédiatement traité par l'infrastructure de prédiction ou de recommandation en fonction du répertoire dans lequel il a été déposé et le format du nom du fichier. Le fichier de résultat est créé en sortie et, selon les règles définies, le fichier d'origine est conservé, déplacé ou effacé et des logs sont créés.

Les formats de fichiers supportés sont le JSON / JSON-schema, CSV / TSV / TXT, ARFF.

Les actions principales, correspondant à des répertoires, sont : entraîner ou enrichir un modèle, prédire ou recommander ou optimiser, chercher le meilleur modèle, charger un modèle, sauvegarder un modèle.

L'intérêt est multiple :

- ✓ Prototypage et tests : l'utilisation simplifiée de l'architecture WSIrma sans écriture de code permet de créer rapidement et simplement de nouveaux modèles ou de faire des tests immédiats.
- ✓ Vitesse pour les fichiers volumineux : l'utilisation de fichiers bruts accélère fortement le traitement dans de nombreux cas où le volume de données est important.
- ✓ Intégration rapide : l'environnement de travail étant Linux, ce module orienté fichier s'intègre facilement avec de nombreux outils puissants du Shell pour réaliser des intégrations multiples.
- ✓ Lots de tests : les nombreux lots de tests de non régression sur les modèles sont réalisés simplement via une archive dont les fichiers sont décompressés dans les répertoires du Poller et immédiatement comparés avec les résultats prévus.

b. Streamer : agrégation et enrichissement rapide des flux d'entrée

Ce module a été conçu suite aux premiers prototypes de sources de données pour la prédiction des prix des hôtels qui dont les prédictions ont été grandement améliorées après l'ajout de données agrégées des hôtels voisins et de multiples agrégations temporelles

(différentes tendances sur 15mn, 1h…1mois). Malheureusement, la création de ces données volumineuses depuis une base de données SQL prenait plusieurs jours si on voulait le lancer sur la totalité des données d'historique disponibles. Après optimisation (étude des indexes, interprétation interne de la requête, génération dynamique de requête, création de fonctions dédiées, modification du moteur), cela ne mettait plus que 2 à 4 heures. J'ai alors développé un prototype en C-Python parallélisé qui créait ce flux à la volé et était capable de générer ces mêmes données en 20 secondes soit 60 000 données secondes, et en utilisant beaucoup moins de ressources. C'était donc une solution parfaite à développer pour notre architecture en quasi temps-réel et qui a été ensuite amélioré avec les fonctionnalités d'agrégation de données d'événements pour les flux de données de comportement pour les recommandations.

Streamer est un module autonome qui non seulement agrège des flux différents de manière définie ou automatiquement, mais aussi les enrichis via différents filtres et l'historique temporel des données (ex. tendances, moyennes pondérées, fréquence….). Cet enrichissement des données via l'historique fournit des informations, pour les recommandations comme pour les prédictions, de qualité à moindre coût. La gestion de flux de donnée est cruciale car de la qualité du flux (variété, pertinence, rapidité, fiabilité) dépendent les prédictions et recommandations délivrées.

Il gère de nombreux formats de données en entrée, accepte les données venant du Poller, et possède un connecteur RabbitMQ afin de pouvoir consommer la file de message de données venant de BeeSpy2 (architecture de collecte de données de la concurrence par robots).

Il offre de nombreuses autres possibilités qui ont été ajoutés au fur et à mesure : gestion de la pondération des données d'apprentissage (weighted exemples) pour les données rares ou les objectifs à prioriser, support de la fameuse librairie Pandas pour effectuer de nombreuses autres manipulations de données sur les flux.

Le Streamer est flexible car il s'intègre avec WSIrma, Poller et a été adapté pour s'intégrer dans une architecture Storm ou Spark.

c. Storm Streamer (prototype)

Ce module est une version isofonctionelle du module Streamer. Il bénéficie ainsi de l'architecture Storm, gestion de flux de données en réseau, et du Streamer pour bénéficier de ses fonctionnalités d'agrégation temporelles très flexibles. Il est à l'état de prototype fonctionnel et la décision future de le déployer, avec une phase préalable nécessaire de refactoring, dépendra de la taille de l'équipe et de l'architecture dédiée au service prédiction et recommandation. Il permettra d'avoir un streamer plus scalable (meilleur capacité de mis à l'échelle) et non propriétaire. Un désavantage aujourd'hui de Storm est que la modification d'un traitement de flux impose de stopper et de redémarrer sa topologie (architecture de gestion d'un flux).

d. OPRA : SOAP/CXF et REST/Node.JS

Ce module est un connecteur existant en 2 versions pour exploiter WSIrma en REST ou SOAP. Le module REST en Node.js autorise l'ajout de logiques métiers, c'est l'interface la plus utilisée car REST est la méthode préférée de communication des applications de l'entreprise. Le module SOAP a été développé au début du projet pour fonctionner avec ADAMS, un logiciel Java permettant de créer en quelques clicks des workflows de données plus ou moins complexes et d'exposer ses services en webservice ou script. ADAMS a été très utile en début de projet mais nous ne l'utilisons plus actuellement, il est possible qu'il soit réutilisé dans un futur projet.

4. Prédictions et recommandations

Les prédictions et recommandations prévues dans les objectifs utilisent donc cette architecture.

a. Prédictions "Price change trigger" et "Overbooking risk & room availability"

La prédiction "Price change trigger" vise à savoir quand est-ce qu'un prix de chambre va changer et vers quelle valeur approximative pour s'assurer toujours d'être moins cher que la concurrence ou désactiver le produit. La prédiction "Overbooking risk & room availability" vise à savoir quelle est la réelle disponibilité des chambres, car souvent l'hôtelier ne prend pas le temps de la mettre à jour, pour maximiser l'offre de VeryLastRoom sans prendre de risques.

Ces prédictions, similaires dans leur implémentation, modélisent donc les différentes règles globales des stratégies des hôteliers comme leurs choix individuels face à la demande. J'ai donc créé un système capable de prédire par combinaison de modèles sur la base de modèles individuels par hôtel, par groupe d'hôtels similaires (pour modéliser les phénomènes locaux) et un modèle global.

Les rôles des différents éléments de l'architecture sont les suivants :

- ✓ Le « Streamer » génère des données enrichies pour l'entrainement et la prédiction. Il transmet d'un côté les données d'entraînement à « WSIrma » et les sauvegarde au fur et à mesure dans un fichier historique d'entraînement. D'un autre côté, il fournit les dernières données disponibles par hôtel au composant « OPRA / REST » car nécessaires pour des demandes de prédiction.
- ✓ « OPRA / REST » fournit l'application métier « Price change trigger » et « Overbooking risk & room availability » en s'appuyant sur un algorithme métier et les prédictions fournies par « WSIrma » nécessitant les dernières données disponibles par hôtel provenant du « Streamer ». C'est lui qui fournit donc le principal service délivré à l'extérieur de l'architecture.
- ✓ « WSIrma » est utilisé pour :

- créer les modèles initiaux optimaux depuis les données historiques des hôtels générées par le « Streamer » (ou les recréer à la demande du processus de maintenance).
- entrainer et mettre à jour les modèles par les données fournies en permanence par le « Streamer ».
- répondre aux prédictions demandées par le module « OPRA / REST »
- la gestion éventuelle de multiples modèles pour un même type de prédiction.

✓ « Poller » déclenche la création de modèles optimaux depuis le fichier historique d'entraînement.

✓ Un script de maintenance relance les demandes de création de modèles optimaux à fréquence définies en utilisant les fichiers de données historiques d'entraînement avec le « Poller ». Il sera dans le futur remplacé par des comportements intelligents offerts par les futurs composants « tuner » et concept drift detector ».

✓ Les logs des modules OPRA / REST, WSIrma et Poller fournissent des statistiques de prédiction et de fonctionnement de l'architecture. Ces logs peuvent être analysés via le système de centralisation des logs Elasticsearch / Kibana (développé dans un autre projet).

b. Recommandations "Autoyield recommandation" et "Customer recommended offers"

Ce service nécessite de pouvoir recommander sur la base de différents contextes (lieu, heure, concurrence locale) et avec des données d'apprentissage qui peuvent être quasi inexistantes ou nombreuses. Les recommandations classiques ne prennent pas en compte le contexte.

J'ai donc étudié plusieurs papiers de recherche et réalisé plusieurs prototypes pour développer :

- un système sur mesure prenant en compte les différents types de contextes (lieu, heure, concurrence) tout en s'adaptant aux différentes phases liées à la disponibilité de données : cold start, recommandation orientée "utilisateur", recommandation orientée "produit".
- la collecte des données historique de comportement et les conversions (achat ou presque-achat) : un flux d'apprentissage collectant intelligemment dans le temps le lien entre action et cible souhaitée (ex. du « click » et temps sur différentes pages à une commande ou autre type de conversion). Cela a conduit à l'extension des possibilités du « Streamer ».
- un système de simulation de stratégie de prix pour pouvoir simuler et découvrir les meilleures stratégies de tarification pour "Autoyield recommandation". Cela a conduit à la réalisation du module métier de WSIrma « Backtesting ».
- des lots de tests pertinents pour évaluer la pertinence des recommandations "Autoyield recommandation" et "Customer recommended offers". Des scripts dédiés de génération de ces données ont été créés.

Ces recommandations utilisent un processus similaire à celui décrit dans « Price change trigger » et « Overbooking risk & room availability » mais au niveau de WSIrma s'appuient sur le moteur de recommandation basé sur « Lenskit » (voir IV.B.4.d) dont les différents processus de sélection du modèle, d'entraînement, de mise à jour et de recommandations sont décrits dans les schémas en annexe.

c. Prédiction "Customer booking probability"

La prédiction "Customer booking probability" souhaite prédire en temps réel quand est-ce qu'un client prévoit de réserver une chambre d'hôtel chez la concurrence ou sur VeryLastRoom. C'est un problème qui peut être traité comme les prédictions pour les hôtels ("Price change trigger" et "Overbooking risk & room availability") mais appliqué aux utilisateurs. La direction de l'entreprise a décidé de reporter cet élément du projet car elle impliquait des développements pour l'exploiter[9]. Un premier prototype fonctionnel a tout de même été créé mais il sera probablement nécessaire de le transformer pour le nombre d'utilisateurs visés (1 million). Il pourrait ainsi être nécessaire de faire des clusters automatiques d'utilisateurs.

5. Spark (à l'étude)

Apache Spark est un framework de traitement de big data intégrant la gestion des flux et du machine learning. Il a remplacé beaucoup de projets qui utilisaient le traitement par lot en MapReduce car Spark est souvent beaucoup plus rapide, moins gourmand en ressources, et simplifie la migration par sa proximité en termes de conception. Le lot en « batch » de MapReduce est remplacé par un traitement en flux sur Spark qui est en fait du « micro batch » (micro lots de données) autorisant ainsi un code très similaire. Il est sur la feuille de route de très nombreux projets d'entreprise phares dans ce domaine. Après étude, il ne convient aujourd'hui pas à nos besoins et contraintes, notamment à cause des limites sur la gestion dynamique des modèles en mémoire, nécessaire pour avoir une architecture minimale, et des données temporelles. Il pourrait être un remplaçant d'une importante partie d'architecture dans le future si l'entreprise veut investir dans une architecture un peu plus coûteuse en terme de capacité mémoire et de calcul pour simplifier l'architecture et la rendre plus standard.

C. Réalisation

1. Planification initiale

a. Utilisation de la planification initiale

La réalisation du projet (voir "solutions retenues") s'est architecturée sur la base de la planification initiale qui a évolué par des itérations contrôlées suites aux résultats de la R&D et des phases de développement. Cette planification initiale a été réalisée à la suite de la

[9] Développements nécessaires : système de collecte des données du comportement de l'utilisateur et modification des applications sur smartphone pour qu'elles prennent en compte ces recommandations.

validation de la charte de projet et le plan de gestion de projet (Project management plan) avec une hypothèse d'une ressource principale (moi), de deux ressources externes en développement ainsi que l'aide ponctuelle de membres de l'équipe de développement.

Les grands points de la planification initiale :

- réalisation d'une première version du projet remplissant les différents objectifs du projet en 9 mois (sous condition de disponibilité des différentes ressources humaines et matérielles) découpée comme suit :
    - 3 mois pour concevoir la première version de l'architecture avec la première prédiction "Price change trigger".
    - R&D et réalisation des objectifs de prédiction pendant les 6 mois qui suivent.
    - utilisation de ressource externe en parallèle sur :
        - les prototypes et développement pour les objectifs en termes de systèmes de recommandation.
        - les prototypes et développement pour la mise à l'échelle de l'architecture (scaling).
- utilisation de 2 ressources externes qui se sont traduites par une sous-traitance distante (développeur expert Java en Russie) et un stagiaire qui a été prolongé par un contrat à durée déterminée.
- utilisation ponctuelle d'autres ressources internes à l'entreprise pour l'intégration entre les APIs de l'entreprise.

b. Diagramme de Gantt prévisionnel détaillé

Le projet a été décomposé en tâches (WBS) et a permis la réalisation du diagramme de Gantt ci-dessous. Le temps de réalisation de chaque tâche a été estimé par analogie ou, quand cela n'était pas possible, par jugement expert.

Name	Work
Valider la charte de projet	2d
Développer le "Project Management Plan"	**3d 1h**
Project Plannification / Preparation & finalisation recrutement initial	3d
Irma et prédiction changement prix	**49d**
Java command-line trainer	**9d**
Analyse complémentaire - structure & plan	2d
Java command-line trainer via CSV	2d
add full support of Weka GUI command line	1d
tests & corrections	2d
Documentation	2d
Java command-line prediction	**8d**
Analyse complémentaire - Definir structure & plan	2d
command-line prediction via CSV OR Inline/JSON?	2d
tests & corrections	2d
Documentation	2d
Automatisaton data preparation	**5d**
Analyse complémentaire - DB Schema & plan	1d
DB update/sync	1d
Automatisation & tests	2d
Documentation	1d
Automatisation training	**4d**
Analyse complémentaire & DB Schema	1d
Automatisation & tests	2d
Documentation	1d
Finalisation 1ère prédiction	**4d**
Sélection/correction objectif parmi déjà créés	1d
Nettoyage/amélioration SQL existant	1d
Tests pour validation & collecte données choix comité	1d
Présentation résultats comité & préparation	1d
Objectif 1 intermediaire	
Symfony Web Service	**11d 4h**
Définir Schema generique et Interrogation type Google Prediction IO	1d
DB Schema services & tests	1d
Delegated	**5d 4h**
Single prediction	1d
Multiple predictions	2d
Prediction dictionary	4h
Test suit	2d
Collaboration to integrate with Vlr backoffice	1d
Réunion projet interm 1er projet & prep	1d
Documentation	2d
Performance & issue logging	**3d**
trainer: DB management class & process	2d
prediction: DB management class & process	1d
Prediction to production	**4d 4h**
Data update & agregation	1d 4h
Scheduling & automation	1d
Tests / Validation	2d
Generalisation et documentation howto	5d
Réunion validation 1er objectif & preparation	1d
Objectif 1 final	
2ème prédiction: dispo maximale des chambres	**23d**
analyse des données	1d
agrégation	1d
étude, recherches	2d
Réunion décision 2ème prédiction & préparaiton	4h
Objectif 2 intermediaire	
création/validation des jeux de test	5d
création/sélection du model	3d
creation du webservice	1d
automatisation & doc	4d
to production	4d 4h
Réunion validation 2ème objectif & préparation	1d

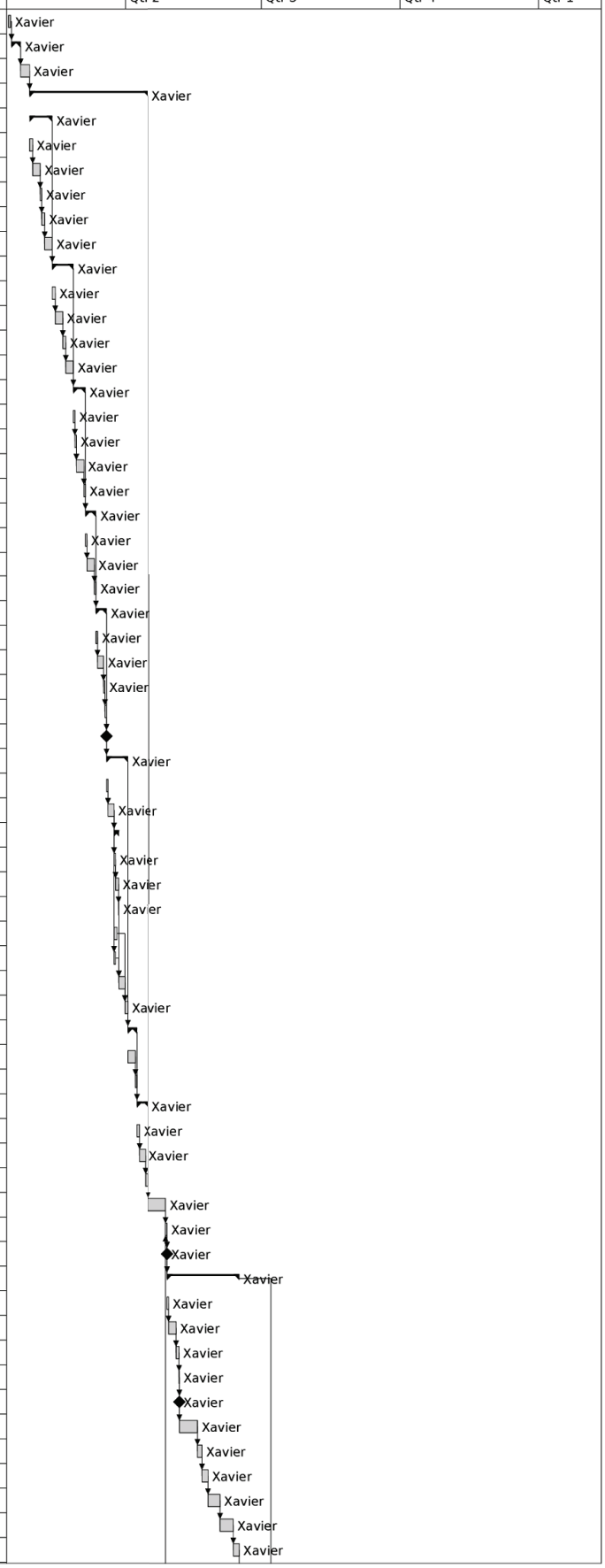


Figure 25 : GANTT de planification initiale 1/2

Fin Objectif 2	
3ème prédiction: recommandation "autoyield recommandation"	**8d**
support version 2.0	8d
Fin Objectif 3	
4ème prédiction: "customer booking probability":	**23d**
version 1.0	12d
version 2.0	8d
production	3d
Fin Objectif 4	
5ème prédiction: recommandation "customer recommended offers'	**8d**
support version 2.0	8d
Bilan de projet	2d
Optimisation & scaling création de modèles (Stage 1)	**97d**
Kick-off, prep, analyse & planning	5d
Itération 1: 20/80	**31d**
Dev	**25d**
Finalisation production Iteration 1	5d
Présentation nouvelle architecture & préparation	1d
Itération 2: strong	**40d**
Iteration 2a	20d
Iteration 2b	20d
Logs, monitoring	10d
Documentation/tuts dev & utilisateurs	10d
Fin Objectif architecture scaling & présentation	1d
Création d'un système de recommandation (Stage 2)	**107d**
Kick-off, prep, analyse & planning	4d
R&D, Archi	**23d**
Recherche, comparaison, tests	5d
Compréhension objectifs pour rec type 1 & 2	2d
Tests avancés sur solution choisie & rec type 1 & 2	4d
1er plan d'archi, implémentation détaillée	4d
création plateforme basique & tests	4d
création web service & tests	4d
rec type 1 & 2 version 1.0	**30d**
dev	**25d**
creation recommandation type 1 version 1.0 (obj 3)	10d
creation recommandation type 2 version 1.0 (obj 5)	10d
scaling 1.0	5d
prod & tests réels	5d
rec type 1 & 2 version 2.0	**30d**
dev	**25d**
création recommandation type 1 version 2.0	10d
création recommandation type 2 version 2.0	10d
scaling 2.0	5d
prod & tests réels	5d
Logs, monitoring	10d
Documentation/tuts dev & utilisateurs	10d

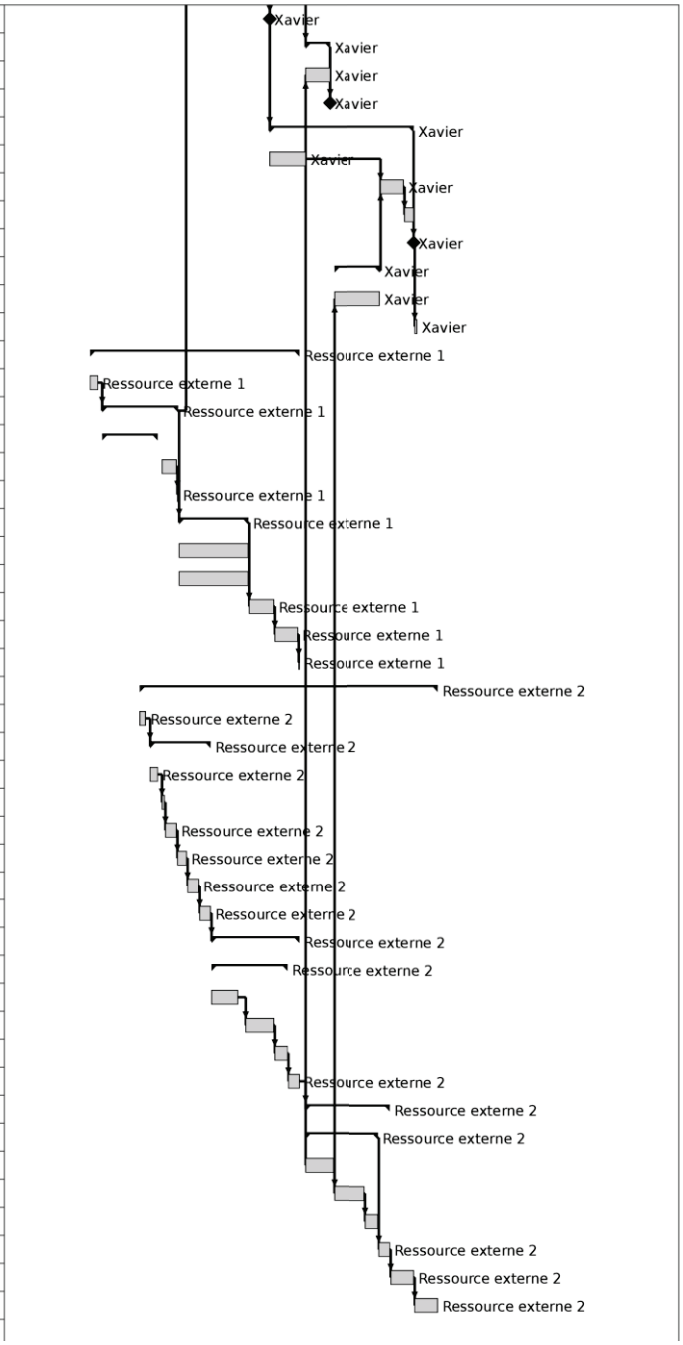


Figure 26 : GANTT de planification initiale 2/2

2. Historique de la réalisation

La réalisation a atteint les objectifs du projet en suivant la planification initiale avec les objectifs rectifiés (voir "modification des objectifs du projet"), les méthodes et outils détaillés en début de chapitre, et quelques différences majeures ci-dessous :

a. Ajout d'étapes

En début de projet (fin du premier mois) : bien qu'étant dans une phase initiale non orientée performance et mise à l'échelle (scaling), je me suis aperçu, en contrôlant mes estimations à partir des premiers résultats, que la conception envisagée ne pourrait répondre

aux temps de réponse exigés au final (max 10 ms pour 1000 prédictions). Une phase importante de R&D sur le choix intermédiaire d'architecture a été ajoutée immédiatement pour ne pas repousser un problème qui pourrait être bloquant plus tard. L'étape de création d'une architecture orientée service a donc été anticipée.

Cela a allongé de 5 semaines cette première phase cruciale du projet mais a permis d'éliminer un risque majeur et d'avoir une base déjà solide et flexible.

Au 6ème mois du projet, fort des premiers résultats du projet, j'ai eu pour mission de constituer un dossier, pour un appel à projet pour des fonds européennes, sur l'état d'avancement et les résultats intermédiaires de ce projet R&D ainsi que sa feuille de route.

Cela a allongé de 2 semaines cette phase du projet.

Il n'y a pas eu de recherche dans la compensation du temps additionnel ajouté par ces étapes car le projet était en avance par rapport à la disponibilité des projets dont il dépendait. Cela a d'autant moins été un problème que ce travail additionnel a aider à constituer un dossier solide pour cet appel à projet qui a été remporté par l'entreprise.

b. Modification des objectifs du projet

Deux modifications majeures ont eu lieu.

Production simulée : les projets annexes de l'entreprise qui devaient alimenter les flux de données nécessaires (BeeSpy2) et supporter l'architecture requise pour la mise en production de ce projet n'ont pu être mis à disposition par l'entreprise. Les phases dites de "production" ont donc été remplacées par des environnements de simulation avec des flux similaires.

Report prédiction "Customer booking probability" : la R&D avec prototype fonctionnel pour la prédiction 5 a été réalisée mais le développement nécessaire a été reporté par l'entreprise car le développement nécessaire pour exploiter cette prédiction ne pouvait être programmé avant longtemps.

c. Suspension temporaire du projet

Ce projet a été interrompu pendant 4 mois à la demande de l'entreprise pour m'investir dans le projet de conception de la nouvelle architecture globale virtualisée de l'entreprise.

d. Diagramme de Gantt réel

Gant réel après réalisation :

Name	Work
Corps du projet	**238d...**
Création charte de projet	2d
Validate Project Charter	1d
Create Project Management Plan	**3d 1h**
Project Plannification / Preparation & finalisation recrutement initial	4d
Irma et prédiction changement prix	**80d 4h**
Moteur beta & validation choix d'architecture	**41d**
Fiabilisation archi	**23d**
Finalisation 1ère prédiction	**12d**
Prediction to production	**4d 4h**
Generalisation et documentation howto	5d
Réunion validation 1er objectif & preparation	1d
Objectif 1 final	
2ème prédiction: dispo maximale des chambres	**23d**
3ème prédiction: recommandation "autoyield recommandatio	**8d**
Fin Objectif 3	
4ème prédiction: "customer booking probability":	**23d**
Fin Objectif 4	
PROJET ARCHITECTURE VIRTUALISEE	**80d**
5ème prédiction: recommandation "customer recommended offers"	6d
Bilan de projet	2d
Optimisation & scaling création de modèles (Sous-traitance)	**97d**
Kick-off, prep, analyse & planning	5d
Itération 1: 20/80	**31d**
Itération 2: strong	**40d**
Logs, monitoring	10d
Documentation/tuts dev & utilisateurs	10d
Fin Objectif architecture scaling & présentation	1d
Création d'un système de recommandation (Stage + CDD)	**107d**
Kick-off, prep, analyse & planning	4d
R&D, Archi	**23d**
rec type 1 & 2 version 1.0	**30d**
rec type 1 & 2 version 2.0	**30d**
Logs, monitoring	10d
Documentation/tuts dev & utilisateurs	10d

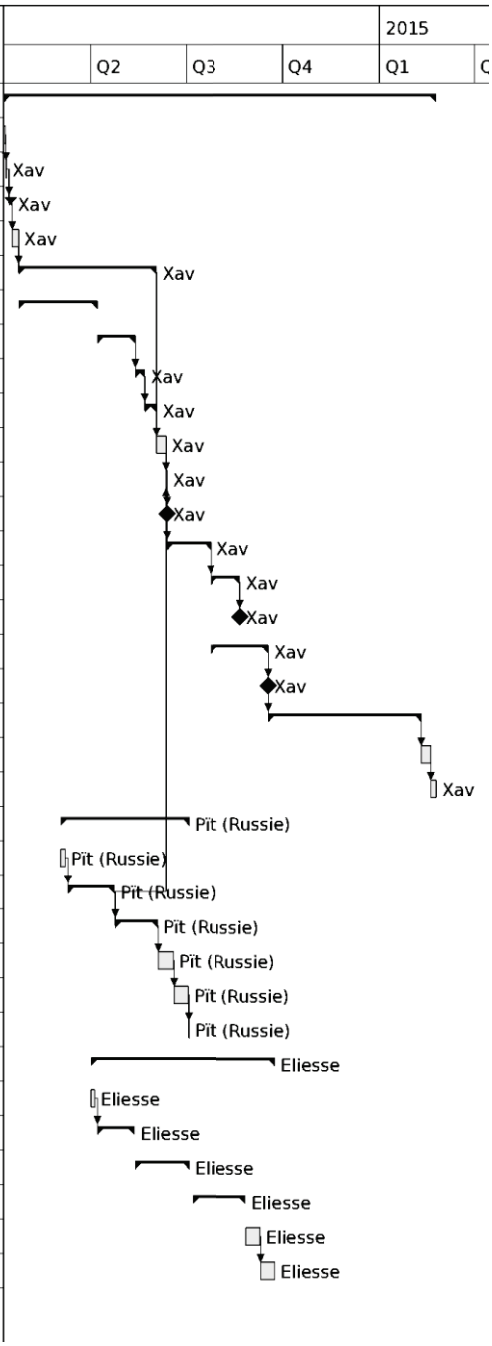


Figure 27 : GANTT de la réalisation effective (historique)

D. Résultats et retour d'expérience

1. Résultats du projet

Tous les livrables définis avec l'entreprise ont été livrés avec la qualité définie en environnement simulé. Ils ne sont pas encore en production ni en pré-production car les projets qui en dépendent ne sont pas encore disponibles. Un succès intermédiaire est que le projet R&D et son programme ont permis d'obtenir un financement de 400 k€ pour l'entreprise.

a. Etat des livrables

- Plateforme avec base de données agrégées et enrichies automatiquement : livré.
- API Web Service de prédiction : livré.
- Fabrication automatique des modèles : livré.
- Méthode semi-automatique de création de nouveaux modèles et prédiction : livré.
- Documentation de création de modèles et prédiction : livré.
- Architecture distribuée adaptée au Machine Learning et BigData : prototype livré.
- Prédiction 1 "Price change trigger" : livré.
- Prédiction 2 "Overbooking risk & room availability" : livré.
- Prédiction 3 "Autoyield recommandation" : prototype livré.
- Prédiction 4 "Customer booking probability" : reporté par l'entreprise.
- Prédiction 5 "Customer recommended offers" : prototype livré.

Tous les livrables prévus dans le projet ont été réalisés avec les indicateurs de qualité de développement définis (code, documentation, tests).

Pour des raisons techniques incontournables et non gérées par ce projet[10] et des priorités financières, la mise en production des produits développés dans ce projet n'est aujourd'hui pas planifiée. Il sera donc difficile dans l'immédiat de mesurer son efficacité en production.

b. Couverture délais et budget

Le projet a pris un retard cumulé de 7 semaines par rapport au planning initial sur 39 semaines soit un écart de +18%. Il a été anticipé au plus tôt et l'amélioration substantielle de la qualité de l'architecture et le temps alloué pour la rédaction du dossier de subvention ont contribué à l'obtention d'importants fonds pour l'entreprise.

Le budget du projet est sensiblement celui estimé au départ car le retard de +18% impactait uniquement mon affectation en tant que ressource et est compensé par la très faible utilisation des ressources internes, prévues à hauteur de 10% d'un développeur et 15% du CTO, et le budget de dépenses diverses de 5000 € qui a été utilisé à moins de 50%.

[10] Livraisons ou développements externes nécessaires au projet qui ont été mise en attente : BeeSpy2, l'architecture virtualisée, les fonctionnalités backoffice et frontoffice exploitant les prédictions

En accord avec l'entreprise, les produits ont été livrés en environnement simulé. La mise en production future des produits développés dans ce projet sera un nouveau projet avec une phase de redémarrage intégrant notamment les mises à jour nécessaires et probablement des objectifs révisés.

Les temps mesurés comme base d'évaluation pour de futurs projets devront être utilisés avec précaution sur la phase de mise en production qui n'était que sur des données simulées. Les corrections nécessaires suite à la mise en production réelle plutôt que simulée auraient pu être importantes car les données des prédictions 3 à 5 ne sont pas basées sur des réelles données d'historique.

c. Fiabilité et performance des prédictions

i. Prédiction 1 "Price change trigger"

Ces résultats sont les moyennes de précision sur les modèles hôtel sachant que chaque hôtel a son modèle, un modèle de groupe et appartient à un modèle global mais un seul méta-modèle héberge l'ensemble :

- ✓ Zone ROC >= 97%
- ✓ Précision TP (True Postive) >= 98%
- ✓ Précision TN (True Negative) >= 97%

Débit et ressources utilisées :

- ✓ Prédictions à la seconde : > 30 000 prédictions / seconde
- ✓ RAM moyenne : 2 GB
- ✓ Moteur de machine learning : Vowpal Wabbit (validé aussi avec MOA)

ii. Prédiction 2 "Overbooking risk & room availability"

Comme pour la prédiction 1, chaque hôtel a son propre modèle et appartient à un groupe et au modèle global et le tout est contenu dans un méta-modèle. Les moyennes de précision sur les modèles hôtel pour risque d'overbooking < 1% sont :

- ✓ Modèle régression : RMSE < 0.3
- ✓ Modèle multi-classe : moyenne par classe de TP > 97%

Débit et ressources utilisées :

- ✓ Prédictions à la seconde : > 30 000 / seconde
- ✓ RAM moyenne : 2 GB
- ✓ Moteur de machine learning : Vowpal Wabbit (validé aussi avec MOA)

iii. Prédiction 3 à 5

Contrairement aux prédictions 1 et 2 qui utilisent des données simulées sur la base de vraies données d'historiques, les données utilisées pour ces prédictions sont virtuelles donc communiquer les chiffres de fiabilité (RMSE et Top@N pour les recommandations) de ces prédictions et recommandations simulées ne serait pas représentatif d'une réalité.

Débit et ressources utilisées :

Prédictions 3

- ✓ Prédictions à la seconde : > 3000 recommandations / seconde
- ✓ RAM moyenne : 3 GB
- ✓ Moteur de machine learning : Lenskit

Prédictions 5 pour 100 000 utilisateurs :

- ✓ Prédictions à la seconde : > 3000 recommandations / seconde
- ✓ RAM moyenne : 4 GB
- ✓ Moteur de machine learning : Lenskit

Prédiction 4 pour 100 000 utilisateurs :

- ✓ Prédictions à la seconde : > 30 000
- ✓ RAM moyenne : 7 GB
- ✓ Moteur de machine learning : Vowpal Wabbit

d. Les indicateurs de réussite non encore mesurables

Le fruit de ce projet n'étant pas à ce jour en ligne, certains indicateurs clés de succès définis dans la charte de projet ne peuvent être évalués : satisfactions des utilisateurs, indicateurs de performance (KPI) de l'entreprise relatifs à ces prédictions et recommandations.

e. Facilité de création de nouvelles prédictions

Le système a été conçu et réalisé pour pouvoir créer des modèles de prédiction autant par des novices avec Weka (interface intuitive facile d'accès) que depuis des solutions plus avancées avec MOA, Scikit, Vowpal Wabbit. Les modèles comme les données nécessaires peuvent être fournies via l'API, une queue RabbitMQ, un pipeline connecté au streamer, ou par simple dépôt des fichiers de données et de modèles dans le répertoire du système Poller.

f. Architecture validée et optimisée en simulation

L'architecture n'est pas validée en production réelle mais les nombreux tests de charges simulés ont montré la capacité de soutenir une charge bien au-delà de ce dont a besoin VeryLastRoom actuellement (voir performance des prédictions). A noté qu'a été conçu un un prototype pour l'optimisation de l'utilisation des robots de collecte en fonction de la charge et que cette technique pourrait aussi être utilisée pour améliorer certains traitements de l'architecture.

g. Un système facile et évolutif pour le future

L'architecture orientée service multi-langages comme le module WSIrma autorisent l'ajout rapide de services et même de plugins (sur WSIrma) sans interruption du service et autorisant des versions différentes d'objets sur l'architecture à coexister.

h. Nouveau produit issu de la R&D : optimisation dynamique de l'architecture

Le module d'optimisation dynamique de l'architecture des robots de collecte en fonction de l'objectif et des ressources restreintes est né pendant le processus R&D. Il est au stade de prototype, sa finalisation attend un test nécessaire après la mise en production du projet BeeSpy2.

i. Le partage de connaissance avec le reste de l'équipe

Les connaissances développée dans ce projet R&D ont été partagées avec le reste de l'équipe notamment pour la création d'un système de notation des hôtels par les utilisateurs intelligent et l'analyse de données par le service marketing en utilisant des outils simples de machine learning (BigML), probablement un bon premier pas pour réfléchir à de futures applications plus avancées.

2. Retour d'expérience

a. Les difficultés du projet

Comme dans tout projet, j'ai rencontré différentes difficultés formatrices pour mes futurs projets.

i. Répartition des temps du cycle d'un projet de « machine learning » et simplification

Je me suis rapidement aperçu que ma répartition des temps entre les différents activités de machine learning étaient surdimensionnées pour la recherche des modèles optimaux et sous-dimensionné pour les phases de préparation des données et du problème à résoudre.

Une fois qu'une architecture de machine learning est en place, la collecte et le traitement des données représentent en général bien plus de travail que la création de l'algorithme d'apprentissage.

De même, je me suis aperçu que mon premier problème (anticipation des prix à la place du changement de prix) n'avait pas été bien assez défini et pouvait être simplifié.

Investir du temps en amont pour simplifier au maximum le problème à résoudre et le découper en sous-problèmes et faire d'éventuelles réductions (transformer un problème en un autre équivalant – ex. maximisation transformée en minimisation, classification multi-labels transformée en plusieurs classifications binaires) réduira grandement le risque et le travail à accomplir.

ii. Risque de dépendance non maîtrisé

Malgré la réussite technique du projet R&D et les retombées financières, les produits du développement ne sont pas actuellement en production car ils nécessitent le projet annexe BeeSpy2 et la nouvelle architecture qui ne sont toujours pas disponibles. L'entreprise devant faire face à l'augmentation des coûts marketing, élément stratégique pour une startup, qui

ont été multipliés par plus de 6 en 1 an : ce programme R&D a donc été mis en pause. C'est un choix assumé de l'entreprise largement compensé par les retombées financières mais ne pas pouvoir exploiter les produits réalisés est forcément un regret personnel. Dans mon analyse de risque cela avait été mentionné mais pas estimé assez probable pour exiger une solution de remplacement de ma part surtout étant nouveau dans la société. J'ai demandé plusieurs fois en cours de projet une solution alternative mais n'ai pas pu l'obtenir car l'entreprise pensait encore pouvoir livrer les projets attendus. Cette situation aurait probablement pu être évitée si j'avais négocié une alternative au moment de la rédaction et de la signature de la charte de projet.

iii. Données temporelles et performances individuelles : méthode d'évaluation

En début de projet, j'avais obtenu un très haut taux de prédiction sur les prix futurs des chambres d'hôtel. J'ai été bien surpris quand je me suis aperçu que mes jeux de tests étaient automatiquement mélangés avant entraînement et donc que j'apprenais sur des données futures. De même, je me suis aussi aperçu que les critères d'évaluation choisis ne me permettaient pas de constater les contre performances pour certains hôtels (ou quand les changements de prix étaient rares). Les critères d'évaluation ont donc été revus : évaluation de la performance du modèle par hôtel et utilisation de l'indicateur de Kappa.

La définition rigoureuse des bons critères d'évaluation et protocoles de test sont ainsi un travail majeur de préparation pour ne pas partir dans une mauvaise direction.

iv. Importante courbe d'apprentissage et retour des mathématiques

Le "machine learning" requiert un important corpus de connaissance dans différents domaines (statistiques et optimisation, développement d'algorithme, architecture de données) à connaître pour résoudre efficacement un problème et son architecture. Pour des novices et si le projet est ponctuel, il vaut mieux s'orienter directement vers des solutions prêtent à l'emploi et reprendre les cas d'utilisation déjà bien étudiés.

Le retour nécessaire des mathématiques pour les analyses de données et la compréhension approfondie des algorithmes a été difficile malgré un fort intérêt pour cette discipline. Après cette année d'immersion, il me faudra à nouveau revoir tous les concepts abordés pour ancrer ces connaissances à plus long terme.

v. ML en flux : Anticipation des ressources et des latences

En début de projet, j'ai eu à repenser la planification du projet de manière importante suite à l'estimation du temps de traitement en production calculé sur la base d'un prototype. Plus tard dans le projet, j'ai pu remplacer un élément majeur de l'architecture très gourmant en mémoire et en traitement par une solution plus adaptée, moins gourmande en mémoire et bien plus rapide (de 1000 prédictions en 10ms à > 30 000 prédictions en 1s).

Cette anticipation est un prérequis dans la conception d'une architecture mais quand la connaissance est très limitée comme le temps, il faut organiser son projet en conséquence.

vi. Gestion fine de la mémoire en Java

Les différents techniques de la mesure de la mémoire, globale ou par objet, disponibles en Java ont soit une certaine latence soient nécessitent d'importants calculs pour des objets complexes comme pour les modèles de machine learning. Sachant que je devais pouvoir gérer de nombreux modèles ne pouvant être gardé simultanément en mémoire, j'ai investi du temps et des ressources pour concevoir une gestion intelligente des modèles à charger (collection « LRU » modifiée), capable d'anticiper la mémoire et de distribuer cette mémoire sur d'autres machines.

vii. Difficile mariage entre R&D et développement des livrables

J'étais à la fois impliqué en même temps dans la R&D et ses prototypes d'un côté et dans le développement pour la mise en production donc avec une exigence de qualité différente. La R&D et ses prototypes demandent de cerner de nombreux concepts transverses et de réaliser rapidement des prototypes, l'autre demande une rigueur constante et suit une certaine routine. Ces deux activités se parasitent facilement : le prototypage rapide de la R&D peut faire prendre de mauvaises habitudes pour le développement d'application finie, le développement peut faire perdre l'agilité et la mémorisation des nombreux concepts à garder en mémoire pour avancer rapidement en R&D. Basculé d'un type d'activité à l'autre m'a parfois fait perdre pas mal de temps. J'ai donc essayé de bien alterner les deux et sous-traiter le développement lourd dès que possible.

viii. Attention aux mauvais outils : Microsoft Project m'a manqué

N'ayant pas accès à MS Project que j'utilise couramment pour la gestion de projet, j'ai utilisé le logiciel open-source « Planner Project Management » sous Ubuntu. J'ai non seulement perdu pas mal de temps pour faire des planifications de moindre qualité mais sachant le temps que je perdrai et ce qui n'était pas possible de faire ou trop difficilement, j'ai souvent opté pour de la saisie manuelle dans mes cartes mentales pour le suivi courant.

b. Leçons à exploiter

Toutes les difficultés du projet présentées dans le chapitre précédent sont déjà de bonnes leçons à exploiter mais ce projet m'a aussi enseigné d'autres leçons présentées ci-dessous.

i. Github, Travis, Gestion de dépendances : Maturation du développement collaboratif et réparti

Github et Travis sont des outils de travail collaboratif et d'automatisation en développement bien plus efficaces que ce que j'ai pu utiliser jusqu'à présent. Ces nouveaux gestionnaires de version et leurs différentes possibilités de lien avec d'autres services notamment pour le test et l'intégration automatique, simplifient et améliorent grandement la gestion du code (versions, branches, corrections, collaboration, bugs, tests, qualité, documentation) et le travail d'équipe. La facilité à collaborer sur des projets open-source en assurant de bonnes méthodes de contrôle de qualité est pour moi une révolution nécessaire dans le développement.

Cela ainsi été un réel plaisir de gérer mes projets en interne et en sous-traitance, et d'une grande facilité pour contribuer à l'amélioration de projets externes de renommé tels que Vowpal Wabbit (Microsoft Labs), GraphChi (ex CMU NYC). Pour la sous-traitance cela permet de partager des outils universels et efficaces qui peuvent être configurés pour maître en place des standards élevés de qualité avec un fort degré d'optimisation. De nombreux services annexes s'intègrent avec GitHub.

Couplé aux gestionnaires de dépendance (ou de paquets) qui sont en voix de normalisation, le développement collaboratif et réparti semble atteindre une certaine maturité assurant qualité et simplicité.

ii. Performance vs dimensionnement

Il est souvent possible d'améliorer grandement les performances d'un modèle en machine learning mais cela va souvent au détriment des ressources nécessaires et de la complexité de l'architecture. C'est l'exemple du challenge Netflix où le gagnant a surpassé tout ce qui avait été fait par la combinaison de plus de 100 modèles. Cela n'a jamais été mis en production car trop coûteux. De nombreux problèmes de machine learning ou le big data n'ont pas forcément besoin d'un cluster (grappe d'ordinateurs en réseau), une bonne ingénierie du flux de données et l'utilisation d'algorithmes en ligne (ex. GraphChi, Vowpal Wabbit) peut être un peu moins performant théoriquement et surpasser une grosse architecture une fois en production. Par ailleurs, la baisse constante des coûts des ressources informatiques comparé aux coûts de développement et de maintenance, la nécessaire anticipation d'un projet à une échelle plus massive, l'évolution vers les architectures distribuées comme norme mettent en évidence la nécessité de concevoir des architectures compatibles avec une architecture distribuée.

iii. Gestion du cycle de projet en machine learning

Je pense avoir développé un bon cycle de gestion de projet de machine learning que je réutiliserai et améliorai (voir III.D).

iv. Architecture de machine learning : sur-mesure vs standards

Le développement de cette architecture sur-mesure et les tests sur les nombreux composants d'architecture m'ont aidé à comprendre en profondeur les implications des différents choix techniques et des algorithmes. J'ai appris qu'utiliser uniquement des composants standards n'était à ce moment pas possible face aux contraintes du projet. Par ailleurs, le « machine learning » et le « big data » sont un secteur en pleine évolution ayant la volonté de faire émerger des composants d'architecture de plus en plus solides et intégrés. Je suis en veille sur ces composants afin de pouvoir réduire au maximum les parties non standards dans mes futurs projets.

v. Recrutement par challenge sur le futur poste

Le recrutement par tests sur les connaissances et compétences à développer pour un poste sous forme de compétition a été relativement rapide et très efficace. C'est donc un bon moyen de recruter dans un temps maîtrisé des développeurs de qualité.

c. Améliorations à étudier

i. Migration vers Apache Spark

La migration de l'architecture en utilisant majoritairement Apache Spark reste une option future à étudier suivant l'évolution de ce produit. Le récent investissement financier colossal d'IBM dans ce projet open source montre clairement que ce projet va rapidement pouvoir palier a ses défauts et devenir une option pérenne d'architecture. Il faut rester en veille car une meilleure gestion de nombreux modèles en concurrence (et de leur mémoire), l'intégration d'autres librairies d'algorithmes en ligne, et de solides possibilités d'agrégations temporelles optimisées pour les flux sont des fonctionnalités qui arriveront certainement rapidement.

ii. Gestion de la connaissance : gestion collaborative, iPython Notebooks

Comme présenté plus haut dans ce rapport, un important travail a été réalisé pour acquérir la connaissance nécessaire et la structurer. Pour une équipe plus importante, ce processus n'est pas adapté et devra être modifié pour aussi profiter d'une véritable gestion collaborative de la connaissance. Il sera bien aussi de penser a profiter d'outils comme les iPython Notebooks qui facilitent le mélange de documentation, expériences, données et prototypes de code dans des documents très simples à réaliser, compatibles avec les gestionnaires de version comme GitHub et efficaces pour mémoriser et diffuser ce type de connaissance.

Conclusion

Cette expérience a été intense. Le machine learning est un sujet passionnant entre la théorie et la réflexion pratique des domaines qu'il peut aborder. Je me suis beaucoup investi dans le développement de mes connaissances en machine learning car les produits actuels ont chacun d'importantes limitations et bien comprendre la théorie m'a aidé à concevoir des solutions efficaces palliant ces limitations et répondant aux contraintes fortes du projet.

L'utilisation de mon expérience en gestion de projet dans une configuration plus R&D, dans un nouveau secteur, un nouveau domaine, une nouvelle entreprise m'a obligé à faire face à beaucoup de nouvelles problématiques. Je n'avais plus l'habitude de travailler seul ou en très petite équipe. Gérer un projet tout en effectuant de la R&D et du développement classique avec peu de ressources a été difficile et formateur car les rythmes de ces deux activités ne sont pas toujours facile à conjuguer.

Aujourd'hui j'ai envie de continuer à approfondir mes connaissances en intelligence artificielle appliquées aux données dans de nouveaux projets. Après mes années en conception de systèmes télématiques, j'imagine les nombreuses applications du machine learning à l'Internet des objets (IoT). Les opportunités ouvertes par l'intelligence artificielle sont aujourd'hui dans tous les domaines que ce soit pour la création de nouveaux logiciels, produits et services ou l'amélioration de ceux qui existent.

Bibliographie

Voir aussi « Détail des sources et vecteurs de formation »

Ressources "métier"

- Prédiction billets d'avion : Mining Airfare Data to Minimize Ticket Purchase Price - 2003 - Etzioni
- Prédiction pour yield management : Dynamic Pricing for Hotel Revenue Management Using Price Multipliers - 2013 - Abd El-Moniem Bayoumi, Mohamed Saleh, Amir Atiya, Heba Abdel Aziz, Amir Atiya
- Dynamic Pricing and Automated Ressource Allocation for Complex Information Services - 2007 - Goethe University, Michael Schwind
- Recommendation en contexte secteur hôtelier : www.slideshare.net/OssiMokryn/cold-start-context-aware-hotel-recommender-system
- Recommendation site web et e-commerce :
    - Evaluating Various Implicit Factors in E-commerce - 2012 - Ladislav Peska, Peter Vojtas
    - Toward a New Protocol to Evaluate Recommender Systems - Orange Labs, Frank Meyer, Françoise Fessant, Fabrice Clérot , Eric Gaussier
- Recommendation en contexte:
    - A survey of context-aware recommender systems - 2013 - Chuong C. Vo, Torab Torabi, Seng W. Loke : http://homepage.cs.latrobe.edu.au/ccvo/papers/16recommendation.pdf
- Machine learning et économétrie :
    - Machine Learning and Econometrics - Google, 2014
    - http://web.stanford.edu/class/ee380/Abstracts/140129-slides-Machine-Learning-and-Econometrics.pdf

Liens techniques principaux

- Répertoire exhaustif des librairies, logiciels , framework du machine learning : https://github.com/josephmisiti/awesome-machine-learning
- Bibliothèques d'algorithmes de Machine learning en flux avec gestion du concept drift (évolution des modèles et mutation des données) - Université de Waikato : http://moa.cms.waikato.ac.nz/details/
- Architecture distribuée pour Machine learning en flux - Yahoo et université de Waikato : http://samoa-project.net/
- Bibliothèques et framework pour systèmes de recommandation - Université du Minnesota: http://lenskit.org
- Bibliothèque C++ de machine learning online out-of-core très rapide: https://github.com/JohnLangford/vowpal_wabbit
- Étude pour service de création de modèles optimaux sous contraintes de temps et de ressources - Université de Berkeley : http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-122.pdf

- Bibliothèques d'algorithmes optimisés pour le traitement de graphes en ressources limitées - projet de l'Université Carnegie Mellon NY repris par GraphLab: http://graphlab.org/projects/graphchi.html
- Recherche parallélisée d'optimisation des hyperparamètres d'algorithme - projet Open Source, post doc en deep learning : https://github.com/hyperopt/hyperopt/wiki
- Stockage des modèles en grille - projet Open Source : http://infinispan.org

Divers Livres

- Bayesian Reasoning and Machine Learning - David Barber
- Artificial Intelligence, a modern approach - 3rd Edition - [Russell, Norvig] (2010)
- Data Mining- Practical Machine Learning Tools and Techniques, Third Edition (Weka) (2011)
- Artificial Intelligence For Games - Ian Millington, John Funge Morgan Kaufmann (2009)
- Data Mining et Statistique décisionnelle.
- Machine Learning, a Probabilistic Perspective - Kevin P. Murphy (2012)
- QuantStart, Algo Trading.
- Programming Collective Intelligence - Building Smart Web 2.0 - Toby Segaran (2007)
- Adaptive Stream Mining Pattern Learnin - [Albert Bifet] (2010)
- MOA - StreamMining - [Albert Bifet] (2011)
- Dynamic Pricing and Automated Resource Allocation for Complex Information Services (Reinforcement Learning and Combinatorial Auctions) - [Michael Schwind] (2007)
- Handbook of Metaheuristics - Springer (2010)
- Metaheuristics for Dynamic Optimization
- Python for Data Analysis
- Recommender Systems Handbook (2011)
- Learning Storm (2014)

Papiers & Thèses

- A Classification Framework For Imbalanced Data - [Piyaphol Phoungphol]
- Learning to Classify Data Streams with Imbalanced Class Distributions - [Ryan N. Lichtenwalter, Nitesh V. Chawla]
- Learning from streaming data with concept drift and imbalance - [T. Ryan Hoens · Robi Polikar · Nitesh V. Chawla]
- Automatic Selection of Machine Learning Models for Compiler Heuristic Generation - [Paul Lokuciejewski, Marco Stolpe, Katharina Morik, Peter Marwedel]
- Online Bagging and Boosting for Imbalanced Data Streams - [Boyu Wang and Joelle Pineau]
- streamMOA: Interface to Algorithms from MOA for stream - [Matthew Bola, John Forrest]
- Active Learning with Drifting Streaming Data - [Indre Zliobaite, Albert Bifet, Bernhard Pfahringer, and Geoff Holmes]
- Temporal Aggregation over Data Streams using Multiple Granularities - [Donghui Zhang, Dimitrios Gunopulos, Vassilis Tsotras, Bernhard Seeger]
- Distributed Decision Tree Learning for Mining Big Data Streams - Arinto Murdopo

- Recommandation Locale Et Sociale Dans Les Architectures Décentralisées [Simon Eyffret M]

Divers

Cycle de vie d'un projet Data Mining ou Machine Learning :

datasciencecentral.com/profiles/blogs/life-cycle-of-data-science-projects
en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining
the-modeling-agency.com/crisp-dm.pdf

Livre blanc sur le "big data" : http://big-project.eu/sites/default/files/BIG_D2_2_2.pdf

Annexes

Architecture du service de recommandation de Netflix

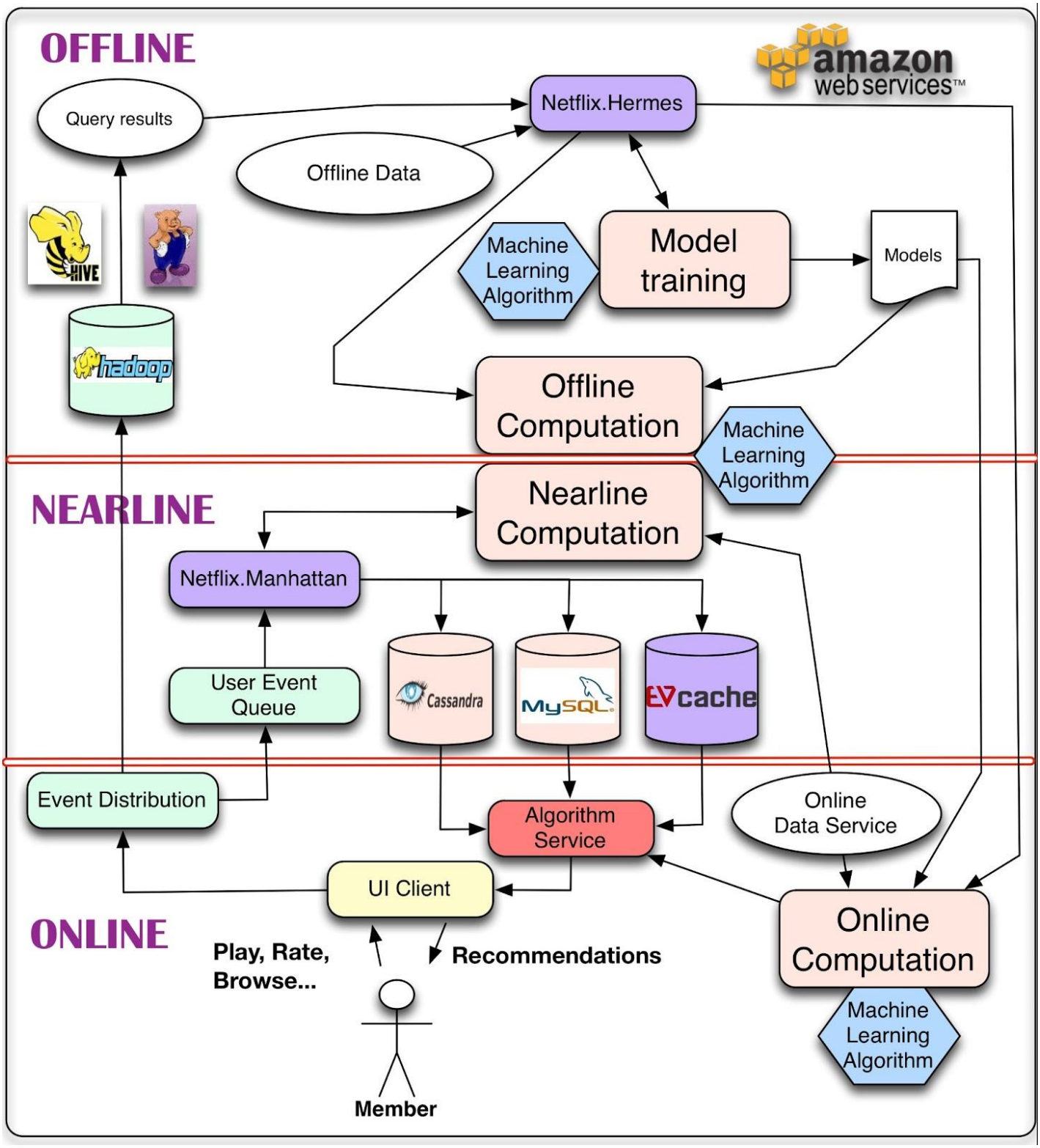
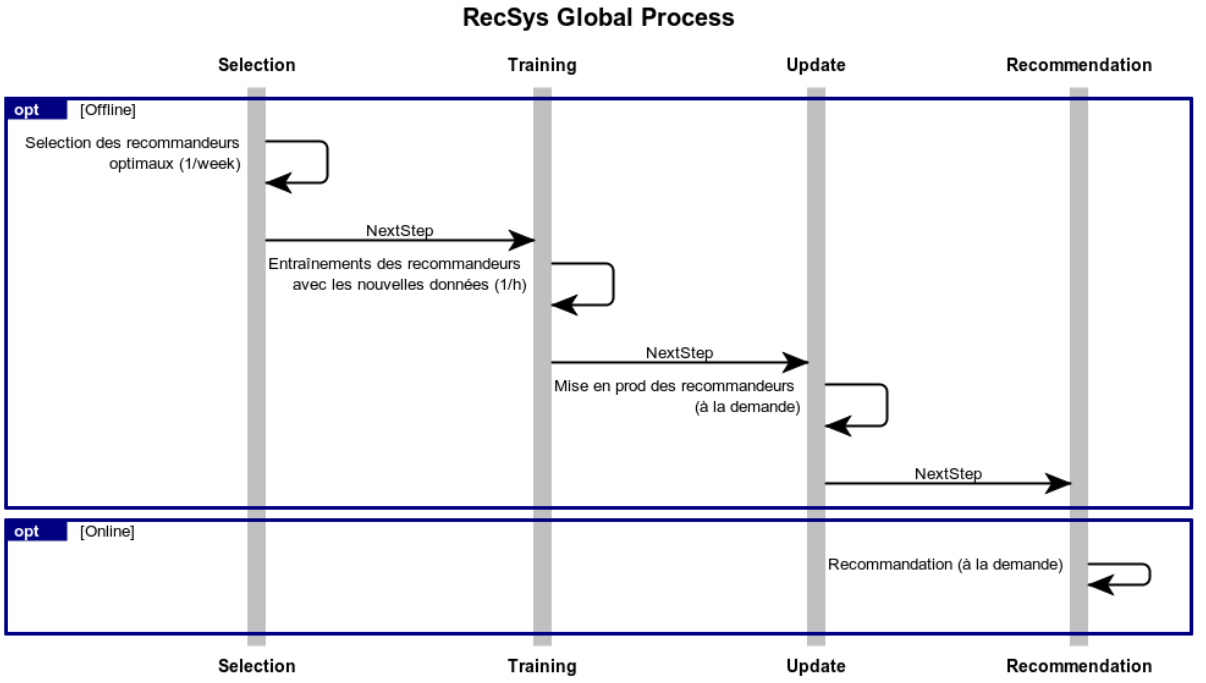


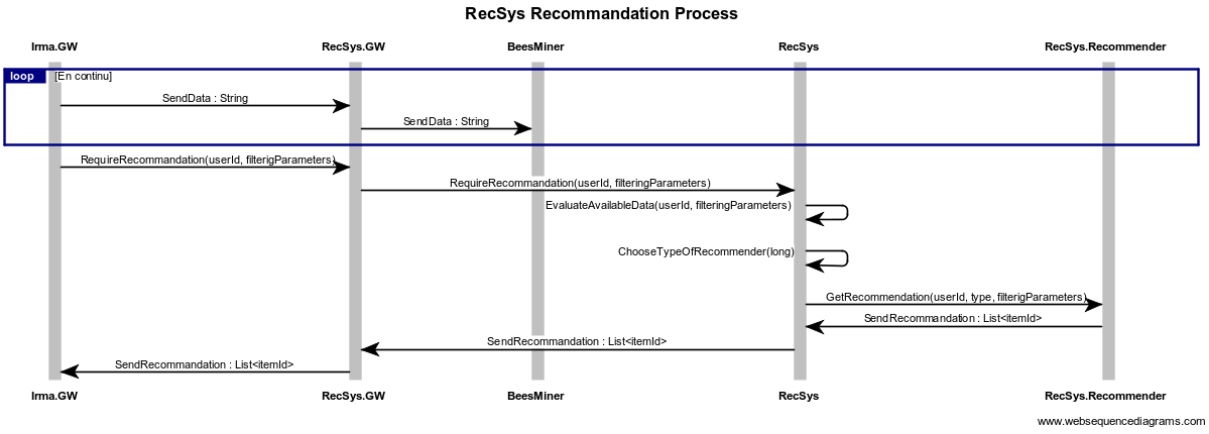
Figure 28 : Architecture du service de recommandation de Netflix

Fonctionnement du moteur le recommandation basé sur « Lenskit » RecSys

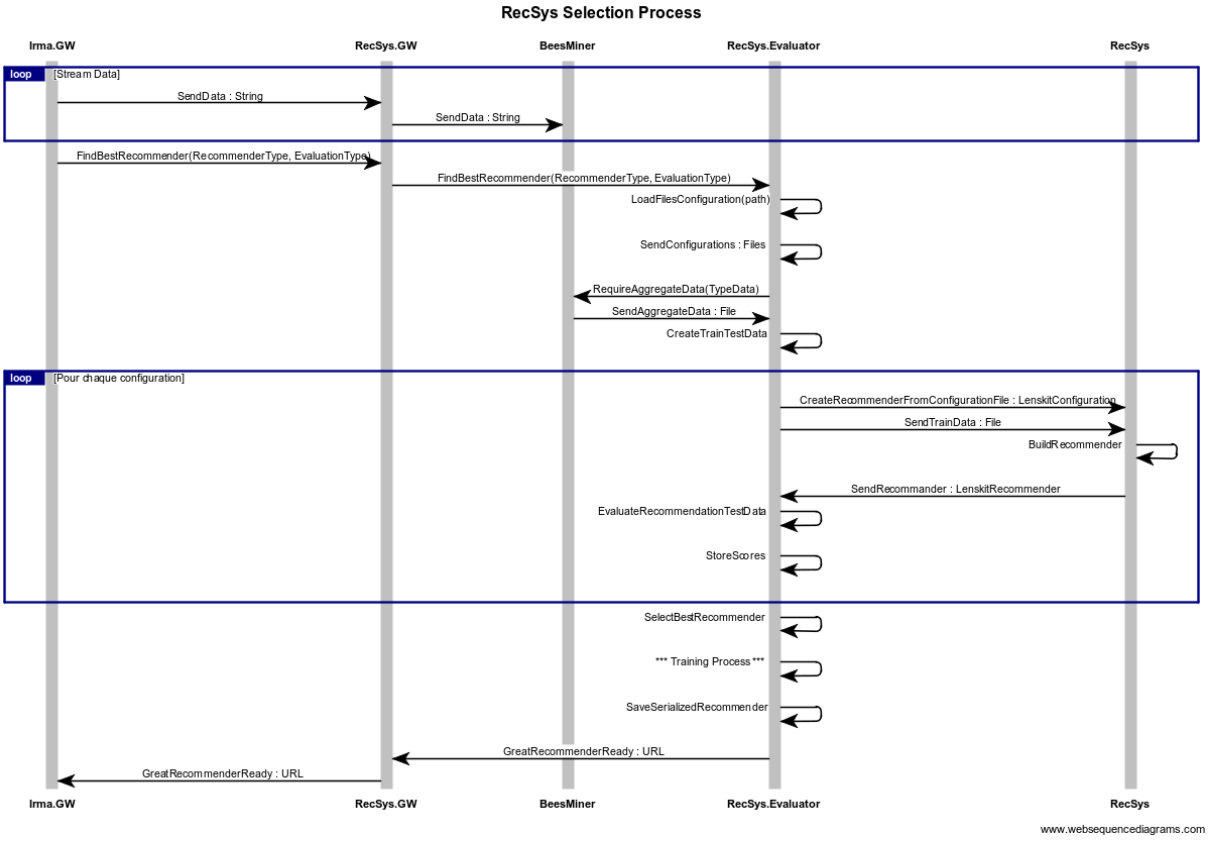
Processus global



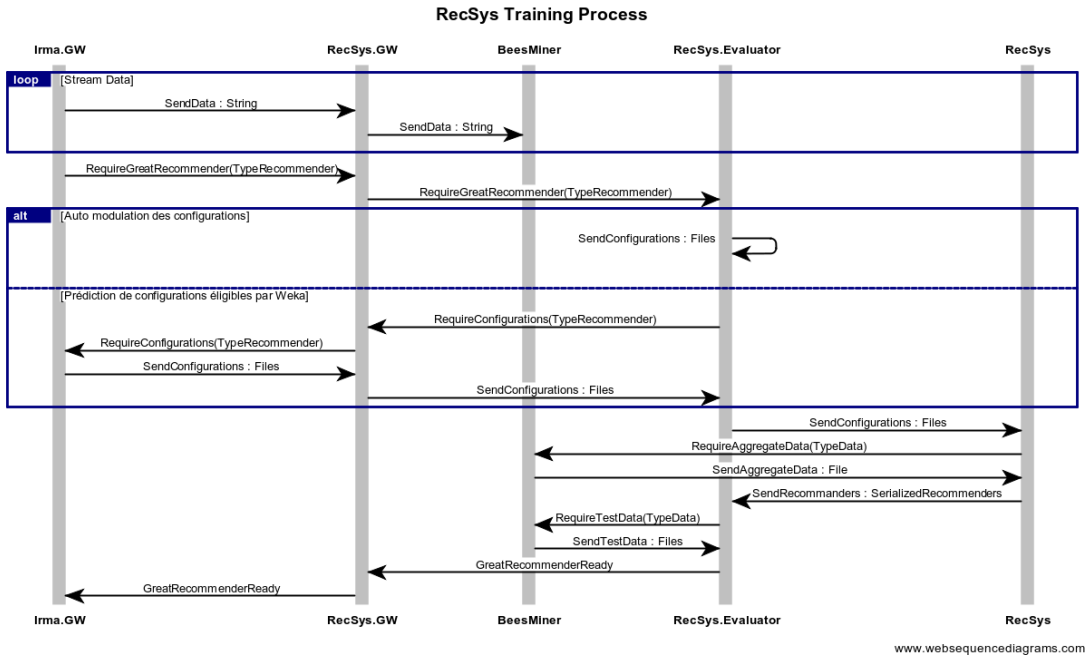
Processus de « Recommandation »



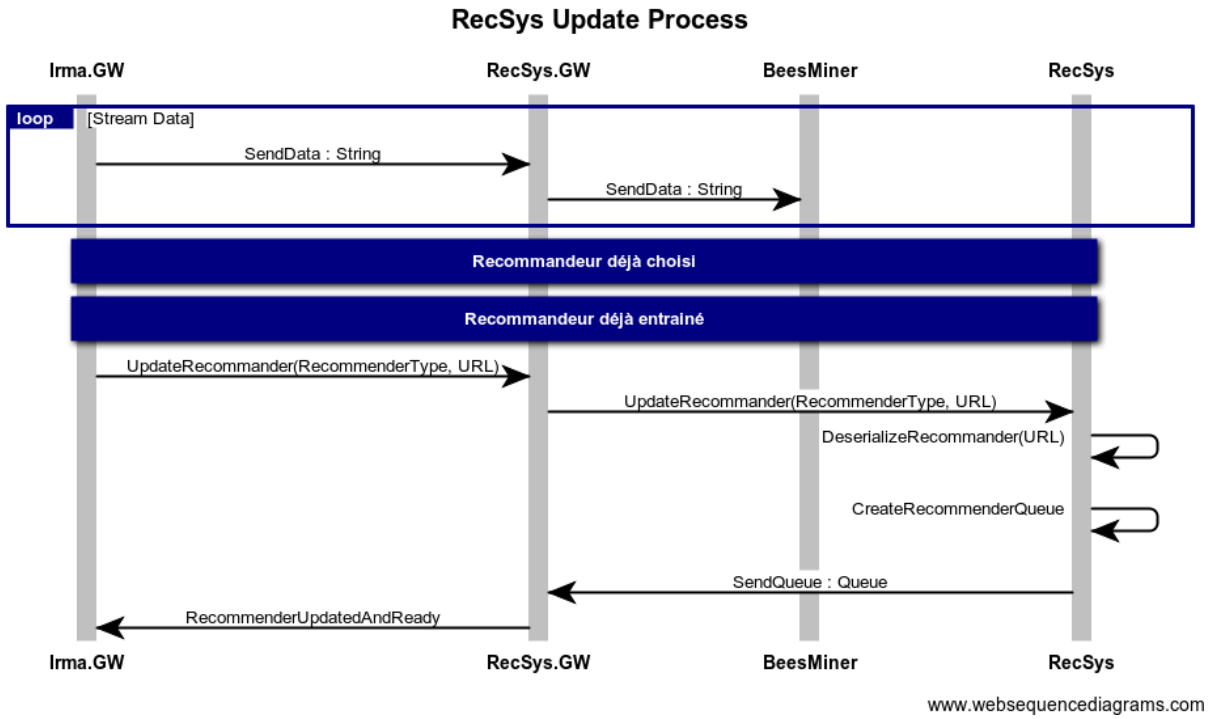
Processus de sélection optimale du modèle



Processus d' « entraînement du modèle »



Processus de « mise à jour du modèle »



Détail des sources et vecteurs de formation

- Cours de Mathématique avec un professeur d'université spécialisé en probabilités et statistiques.

- Suivi de différents MOOC :
    - Stanford “Machine Learning” : très bonne base et bonne pédagogie pour avoir une vue d’ensemble du machine learning et de ses bases mathématiques et algorithmiques.
    - Indian Institute of Technology “Intelligence Web et Big data” : bonne base pour avoir une vue d’ensemble des architectures big data pour le web.
    - Pennsylvania University “Recommenders Systems” : très bonne base et bonne pédagogie pour découvrir les système de recommandation bien illustré avec la librairie Java Lenskit.
    - Waikato University “Data Mining with Weka” and “Advanced Data Mining with Weka” : très bonne base et bonne pédagogie pour avoir une vue d’ensemble du data mining et du machine learning. Il offre un panorama beaucoup plus complet sur les algorithmes de machine learning que dans le MOOC de Stanford mais pas avec de vraies démonstrations mathématiques et algorithmiques.
    - Université de Sherbrooke “Apprentissage Automatique” : très bonne base et éléments avancés du machine learning. Moins pédagogique que les autres MOOC de machine learning mais c’est en français.
    - Columbia University “Big Data in Eduction”. Je l’ai arrêté car pas très pertinent.

- Lecture de différents livres de référence conseillés sur les forums spécialisés :
    - Machine Learning - Peter Flach, Cambridge : très bon panorama du machine learning et explication des différents algorithmes tout en gardant une bonne approche pédagogique.
    - Data Mining et Statistique décisionnelle : une référence en français sur les statistiques, le data mining, le machine learning (mais n’utilise pas les récentes avancées dans le domaine ni n’aborde l’apprentissage en flux ou en ligne).
    - Bayesian Reasoning and Machine Learning - David Barber : très bon investissement pour le futur du machine learning.
    - QuantStart, Algo Trading : par association, je l’ai utilisé pour approfondir les bonnes pratiques utilisées pour valider des stratégies autour de prédictions.
    - Mastering Scikit-learn : beaucoup trop simplifié.

- Abonnements à des flux RSS pour la veille dans le domaine et pour développer une culture plus long terme :
    - FastML : revue généraliste sur le domaine du machine learning.

    - Vowpal Wabbit : groupe sur la librairie novatrice en termes de performance et d'architecture. J'en suis devenu modérateur.
    - MachineLearning : revue universitaire sur le domaine du machine learning
    - MOA et SAMOA : groupe sur les librairies dédiées au machine learning en ligne intégrant fédérant de nombreuses recherches universitaires.

- Lecture de thèses et articles sur le machine learning appliqué au :
    - Yield management
    - L'hôtellerie ou des secteurs similaires
    - Analyse e-commerce
    - La recherche et l'optimisation automatique de modèles.

- Réalisation systématique de mini projets sur les librairies de machine learning pour les comprendre et les comparer :
    - Shark, R, Mahoot, Scikit, Weka, Vowpal Wabbit, Torch, Caffé, Theano, Orange, MOA / SAMOA, Panda, Numpy/Scipy

- Pratique et amélioration des connaissances :
    - **Kaggle.com** est la référence pour pratiquer le machine learning autour de challenges et progresser.
    - De nombreux tutoriaux et expériences sont diffusés sous le format iPython Notebooks qui sont des documents mélangeant explication et code intégré qui peut être exécuté et modifié de manière interactive.

- Conférence : PAPIs.io 2014 à Barcelone

- Sites de Questions / Réponses auprès de spécialistes
    - stackoverflow.com
    - stats.stackexchange.com (Cross-Validated)

- Développement d'un réseau en devenant modérateur, contributeur, membre :
    - je suis devenu modérateur du forum de Vowpal Wabbit
    - j'ai noué des échanges avec des personnes clés de : l'université de Waikato, Yahoo Barcelone R&D, l'université du Minnesota, des chercheurs de la Melon University NYC, l'organisateur de la première conférence sur les API de machine learning.

- Twitter :
    - je suis des personnalités connues ou des journaux de référence.

- Sous-traitance auprès de spécialistes :
    - j'ai fait appel à un spécialiste pour l'analyse de l'architecture lambda utilisant Storm ou Spark, je l'ai découvert via son activité sur GitHub.

Extrait charte de projet (1ère page)

Rezzza SAS - www.verylastroom.com

Project Charter

-

Projet Irma 1.0

Naissance

Nom: Project Charter
Propriété de: Rezzza SAS (VeryLastRoom.com)

Auteur: Xavier Daull
Contributeurs: Sébastien Houzé, Nicolas Salin

Version document: 1.1
Diffusion: CONFIDENTIEL, CEO, CTO, Chef projet

Créé le: 08/01/2014
Dernière mise à jour le: 13/01/2014

Introduction

Le projet "Irma 1.0 - Naissance" a pour objectif de créer un service de prédiction (type Machine Learning) et de recommandation en temps réel offrant différentes informations stratégiques d'anticipation, d'optimisation pour le back-office et autres applications de VeryLastRoom.

Le projet de prédiction Irma 1.0 exploitera de nombreuses données de la concurrence issues du projet BeeSpy2 et fournira notamment au back-office des informations d'anticipation et d'optimisation.

Sommaire

Extrait du « Process Management Plan » (1ère page)

Project Management Plan

-

Projet Irma 1.0

Naissance

Nom: Project Management Plan
Propriété de: Rezzza SAS (VeryLastRoom.com)
Auteur: Xavier Daull
Contributeurs: Sébastien Houzé
Version document: 1.0
Diffusion: CONFIDENTIEL, CEO, CTO, Chef projet
Créé le: 13/01/2014
Dernière mise à jour le: 15/01/2014

SOMMAIRE

Liste des tableaux et figures

Note : nombreuses de ces illustrations viennent de ressources sur Internet. Les sources n'ont pas été indiquées mais vous pouvez les retrouver via une recherche par image sur le service images.google.com (il utilise un bel algorithme de machine learning !)