Université de Montréal

Faculté des arts et des sciences

Département d'informatique et de recherche opérationnelle (DIRO)

Rapport de stage
Évaluation du risque de retour à la maison

Auteur :
Vilon Saint-Fleurose
MSc, informatique
Université de Montréal

Directeur de recherche :
Dr. Michalis Famelis
Professeur adjoint
Université de Montréal

Superviseur :
Nicolas Coallier
Vice-Président Exécutif,
TIC
ML+

3 octobre 2018

Université de Montréal

Table des matières

Remerciements

Je veux commencer d'abord par remercier le Grand Dieu Tout-Puissant, le Créateur de l'univers, des cieux et de la terre qui m'a donné la vie, la santé, les opportunités et tout ce dont j'avais besoin pour faire cette grande et belle étude à l'université de Montréal. Il a rendu toutes choses possibles en ma faveur, moi qui suis pécheur et désobéissant ; immérité de toutes ces grâces. Il m'accompagnait toujours dans les moments les plus difficiles de ma vie, Il ne m'a jamais laissé seul ; surtout dans les moments où je devais payer les frais de scolarité qui étaient si énormes et impossibles à payer de mon propre compte. A un Dieu si merveilleux et si bon, je Lui dois beaucoup de reconnaissance.

Je remercie aussi ma femme qui m'a beaucoup supporté pendant plus de deux années d'études. Elle n'a jamais murmuré, ni découragé quand nous devions passer par des moments difficiles de notre vie conjugale à cause de ces études. Elle a mis toutes ses ressources disponibles pour entretenir la famille et payer mes études quand j'étais moi-même dans l'impossibilité de travailler. Vraiment, ma femme est une bénédiction dans ma vie, un cadeau venant de Dieu. Je t'aime ma chérie.

Je tiens à remercier le professeur Michalis Famelis d'avoir accepté être mon directeur de recherche et supervisé ce stage, il est toujours là pour m'encourager et me pousser vers l'avant. Il répond toujours présent à tous mes appels, il est toujours disponible pour me rencontrer, me parler et me conseiller ; même en dehors du cadre universitaire.

Je remercie Nicolas Coallier et toute l'équipe ML+ qui ont accepté que je sois leur stagiaire, ils ont placé leur confiance en moi quoiqu'ils ne me connaissaient pas encore. Cette équipe, quoique jeune, est très dynamique et chaleureuse, c'est une équipe motivante qui stimule la connaissance. J'ai dû apprendre beaucoup de choses par rapport à eux. Finalement, je présente mes sincères remerciements à toute la communauté universitaire, à DIRO en particulier. Merci pour la formation prestigieuse que vous m'avez fournie. Cette formation est si solide qu'elle m'aidera rapidement à intégrer le marché du travail sans perdre de temps.

Préface

Nous développons ce rapport en vue d'achever notre étude de maîtrise en informatique faite à l'université de Montréal. Nous avons inscrit notre maîtrise avec option stage, ce qui exige la redaction d'un rapport au terme de ces études. Nous avons, d'abord réalisé notre stage chez ML+, une entreprise située à l'adresse suivante : 338, rue St-Antoine est, Montréal. C'est une jeune entreprise dynamique et spécialisée en science de données. Elle développe des outils pour aider les entreprises à prendre leurs décisions stratégiques tout se basant sur leurs données historiques.

C'est une expérience enrichissante d'avoir eu l'opportunité de faire ce stage dans cette entreprise, j'ai pu apprendre beaucoup de choses tant sur l'apprentissage automatique que sur les sciences de données en général. Ces connaissances acquises m'ont ouvert la porte pour pouvoir rédiger ce rapport. Ce dernier permet d'évaluer le risque d'un patient de retourner chez lui après une opération contre l'arthrose (douleur ressentie au niveau des os).

Table des figures

Liste des tableaux

Résumé

L'arthroplastie totale de la hanche et du genou permet à diminuer considérablement la douleur et améliore la fonction chez les personnes atteintes d'arthrose avancée. Selon une étude publiée en 2012 dans le journal ISRN Orthopedics, les auteurs Bronislava Bashinskaya et al. [2] ont constaté que : «Le vieillissement de la génération du "baby boom", combiné au désir de maintenir un mode de vie actif et sans douleur, entraînera une augmentation du nombre annuel de chirurgies de remplacement articulaire pratiquées aux États-Unis». Dans un rapport publié en Avril 2007 par les orthopédistes Steven M Kurtz, Kevin Ong, Edmund Lau et Michael T Halpern [18], membres de l'Académie américaine des chirurgiens orthopédiques, ils ont prédit que : «D'ici 2030, la demande d'arthroplasties totales de la hanche totale devrait augmenter de 174% pour atteindre 572 000. La demande pour les arthroplasties totales du genou primaire devrait augmenter de 673% à 3,48 millions de procédures ». Face à l'augmentation continue du nombre de chirurgies de remplacement, il devient crucial aux administrateurs des hôpitaux de déterminer si les patients doivent rester à l'hôpital pour une réadaptation après leur chirurgie ou s'ils doivent être renvoyés à la maison puisque les coûts associés à une réadaptation doivent être pris en charge par les hôpitaux. Il est important de souligner que la majorité des compagnies d'assurance santé aux États-Unis n'assument pas les coûts liés à la réadaptation des patients. Ces frais qui sont très élevés (entre 15,000 et 30,000 $ par patient) doivent être assurés par les hôpitaux. Pour prendre la décision de retour ou non, les chirurgiens tiennent compte des antécédents médicaux du patient et l'état de santé du patient après l'arthroplastie, ce qui est un travail difficile à effectuer manuellement puisque le nombre de patients dans une base de données orthopédiques sont très nombreux. Dans notre projet de stage, nous avons développé un modèle d'apprentissage automatique qui aide à prendre cette décision de façon automatique. Nous avons aussi construit un web API qui fait la connexion avec le modèle, ce qui est utile pour les orthopédistes qui n'ont pas de connaissances en ML. On a conçu une interface utilisateur qui rend notre système convivial et facile à utiliser.

1 | Introduction générale

1.1 Contexte

Le système de santé américain est contrôlé en majeure partie par des compagnies privées, le gouvernement américain ne finance qu'une partie minoritaire de la population. Ce système repose sur deux sources de financement. D'abord on a le financement public géré par le programme fédéral Medicare qui vise seulement les personnes âgées de plus de 65 ans et celles qui sont gravement handicapées. D'après Europusa [1], une compagnie européenne spécialisée dans l'accompagnement des européens qui veulent immigrer aux États-Unis : «Cette source de financement couvre 26% de la population américaine dont 15 % sont des personnes agées et handicapées et 11% sont des familles pauvres avec enfants. ». La deuxième source de financement qui est un financement privé touche tout le reste de la population soit 74%. L'assurance est donc majoritairement privée aux Etats-Unis. Les Américains sont assurés en général via leurs employeurs ou sinon de manière individuelle lorsque leur employeur ne propose pas d'assurance ou qu'ils travaillent comme travailleurs autonomes. La composante « assurance médicale » dans le choix d'un emploi est donc un critère important. En général le système de santé américain est échoué parce qu'une grande partie de la population se retrouve sans assurance santé. C'est un système libéral fondé sur le principe du marché et de la concurrence qui s'organise autour d'assurances privées souvent liées à l'emploi et d'une assurance maladie obligatoire.

Les compagnies d'assurance santé américaines ne couvrent pas les coûts associés à une réadaptation d'un patient après une arthroplastie [2]. Il revient à l'hôpital de couvrir ces coûts qui sont très élevés. Selon le docteur orthopédiste Jonah Hebert-

1. Site Europusa : https ://www.europusa.com/assurance/pourquoi-et-comment-choisir-une-bonne-assurance-sante/comment-fonctionne-le-systeme-de-sante-et-les-assurances-aux-etats-unis/

2. Opération ayant pour but de rétablir la forme et la mobilité d'une articulation abîmée ou bloquée.

Davies[3], un de nos collaborateurs au stage, qui a fait ses études de spécialité en Orthopédie à l'université de Seattle, Washington aux États-Unis, les coûts estimés sont entre 15,000 $ et 30,000 $ par patient. Ce qui revient à une grande perte financière pour les hôpitaux. La décision de maintenir ou de renvoyer un patient après une arthroplastie devient une question préoccupante dans la mesure où elle pourrait réduire le coût budgétaire des hôpitaux. Si la décision prise est de renvoyer le patient, l'hôpital est le gagnant sinon il est le perdant.

D'autres part, l'arthrose[4] touche environ 27 millions d'adultes aux États-Unis [21]. Ce qui augmente considérablement le nombre d'arthroplastie dans les hôpitaux, et aussi le poids de travail des médecins à savoir quand est-ce qu'il ya une réadaptation ou non.

Il est un travail fastidieux pour les spécialistes orthopédistes de vérifier manuellement les données médicales d'un patient pour pouvoir décider s'il doit être renvoyé ou non. Ce travail consiste en ce que les données médicales d'un dossier patient sont très volumineuses. Seulement pour déterminer les conditions médicales antécédentes du patient, quarante-six colonnes pré-opératoires (conditions de santé du patient avant l'opération) sont prises en compte. Plus tard nous verrons tous les calculs qu'il faut faire avant de prendre cette décision. Dans notre travail nous avons créé des processus automatisés pour faire toutes ces vérifications.

Considérant le nombre grandissant d'arthroplastie dans les hôpitaux américains et tous les travaux manuels qu'un orthopédiste doit réaliser avant de décider le retour ou non d'un patient à la maison, nous estimons que créer un outil pour automatiser cette tâche est un projet pertinent. Le but de notre stage consiste à développer cet outil qui aidera les orthopédistes américains à prendre cette décision de façon automatique et qui les évite à dépenser tout leur temps à analyser un dossier patient parmi plusieurs milliers de dossiers. Dans un premier temps nous avons construit un modèle ML et ensuite nous avons conçu un web API pour faire la connexion avec le modèle. Un interface utilisateur web a été implémenté pour rendre que notre système soit convivial à utiliser.

3. Pour informations sur le docteur, consultez http ://www.orthop.washington.edu/ ?q=faculty-profiles/jonah-hebert-davies-md-frcsc.html

4. Maladie qui touche les articulations et caractérisée par la douleur et la difficulté à effectuer des mouvements articulaires

1.2 Ingénierie des besoins

Nous supposons qu'il existe déjà un jeu de données disponible contenant toutes les informations touchant un grand nombre de patients et que chaque échantillon de ce jeu de données est un dossier patient correct faisant l'objet d'une décision de retour. Nous supposons aussi que ce jeu de données possède des caractéristiques suffisantes pour décider si un patient doit être renvoyé ou non. Nous considérons aussi que ce jeu de données est constitué de patients qui ont subi une décision de retour dans le passé. Si toutes ces conditions sont réunies, nous pouvons profiter des différentes technologies de la science de données, de l'apprentissage automatique et l'ingénierie de logiciels pour développer un outil d'aide à la décision qui aide les orthopédistes à prendre la décision de retour automatiquement.

Les technologies de la science de données sont conçues pour développer des outils qui permettent de prendre des décisions à partir d'un ensemble de données existantes. L'apprentissage automatique est un sous-ensemble de la science de données qui permet de modéliser à partir d'un ensemble de données. Dans le chapitre 2, nous passons en revue le cycle de travail complet du scientifique de données. Ce que nous proposons comme solution dans notre projet. Nous avons suivi ce cycle complet pour implémenter notre système d'aide à la décision.

Il y a deux grandes phases impliquées dans le processus de développement de notre projet : la première phase est la construction d'un modèle d'apprentissage automatique et la deuxième est le déploiement du modèle construit. Pour construire ce modèle nous avons utilisé plusieurs techniques d'apprentissage automatique telles que le re-échantillonnage et l'optimisation. La deuxième phase consiste en la création d'un API nommé Ortho, c'est surtout une phase d'ingénierie de logiciels. Afin de modéliser les deux phases du développement de notre système et permettre de visualiser sa conception à très haut d'abstraction nous avons crée deux diagrammes UML [32] (Unified Modeling Language). Le premier est un diagramme de cas d'utilisation qui donne une vision globale du comportement fonctionnel de notre système et le deuxième est un diagramme d'activités qui représente l'état d'exécution de notre modèle ML. Nous donnons une brève explication de chaque diagramme et leurs différents composants. Pour plus de commodité dans nos schémas, nous choisissons d'écrire les textes en anglais car les textes français sont souvent beaucoup plus longs pour exprimer les idées. Mais dans toutes les explications nous prenons le soin de donner une traduction française pour faciliter la compréhension.

1.2.1 Notre diagramme de cas d'utilisation

Un cas d'utilisation est une situation où un système est utilisé pour décrire les besoins des utilisateurs. On spécifie seulement ce que le système est supposé de faire c'est-à-dire les besoins fonctionnels du système. A ce stade, on évite de spécifier ce que le système ne doit pas faire. Le diagramme des cas d'utilisation est un bon point de départ à propos de toutes les facettes du développement, de la conception, des tests et de la documentation des systèmes orientés objet. Parce qu'il décrit strictement les besoins du système d'un point de vue extérieur ; il spécifie les valeurs que le système offre aux utilisateurs [22]. Notre diagramme de cas d'utilisation décrit la phase de déploiement de notre modèle ML. Ce diagramme montre les différentes fonctionnalités notre système que les orthopédistes américains auront accès. la figure 1.1 donne une vue schématique de notre diagramme ; chaque cas représente une unité discrète d'interaction entre un utilisateur (Orthopédiste ou administrateur) et notre système. Voici une brève explication des acteurs et de chaque cas :

1. **Actors.** Les acteurs sont des orthopédistes et les administrateurs du système qui sont des orthopédistes ayant des droits et privilèges d'évaluer, de suivre et de gérer le modèle déployé ;
2. **Predict.** Les orthopédistes peuvent utiliser cette section en vue de prédire si les patients doivent rester à l'hôpital ou retourner chez eux. Pour accomplir cette tâche, il est obligatoire de remplir un formulaire.
3. **Fill form.** Pour prédire, il faut remplir le formulaire qui contient les informations du dossier médical du patient. On recueille les informations pertinentes pour la prise de décision telles que les conditions médicales, les diagnostiques antérieures, l'indice de masse corporelle (BMI en anglais) etc.
4. **Evaluate.** Permet d'évaluer le modèle sur de nouvelles données prédites. On peut comparer l'évaluation faite au moment de la construction du modèle et celle faite au moment de la prédiction. L'évaluation exige une authentification.
5. **Monitor.** Permet de suivre périodiquement le modèle pour déterminer si les performances s'améliorent ou si elles détériorent. Exige une authentification.
6. **Manage.** Permet d'optimiser le modèle sur de nouvelles données en essayant d'utiliser de nouveaux hyper-paramètres en vue de comparer les résultats. Exige une authentification.
7. **Authenticate.** Permet l'authentification du système. Exige un nom utilisateur et un mot de passe.

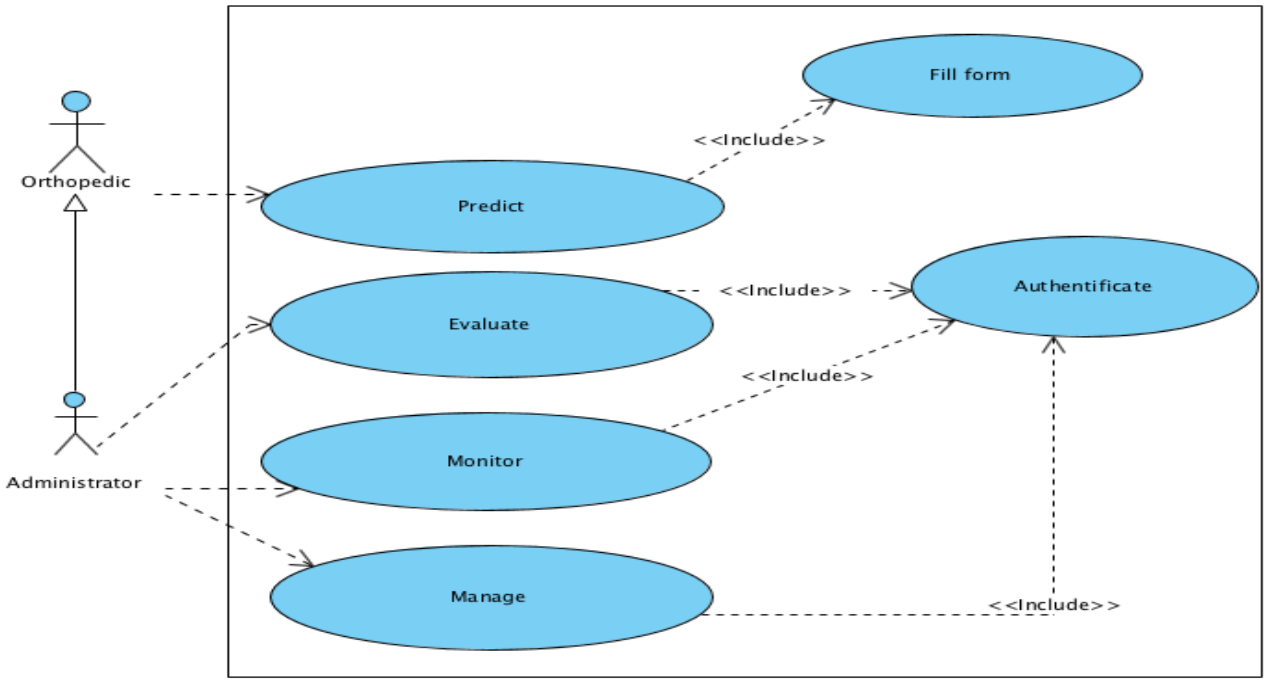

Fig. 1.1 – Diagramme cas d'utilisation de notre système

1.2.2 Notre diagramme d'activités

Un diagramme d'activités est une variante du diagramme d'états-transitions qui montre le comportement discret d'une partie d'un système conçu par des étapes regroupées séquentiellement dans des flots de contrôle. Une transition, dessinée comme une flèche qui connecte les états, représente un changement d'état ou comment passer d'un état source à un état cible. Un état peut être actif ou inactif. Il est actif quand une transition est entrée et devient inactif quand une transition sort [22]. Dans notre diagramme d'activités (voir figure 1.2), nous avons défini plusieurs états et transitions lesquels décrivent les différents comportements de notre système d'apprentissage automatique. Voici un bref résumé de ce que fait chaque état de ce diagramme :

1. **Initial state.** C'est le point de départ du système. Il annonce la construction d'un modèle ML.
2. **Getting historical data.** Permet de récupérer les données à partir d'un fichier d'extension *csv* (comma-separated values). Les données historiques sont un jeu de données disponible avec des valeurs correctes pour entrainer les modèles ;
3. **Explore raw data.** Permet d'explorer les données afin de prendre connaissance de l'ensemble de données. A cet état nous pouvons entre autres voir les

données impertinentes, différentes corrélations qui existent entre les variables et l'étiquette ;

4. **Cleaning data.** Dans cet état nous nettoyons nos données en imputant les données nulles et supprimant les variables que ne sont pas pertinentes ;
5. **Transformation data.** Création de nouvelles variables en fonction de celles qui existent déjà dans l'ensemble des données.
6. **Modeling data.** C'est un état au cours duquel on fait d'appliquer les algorithmes d'apprentissage automatique sur un ensemble de données bien préparé. À ce stade, on définit les quatre sous-états suivants :
   (a) **Training data.** C'est la phase d'entrainement de modèles pour déterminer lequel est meilleur pour notre ensemble de données ;
   (b) **Choosing model.** Sélectionner le meilleur modèle après une évaluation de chaque modèle entrainé ;
   (c) **Prediction.** Utilisation du meilleur modèle sur un ensemble de données de test ;
   (d) **Serialize.** Sauvegarde le meilleur modèle pour son utilisation future.
7. **État final.** Le modèle construit est sérialisé pour le système de déploiement.

La suite de ce rapport est structuré de la façon suivante : Le chapitre 2 donne un contexte théorique des différents concepts de la science de données et de l'apprentissage automatique qui sont importants pour notre projet. Le chapitre 3 décrit en détail la construction de notre modèle d'apprentissage automatique. Le chapitre 4 décrit la création de notre API *Ortho* qui fait la connexion avec le modèle d'apprentissage automatique. Le chapitre 5 montre comment nous évaluons les différents morceaux de notre système. Le chapitre 6 détaille les travaux connexes liés au développement d'API d'apprentissage automatique, la technique de re-échantillonnage, aux algorithmes ensemblistes etc. Le chapitre 7 fait la conclusion générale de notre rapport.

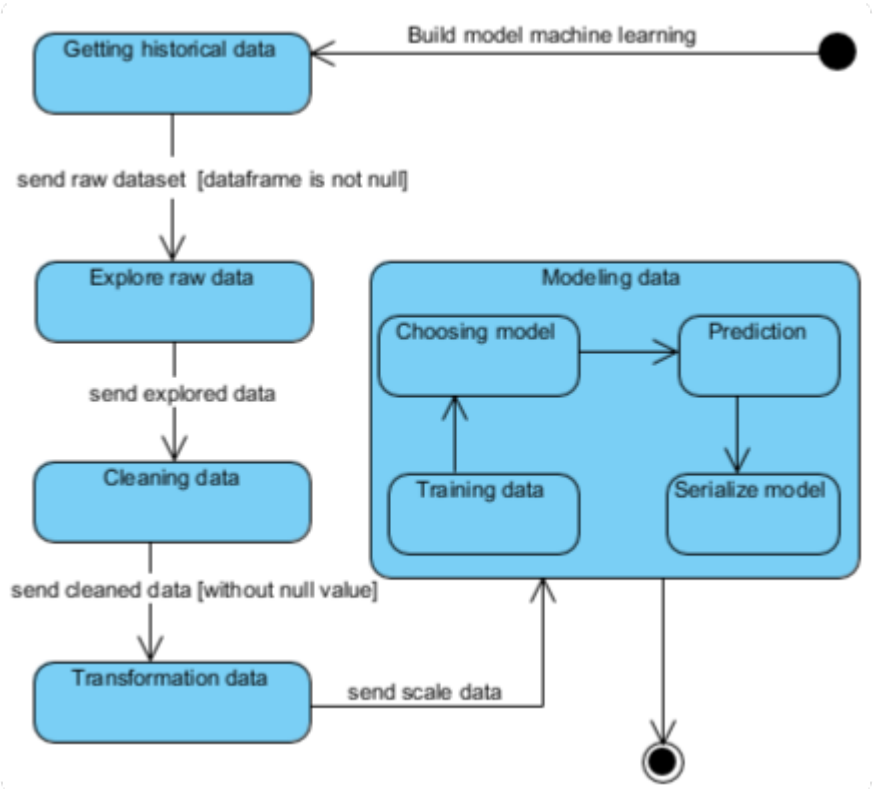

Fig. 1.2 – Diagramme d'activités de notre système ML

2 | Contexte théorique

2.1 Introduction

Tout projet informatique suppose qu'on doit délivrer un produit fini et utilisable accomplissant un certain nombre d'objectifs. Il y a toujours une équipe qui travaille durement pour réaliser ou implémenter ce genre de projets. Tout développement informatique passe nécessairement par la pratique ce qui exige souvent l'application de beaucoup de concepts étudiés en théorie. Dans tout rapport portant sur la réalisation d'un projet scientifique et technique, il s'avère utile et important d'établir un contexte théorique dans le but de faire ressortir les différents cadres théoriques. Nous allons, dans ce chapitre, présenter quelques concepts théoriques importants abordés dans le cadre de notre projet.

Généralement, nous mettons l'accent sur un sous-ensemble de tous les concepts jugés pertinents pour la réalisation de notre projet. Par exemple, l'apprentissage automatique est un champ de recherche informatique de plusieurs décennies de travaux de recherche, il est impossible de l'aborder dans un seul chapitre. Cependant nous focalisons sur les éléments les plus essentiels pour notre projet.

La suite de ce chapitre se subdivise comme suit : la deuxième section est une approche globale sur les sciences de données. Nous estimons que c'est important parce qu'aucun modèle d'apprentissage automatique ne peut être construit sans d'abord réaliser une phase d'exploration de données ; or les sciences de données peuvent être définies comme étant une combinaison entre l'exploration de données et la modélisation de données. La troisième section se porte sur l'apprentissage automatique qui est le cœur même de notre projet ; comme on avait précisé au paragraphe antérieur, nous considérons seulement un sous-ensemble de concepts jugés utiles pour notre projet. La quatrième section traite du problème de classification en ML ; puisque notre projet était un problème de classification, nous décidons de lui consacrer une section indépendante. La cinquième et dernière section porte sur les classes déséquilibrées car notre ensemble de données était déséquilibré.

2.2 Sciences des données

On ne peut pas parler d'apprentissage automatique sans avoir parlé des sciences de données. Car la modélisation des données est une étape des sciences de données, quoique la plus importante. Les sciences de données nous permettent de prendre un ensemble de données brutes, les explorer et les modéliser dans le but d'extraire des connaissances utiles à la prise d'une décision. Comme son nom l'indique, la science des données n'est possible que lorsqu'on dispose d'un jeu de données qu'on appelle données historiques. Ces dernières sont des données recueillies à partir de la collecte d'informations. Les données peuvent être sous diverses formes, structurées ou non. D'une façon générale, pour traiter un problème en sciences de données, trois conditions sont nécessaires : (1) il faut avoir les données disponibles dans un format adéquat ; (2) définir des objectifs précis de ce qu'on veut faire avec les données ; (3) et savoir comment est-ce qu'on va procéder pour réaliser ces objectifs.

Les sciences de données peuvent être considérées comme une jonction entre l'informatique et les statistiques. Du coté de l'informatique, c'est la modélisation de données et tandis que du coté des statistiques, c'est l'exploration des données. Nous définissons la modélisation comme étant l'utilisation d'un algorithme d'apprentissage sur un ensemble de données bien préparé ; on entend par *bien préparé*, quand un jeu de données ne contient pas des données nulles, vides et textuelles. Quant à l'exploration, c'est la phase de visualisation et de préparation des données brutes qui peuvent être sous diverses formes.

Les sciences de données offrent des méthodes automatisées pour l'analyse prédictive d'un ensemble de données généralement massif. Le but est d'apporter des solutions à des problèmes spécifiques. Au début de l'ère de l'intelligence artificielle, l'analyse des données massives était compliquée parce que la capacité de calcul des ordinateurs était faible. Avec l'augmentation exponentielle des capacités des ordinateurs, l'explosion quantitative des données numériques devient de moins en moins un problème. Il faut noter qu'en apprentissage automatique, plus on a de données, plus l'analyse prédictive est fiable. Les algorithmes d'apprentissage fonctionnent mieux avec de grandes quantités de données. Quand le jeu de données est volumineux, on peut faire facilement la validation croisée qui est une technique par laquelle on subdivise aléatoirement un ensemble de données en plusieurs parties dans le but d'obtenir un meilleur apprentissage. La grande quantité de données permet aussi une séparation de données adéquate entre l'ensemble d'entrainement et l'ensemble de test.

Grâce à la croissance de la capacité de stockage des ordinateurs, l'apparition

des algorithmes d'apprentissage puissants et la puissance de calcul des ordinateurs, la profession de *scientifique de données* a vu le jour. Yannis Chaouche, professeur et expert en apprentissage automatique, a publié sur le site web OpenClassrooms, un cours intitulé «*Initiez-vous à l'apprentissage automatique*»où il a défini le cycle de travail d'un scientifique de données. D'après lui ce cycle comprend :

> *1. la récupération des données utiles à l'étude ;*
> *2. le nettoyage des données pour les rendre exploitables ;*
> *3. une longue phase d'exploration des données afin de comprendre en profondeur l'articulation des données ;*
> *4. la modélisation des données ;*
> *5. l'évaluation et interprétation des résultats ;*
> *6. la conclusion de l'étude : prise de décision ou déploiement en production du modèle.* ([25])

La figure 2.1 nous donne une vue générale du travail d'un scientifique de données. Cette figure est une façon de schématiser les différentes étapes qu'on a définies précédemment. Comme stagiaire en science de données, nous avons suivi minutieusement, dans le cadre de notre projet, le cycle de travail complet défini par le professeur Chaouche. Depuis la récupération des données jusqu'au déploiement d'un système en production. Nous expliquerons en détail dans les deux prochains chapitres comment nous implémentons ce cycle au sein de notre projet.

2.2.1 Données.

Comme nous avons précédemment dit, les données sont éminentes pour un travail de scientifique de données. S'il n'y a pas de données ou si elles ne sont pas disponibles, on ne peut rien faire. Dans les organisations, les données n'ont pas les mêmes niveaux d'importance, par exemple les données juridiques sont de haute confidentialité, personne ne peut avoir accès à moins que les concernés donnent une autorisation légale. Il est toujours difficile d'obtenir les données parce que cela peut compromettre une institution ; souvent il faut signer un contrat de confidentialité qui prévoit une sanction sévère s'il y a une violation. Les données constituent la ressource principale pour qu'un scientifique de données puisse effectuer son travail correctement.

2.2.1.1 Types de données

Les données, de part leurs natures, peuvent être quantitatives ou qualitatives. Pour mieux explorer un ensemble de données, il faut savoir distinguer le type de chaque donnée. Quand on parle de type de données ici, il ne faut pas confondre

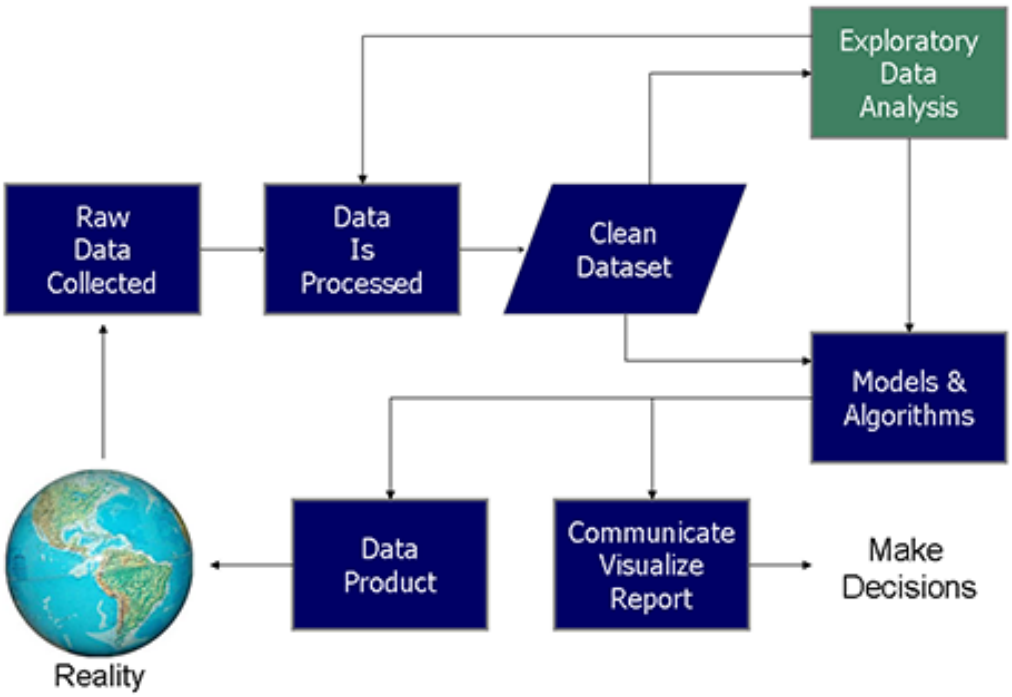

Fig. 2.1 – Cycle de travail d'un scientifique de données [25]

avec type de données qu'on a l'habitude d'utiliser en programmation informatique dans les langages typés ; mais il faut les considérer d'un point de vue ML.

Données quantitatives. Les données quantitatives sont des données qu'on peut mesurer ou qui peuvent prendre une valeur quelconque, avec lesquelles on peut faire des calculs numériques tels que la moyenne, l'addition, la multiplication, la division entre autres. Ces types de données permettent de répondre à la question du type «combien ». Par exemple combien de patients y-a-t-il dans la base de données ? La réponse pourrait être 3200 qui est une donnée quantitative. Il existe deux catégories de données quantitatives : données quantitatives continues et données quantitatives discrètes. Les premières peuvent prendre n'importe qu'elle valeur dans un ensemble de valeurs, par exemple la température, les PIB,le taux de chômage etc. Les deuxièmes ne prennent qu'un nombre limité de valeur dans un ensemble de valeur, par exemple le nombre d'enfants dans une famille, le nombre de chambres dans un appartement etc.

Données qualitatives. Les données qualitatives décrivent les qualités ou les caractéristiques. On ne peut pas les compter ni faire de calcul avec ces données, on peut les comparer ou éventuellement les trier. Les valeurs qualitatives ne doivent pas être des nombres, sinon un ensemble de modalités ; elles permettent de répondre à la question de la forme «Quel type »ou «Quelle catégorie ». On distingue aussi deux catégories de données qualitatives : données qualitatives nominales (ou catégorielles) et données qualitatives ordinales. Les premières catégories sont celles dont les modalités ne peuvent être ordonnées c'est-à-dire on ne peut pas les classer en ordre de grandeur, par exemple la couleur des yeux, le sexe, la région d'appartenance etc. Les deuxièmes catégories peuvent être ordonnées selon un ordre logique, par exemple les tailles des vêtements, le degré d'accord à un test d'opinion (fortement d'accord, d'accord, pas d'accord ou fortement pas d'accord). [28]

2.2.2 Problématique.

Tout projet de science de données suppose la résolution d'un problème, ces genres de projets commencent toujours par le constat d'un problème. Il faut qu'on soit en mesure de définir clairement ce qu'on veut résoudre et comment on va le résoudre. Généralement, cela demande qu'on s'asseye avec les responsables des organisations, de faire des entrevues avec eux afin de prendre connaissance des différents besoins et d'établir le pourquoi du projet. S'il n'y a pas une problématique clairement définie, le projet de science de données est impossible.

Pour conclure la section sur les sciences de données, la chose la plus importante à retenir c'est que deux conditions doivent être réunies pour qu'un projet de sciences de données soit possible : un jeu de données disponible et une définition claire des problèmes à résoudre. Pour notre projet nous avons disposé d'un ensemble de données orthopédiques et tous les objectifs étaient bien élaborés.

Dans la section qui suit nous allons voir une brève introduction sur ce quoi l'apprentissage automatique, ensuite un survol sur les concepts fondamentaux du ML et finalement nous nous concentrerons sur un sous-ensemble de concepts qui touchent notre projet, principalement le problème de classification. Nous n'aborderons pas dans ce rapport, la théorie sur l'exploration des données *(data mining en anglais)*. Nous supposons que ceux qui lisent ce rapport ont déjà des notions élémentaires sur cette question, sinon nous vous proposons de lire [31, 5]

2.3 Vue d'ensemble sur Apprentissage Automatique

Dans la section précédente, nous avons pu y voir plus clair sur le cycle global de travail du scientifique de données. Nous allons maintenant parler de l'apprentissage automatique dans cette section, c'est à dire la modélisation des données. Nous utilisons l'apprentissage automatique probablement des dizaines de fois par jour sans même le savoir. Chaque fois que nous effectuons une recherche Web sur Google ou Bing, cela fonctionne si bien parce que leur logiciel d'apprentissage automatique a trouvé comment classer les pages. Lorsque Facebook ou l'application photo d'Apple reconnaît nos amis dans nos images, c'est aussi de l'apprentissage automatique. Chaque fois que nous lisons nos courriels et qu'un filtre anti-spam nous évite d'avoir à parcourir des tonnes de spam, c'est parce que nos ordinateurs ont appris à distinguer le spam du courriel non-spam. Donc, c'est de l'apprentissage automatique. En 1997, Tom Mitchell, dans son livre «Machine Learning »a proposé une définition plus précise :

> *«On dit qu'un programme informatique apprend de l'expérience* ***E*** *par rapport à une certaine classe de tâches* ***T*** *et à la mesure de performance* ***P****, si sa performance aux tâches dans* ***T*** *, telle que mesurée par* ***P****, s'améliore avec l'expérience* ***E****».* (Tom Mitchell [24, p. 2])

Nous voyons dans cette définition que le programme informatique peut être n'importe quel algorithme d'apprentissage ; l'expérience est considérée comme n'importe quelle source de connaissances auxquelles un algorithme est obligé d'apprendre, par exemple un même algorithme peut apprendre à partir des données de finance, biologie, médecine etc. ; La mesure de performance consiste à améliorer les résultats d'un algorithme tout en conservant la même expérience.

La figure 2.2 nous montre en grosso modo les techniques d'apprentissage automatique à un très haut niveau d'abstraction. Deux grandes techniques d'apprentissage automatique sont définies : Apprentissage supervisé **(Supervised learning en anglais)** et l'apprentissage non-supervisé **(Unsupervised learning en anglais)**. L'apprentissage supervisé est une technique d'apprentissage avec des données étiquetées, il existe deux types de problèmes possibles dans l'apprentissage supervisé : problème de régression qui consiste à prédire une valeur numérique comme le prix d'un appartement dans une région géographique quelconque ; et le problème de classification qui consiste à prédire une valeur booléenne (oui ou non), par exemple on peut vouloir savoir si une campagne marketing pourrait être une réussite ou un échec. L'apprentissage non-supervisé est une technique d'apprentissage non étiquetée, c'est l'idée de groupe **(Clustering en anglais)** qui prédomine.

Cette figure donne une vue générale sur ce qu'est l'apprentissage automatique.

L'apprentissage automatique constitue une manière de modéliser des phénomènes, dans le but de prendre des décisions stratégiques. Les algorithmes utilisés permettent, dans une certaine mesure, à un système piloté par ordinateur (un robot éventuellement), ou assisté par ordinateur, d'adapter ses analyses et ses comportements en réponse, en se fondant sur l'analyse de données empiriques provenant d'une base de données ou de capteurs. La difficulté réside dans le fait que l'ensemble de tous les comportements possibles compte tenu de toutes les entrées possibles devient rapidement trop complexe à décrire **(on parle d'explosion combinatoire)**. On confie donc à des programmes le soin d'ajuster un modèle pour simplifier cette complexité et de l'utiliser de manière opérationnelle. Idéalement, l'apprentissage visera à être non supervisé, c'est-à-dire que la nature des données d'entrainement n'est pas connue [35].

2.3.1 Phases de modélisation d'un problème ML

Nous avons mentionné dans la section 2.2 portant sur les Sciences de données, qu'aucun projet de sciences de données n'est possible s'il n'y pas une problématique bien définie. Et s'il existe un problème, il existe aussi des moyens de le résoudre. Le cycle de travail d'un scientifique de données énuméré dans la section 2.2 peut être comme les principales techniques de résolution d'un problème. Dans ce cycle de travail, il y a la technique de modélisation, voyons maintenant le deux phases nécessaires à la modélisation :

1. **Phase d'entrainement.** Phase dans laquelle on dispose d'un ensemble de données d'entrainement **(trainset en anglais)** constitué de variables prédictives et une variable cible [1] ; les données d'entrainement proviennent des expériences antérieures. Durant cette étape l'algorithme d'apprentissage ne fait que d'apprendre à partir des exemples des données d'entrainement, il ne retourne aucune valeur. C'est pendant cette phase que la fonction d'hypothèse h (ou fonction de prédiction) est sélectionnée dans l'espace des hypothèses $\mathcal{H}$.
2. **Phase de prédiction.** Dans la prédiction, on dispose d'un nouvel ensemble de données où les valeurs cibles ne sont pas connues et ce sont ces valeurs qu'on doit prédire. Dans ce cas la fonction de prédiction est utilisée pour calculer toutes les valeurs possibles de la variable cible et son entrée est l'ensemble des variables prédictives.

Dans la pratique, obtenir de nouvelles données est une tâche très difficile, peut prendre plusieurs mois d'attente. Par conséquent quand on a un ensemble de don-

1. Les variables cibles peuvent aussi appeler *étiquettes*

nées, il est recommandé au scientifique de données, au moment où il construit son modèle, de le répartir en deux : données d'entrainement et données de test. L'ensemble des données d'entrainement est utilisé dans la phase d'entrainement et celui des données de test dans la phase de prédiction. Habituellement on conserve 80% des données pour l'entrainement et 20% pour le test.

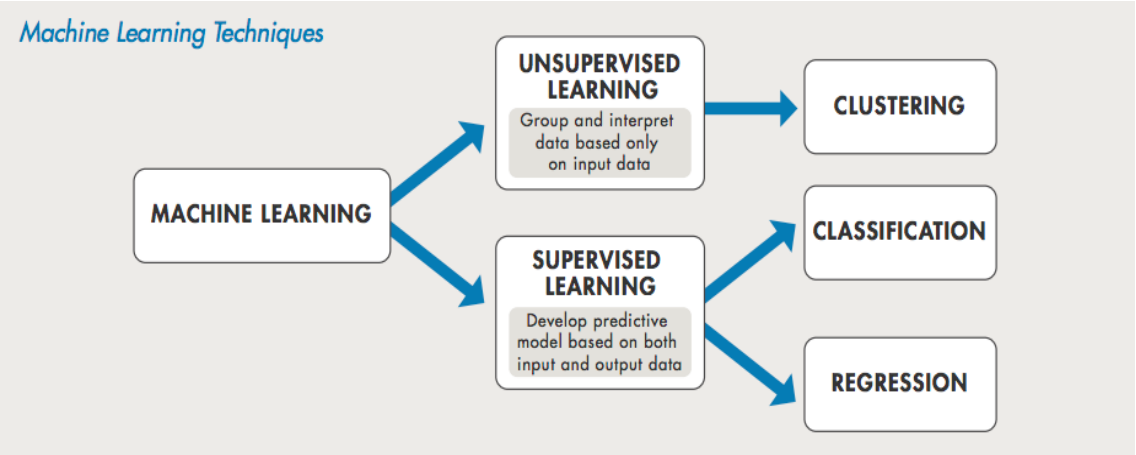

Fig. 2.2 – Vue globale d'Apprentissage automatique [27]

Pour continuer nous allons passer en revue les différents types de techniques d'apprentissage automatique apparues dans la figure 2.2. Elles constituent la base de l'apprentissage automatique. Aussi nous verrons quelques concepts importants découlés de chaque type. Nous donnerons une brève introduction de chaque thème jugé important pour la suite de ce rapport. L'idée n'est pas de tout détailler mais de présenter un aspect global de chaque concept pour faciliter la compréhension.

2.3.2 Apprentissage supervisé

L'apprentissage supervisé est formellement défini comme une fonction $h : x \rightarrow y$ où $h \in \mathcal{H}$ est la fonction d'hypothèse produite par un apprenant qui est un algorithme d'apprentissage, et $\mathcal{H}$ est l'espace des hypothèses d'apprentissage ; x représente l'ensemble de données d'entrée de la fonction et y détermine l'ensemble de données de sortie. La fonction d'hypothèse peut être aussi appelée fonction de prédiction, elle est sélectionnée de telle sorte qu'elle soit très rapprochée de l'ensemble de données. Une autre définition possible de l'apprentissage supervisé est un ensemble de couples d'échantillons d'entrée-sortie de la forme $(x^{(i)}, y^{(i)})$ où $x^{(i)}$ est la description d'un objet dans un espace de représentation ; $y^{(i)}$ est la variable cible désirée de l'objet ; et i est un index défini sur l'intervalle $1 \leq i \leq m$, où m le nombre d'échantillons ou d'observations dans l'ensemble de données. Le nombre de variables prédictives associées à une observation est noté comme un

vecteur $\vec{x} = (x_1, \cdots, x_j)$ où j est un index compris entre $1 \leq i \leq n$ et n représente la taille des variables [23, 26, 19]. Les n variables décrivant les m individus de l'ensemble de données sont représentés sous la forme d'une matrice nommée X de dimensions (m, n) qui est définie de la manière suivante :

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}$$

[28, p. 21]

et un vecteur $\vec{y}$ défini de la manière suivante :

$$\vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

Dans le but de faciliter la compréhension nous estimons qu'il est important de définir certains concepts utilisés dans cette définition :

- **Objet.** Un objet est une instance d'un ensemble de données, on le nomme aussi individu, observation, échantillon ou ligne.
- **Espace de représentation.** L'espace de représentation représente l'ensemble de données.
- **Variables prédictives.** L'ensemble de variables à partir desquelles on peut faire des prédictions.
- **Variable cible.** C'est une variable qu'on souhaite prédire.

La plupart des algorithmes supervisés sont linéaires c'est-à-dire leurs fonctions d'hypothèse h sont une combinaison linéaire des variables prédictives et elle est écrite de la forme : $\theta_1 x_1 + \cdots + \theta_n x_n \equiv \theta \cdot X$. Le vecteur $\vec{\theta}$ est défini de la façon suivante : $\vec{\theta} = \theta_1, \cdots, \theta_n$ qui sont les différentes valeurs possibles de chaque variable prédictive.
La régression logistique et la classification logistique sont des exemple d'algorithmes linéaires.

2.3.3 Type d'apprentissage supervisé

L'apprentissage supervisé, selon le type de sorties, est divisé en deux types de problèmes : problème de classification et problème de régression. La différence entre les deux se situe au niveau du type de problème à traiter. Dans la vie réelle,

les problèmes sont diverses et différents les uns des autres mais une analyse prédictive aide à catégoriser les types de problèmes à traiter et déterminer quels types d'algorithmes supervisés à utiliser. Dans les deux cadres de figure les entrées restent toujours les mêmes, il n'y a aucune distinction à ce niveau. Voyons un peu en détails les caractéristiques de chaque problème et pour pouvoir saisir la différence qui existe entre les deux.

2.3.3.1 Problème de classification

Dans un problème de classification, les types de sorties qu'on traite sont des classes. Les classes sont des variables catégorielles ou discrètes, elles permettent de déterminer si un vecteur x appartient à une catégorie de sortie, dans ce cas la variable cible est qualitative. En d'autres termes, le problème de classification est le problème d'identifier lequel d'un ensemble de catégories (sous-population) appartient à une nouvelle observation, sur la base d'un ensemble de données contenant des observations (ou instances) dont la composition est connue. Une définition plus précise et formelle a été proposée par Tom Michell :

> *«Un exemple est un couple* $(\mathbf{x}, \mathbf{u})$ *où* $\mathbf{x} \in \mathcal{X}$ *est la description ou la représentation de l'objet et* $\mathbf{u} \in \mathcal{U}$ *représente la supervision de* $\mathbf{x}$*. Dans un problème de classification,* $\mathbf{u}$ *s'appelle la classe de* $\mathbf{x}$ *et appartient à un ensemble* $\mathcal{C} = \{\omega_1, \cdots \omega_C\}$*.* $\mathcal{C}$ *designe le nombre de classes possibles pour un objet».* ([23, p. 88])

Dans la pratique, pour parler de classification, l'ensemble $\mathcal{C}$ doit être fini et petit. Si $\mathcal{C} = 2$, on parle de classification binaire qui est le cas spécial de distinguer exactement deux classes (oui/non ou vrai/faux) ; si $\mathcal{C} > 2$, on parle de classification multiclasse. L'exemple d'iris [2] qui est un jeu de données basé sur 50 échantillons d'espèces végétales est un exemple typique de la classification multiclasse. Un exemple de classification binaire est de prédire si un patient d'un certain lieu géographique peut avoir ou non le cancer si l'on dispose d'un ensemble de données médicales des patients de ce lieu. Les algorithmes d'apprentissage qui résolvent des problèmes de classification sont appelés «classificateur ». L'algorithme essaye d'attribuer chaque nouvelle entrée à une classe discrète appartenant au nombre $\mathcal{C}$ de classes. [28, 34, 15]

2.3.3.2 Mesure de performance des algorithmes de classification

L'évaluation de la performance des méthodes d'apprentissage automatique est aussi cruciale que l'algorithme lui-même, car il identifie les forces et les faiblesses

2. page de wikipedia sur iris : https ://en.wikipedia.org/wiki/Iris_flower_data_set

de chaque algorithme d'apprentissage. Différentes mesures de performance sont utilisées pour évaluer différents algorithmes d'apprentissage automatique. Pour ce projet, nous nous concentrons sur celles utilisées dans les problèmes de classification parce que notre projet est problème de classification. Les métriques de classification couramment utilisées pour évaluer la performance des problèmes de classification sont *Log-Loss, Accuracy, AUC(Area under Curve)* etc. Nous pouvons créer nos propres métriques personnalisées selon nos besoins. Le choix des métriques est très important en ML car il permet de détecter les défauts ou les erreurs de nos modèles.

2.3.3.3 Matrice de confusion

La matrice de confusion **(Confusion Matrix en anglais)** est utilisée pour trouver l'exactitude et la précision d'un modèle. Elle doit être utilisée pour des problèmes de classification binaire puisqu'il est basé sur des valeurs positives et négatives.La matrice de confusion en soi n'est pas une mesure de performance en tant que telle, mais presque toutes les métriques de performance sont basées sur la matrice de confusion et les nombres qui s'y trouvent. La figure 2.3 est une représentation de la matrice de confusion. La Actual value (**valeur réelle**) représente la vraie valeur de l'étiquette ou la variable cible tandis que la Predicted value (**valeur prédite**) représente la valeur de prédiction.

True Positives (TP) : True Positives (**vrais positifs, en français**) sont les cas où la classe réelle du point de données était 1 (Vrai) et la prédiction est également 1 (Vrai)

True Negatives (TN) : True Negatives (**vrais négatifs, en français**) sont les cas où la classe réelle du point de données était 0 (Faux) et la prédiction est également 0 (Faux)

False Positives (FP) : False Positives (**faux positifs, en français**) sont les cas où la classe réelle du point de données était 0 (Faux) et la prédiction est 1 (Vrai). Faux parce que le modèle a prédit incorrectement et positivement parce que la classe prédite était positive. (1)

False Negatives (FN) : False Negatives (**faux négatifs, en français**) sont les cas où la classe réelle du point de données était 1 (Vrai) et la prédiction est 0 (Faux). Faux parce que le modèle a prédit incorrectement et négativement parce que la classe prédite était négative. (0)

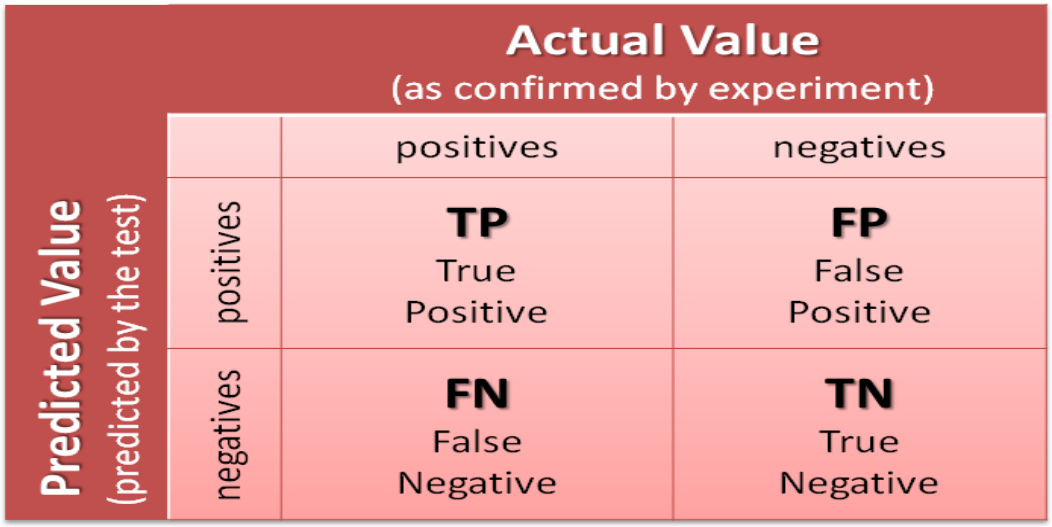

Fig. 2.3 – Confusion Matrix representation [30]

Voyons comment calculer quelques métriques de classification à partir de la matrice de confusion :

1. **Accuracy :** L'exactitude **(Accuracy en anglais)** dans les problèmes de classification est le nombre de prédictions correctes faites par le modèle sur toutes les prédictions faites. Accuracy = $\frac{TP+TN}{TP+FP+FN+TN}$
2. **Precision :** La précision permet de répondre à la question suivante : Quelle proportion d'identifications positives était effectivement correcte ? Precision = $\frac{TP}{TP+FP}$
3. **Recall or Sensitivity :** Le rappel permet de répondre à la question suivante : Quelle proportion de résultats positifs réels a été identifiée correctement ? Recall = $\frac{TP}{TP+FN}$
4. **F1-score :** Nous ne voulons pas vraiment avoir à la fois la précision et le rappel dans nos poches chaque fois que nous faisons un modèle pour résoudre un problème de classification. Donc, il est préférable que nous puissions obtenir un seul score qui représente à la fois la précision (P) et le rappel (R)
   F1score = $\frac{2}{\frac{1}{\text{precision}}+\frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision}+\text{recall}} = \frac{TP}{TP+\frac{FN+FP}{2}}$

2.3.3.4 Problème de régression.

Dans un problème de régression l'espace de sortie est formé par des valeurs continues. C'est le cas où l'espace de représentation $\mathcal{H}$ est un ensemble de fonctions h à valeurs réelles. L'exemple que nous avons pris pour expliquer l'apprentissage supervisé est un problème de régression car les prix des logements sont des valeurs

réelles. Établir la relation entre l'augmentation du prix d'un produit et sa demande, évaluer l'impact d'une campagne publicitaire en fonction des frais engagés sont d'autres exemples d'utilisation de la régression. La différence entre les problèmes de classification et de régression est la suivante : Dans la classification, la variable cible est qualitative c'est-à-dire c'est une variable à valeurs discrètes ou catégorielles tandis que dans la régression la variable cible est quantitative c'est une variable à valeurs continues.

2.3.4 Apprentissage non supervisé

L'apprentissage supervisé est une tâche d'apprentissage automatique qui consiste à inférer une fonction qui décrit la structure des données à variables non-étiquetées c'est-à-dire il n'y a pas de variable cible, toutes les données sont équivalentes (il n'y a que de variables d'entrées, pas de sortie) ; dans ce cas les algorithmes cherchent à organiser les données en groupes (**Clusters en anglais**). Chaque groupe doit comprendre des données similaires et les données distincts doivent se retrouver dans des groupes distincts. S'il existe la possibilité de trouver un partitionnement (**Clustering en anglais**) tel que les similarités des objets dans un groupe sont beaucoup plus grands que les autres groupes, on peut considérer que l'ensemble d'entrainement peut être représenté par un seul groupe structuré. La similarité entre les objets est généralement calculée selon une fonction de distance entre les couples d'échantillons. La distance communément utilisée est la distance euclidienne définie de la manière suivante :

Soit une matrice X à n variables quantitatives

Soient deux échantillons (x_1, x_2)

La distance euclidienne est calculée par la formule suivante :

$$d(x_1, x_2) = \sqrt{\sum_{j=1}^{n}(x_{1j} - x_{2j})^2}$$

Les possibilités d'application de l'apprentissage non-supervisé sont nombreuses, par exemple on peut analyser une base de données marketing pour grouper les clients qui ont des comportements similaires auxquels on pourrait adresser des campagnes commerciales personnalisées. D'autres cas d'application serait le traitement des images, la segmentation des textes, la détection d'une anomalie dans les systèmes de contrôle entre autres. [34, 28]

Deux approches sont proposées pour partitionner un ensemble de données de façon automatique :

1. **Partitionnement hiérarchique.** C'est une approche qui consiste à repartir un ensemble de données dans une suite de groupes emboités les uns dans les

autres ; dans ce cas les relations entre les groupes sont représentés dans une structure arborescente. Plus on est plus bas dans l'arbre, plus les objets se ressemblent. Cette méthode est applicable lorsqu'on ne connait pas à l'avance le nombre de groupes à former.

2. **Partitionnement non-hiérarchique.** Les instances de l'ensemble des données sont réparties en un nombre k de classes. k étant une valeur fixée à l'avance.

Dans les deux approches une bonne classification (**clustering en anglais**) est réussie quand les groupes ont une forte similarité intraclasses (c'est-à-dire les objets d'un même groupe doivent se ressembler) et une fiable similarité interclasses (c'est-à-dire les objets d'un groupe distinct ne doivent pas se ressembler). [28]

2.3.5 Classes déséquilibrées

Nous terminerons la section 2.3 sur l'apprentissage automatique en abordant un sujet très important en apprentissage automatique, surtout dans un problème de la classification qui est "*Jeux de données déséquilibré*" **(Imbalanced datasets or Imbalanced Classification or Imbalanced Classes, ses différentes appellations en anglais)**. C'est un problème très commun en apprentissage automatique, spécialement en classification. Nous abordons ce sujet parce que c'était l'une des principales difficultés de notre projet.

Une distribution de classe déséquilibrée est un scénario où le nombre d'observations appartenant à une classe est significativement inférieur à celui des autres classes. Ce problème prédomine dans les cas où la détection d'anomalies est cruciale comme le vol d'électricité, les transactions frauduleuses dans les banques, l'identification de maladies rares, etc. Dans ce cas, le modèle prédictif développé avec des algorithmes conventionnels pourrait être biaisé et inexact.
Cela arrive parce que les algorithmes d'apprentissage automatique sont généralement conçus pour améliorer la précision en réduisant l'erreur. Ainsi, ils ne tiennent pas compte de la répartition / proportion des classes ou de l'équilibre des classes .

Jason V. Hulse et al. [16] ont proposé plusieurs techniques permettant de manipuler les classes déséquilibrées. Voyons un bref résumé de quelques-unes d'entre elles :

- **Sous-échantillonnage aléatoire.** Le sous-échantillonnage aléatoire (**Random Undersampling, en anglais**) vise à équilibrer la distribution des classes en éliminant de manière aléatoire les exemples de classes majoritaires. Ceci est fait jusqu'à ce que les instances de la majorité et de la classe minoritaire soient équilibrées.

- **Sur-échantillonnage aléatoire.** Le sur-échantillonnage aléatoire (**Random Over-Sampling, en anglais**) augmente le nombre d'instances dans la classe minoritaire en les reproduisant aléatoirement afin de présenter une représentation plus élevée de la classe minoritaire dans l'échantillon.
- **Suréchantillonnage en groupe (Cluster-Based Over Sampling, en anglais).** Dans cette technique, l'algorithme de clustering K-means est appliqué indépendamment aux instances de classe minoritaire et majoritaire. Cela permet d'identifier les clusters dans l'ensemble de données. Par la suite, chaque grappe est suréchantillonnée de sorte que tous les clusters de la même classe ont un nombre égal d'instances et toutes les classes ont la même taille.
- **Synthetic Minority Over-sampling Technique(SMOTE) :** Cette technique est suivie pour éviter le surapprentissage qui se produit lorsque des répliques exactes d'instances minoritaires sont ajoutées à l'ensemble de données principal. Un sous-ensemble de données est pris à partir de la classe minoritaire à titre d'exemple, puis de nouvelles instances similaires synthétiques sont créées. Ces instances synthétiques sont ensuite ajoutées à l'ensemble de données d'origine. Le nouvel ensemble de données est utilisé comme un échantillon pour former les modèles de classification.
- **Modified synthetic minority oversampling technique (MSMOTE) :** C'est une version modifiée de SMOTE. SMOTE ne tient pas compte de la distribution sous-jacente de la classe minoritaire et des bruits latents dans l'ensemble de données. Pour améliorer les performances de SMOTE, une méthode modifiée MSMOTE est utilisée.

2.3.6 Méthodes d'ensemble

L'objectif principal de la méthodologie d'ensemble **(Ensemble methods en anglais)** est d'améliorer la performance des classificateurs uniques. L'idée de base de l'apprentissage d'ensemble est de combiner plusieurs algorithmes simples afin d'obtenir un qui soit plus complexe dans le but de générer de meilleurs résultats. Ces résultats sont beaucoup mieux que lorsqu'on utilise un seul algorithme. L'approche consiste à construire plusieurs classificateurs à deux étapes à partir d'un ensemble de données original, puis à agréger leurs prédictions [1, 19]. Les méthodes ensemblistes fonctionnent aussi bien sur de petit ensemble de données que sur de grand volume de données.

Les méthodes d'ensemble sont nombreuses, mais nous allons brièvement définis les trois plus importantes : Bagging, Boosting et Random Forest.

Bagging. Bagging (Bootstrap aggregating) est une approche de construction d'ensemble qui utilise différents sous-ensembles de données d'apprentissage avec

une méthode de classification unique. Étant donné un ensemble d'apprentissage de taille **t**, Bagging attire des instances aléatoires **t** de l'ensemble de données avec remplacement (à l'aide d'une distribution uniforme). Ces **t** instances sont apprises, et ce processus est répété plusieurs fois. Étant donné que le tirage au sort est effectué avec remplacement, les instances tirées contiendront des doublons et des omissions par rapport à l'ensemble d'apprentissage initial. Chaque cycle à travers le processus aboutit à un classificateur. Après la construction de plusieurs classificateurs, les sorties de chaque classificateur sont combinés pour produire la prédiction finale [36].

Boosting. Une autre approche appelée «Boosting» utilise également une méthode d'apprentissage unique avec différents sous-ensembles de données d'apprentissage. Sa structure globale est similaire à celle de la méthode Bagging, à la différence qu'elle conserve la trace de la performance de l'algorithme d'apprentissage et se concentre sur les cas qui ne sont pas correctement appris. Au lieu de choisir les instances **t** d'apprentissage à l'aide d'une distribution uniforme de manière aléatoire, les exemples d'apprentissage sont sélectionnés en favorisant les instances quine sont pas bien classées. Après plusieurs cycles, la prédiction est réalisée selon un vote pondéré des prédictions de chaque classificateur. Ainsi, les poids sont proportionnels à la précision de chaque classificateur sur son ensemble d'apprentissage. L'algorithme le plus connu de l'approche Boosting, appelée « AdaBoost » [36].

Stacking. Le stacking est similaire aux boosting et bagging : le but est de combiner également plusieurs modèles à un ensemble données d'origine. La différence est que le stacking permet d'explorer différents modèles pour le même problème. Il est possible d'entrainner différents types d'apprenants simultanément. Le stacking divise l'ensemble de données en deux sous-ensembles, le premier est utilisé durant la phase d'entrainement et le deuxième est conservé pour le test. Le mécanisme est que la sortie d'un classifier est utilisé comme données d'entrainement pour un autre classifier.

Random Forest. Les forêts aléatoires(plus connus sous **Random Forest**) sont une combinaison d'arbres de décision, où chaque arbre dépend des valeurs d'un vecteur aléatoire indépendamment échantillonné et avec la même distribution pour tous les arbres de la forêt.L'erreur de généralisation d'une forêt d'arbres dépend de la force des arbres individuels dans la forêt et de la corrélation entre eux. L'utilisation d'une sélection aléatoire de caractéristiques pour diviser chaque nœud donne des taux d'erreur qui se comparent favorablement à AdaBoost. [1]

2.4 Conclusion

Le chapitre 2 a présenté une vue théorique de quelques grands concepts d'apprentissage automatique et de sciences de données, ceux que nous jugeons être importants pour la réalisation de notre projet. Nous avons donné une brève définition de chaque concept tout en essayant d'être le plus clair et précis possible. Nous avons vu un sous-ensemble de sujets importants des différents concepts puisque ces champs d'études sont très vastes et contiennent beaucoup de choses intéressantes, mais nous avons mis accent sur ce qui nous intéresse pour notre projet. Par exemple notre projet est un problème de classification en apprentissage automatique, pour cela nous avons dédié une section à une étude brève du problème de classification Tout en ayant le soin de définir et d'expliquer les thèmes les plus pertinents. Aussi notre dataset est déséquilibré, par conséquent nous avons mis l'accent sur c'est quoi les classes déséquilibrées et les différentes techniques permettant de les manipuler. Et en dernier lieu nous avons passé en revue les méthodes ensemblistes puisque les algorithmes d'ensemble sont utilisés pour attaquer le problème de classes déséquilibrées. Le chapitre 3 qui suit présentera les détails techniques de notre projet, le but est de faire sortir les différentes étapes que nous avons suivies pour arriver à un produit fini et utilisable.

3 | Approche : Construction du modèle d'apprentissage automatique

La construction d'un modèle d'apprentissage automatique est la première grande phase dans l'implémentation de notre solution. Le modèle estimateur (meilleur modèle entrainé en fonction d'un ensemble de données) va être utilisé pour introduire la deuxième phase, celle de la création de l'API *Ortho*. Toute notre solution a été implémentée en utilisant le langage de programmation *Python*. Les librairies d'apprentissage automatique utilisées dans ce projet sont toutes écrites en *Python*.

Comme nous avons déjà vu dans l'introduction un diagramme d'activités qui décrit le comportement de notre système d'apprentissage automatique. Maintenant nous proposons un diagramme de classe (figure 3.1) qui implémente les différents états que nous avons décrits. Nous tenons à souligner que dans notre diagramme de classe nous ne prenons pas en compte les classes prédéfinies dans les librairies de Python. Par exemple le **DataFrame** est une classe prédéfinie dans la librairie **pandas**. Nous l'utilisons dans notre classe **Gather** mais elle n'est pas définie dans notre diagramme de classe.

Dans notre projet, le problème d'apprentissage est un problème de classification parce que notre variable cible est une étiquette discrète et c'est binaire parce qu'on doit décider si oui ou non le patient reste à l'hôpital après une intervention chirurgicale. Si le patient reste, la valeur de la classe est positive (c'est-à-dire 1), s'il ne reste pas ou il est renvoyé, la valeur de la classe est négative (c'est-à-dire 0). Puisque dans la majorité des cas les médecins ne gardent pas les patients à l'hôpital après l'arthroplastie, alors notre ensemble de données est déséquilibré. Plus de 80% des patients sont retournés à la maison. C'est normal parce que s'ils restent, les coûts sont couverts par l'hôpital. Constatant ce déséquilibre, nous sommes obligés d'appliquer une technique de re-échantillonnage afin d'éviter que notre modèle soit biaisé. Nous avons opté d'équilibrer notre ensemble de données

en appliquant la technique *SMOTE* parce qu'elle a l'avantage d'atténuer le problème de sur-adaptation causé par le sur-échantillonnage aléatoire. Ce dernier est une technique traditionnelle qui a beaucoup d'inconvénients dont nous ne traitons pas dans ce rapport. Si vous voulez avoir plus de connaissances sur ces techniques, vous pouvez lire ces articles [16, 6, 14].

Dans les sections qui suivent nous allons voir les différentes étapes d'implémentation de notre modèle d'apprentissage automatique. Nous commençons d'abord par l'obtention des données, pour ensuite continuer avec la visualisation, le nettoyage, la transformation et enfin la modélisation. Ce qui donne quasiment le cycle complet d'un travail de scientifique de données.

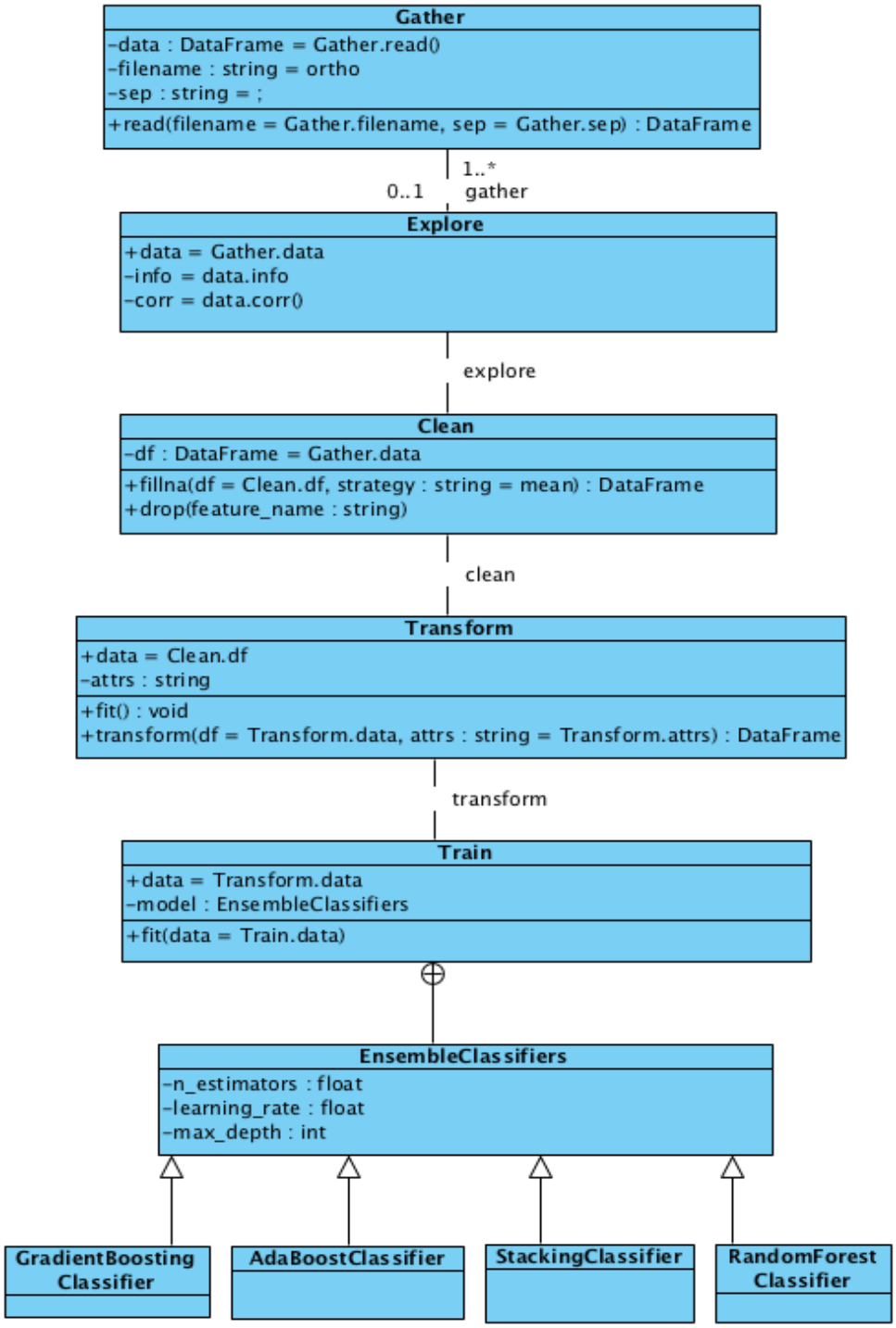

Fig. 3.1 – Diagramme de classe de notre modèle d'apprentissage automatique

3.1 Collecte de notre ensemble de données

Une fois qu'on décide d'attaquer un problème d'apprentissage automatique, la première chose à faire est d'explorer toutes les pistes possibles pour récupérer les données. En effet, les données constituent l'expérience, les exemples qu'on va fournir à notre algorithme d'apprentissage automatique afin qu'il puisse apprendre et devenir plus performant. D'après la définition d'apprentissage automatique de Tom Mitchell qu'on a vue à la section 2.3, l'expérience E constitue la source des connaissance ou la source des données. Suivant cette définition pour qu'il y ait l'apprentissage, il faut disposer d'une source de données.

Nos données proviennent du département orthopédique de l'université de Seattle, Washington, États-Unis. Elles sont au nombre de 2718 lignes et 92 colonnes. C'est un ensemble de données orthopédique très sensible. Ces données ne sont pas accessibles au public, elles sont hautement confidentielles ; pour avoir accès à ces données nous avons signé un contrat de confidentialité dont la clause est de sauvegarder secrètement ces données. En général les bases de données médicales sont des données privées et extrêmement sensibles ; d'après la loi sur la protection des informations privées, on ne peut pas accéder à ces données sans l'autorisation de leurs propriétaires.

Avant de nous donner accès à ces données, on a supprimé toutes les informations personnelles du patient sauf son âge qui est une donnée pertinente pour l'analyse prédictive. Les données personnelles qu'on a supprimées sont entre autres : le nom, le prénom et l'adresse complète. L'objectif est de protéger le plus possible les informations des patients.

L'ensemble de données est hébergé sur le serveur de la compagnie pour laquelle je fais mon stage et il est disponible dans un fichier de format *csv*. La compagnie les a obtenues de l'université Seattle, assurément après une entente entre les deux entités. Nous avons eu un accès direct au serveur pour pouvoir télécharger ces données et les utiliser pour construire notre modèle.

Nous avons divisé notre ensemble de données en deux parties : un ensemble d'entrainement et un ensemble de test. 80% de l'ensemble des données est conservé pour la phase d'entrainement et 20% pour la phase de prédiction. La plus grande difficulté c'est que l'ensemble de données est peu nombreux pour construire un modèle prédictif hautement performant. En ML plus les données sont nombreuses, plus on est capable de construire de bon modèle.

Comme vous pouvez remarquer dans la diagramme de classe de l'apprentissage automatique (figure 3.1), on a implémenté un classe nommant *Gather*. Celle-ci nous permet d'obtenir les données brutes à travers la méthode *read* qui prend en paramètre une chaine qui contient le chemin et nom de fichier (*filename*) et un séparateur (**sep**). Le fichier contient toutes les données disponibles pour la construction du modèle et le séparateur permet de préciser comment les colonnes sont séparées entre elles, dans notre cas c'est un point virgule (*;*) qui les sépare.

Après avoir obtenu ces données, nous devons maintenant les explorer pour avoir des idées claires sur l'ensemble des données. La section suivante est l'étape d'exploration des données, les données obtenues sont transférées à la classe Exploration de données.

3.2 Exploration de notre ensemble de données

L'analyse exploratoire est une technique permettant de visualiser l'ensemble de données dans le but d'en prendre connaissance. Elles permettent de déterminer les relations qui existent entre les variables de notre ensemble de données. L'intérêt de la visualisation est de détecter les valeurs aberrantes et vérifier certaines hypothèses avant de choisir notre modèle comme la linéarité de certaines valeurs. Nous faisons cette démarche à l'avance parce que nous voulons obtenir de précieux conseils pour le nettoyage de données et aussi avoir des idées claires pour l'ingénierie des caractéristiques **(Feature Engineering en anglais)**.

Pour l'analyse exploratoire de notre ensemble données, Tout d'abord, nous commençons par une analyse basique de l'ensemble de nos données. La table 3.1 donne une description de notre ensemble de nos données utilisé pour notre projet et la figure 3.2 donne une vue de notre ensemble de données, leur représentation, et leurs valeurs possibles. Si nous nous référons à notre classe diagramme 3.1), c'est l'attribut info de la classe Explore qui nous fournit les informations sur notre dataset. La méthode head nous retourne les cinq premières lignes du jeu de données.

Voyons une brève description de chaque composante de notre ensemble de données après une analyse basique de notre exploration :

Nombre de variables. Représente le nombre totale de variables ou colonnes de notre ensemble de données. Les variables prédictives sont importantes pour un modèle d'apprentissage automatique puisque le modèle les utilise pour prédire des événements. Dans l'annexe A, la table A.1 donne une description de tous les variables de notre ensemble de données. Le nombre total des variables est

92. La colonne **Feature name** représente le nom de chaque colonne. La colonne **datatype** représente le type de données et les colonnes. Et les colonnes **non-null** et **null** représentent respectivement le nombre d'échantillons non-nulls et le nombre d'échantillons nulls. Référez-vous à la table A.1, pour prendre connaissance des nom de colonnes car nous allons les utiliser plus tard.

Nombre de variables	92
Nombre d'observations	2718
Nombre de variables numériques	87
Nombre de variables discrètes ou catégorielles	5
Variable cible	Discharge

TABLE 3.1 – Exploration basique de notre ensemble de données

Nombre d'observations. Représente le nombre d'échantillons **(samples or rows en anglais)** de notre ensemble de données.

Variables numériques. Ce sont les variables dont les valeurs sont numériques ou continues dans notre ensemble de données.

Variables discrètes. Ce sont les variables qui viennent sous forme de texte. Plus loin dans la section de la transformation, nous convertirons ces variables en données numériques car la majorité des algorithmes d'apprentissage automatique ne reconnaissent pas les données discrètes ou textuelles.

Variable cible. C'est l'étiquette (**label or target en anglais)** de notre ensemble de données. Comme nous avons vu l'étiquette est une variable très importante pour les algorithmes supervisés car elle permet d'évaluer la valeur de sortie d'une décision. Dans notre cas c'est le *Discharge* qui est notre variable cible, elle permet de dire si oui ou non le patient doit rester à l'hôpital pour une réadaptation.

La table 3.2, quant à elle, c'est la description de notre variable target value. Dans l'ensemble de données, elle s'appelle *Discharge*. On peut remarquer que les classes positives sont significativement plus nombreuses que les classes négatives, ce qui produit un déséquilibre au niveau de l'ensemble de données. Nous avons déjà discuté des classes déséquilibrées dans la sous-section 2.3.5 et les différentes techniques pour les équilibrer.

Nom d'étiquète	Discharge
Nombre de classes positives	2421
Pourcentage de classes positives	89.072848 %
Nombre de classes négatives	297
Pourcentage de classes négatives	10.927152 %
Nombre total classes	2718

TABLE 3.2 – Description de notre variable cible

3.2.1 Corrélation entre nos variables

Nous terminons la section d'exploration de données avec les corrélations. Ces dernières nous permettent d'examiner les relations entre les variables numériques de notre ensemble de données. La corrélation est une valeur comprise entre -1 et 1 qui représente à quel point deux entités se déplacent à l'unisson. Deux variables sont positivement corrélés lorsque l'une augmente et l'autre augmente aussi. Par exemple, Le BMI d'un patient et son poids **(Weight dans le dataset)** sont positivement corrélés, ce qui veut dire le BMI n'augmente que lorsque son poids augmente.

Dans la corrélation négative, lorsqu'une variable augmente, l'autre diminue. Par exemple, dans notre ensemble de données quand l'âge du patient augmente son poids diminue. Les corrélations proches de -1 ou 1 indiquent une relation forte entre les variables. Les plus proches de 0 indiquent qu'il y a une relation faible. 0 indique qu'il n'y aucune relation.

La figure 3.3 nous donne une idée comment nos variables sont corrélées entre elles. Par exemple, le poids **(Weight dans notre ensemble de données)** et le **BMI** ont une corrélation de 0.81 très proche de 1, c'est-à-dire qu'ils sont fortement relationnés. Nous pouvons déduire que les patients qui ont leur BMI plus élevé sont les plus pesants. L'objectif de la corrélation est de gagner en intuition sur les données, ce qui nous aidera tout au long du flux de travail. Pour des raisons d'espace nous ne pouvions pas présenter la corrélation entre les 92 variables de notre ensemble de données, par contre nous avons sélectionné les plus pertinentes pour montrer dans la figure 3.3.

Pour calculer la corrélation, nous avons utilisé la méthode **corr** de la classe **Explore** de notre diagramme de classes. Nous avons passé un peu de style en paramètre pour avoir les couleurs.

L'exploration nous donne une idée générale de notre ensemble de données, elle

nous rend capable de poursuivre les autres étapes avec une marge d'erreur très faible. Nous avons vu l'exploration basique et la corrélation entre nos variables, ces résultats sont utilisés à la section 3.4. Maintenant nous poursuivons donc avec la phase de nettoyage. La prochaine section est consacrée au nettoyage de notre ensemble de données.

3.3 Nettoyage de notre ensemble de données

Une fois l'étape exploratoire des données est terminée et que nous ayons suffisamment de connaissances des données, c'est le temps maintenant de les nettoyer. Le nettoyage des données **(Data cleaning en anglais)** consiste à éliminer les observations qu'on ne souhaite pas conserver pour la suite de l'analyse prédictive. Ce sont des informations qui peuvent être erronées, inexactes ou sans intérêt pour la phase de modélisation. Les scientifiques de données consacrent habituellement une très grande partie de leur temps au nettoyage des données parce que même les algorithmes simples peuvent apprendre des informations impressionnantes à partir d'un jeu de données correctement nettoyé.

Nous avons suivi les étapes suivantes pour nettoyer notre ensemble de données :

- **Suppression des observations indésirables.** Les observations indésirables (**Irrelevant observations, en anglais**) sont celles qui ne sont pas pertinentes pour une analyse prédictive c'est-à-dire elles n'ont aucune influence sur la prédiction, on suggère de les supprimer afin que notre modèle ne soit pas biaisé. Dans notre ensemble de données, nous n'avons pas constaté d'observations indésirables parce que chaque observation est un patient qui a fait l'objet d'une décision de retour. Toutes les observation étaient importantes pour notre analyse.
- **Observations en double.** Les observations en double (**Duplicate observations, en anglais**) surviennent le plus souvent lors de la collecte des informations, plus précisément lorsque les données proviennent de plusieurs endroits différents et qu'on essaye de les combiner pour avoir un seul ensemble de données. Dans notre cas, l'ensemble de données provient d'un seul endroit (université de Seattle), par conséquent il n'y avait pas de doublons. En plus c'est très difficile de constater des doublons dans un jeu de données médical parce que c'est très dangereux d'avoir deux instances pour un même patient.
- **Données manquantes.** Dans la vie réelle, un ensemble de données contient toujours des instances où un élément particulier peut être absent, ceci est dû parce que les administrateurs de base de données, en raison de concep-

tion, admettent que certains attributs peuvent être vides ou nulles. Pour des raisons très pratiques, la plupart des algorithmes d'apprentissage ne fonctionnent pas avec des valeurs nulles, il faut, d'une manière ou d'une autre, trouver un moyen de les gérer. Deux stratégies sont couramment recommandées pour traiter les valeurs manquantes :

1. Suppression des observations qui ont des valeurs manquantes **(Dropping en anglais)**. On élimine toutes les instances qui ont des données vides ou nulles.
2. Imputation des valeurs manquantes en fonction d'autres observations où toutes les valeurs sont présentes **(Imputing en anglais)**. Dans l'imputation, il y a plusieurs méthodes de remplissage : On peut remplir en remplaçant les valeurs nulles soit par "**zéro (0**)", soit par la valeur moyenne "**mean**" ou par la valeur moyenne "**median**".

Après avoir essayé les deux techniques de traitement des valeurs manquantes, nous nous rendons compte que la stratégie d'**imputation** nous donne de meilleurs résultats. Pour arriver à cette affirmation, nous avons essayé plusieurs modèles avec leurs différentes évaluations, puis nous comparons les résultats pour décider lequel est meilleur.

En résumé pour nettoyer notre ensemble de données nous avons rempli les valeurs vides et nulles en utilisant la méthode du calcul de la moyenne car elle nous donne de meilleurs résultats toujours en comparant les différents résultats de chaque méthode en vue de sélectionner le meilleur. Nous avons développé une fonction qui fait automatiquement le choix du meilleur résultat. Nous tenons à souligner que notre ensemble de données a eu beaucoup de valeurs nulles, c'est la deuxième difficulté de notre projet après les classes déséquilibrées. La méthode **fiilna** de la classe **Clean** a été utilisée pour remplir les données manquantes de notre ensemble de données. Elle prend en paramètres un jeu de données et la stratégie de remplissage et retourne un jeu de données sans données manquantes. Une fois qu'on a un ensemble de données correctement nettoyé, on peut poursuivre avec la phase de transformation de données.

3.4 Transformation de notre ensemble de données

La transformation de données se repose sur l'ingénierie des caractéristiques **(Feature Engineering en anglais)** qui est une technique qui consiste à créer de nouvelles variables prédictives à partir de celles qui sont déjà disponibles ou à

supprimer des variables impertinentes. La transformation permet soit l'ajout ou la suppression de variables non-étiquetées, mais généralement elle consiste en la création de nouvelles variables surtout quand on dispose d'une petite quantité de variables. La transformation est aussi une tâche précieuse qui permet au scientifique de données d'améliorer la performance d'un modèle puisqu'elle permet de mettre en évidence les informations clés, de se concentrer sur ce qui est important et d'apporter une certaine expertise au domaine.

Dans notre projet, nous avons trouvé 34 variables impertinentes. Par exemple, pendant la phase d'exploration nous avons remarqué que le "**Weight**" est une variable impertinente car elle est très corrélée avec le "**BMI**". La raison est parce que deux variables prédictives ne doivent pas avoir une trop grande corrélation, cela peut biaisé le modèle et nous donne des résultats trompeurs. Pour cette raison, on a décidé de maintenir le **"BMI"** et supprimer le **"Weight"**. Du même coup nous nous sommes rendus compte que le **"RawDX"** et **"GHOA"** stockent des données similaires sauvegardées dans des formes différentes. Et on a maintenu le **"RawDX"** et laissé tomber le **"GHOA"**. La méthode **drop** de la classe **Clean** est utilisée pour supprimer les features non pertinents.

L'ingénierie des caractéristiques est l'étape qui précède la modélisation. Il prend en entrée le jeu de données nettoyé et retourne un jeu de données préparé pour la phase de modélisation. Les algorithmes d'apprentissage prennent en entrée le jeu de données produit par le processus de transformation.

Pour faire l'ingénierie de caractéristiques nous avons procédé de deux manières :

I. Premièrement, les variables **Dx1, Dx2, Dx3** qui sont déjà disponibles dans notre ensemble de données, sont combinés entre elles pour créer un nouvelle variable appelée **degree_dx** (le niveau de diagnostique général du patient). Nous soulignons que **Dx1, Dx2, Dx3** représentent respectivement : premier niveau de diagnostique, deuxième niveau de diagnostique et troisième niveau de diagnostique du patient. En combinant ces trois diagnostiques, nous créons un diagnostique général du patient.
   Logique de création : Pour chaque ligne de l'ensemble de données on regroupe les colonnes **Dx1, Dx2** et **Dx3** de telle sorte qu'elles suivent les quatre logiques suivantes :
   (a) Si les trois sont présentes (c'est-à-dire si leurs valeurs ne sont pas nulles ou vides), alors la nouvelle variable **degree_dx** prend la valeur 3 ;
   (b) si seulement les diagnostiques **Dx1** et **Dx2** sont présentes et la troisième est absente[1], alors **degree_dx** prend la valeur 2 ;

1. Absente veut dire valeur nulle ou vide

(c) si seulement **Dx1** est présente et les deux autres sont absentes, alors **degree_dx** prend la valeur 1 ;

(d) si les trois diagnostiques sont absentes, alors **degree_dx** prend la valeur 2.

À la fin de ce processus, nous conservons la variable résultante **degree_dx** dans notre ensemble de données et nous supprimons **Dx1, Dx2** et **Dx3**.

II. Deuxièmement, nous créons une autre nouvelle variable nommée **medcond** qui détermine les conditions médicales du patient après une arthroplastie. Nous nous basons sur l'ensemble des variables pré-opératoires du patient qui sont au nombre de 46 colonnes (Les lignes 30 à 76 de la table A.1). Ces variables déterminent les conditions pré-médicales du patient. Par exemple la ligne 30 représente les quantités d'hémoglobine blanches, la ligne 33 son niveau de glucose et la ligne 56 s'il est obèse. Référez-vous à la table A.1 si vous voulez avoir plus de connaissances sur les préopératoires.
Logique de création : On attribue la valeur 1 à une colonne pré-opératoire lorsqu'on trouve une valeur non-nulle sinon la valeur zéro est attribuée. à chacune des conditions sauf pour les colonnes suivantes :

- [**PreOpHgb, PreOpGlucose, pulm circ, other neuro, chronic pulm**] pour lesquelles on attribue la valeur 2.
- [**PreOpC, Paralysis, renal failure, liver failure**] pour lesquelles on attribue la valeur 3.

À la fin de ce processus, on fait la somme de toutes les valeurs pré-opératoires, on attribue le résultat à la nouvelle variable **medcond** et on supprime toutes les colonnes pré-opératoires.

À la fin de l'étape de transformation, 2 nouvelles variables sont créées et 48 sont supprimées. La classe **Transform** de notre diagramme de classe fait toutes les transformations que nous venons de décrire dans cette section. La méthode **transform** prend en paramètre le jeu de données nettoyé et les attributs à transformer et elle retourne un jeu de données transformé lequel nous allons utiliser comme entrée dans la phase modélisation.

3.5 Modélisation de notre ensemble données

La modélisation est une technique qui consiste à entrainer un ou plusieurs algorithmes d'apprentissage sur un ensemble de données en vue de déterminer un bon modèle prédictif pour cet ensemble de données. Il est question de décrire comment choisir le meilleur modèle qui s'adapte à la problématique qu'on étudie et les raisons pour lesquelles le modèle choisi est meilleur. Ici, meilleur modèle ne veut

pas dire qu'un algorithme est meilleur qu'un autre mais tout dépend du cas qu'on étudie et les résultats des métriques d'évaluation. Pour comparer les modèles on peut utiliser des métriques de classification telles que précision, rappel, f1-score etc.

Toutes les étapes que nous venons de détailler dans les sections précédentes constituent des étapes préparatoires pour la modélisation. Dans l'étape d'exploration de données, nous nous sommes rendus compte que notre ensemble de données était déséquilibré. Par conséquent, nous ne pouvons pas utiliser les algorithmes de classification simples tel que **LogisticRegression**, **TreeDecisionClassifier**, **LinearClassifier** etc. Nous avons utilisé l'approche des méthodes ensemblistes qui consiste à combiner plusieurs classificateurs simples pour donner un classificateur fort comme nous avons vu à la sous-section 2.3.6 . L'image 3.4 nous donne une approche générale sur le fonctionnement et la construction des méthodes d'ensemble. **Data** représente l'ensemble de données d'origine, (**C1, C2, ... , Cn**) sont les différents algorithmes de classification faibles qu'on combine pour donner en entrée à **Vote Classifier**. **Strong Classifier** est l'algorithme de classification fort qu'on obtient.

Au chapitre 2 traitant le *Contexte théorique*, nous avons vu comment manipuler les données déséquilibrées en appliquant l'une des différentes techniques de ré-échantillonnage **(resampling en anglais)** sur l'ensemble de données d'origine. Nous avons vu les avantages à utiliser la technique de *SMOTE* que les autres. Aussi nous avons brièvement examiner une autre approche, à savoir la méthodologie d'ensemble qui est une manière de combiner plusieurs algorithmes de classification faibles afin d'obtenir un algorithme fort qui soit approprié à notre ensemble de données déséquilibrés.

Pour notre projet nous avons entrainé quatre algorithmes ensemblistes comme on peut constater dans notre classe diagramme 3.1 ; deux d'entre eux sont des algorithmes de boosting, un algorithme stacking et un algorithme de forêt aléatoire. La table 3.3 présente une comparaison des différents résultats obtenus en entrainant ces quatre types d'algorithmes ensemblistes différents. Pour sélectionner le meilleur algorithme, nous avons développé des fonction en Python pour automatiser cette tâche tout en nous basant sur les scores de chaque modèle entrainé. Le meilleur modèle est celui du meilleur score obtenu.

Nous tenons compte aussi des métriques d'évaluation telles que *Precision* et *Recall* pour sélectionner le meilleur modèle. Nous avons vu en détail ces deux métriques d'évaluation dans le chapitre 2. Ces derniers nous permettent de déterminer la performance de notre modèle à classifier les classes positives et celles qui sont négatives. Le plus grand défi de notre projet était d'obtenir une précision au-

Algorithm	**Metrics**		
	Accuracy score	**Precision**	**Recall**
GradientBoostingClassifier	0.83	0.93	0.89
AdaBoostClassifier	**0.85**	**0.93**	**0.90**
StackingClassifier	0.78	0.91	0.84
RandomForestClassifier	0.82	0.91	0.90

TABLE 3.3 – Algorithmes ensemblistes utilisés dans notre projet

dessus de 90% et un rappel au-dessus de 60%. Généralement, dans la pratique on considère qu'une telle précision et qu'un tel rappel sont acceptables pour pouvoir affirmer qu'un modèle performant. Pour notre rappel nous avons obtenu 90%, un très bon résultat par rapport à la valeur moyenne acceptable.

La table 3.3 nous montre les différents résultats obtenus pour chaque algorithme. Pour faire le choix du bon algorithme, notre système d'apprentissage automatique est capable d'évaluer chaque algorithme de façon indépendante et itérative pour déterminer quel est le modèle estimateur, tout en combinant les différents résultats de chaque métrique de classification. Dans ce cas, c'est l'algorithme *AdaBoostClassifier* qui est le modèle estimé meilleur.

Après avoir déterminé le meilleur modèle à partir des processus automatisées. La dernière étape qui nous reste à faire est la sérialisation de notre modèle pour son utilisation future soit dans la phase de déploiement. Parce qu'un modèle doit être capable de prédire de nouvelles données (**online data**). La section qui vient expliquera en détail comment nous avons procédé pour déployer notre modèle c'est-à-dire le rendre accessible à des utilisateurs qui ne connaissent rien en apprentissage automatique.

	AGE_AT_ADMIT	ASA_SCORE	Female	Height	Weight	BMI	PreOpHgb	PreOpCr	PreOpGlucose	Warfarin	Discharge
0	57.073238	1.0	0.0	NaN	NaN	NaN	NaN	NaN	NaN	0.0	1
1	87.006160	2.0	1.0	154.9	60.7	25.297981	NaN	NaN	NaN	0.0	1
2	77.623546	3.0	0.0	177.8	90.7	28.690874	NaN	NaN	NaN	0.0	1
3	69.054073	2.0	1.0	162.5	68.9	26.092308	NaN	NaN	NaN	0.0	1
4	53.396304	2.0	1.0	167.6	83.9	29.868536	NaN	NaN	NaN	0.0	1

Fig. 3.2 – Échantillon de notre ensemble de données

Table of correlation between some features of our dataset

	AGE_AT_ADMIT	ASA_SCORE	Female	Height	Weight	BMI	PreOpHgb	PreOpCr	PreOpGlucose	Warfarin	Discharge
AGE_AT_ADMIT	1	0.081	0.12	-0.22	-0.22	-0.11	-0.2	0.17	0.16	0.044	-0.25
ASA_SCORE	0.081	1	0.027	-0.035	0.037	0.064	-0.21	0.04	0.058	0.03	-0.057
Female	0.12	0.027	1	-0.7	-0.35	0.049	-0.31	-0.24	0.073	-0.029	-0.16
Height	-0.22	-0.035	-0.7	1	0.48	-0.11	0.26	0.24	-0.0078	0.029	0.15
Weight	-0.22	0.037	-0.35	0.48	1	0.81	0.24	0.09	-0.047	0.022	0.063
BMI	-0.11	0.064	0.049	-0.11	0.81	1	0.087	-0.053	-0.012	0.021	-0.023
PreOpHgb	-0.2	-0.21	-0.31	0.26	0.24	0.087	1	-0.2	-0.14	-0.18	0.2
PreOpCr	0.17	0.04	-0.24	0.24	0.09	-0.053	-0.2	1	-0.036	0.06	-0.14
PreOpGlucose	0.16	0.058	0.073	-0.0078	-0.047	-0.012	-0.14	-0.036	1	0.04	-0.24
Warfarin	0.044	0.03	-0.029	0.029	0.022	0.021	-0.18	0.06	0.04	1	-0.11
Discharge	-0.25	-0.057	-0.16	0.15	0.063	-0.023	0.2	-0.14	-0.24	-0.11	1

Fig. 3.3 – Corrélation entre certaines de nos variables

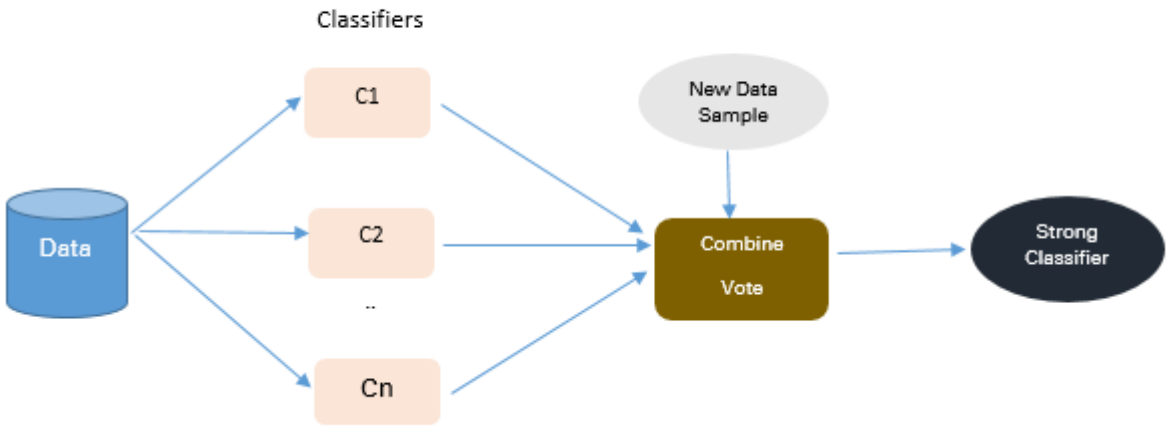

Fig. 3.4 – Approche basée sur la méthodologie des ensemble [33]

4 | Implémentation de l'outil Ortho

Les systèmes modernes d'apprentissage automatique facilitent la construction d'un système de décision de base. Cette facilité, cependant, est un peu décevante. Construire et déployer un premier système de décision a tendance à bien fonctionner, et les résultats peuvent être assez impressionnants pour les bonnes applications. L'ajout des autres systèmes pourraient causer d'étranges interactions qui seraient difficiles à intégrer. Changer une partie du système affecte une autre partie du système, même si des tests isolés peuvent suggérer que cela est impossible.

Le problème est que les systèmes basés sur l'apprentissage automatique peuvent avoir des propriétés très subtiles qui sont très différentes des systèmes logiciels plus traditionnels. En partie, cette différence vient du fait que les sorties des systèmes d'apprentissage automatique ont des comportements beaucoup plus complexes que les composants logiciels typiques. Cela vient aussi en partie du fait de la nature probabiliste des jugements que de tels systèmes sont appelés à faire.

Cette complexité et cette subtilité rendent la gestion de ces systèmes plus délicate que la gestion de systèmes logiques traditionnels, bien modularisés. Les systèmes complexes d'apprentissage automatique peuvent évoluer pour montrer des comportements pathologiques, par exemple changer quoi que ce soit pourrait impliquer qu'on doit tout changer, même s'ils apparaissent superficiellement comme des micro-services bien conçus avec de hauts degrés d'isolation[8].

Compte tenu de cette complexité qui existe au niveau des applications d'apprentissage automatique, nous avons décidé de construire un API Ortho pour faire le déploiement de notre modèle, de telle sorte que s'il faut faire un changement dans la phase d'entrainement du modèle, cela n'affectera pas tout le système en général.

4.1 Construction de l'API Ortho

Les API (Application Program Interfaces) sont des méthodes de communication entre les logiciels développées sur une norme particulière. Une API est une interface logicielle-logiciel qui définit le contrat pour les applications à se communiquer sur un réseau sans l'intervention humaine [7]. De nombreuses entreprises ont leurs propres API publics qui résolvent des problèmes spécifiques pour les développeurs d'applications. En transmettant à leur API un paramètre en entrée, l'utilisateur peut recevoir une sortie sans avoir besoin de savoir (ou de comprendre) comment la tâche sous-jacente est effectuée. Les *API Google Maps* pour localiser un lieu sur une carte, les *API Facebook* pour les jeux ou le partage de contenu et les *API Amazon* pour les informations sur les produits sont quelques exemples d'API très utilisés de nos jours. as MLaaS [20] (Machine Learning as a Service) ou AIaaS [29] (Artificial Intelligence as a Service) sont des exemples d'API d'apprentissage automatique qu'un scientifique de données puisse utiliser pour déployer son modèle. Nous, dans notre projet, avons préféré de développer notre propre API qui soit spécifique à nos besoins.

Pour construire notre API nous avons utilisé le framework web *Django* dont l'architecture est présentée dans la figure 4.1 et le langage de programmation Python. Django est aussi un framework écrit en Python. Voyons une brève description de chaque composante de son architecture :

User. Ce sont les utilisateurs des systèmes développés. Ils utilisent un interface web de Django pour réaliser leurs tâches.

Django. C'est le moteur web de Django. Il se compose de toutes les librairies internes du framework.

URL. C'est un fichier de python contenant tous les URLs utilisés par l'application dont nous développons.

View. C'est la couche metier de toute application Django. Il fait la liaison entre le model et le template.

Model. Le modèle est utilisé pour stocker et maintenir les données. C'est le backend où la base de données est définie. Dans notre cas, le model va être le modèle d'apprentissage automatique que nous avons construit.

Template. Tout ce qui a trait à la présentation. Tout ce que l'utilisateur peut voir.

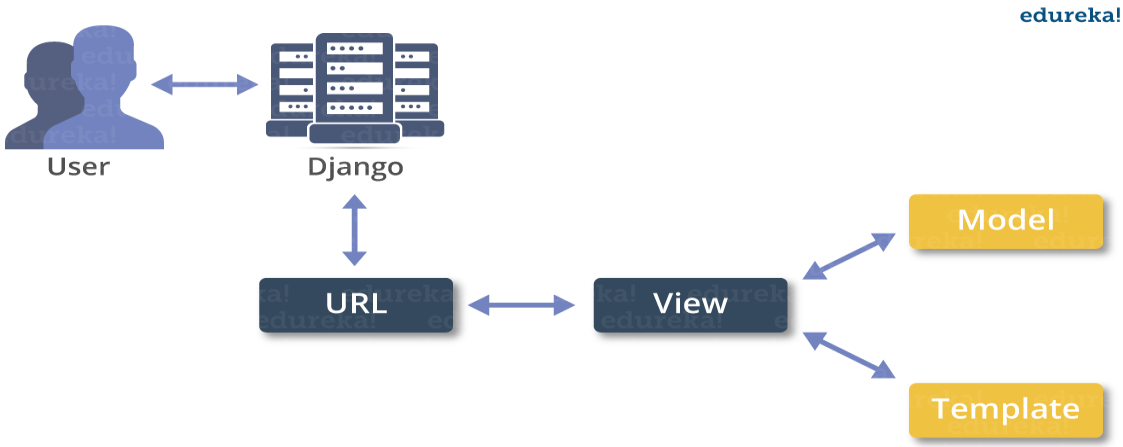

Fig. 4.1 – Architecture de Django [17]

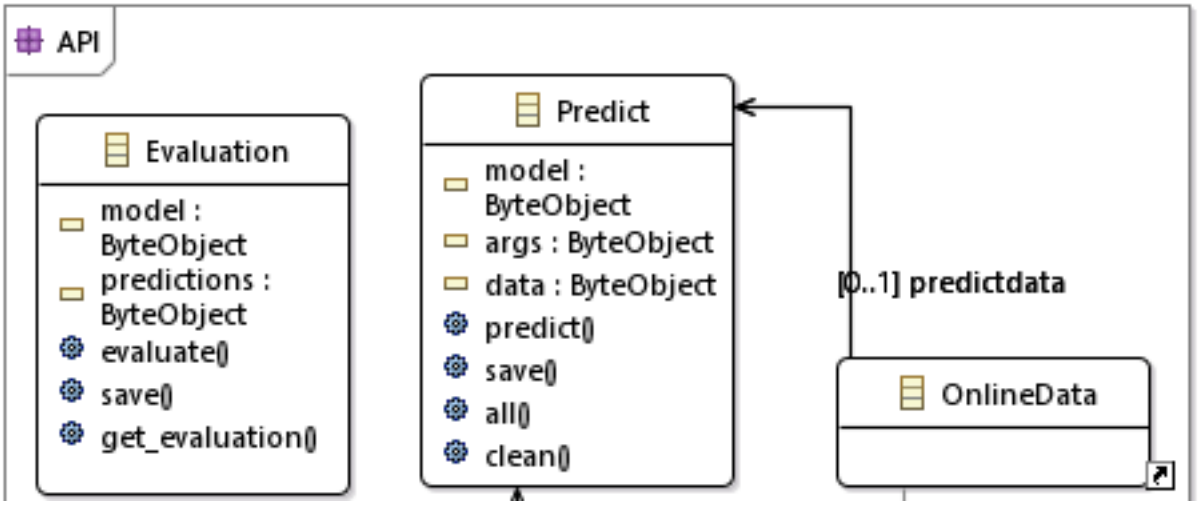

Fig. 4.2 – Diagramme de classe de notre Ortho API

Notre API prend un formulaire web de Django en entrée et retourne un message de décision. Le framework django s'en charge de faire la validation du formulaire avant d'appeler l'API. Si tous les champs du formulaire sont bien validés et toutes les valeurs obligatoires sont présentes, alors l'API prend le formulaire en entrée et fait un pré-traitement des données. S'il trouve des données optionnelles qui sont vides ou nulles, il les remplace par **NaN** et s'il trouve des champs booléens, il les

remplace par **0** si c'est **No** et par **1** si c'est **Yes**.

Le diagramme de classe présenté dans la figure 4.3 permet de répondre à deux des cas d'utilisation que nous avons défini dans le diagramme de cas d'utilisation 1.1. Il nous reste encore deux cas d'utilisation à traiter : **Monitor** et **Manage** , nous les laissons comme travaux futurs ; par contre notre système peut fonctionner normalement dans la pratique sans ces deux fonctionnalités. Voyons brièvement ce que font les deux classes dans ce diagramme de notre API :

Predict. La classe qui répond à l'exigence **Predict** défini à l'introduction. Elle a trois attributs et quatre méthodes. L'attribut **model** est le modèle d'apprentissage automatique désérialisé. L'attribut **args** est un dictionnaire qui stocke la décision rendue par le modèle et un message de prédiction. La méthode **predicit** calcule la prédiction en fonction des données du formulaire. La méthode **save** permet de sauvegarder les prédictions après avoir rendu une décision.

Evaluation. La classe d'évaluation permet de répondre à l'exigence **evaluate** défini dans le diagramme de cas d'utilisation à l'introduction générale. Elle contient deux attributs et trois méthodes. L'attribut **model** obtient le modèle de l'apprentissage automatique construit. L'attribut **predictions** est une liste de toutes les prédictions qui ont déjà traitées par le modèle. L'opération **evaluate** permet de d'évaluer le modèle en fonction de l'ensemble de **prédictions**. L'opération save sauvegarde le résultat de l'évaluation.

La construction de l'API Ortho est un travail d'ingénierie de logiciels, ce qui permet de combiner ensemble deux champs d'études de l'informatique (**Apprentissage automatique et Software Engineering**) tel qu'apparu dans la classe digramme présentée dans la figure 4.3. Il faut relater que les modèles d'apprentissage automatique ont besoin d'une couche d'ingénierie de logiciels pour leur rendre accessible aux utilisateurs qui ne comprennent rien en apprentissage automatique. Dans la vraie vie, ceux qui utilisent les modèles ne connaissent rien ni en apprentissage automatique ni en génie de logiciel. C'est pourquoi dans tout lieu où il y a une équipe de scientifiques de données qui travaillent, il y a aussi une équipe de développement de logiciels.

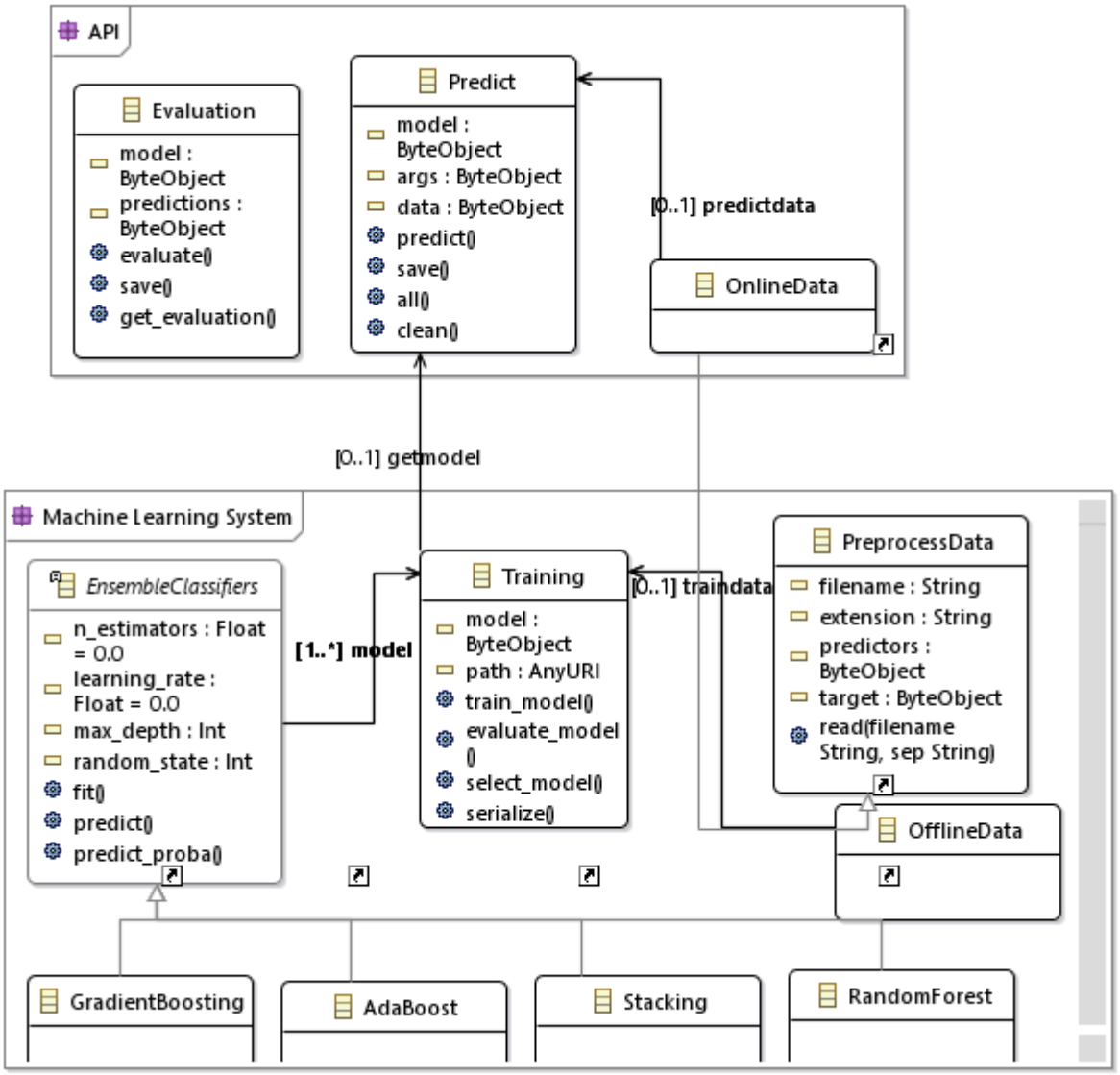

Fig. 4.3 – Connexion entre notre API et notre système d'apprentissage automatique

4.2 Déploiement de notre système

Le système que nous développons a eu deux grandes phases de développement comme mentionné dans l'introduction. En plus, nous avons utilisé Django comme framework web qui a des fonctionnalités prédéfinies que nous utilisons dans notre projet. Maintenant nous allons montrer comment nous combinons tous ces morceaux ensemble pour donner un système. Le diagramme de déploiement présenté à la figure 4.4 donne une vue générale des différentes connexions de chaque système. Voyons une brève description de chaque noeud de notre diagramme de déploiement :

Seattle University Data Server. C'est le serveur de données de l'université Seattle. Ce serveur contient la base de données des patients qui ont subi des arthroplasties aux États-Unis. C'est notre source de données.

Machine Learning System. C'est notre système ML que nous avons développé dans le chapitre 3. Il contient cinq composantes.

Web Repository. C'est le dépôt web qui contient notre modèle d'apprentissage automatique.

Deployment System. Le système de construction de notre API *Ortho* et l'environnement Django, comme décrit dans la section 4.1.

Server web. L'hébergeur du site web Ortho predictions. Le service va être en ligne.

Ce chapitre a détaillé comment nous avons procédé pour implémenter notre outil *Ortho* et le déploiement de notre système complet. C'est la phase d'ingénierie de logiciel de notre projet. Le chapitre qui suit va expliquer comment nous avons fait pour évaluer notre modèle d'apprentissage automatique et notre outil *Ortho* pour savoir si notre système est fiable ou s'il répond aux résultats escomptés.

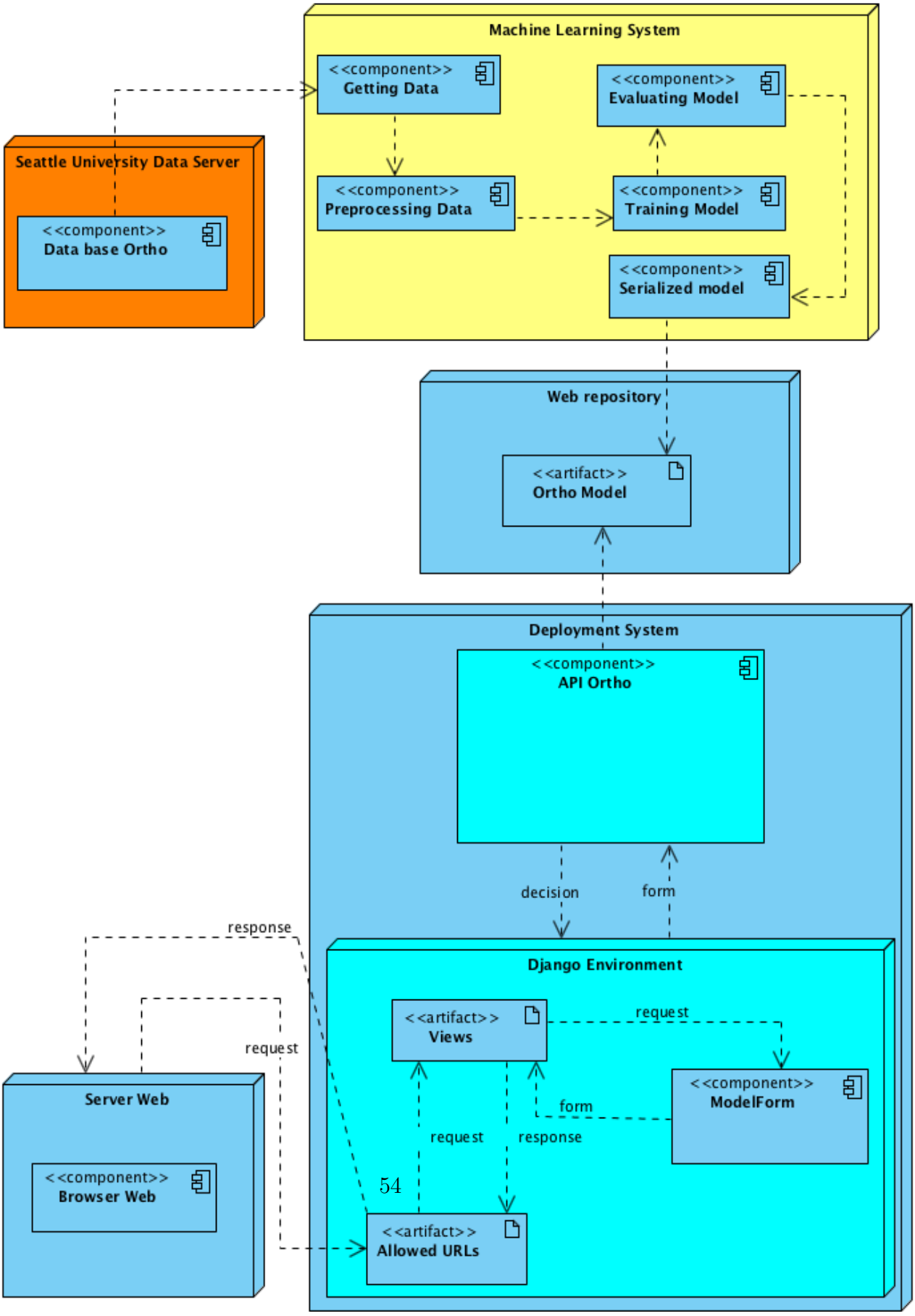

Fig. 4.4 – Diagramme de déploiement de notre système complet

5 | Évaluation

5.1 Évaluation du modèle d'apprentissage automatique

L'évaluation des algorithmes d'apprentissage automatique est une partie essentielle de tout projet. Le modèle peut nous donner des résultats satisfaisants lorsqu'il est évalué à l'aide d'une métrique, par exemple **precision_score**, mais il peut donner de mauvais résultats lorsqu'il est évalué par rapport à d'autres mesures, telles que **recall** ou toute autre mesure de même type. Évaluer un projet d'apprentissage automatique est différent d'évaluer les projets de génie logiciels traditionnels. La différence est que pour évaluer de tels projets, il faut tenir compte des différentes théories d'apprentissage automatique permettant de faire des évaluations sur des modèles. Ce que nous avons déjà dans la sous-section 2.3.3.2 portant sur la "**Mesure de performance des algorithmes de classification**" au chapitre 2 portant sur le "**Contexte théorique**".

Pour évaluer notre modèle, nous avons séparé notre ensemble de données en deux (Train Set et Test Set). Le train set a été utilisé pour construire le modèle et le test set pour faire l'évaluation du modèle. Nous avons séparé le jeu de données comme suit : 80% pour le train set et 20% pour le test set. Puisque notre jeu de données était déséquilibré, nous n'avons pas ré-échantillonné **(resampling en anglais)** notre ensemble de test, parce que les nouvelles données que le modèle va prédire ne seront jamais ré-échantillonné. D'autres raisons de ce choix est que si nous ré-échantillonnons l'ensemble de test, il est fort probable que le modèle soit biaisé, ce qui provoquera de fausses prédictions.

La table 5.1 donne les résultats de l'évaluation de notre modèle sélectionné comme le meilleur. On pourrait constater que cette table est ressemblée à la table 3.3 qui compare les différents algorithmes entrainés dans notre projet. La raison est que pour comparer et sélectionner les différents algorithmes, il faut d'abord évaluer chaque algorithme. C'est comme une évaluation anticipée. Une interpréta-

Model's name	AdaBoostClassifier
Precision score	0.9291845493562232
Recall score	0.9002079002079002
Area under Curve	0.8511029411764706
F1 score	0.914466737064414

TABLE 5.1 – Résultats de l'évaluation de notre modèle

tion sommaire de cette évaluation est que 8% des prédictions faites par le modèle est erroné et 92% sont correctement classées. Pour faire cette interpretation, nous tenons compte du " **Precision score** " de la table 5.1.

5.2 Evaluation de notre API Ortho

Pour évaluer notre API, nous avons simulé un diagramme de séquence [32] qui montre comment les orthopédistes peuvent utiliser l'API pour prédire un retour. Et l'API à son tour se connecte au dépôt de l'apprentissage automatique **(repository machine learning en anglais)** pour charger le modèle sérialisé par le système d'apprentissage automatique. La figure 5.1 est notre diagramme de séquence qui montre les interactions entre les orthopédistes et ses différents objets impliqués dans le scénario de la prédiction.

Nous allons décrire brièvement les différents messages qui permettent à un orthopédiste de réussir une prédiction.

1. **Fill form and press predict.** Quand l'orthopédiste accède à la page de la prédiction, il doit remplir le formulaire et ensuite appuyer sur le bouton Predict pour envoyer le formulaire ;
2. **Validate Form.** Le moteur Django est engagé de faire la validation automatique du formulaire selon les critères qu'on avait précisés dans l'étape de conception du formulaire ;
3. **Form is validated.** Si le formulaire est validé, Django retourne un message que la validation est vraie c'est-à-dire le formulaire est bien rempli ;
4. **Predict decision using the validated form.** Le formulaire rempli est passé à l'API comme paramètre. L'API fait le traitement du formulaire tel que nous avons vu dans la phase de construction de l'API.

5. **Load model.** L'API charge le modèle qui est hébergé dans le repository d'apprentissage automatique. Le repository peut être le serveur de la compagnie ou le cloud ;
6. **Model loaded.** L'API utilise le modèle pour faire la prédiction utilisant le formulaire pré-traité et rendu sous forme d'un dataframe (un objet de la classe DataFrame de la librairie Pandas) ;
7. **Decision predicted.** L'API rend la décision si le patient doit retourner ou rester à l'hôpital ;
8. **A message with the decision.** Une réponse en format de chaine de caractère est rendu à l'utilisateur dans ce cas l'orthopédiste. Cette réponse est le message de la décision.

5.3 Limitations et travaux futurs

Notre projet est loin d'être exhaustif, il reste beaucoup de choses à faire mais les parties essentielles du projet sont réalisées. Nous pouvons considérer tout ce que nous avons fait jusqu'à présent dans ce projet comme étant une première version du système. Pour le moment, on peut utiliser le système pour faire la prédiction et l'évaluation de notre modèle déployé. C'est-à-dire n'importe quel orthopédiste peut utiliser le système pour prédire si son patient doit rester à l'hôpital et s'il doit retourner chez lui après une arthroplastie. Pour que le médecin orthopédiste puisse faire ces prédictions, il doit, préalablement remplir un formulaire web qui contient les champs nécessaires pour compléter un dossier médical.

Comme travaux futurs, nous prévoyons inclure les modules qui gèrent le monitoring et management du modèle. Et aussi la gestion des sessions puisque tous les orthopédistes peuvent avoir accès au module de **prédictions** mais les modules **monitoring** et **management** sont réservés aux orthopédistes qui soient administrateurs du système.

Même s'il nous reste ces travaux à faire mais notre outil peut satisfaire les exigences des orthopédistes.

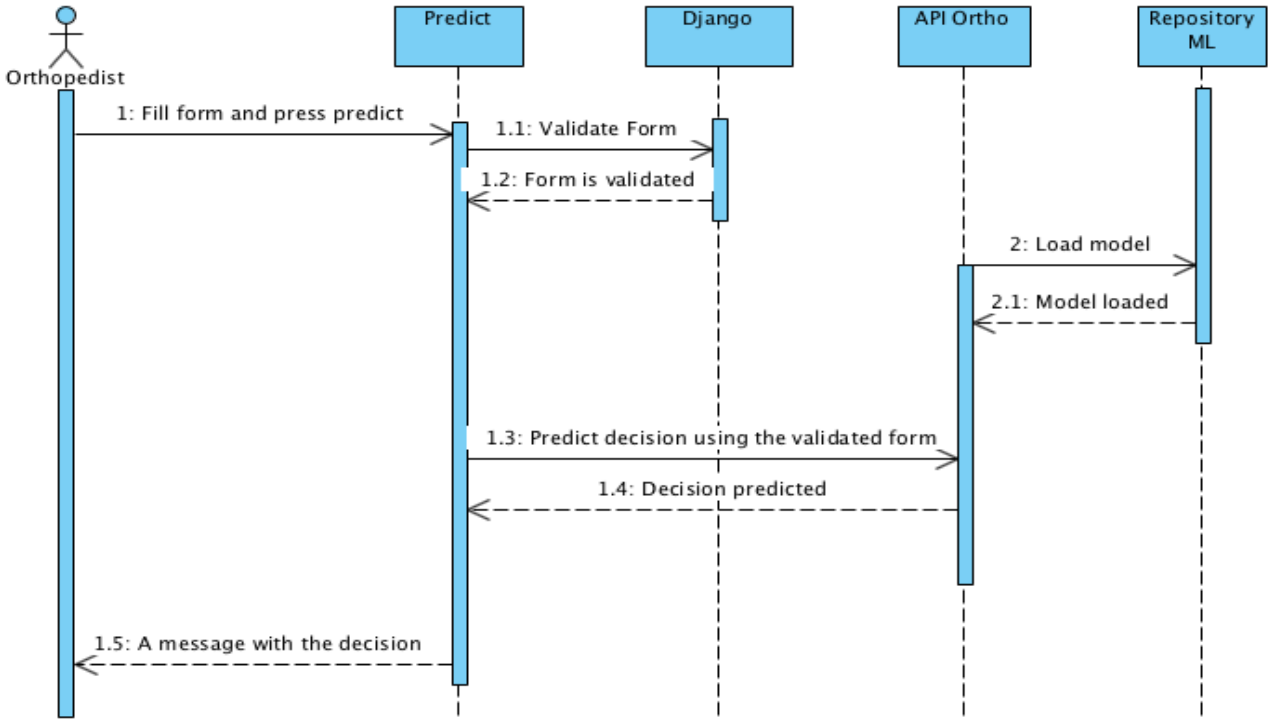

Fig. 5.1 – Diagramme séquence de notre API tool

6 | Travaux connexes

Nous n'avons pas trouvé de travaux connexes directement liés à notre domaine c'est-à-dire aucun organisme n'a encore développé de systèmes liés au risque de retour des patients après une arthroplastie. Par contre nous avons trouvé des projets qui ont des points similaires par rapport à notre projet tels que les classes déséquilibrées, la technique SMOTE, le problème de classification et les algorithmes ensemblistes. Aussi nous avons trouvé des APIs conçus pour être utilisés dans l'apprentissage automatique.

MLlib, une bibliothèque d'apprentissage automatique distribuée de Apache Spark (plateforme populaire adaptée aux tâches d'apprentissage machine itératives) a été présenté par Xiangrui Meng et al. [13]. La bibliothèque cible les paramètres d'apprentissage à grande échelle qui bénéficient du parallélisme des données ou du parallélisme des modèles pour stocker et exploiter des données ou des modèles. MLlib consiste en des implémentations rapides et évolutives d'algorithmes d'apprentissage standard pour les paramètres d'apprentissage communs, y compris la classification, la régression, le filtrage collaboratif, la mise en grappe et la réduction de la dimensionnalité. Il fournit également une variété de statistiques sous-jacentes, d'algèbre linéaire et de primitives d'optimisation. Écrit dans Scala et utilisant des bibliothèques d'algèbre linéaire (basées sur C ++) sur chaque nœud, MLlib inclut des API Java, Scala et Python, et est publié dans le cadre du projet Spark sous la licence Apache 2.0. MLlib fournit une API de haut niveau qui exploite l'écosystème riche de Spark pour simplifier le développement de pipelines d'apprentissage automatique de bout en bout. Dans notre projet, nous avons conçu notre propre API spécifique à nos besoins.

MXNet est une bibliothèque d'apprentissage automatique multi-langages pour faciliter le développement d'algorithmes ML, en particulier pour les réseaux neuronaux profonds. Incorporé dans le langage hôte, il mélange l'expression symbolique déclarative avec le calcul tensoriel impératif pour maximiser l'efficacité et la flexibilité. Il offre une différenciation automatique pour dériver des dégradés. MXNet est efficace en termes de calcul et de mémoire et fonctionne sur divers

systèmes hétérogènes, allant des périphériques mobiles aux clusters GPU distribués. La conception de l'API et l'implémentation du système de MXNet ont été réalisés par Tianqi Chen et al. [12]. Nous avons préféré de développer notre propre API en fonction de nos besoins et notre ensemble de données. En apprentissage automatique, il n'y a pas d'API meilleur que d'autres, tout dépend du contexte de développement qu'on se trouve et la situation à laquelle on fait face. Souvent, il vaut mieux de développer ses propres outils spécifiques et adaptés à ce qu'on veut faire.

Une étude a été réalisée par Lars Buitinck et al. [10] sur l'API scikit-learn et de la manière dont elle mappe les concepts et les tâches d'apprentissage automatique sur les objets et les opérations dans le langage de programmation Python. Ils ont montré comment une API cohérente à travers le paquet rend scikit-learn très utile dans la pratique : expérimenter avec différents algorithmes d'apprentissage est aussi simple que de substituer une nouvelle définition de classe. Grâce à des interfaces de composition telles que Pipelines, Feature Unions et méta-estimateurs, ces blocs de construction simples conduisent à une API puissante et capable d'accomplir une grande variété de tâches d'apprentissage dans une petite quantité de code facile à lire. Par le biais du typage de canard, l'API cohérente conduit à une bibliothèque qui est facilement extensible, et permet aux estimateurs définis par l'utilisateur d'être incorporés dans le flux de travail scikit-learn sans héritage d'objet explicite. Notre API est aussi connecté à l'API scikit-learn pour donner de bons résultats dans notre projet. Par exemple, notre API doit accéder à notre modèle d'apprentissage automatique conçu en utilisant les algorithmes classificateurs de la librairie scikit-learn. Cette techinque produit de bons résultats puisque l'API scikit-learn a été conçu pour être utilisé de cette manière.

Alberto Fernández et al. [9] ont fourni une analyse expérimentale approfondie qui permettra de déterminer le comportement des différentes approches proposées dans la littérature spécialisée. Tout d'abord, ils ont utilisé des schémas de binarisation, c'est-à-dire, un par rapport à un et un par rapport à tous, afin d'appliquer les approches standard à la résolution de problèmes de classes binaires déséquilibrées. Deuxièmement, ils ont appliqué plusieurs procédures ad hoc qui ont été conçues pour le scénario d'ensembles de données déséquilibrés avec plusieurs classes. Leur étude expérimentale inclut plusieurs algorithmes bien connus de la littérature tels que les arbres de décision, les machines à vecteurs de support et l'apprentissage par instance, avec l'intention d'obtenir des conclusions globales à partir de différents paradigmes de classification. Tandis que dans notre projet nous avons utilisé les algorithmes ensemblistes qui donnent de meilleurs résultats que les algorithmes simples utilisés dans cette étude.

Sreejita Ghosh et al. [11] ont réalisé un travail dans le domaine biomédical, un taux de détection élevé de maladies éventuellement rares est généralement hautement souhaitable alors que des erreurs dans la classe majoritaire (par exemple des témoins sains) peuvent être plus acceptables. Par conséquent, l'optimisation de la précision prédictive globale est souvent inadaptée. Ils ont analysé un grand ensemble de données de GC / MS urinaires de 829 contrôles et 68 patients souffrant de l'un des trois troubles stéroïdiens innés. Ils ont utilisé 2 algorithmes comparables capables de gérer de grandes quantités de données manquantes. En outre, ils ont comparé différentes stratégies pour traiter les données fortement déséquilibrées, y compris le sous-échantillonnage, le sur-échantillonnage et l'introduction de coûts par classe. Dans cette étude, ils ont utilisé SMOTE pour équilibrer mais ils n'ont pas utilisé les algorithmes ensemblistes qui sont conçus pour être adaptés aux jeux de données déséquilibrés.

Rukshan Batuwita and Vasile Palade [3] ont démontré que les SVM (Support Vector Machines) pourraient produire des résultats sous-optimaux avec des jeux de données déséquilibrés, malgré c'est une technique d'apprentissage automatique très populaire. Autrement dit, un classificateur SVM formé sur un ensemble de données déséquilibré peut produire des modèles sous-optimaux qui sont biaisés vers la classe majoritaire et ont une faible performance sur la classe minoritaire, comme la plupart des autres paradigmes de classification. Diverses techniques de pré-traitement et d'algorithmique des données ont été proposées dans la littérature pour atténuer ce problème pour les SVM. Il existe des méthodes qui ont utilisé la combinaison des deux méthodes externes et internes pour résoudre le problème de déséquilibre de classe pour les SVM. La méthode hybride de l'ensemble des machines à noyau (HKME) combine un SVM binaire standard et un classificateur SVM à une classe pour résoudre le problème du déséquilibre de classes. La différence avec notre projet, c'est que nous avons utilisé des techniques dédiées pour faire le resampling, non pas des techniques combinées comme proposé dans cet article.

Rok Blagus and Lara Lusa [4] ont appliqué SMOTE (Synthetic Minority Oversampling TEchnique) à des données déséquilibrées de classe à haute dimension (à la fois simulées et réelles) et ont également utilisé des résultats théoriques pour expliquer le comportement de SMOTE. Les principales conclusions de leur analyse sont :

- dans le réglage de faible dimension SMOTE est efficace pour réduire le problème de déséquilibre de classe pour la plupart des classificateurs ;
- SMOTE n'a pratiquement aucun effet sur la plupart des classificateurs formés

aux données de grande dimension ;

- lorsque les données sont de haute dimension SMOTE est bénéfique pour les classificateurs k-NN si la sélection de variables est effectuée avant SMOTE ;
- SMOTE n'est pas bénéfique pour les classificateurs d'analyse discriminante, même dans le cadre de basse dimension ;
- sous-échantillonnage ou, pour certains classificateurs, l'ajustement de coupure est préférable à SMOTE pour les tâches de prédiction de classe de grande dimension.

Dans cette étude, les auteurs ont décrit les avantages d'utiliser SMOTE mais ils n'ont pas proposé les algorithmes qui fonctionnent mieux avec cette technique. Si un scientifique de données utilise *SMOTE* dans sont projets, il ne peut pas utiliser les algorithmes classiques d'apprentissage automatique parce que les résultas peuvent être biaisés facilement. Il faut toujours utiliser les classificateurs fort comme les algorithmes ensemblistes.

7 | Conclusion générale

Nous avons développé un outil qui permet aux orthopédistes américains de déterminer de façon automatique quand un patient doit retourner chez lui après une arthroplastie et quand il doit rester à l'hôpital pour une réadaptation. Le projet était important parce que les compagnies d'assurance de santé américaine ne couvrent pas les frais relatifs à la réadaptation d'un patient qui sont très élevés soit environ $ 30,000 par réadaptation. Ces coûts doivent être couverts par les hôpitaux et les compagnies d'assurance ne couvrent que les arthroplasties qui sont les opérations. Nous avons dit également, il est très fastidieux pour un médecin de vérifier manuellement un dossier patient compte tenu du volume des informations. Grâce à notre outil les docteurs orthopédistes américains peuvent prédire automatiquement la décision à prendre après une arthroplastie.

Pour développer cet outil, nous avons démontré comment nous pouvons combiner les techniques d'apprentissage automatique et celles du génie de logiciel pour aboutir à l'implémentation d'un système complet. Par conséquent, notre projet a été divisé en deux grandes phases qui, elles-mêmes, contiennent plusieurs étapes. Dans la première phase du projet, on a réalisé le cycle de travail complet d'un scientifique de données depuis l'obtention des données jusqu'à la modélisation (étape qui consiste à entrainer différents modèles d'apprentissage automatique en vue de sélectionner le meilleur qui répond à notre problématique). Cette phase se termine par la sérialisation du modèle estimé meilleur en vue de son utilisation future. Dans la deuxième phase, nous avons vu le déploiement de notre modèle d'apprentissage automatique sérialisé. Cette phase est la conception de notre API Ortho. Ce dernier a été conçu en utilisant le framework web Django que nous avons décrit dans le chapitre 4.

Pour chaque phase de notre projet, nous avons construit un modèle d'UML expliquant les différentes exigences que notre système doit satisfaire et nous avons donné une brève explication de chaque exigence dans l'introduction générale. Dans les chapitres 3 et 4 nous avons dessiné deux diagrammes de classe, un pour chaque diagramme de cas d'utilisation dans le but d'expliquer les différentes opérations

nécessaires pouvant satisfaire les exigences que nous avons décrites. Nous avons vu un diagramme de déploiement permettant de connecter les différents noeuds facilitant le déploiement du système.

Difficultés rencontrées. Les développements des projets d'apprentissage automatique peuvent avoir des problèmes différents l'un de l'autre. Les facteurs qui déclenchent ces problèmes peuvent être liées soit au développement technique du projet, soit à l'administration ou aux interdictions légales. Pour réaliser ce projet de stage, nous n'étions pas exempts de ce problème, nous avons fait face à des difficultés techniques et parfois même administratives comme tout autre projet dans un sens général. La liste suivante donne une vue générale de quelques-unes que nous avons rencontrées :

- La première difficulté que nous avons eu c'est de combiner l'apprentissage automatique et le génie de logiciel. Puisqu'au début du stage je n'avais pas eu de connaissances en apprentissage automatique et les logiciels pertinents ;
- choisir la meilleure technique de ré-échantillonnage (resampling) est une difficulté car aucune technique n'est supérieure à d'autres, tout dépend de la problématique étudiée. Pour choisir le SMOTE, il a fallu qu'on essaye plusieurs techniques et comparer les résultats pour déterminer le meilleur en fonction de notre jeu de données ;
- le choix de l'algorithme était une autre difficulté puisque les algorithmes traditionnels ne fonctionnent pas sur un jeu de données déséquilibré. Même pour choisir un algorithme ensembliste est difficile car chacun a une approche différente l'une de l'autre. Par exemple les algorithmes boosting sont séquentiels tandis que les bagging sont avec remplacement. Donc le choix devient difficile ;
- l'ensemble de données était très petit pour faire l'entraînement des modèles. En apprentissage automatique plus votre ensemble de données est petit, plus votre modèle n'est pas fiable. Trouver de nouvelles données est difficile puisque ça doit prendre six mois environ à cause des démarches administratives à entreprendre ;
- les données médicales sont des données très sensibles. Avoir accès à ces données est difficile car c'est interdit par la loi sans autorisation.

Leçons apprises. Puisque nous soutenons que nous avons rencontré des problèmes dans le développement de notre projet, il est primordial de faire ressortir les différentes leçons que nous avons apprises à partir de ces difficultés. Voici deux grandes leçons que j'ai apprises au cours du développement de ce projet :

- **Combinaison de plusieurs spécialités.** Nous avons appris qu'il est possible de combiner plusieurs spécialités en informatique dans la réalisation d'un même projet. Par exemple un API développé en Software engineering peut être utilisé pour connecter à des modèles développés en apprentissage automatique. On peut aussi utiliser les diagrammes UML pour décrire un système d'apprentissage automatique ou n'importe quel autre système en général. De même que les sciences ne sont pas isolées entre elles, de même aussi les spécialités informatiques ne sont pas isolées entre elles ;
- **Technologies apprises.** Au commencement de notre stage, nous ne connaissions rien ni en apprentissage automatique ni en langage Python. Maintenant nous savons utiliser la majorité des librairies d'apprentissage automatique développées en Python telles que **sckit-learn**, **mlxtend**, **pandas**, **matplotlib**, **numpy**, **scipy** etc. Nous avons appris aussi à développer des Web API en Django, un framework web en Python.

Notre outil va être déployé sur le serveur web de la compagnie pour laquelle nous faisons le stage (ML+). L'outil va être en ligne et son utilisation sera gratuite pour tous les orthopédistes américains qui veulent uniquement prédire un retour. Mais ceux qui veulent évaluer, suivre et gérer un modèle doivent être d'abord administrateur du système. Pour être administrateur, il faut contacter l'équipe de ML+ et vous aurez un compte qui vous donnera accès ; néanmoins certaines conditions peuvent être appliquées tel que un frais unique. Le code de source de notre projet est disponible sur Github et le lien pour y accéder est le suivant : https ://github.com/top1986/NewIntershipProject.

A | Annexe A

ID	Feature name	datatype	non-null	null	Comments
1	Side	Categorical	2718	0	Détermine si c'est à gauche ou à droite du patient a eu son arthroplastie. Deux valeurs possibles : R **(pour Right en anglais)** et **L (Pour Left en anglais)**
2	Procedure	Boolean	2718	0	Colonne de type booléen qui indique si le patient a eu une procédure dans le passé
3	RawDx	Categorical	2718	0	Diagnostique brute du patient
4	PriorContralateral	Boolean	2703	15	Colonne de type booléen qui indique si le patient souffre de *PriorContralateral*
5	TSA	Boolean	2718	0	Colonne de type booléen qui indique si le patient souffre de *TSA*
6	RTSA	Boolean	2718	0	Colonne de type booléen qui indique si le patient souffre de *RTSA*
7	Tendon Transfer	Boolean	2718	0	Colonne de type booléen qui indique si le patient souffre de *Tendon Transfer*
8	Dx1	Categorical	2718	0	Représente diagnostique 1
9	Dx2	Categorical	176	2542	Représente diagnostique 2
10	Dx3	Categorical	2	2716	Représente diagnostique 3
11	GHOA	Boolean	2718	0	Colonne de type booléen qui indique si le patient souffre de *GHOA*

12	AVN	Boolean	2718	0	Colonne de type booléen qui indique si le patient souffre de *AVN*
13	RCT	Boolean	2718	0	Colonne de type booléen qui indique si le patient souffre de *RCT*
14	FailedRCR	Boolean	2718	0	Colonne de type booléen qui indique si le patient souffre de *FailedRCR*
15	ComboRCT	Boolean	2718	0	Colonne de type booléen qui indique si le patient souffre de *ComboRCT*
16	RCTA	Boolean	2718	0	Colonne de type booléen qui indique si le patient souffre de *RCTA*
17	IA	Boolean	2718	0	Colonne de type booléen qui indique si le patient souffre de *IA*
18	PTA	Boolean	2718	0	Colonne de type booléen qui indique si le patient souffre de *PTA*
19	PHFx	Boolean	2718	0	Colonne de type booléen qui indique si le patient souffre de *PHFx*
20	PHFxSequelae	Boolean	2718	0	Colonne de type booléen qui indique si le patient souffre de *PHFxSequelae*
21	Other	Boolean	2718	0	Colonne de type booléen qui indique si le patient souffre de *Other*
22	AGE_AT_ADMIT	Numerical	2718	0	L'âge du patient
23	ASA_SCORE	Numerical	2716	2	Nous n'avons pas trouvé l'explication de ces colonnes
24	Gender	Boolean	2718	0	Colonne de type booléen qui indique si le patient est un *Gender*
25	Female	Boolean	2718	0	Colonne de type booléen qui indique si le patient est un *Female*
26	Discharge	Boolean	2718	0	Colonne de type booléen qui indique si le patient doit retourner chez lui
27	Height	Numerical	2293	425	La hauteur du patient

28	Weight	Numerical	2301	417	Le poids du patient
29	BMI	Numerical	2283	435	L'indice de masse corporelle (IMC)- BMI en anglais
30	PreOpHgb	Numerical	272	2446	Hémoglobines blanches du patient
31	PreOpCr	Numerical	255	2463	Conditions pré-opératoires du patient
32	PreOpALC	Numerical	157	2561	Conditions pré-opératoires du patient
33	PreOpGlucose	Numerical	253	2465	Glucose du patient
34	arrhythmias	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *arrhythmias*
35	valvular	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *valvular*
36	pulm circ	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *pulm circ*
37	PVD	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *PVD*
38	HTNw/oCx	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *HTNw/oCx*
39	HTNw/Cx	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *HTNw/Cx*
40	Paralysis	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *Paralysis*
41	other neuro	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *other neuro*
42	chronic pulm	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *chronic pulm*

43	DMw/oCx	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *DMw/oCx*
44	DMw/Cx	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *DMw/Cx*
45	hypothyroid	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *hypothyroid*
46	renal failure	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *renal failure*
47	liver failure	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *liver failure*
48	PUD	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *PUD*
49	AIDS	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *AIDS*
50	L0oma	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *L0oma*
51	Mets	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *Mets*
52	solidCaw/oMets	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *solidCaw/oMets*
53	RA/CVD	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *RA/CVD*
54	coagulopathy	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *coagulopathy*
55	obesity	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *obesity*

56	weightLoss	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *weightLoss*
57	fluidElectrolyte	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *fluidElectrolyte*
58	bloodLossAnemia	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *bloodLossAnemia*
59	DeficiencyAnemia	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *DeficiencyAnemia*
60	EtOH	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *EtOH*
61	Drugs	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *Drugs*
62	Psychosis	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *Psychosis*
63	Depression	Boolean	2092	626	Colonne de type booléen qui indique si le patient souffre de *Depression*
64	Plavix	Numerical	2482	236	Nous n'avons pas trouvé l'explication de ces colonnes
65	Dabigatran	Boolean	2482	236	Colonne de type booléen qui indique si le patient souffre de *Dabigatran*
66	Enoxaparin	Numerical	2482	236	Nous n'avons pas trouvé l'explication de ces colonnes
67	Rivaroxaban	Numerical	2482	236	Nous n'avons pas trouvé l'explication de ces colonnes
68	Warfarin	Numerical	2482	236	Nous n'avons pas trouvé l'explication de ces colonnes
69	PreOpNarcotic	Boolean	900	1818	Colonne de type booléen qui indique si le patient souffre de *PreOpNarcotic*

70	PreOpBloodThinner	Boolean	900	1818	Colonne de type booléen qui indique si le patient souffre de *PreOpBloodThinner*
71	PreOpSteroids	Boolean	900	1818	Colonne de type booléen qui indique si le patient souffre de *PreOpSteroids*
72	PreOpInsulin	Boolean	900	1818	Colonne de type booléen qui indique si le patient souffre de *PreOpInsulin*
73	PreOpDMMeds	Boolean	900	1818	Colonne de type booléen qui indique si le patient souffre de *PreOpDMMeds*
74	Oxycodone	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de l'*Oxycodone*
75	Hydrocodone	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de l'*Hydrocodone*
76	Vicodin	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *Vicodin*
77	Percocet	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *Percocet*
78	Oxycontin	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de l'*Oxycontin*
79	Dilaudid	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *Dilaudid*
80	Aspirin	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de l'*Aspirine*
81	Warfarin.1	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *Warfarin.1*
82	Coumadin	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *Coumadin*

83	Plavix.1	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *Plavix.1*
84	Clopidogrel	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *Clopidogrel*
85	Prednisone	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *Prednisone*
86	insulin	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *insulin*
87	humulin	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *humulin*
88	lantus	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *lantus*
89	lispro	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *lispro*
90	aspart	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *aspart*
91	metformin	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *metformin*
92	glipizide	Boolean	900	1818	Colonne de type booléen qui indique si le patient avait pris de *glipizide*

TABLE A.1 – Notre ensemble de données complet

Bibliographie

[1] D. Amina and D. Soumia. Etude comparative des methodes ensemblistes de classification des donnes medicales. *Universite Abou Bakr Belkaid de Tlemcen*, 2017.

[2] B. Bashinskaya, R. M. Zimmerman, B. P. Walcott, and V. Antoci. Arthroplasty utilization in the united states is predicted by age-specific population groups. *ISRN Orthopedics*, 2012.

[3] R. Batuwita and V. Palade. Class imbalance learning methods for support vector machines. *Singapore-MIT Alliance for Research and Technology Centre*, 2012.

[4] R. Blagus and L. Lusa. Smote for high-dimensional class-imbalanced data. *BMC Bioinformatics*, February 2013.

[5] M. Bramer. *Principles of data mining*. Springer eBooks, 2016.

[6] P. K. Chan and S. J. Stolfo. Toward scalable learning with non-uniform class and cost distributions : A case study in credit card fraud detection. *American Association for Artificial Intelligence*, 1998.

[7] B. De. *API Management : An Architect's Guide to Developing and Managing APIs for Your Organization*. Springer Science+Business Media New York, 2017.

[8] T. Dunning and E. Friedman. *Machine Learning Logistics*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2017.

[9] A. F. et al. Analysing the classification of imbalanced data-sets with multiple classes : Binarization techniques and ad-hoc approaches. *ELSEEVIER*, 2013.

[10] L. B. et al. Api design for machine learning software : experiences from the scikit-learn project. *arXiv*, 2013.

[11] S. G. et al. Comparison of strategies to learn from imbalanced classes for computer aided diagnosis of inborn steroidogenic disorders. *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2017.

[12] T. C. et al. Mxnet : A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv*, 2015.

[13] X. M. et al. Mllib : Machine learning in apache spark. *Journal of Machine Learning Research*, 2016.

[14] Y. T. et al. Spam sender detection with granular svm on highly imbalanced mail server behavior data. *International Conference on Artificial Intelligence and Pattern Recognition*, April 2008.

[15] A. C. M. . S. Guido. *Introduction to machine learning with Python*. O'REILY, 2016.

[16] T. M. K. . A. N. Jason Van Hulse. Experimental perspectives on learning from imbalanced data. *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[17] A. Johari. Django tutorial – web development with python django framework. https ://www.edureka.co/blog/django-tutorial/, September 2017.

[18] C. J. Lavernia, V. H. Hernandez, and M. D. Rossi. Payment analysis of total hip replacement. *Centers for Medicare and Medicaid Services*, pages http ://www.larkinhospital.com/larkinorthopedics/wp–content/uploads/2014/01/Payment–Analysis–of–Total–Hip–Replacement1.pdf, 2007.

[19] S. Marsland. *Machine Learning An algorithme perspective*. Chapman & Hall/CRC, 2009.

[20] K. G. . M. A. C. Mauro Ribeiro. Mlaas : Machine learning as a service. *International Conference on Machine Learning and Applications*, 2015.

[21] W. J. H. Michael Masaracchio and al. Timing of rehabilitation on length of stay and cost in patients with hip or knee joint arthroplasty : A systematic review with metaanalysis. *PLOS ONE*, 2017.

[22] R. Miles and K. Hamilton. *Learning UML 2.0*. Sebastopol, CA : O'Reilly c2006, 2006.

[23] T. Mitchell. *Apprentissage artificiel, concepts et algorithmes*. Groupe Eyrolles, 2003.

[24] T. M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineer ing/Math ; (March 1, 1997), 1997.

[25] Openclassrooms. Initiez-vous au machine learning. https ://openclassrooms.com/courses/initiez-vous-au-machine-learning/comment-resoudre-un-probleme-de-data-science, 2014.

[26] M. M. e. J.-L. R. Pirmin Lemberger, Marc Batty. *Big Data et Machine Learning*. Maitrise d'ouvrage des projets informatiques, 2014.

[27] M. K. A. . C. Ram. A review on machine learning : Trends and future prospects. *Research Cell : An International Journal of Engineering Sciences*, 25(63019) :http ://ijoes.vidyapublications.com/, November 2017.

[28] Éric Biernat et Michel Lutz. *Data science : Fondamentaux et études de cas*. Yann LeCun, 2015.

[29] Z. U. Q. . T. D. Sonia Shahzadi, Muddesar Iqbal. Infrastructure as a service (iaas) : A comparative performance analysis of open-source cloud platforms. *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks*, 2017.

[30] M. J. Towler. Confusion matrix. https ://alearningaday.com/2016/09/14/confusion-matrix/, September 2014.

[31] S. Tuffery. *Data mining and statistics for decision making*. Rod Riesco, 2011.

[32] B. Unhelkar. *Software Engineering with UML*. OCRC Press, 2018.

[33] A. Vidhya. How to handle imbalanced classification problems in machine learning ? https ://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/, 2011.

[34] F. C. . A. Vinciarelli. *Machine Learning for Audio, Image and Video Analysis*. British Library Cataloguing, 2008.

[35] Wikipedia. Apprentissage automatique. https ://fr.wikipedia.org/wiki/Apprentissage_automat

[36] H. K. Zouari. *Contribution à l'évaluation des méthodes de combinaison parallèle de classifieurs par simulation*. Laboratoire PSI - FRE CNRS 2645, Décembre 2004.