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+

5 juillet 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 le stage à l'entreprise ML+ 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 des données historiques des entreprises.

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 machine learning 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. [**?**] 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 dans États Unis». Dans un rapport publié en Avril 2007 par les orthopédistes Steven M Kurtz, Kevin Ong, Edmund Lau et Michael T Halpern [**?**], 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 à 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 de machine learning (ML) 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 un 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 [**?**], 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 (Opération ayant pour but de rétablir la forme et la mobilité d'une articulation abîmée ou bloquée.). Il revient à l'hôpital de couvrir ces coûts qui sont très élevés. Selon le docteur orthopédiste Jonah Hebert-Davies qui a fait ses études de spécialité en Orthopédie à l'université de Seattle, Washington aux États-Unis [1], les coûts estimés sont entre

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

15,000 $ à 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 (maladie qui touche les articulations et caractérisée par la douleur et la difficulté à effectuer des mouvements articulaires [**?**]) touche environ 27 millions d'adultes aux États-Unis [**?**]. 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 (46) colonnes pré-opératoires (C'est-à-dire les conditions de santé du patient avant une opération arthroplastie) doivent être prises en compte. Pour calculer les conditions médicales, le médecin orthopédiste doit additionner toutes les valeurs des colonnes pré-opératoires après avoir substitué ces valeurs soit par un(1), soit par deux(2) ou par trois(3) comme expliqué à la section de feature engineering. En plus de calculer les conditions médicales, il y a d'autres calculs à faire comme déterminer le niveau diagnostique du patient en se basant sur les trois niveaux diagnostiques existant dans l'ensemble de données. Toutes ces démarches sont expliquées en détail dans la section de feature engineering.

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-là qui aidera les orthopédistes américains à prendre cette décision de façon automatique et sans perdre de temps.

1.2 Ingénierie des exigences

Nous supposons qu'il existe déjà un jeu de données disponibles contenant toutes les informations touchant un grand nombre de patients et que chaque échantillon de ce jeu de données est un dossier patient. 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 composé de patients qui font l'objet d'une décision de retour. Si toutes ces conditions sont réunies, nous pouvons profiter des différentes technologies de la science de données, du marching learning 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. Le machine learning qui est un sous-ensemble de la science de données permet de modéliser à partir du même ensemble de données. Dans le chapitre 2, nous expliquons le cycle de vie du travail du data scientist en détail. Ce que nous proposons comme solution dans notre projet. Nous avons réalisé 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 machine learning et la deuxième est le déploiement du modèle construit, cette phase consiste en la création d'un API nommé Ortho, c'est une phase d'ingénierie de logiciels. Pour les deux phases, nous avons crée deux diagrammes, un diagramme de cas d'utilisation qui donne une vision globale du comportement fonctionnel de notre notre système et un diagramme d'états qui explique le processus du développement du modèle. Nous donnons une brève explication de chaque diagramme et leurs différents composants.

Nous commençons d'abord par le diagramme de cas d'utilisation. Il consiste en la phase de déploiement du modèle. Ce diagramme explique comment les orthopédistes peuvent utiliser notre système. la figure 1.1 représente une vue 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.

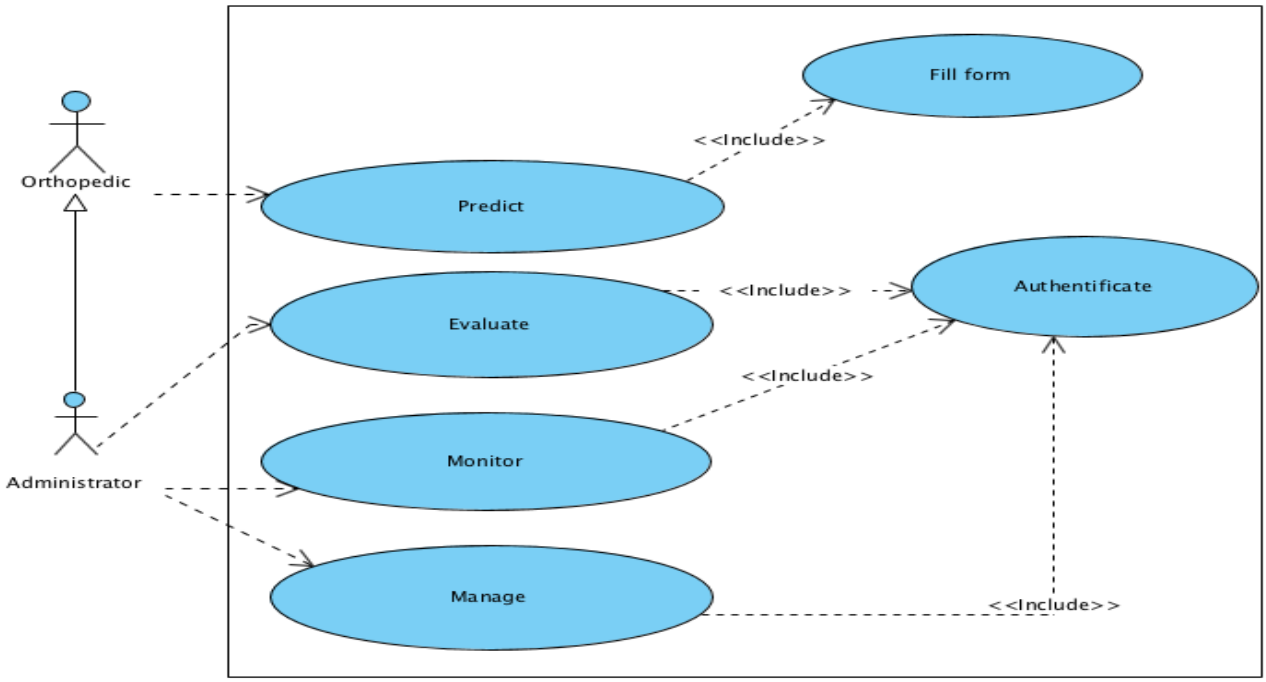

Fig. 1.1 – A use-case diagram for describing our API Ortho

Nous poursuivons avec le diagramme d'états présenté dans la figure 1.2. Il décrit les différents états qui constituent le développement de notre modèle ML. Voici un bref résumé de ce fait chaque état :

1. **API Ortho.** L'API fait la connexion entre le système du machine learning et le système du déploiement. Il accède au modèle qui a été sérialisé.

2. **Gather.** Permet de récupérer les données du dataset ;
3. **Clean.** Permet de nettoyer les données en imputant les données nulles et en supprimant les variables que ne sont pas pertinentes ;
4. **Explore.** Permet d'explorer les données pour voir les différentes corrélations qui existent entre les variables et l'étiquette ;
5. **Transform.** Création de nouvelles variables en fonction de celles qui existent déjà dans l'ensemble des données.
6. **Preprocess.** Un pré-traitement des données avant de les envoyer à l'algorithme de machine learning. Par exemple, les algorithmes de machine learning ne traitent pas les textes, il faut les convertir en valeurs numériques avant de les envoyer aux algorithmes.
7. **Train.** C'est la phase d'entrainement de modèles pour déterminer lequel est meilleur pour notre ensemble de données.
8. **Serialize.** Sauvegarde le meilleur modèle pour son utilisation future.

La suite de ce rapport est composé de plusieurs chapitres et chaque chapitre a des sections, subsections etc. Le chapitre 2 donne un contexte théorique des différents éléments qui sont importants pour notre projet. Le chapitre 3 décrit en détail la construction de notre modèle machine learning. Le chapitre 4 décrit la création de l'outil Ortho qui fait la connexion avec le modèle machine learning. Le chapitre 5 montre comment nous évaluons les différents évaluations morceaux de notre système. Le chapitre 6 détaille les travaux connexes liés au développement d'API machine learning, la technique de re-échantillonnage, aux algorithmes ensemblistes etc. Le chapitre 7 fait la conclusion de notre projet.

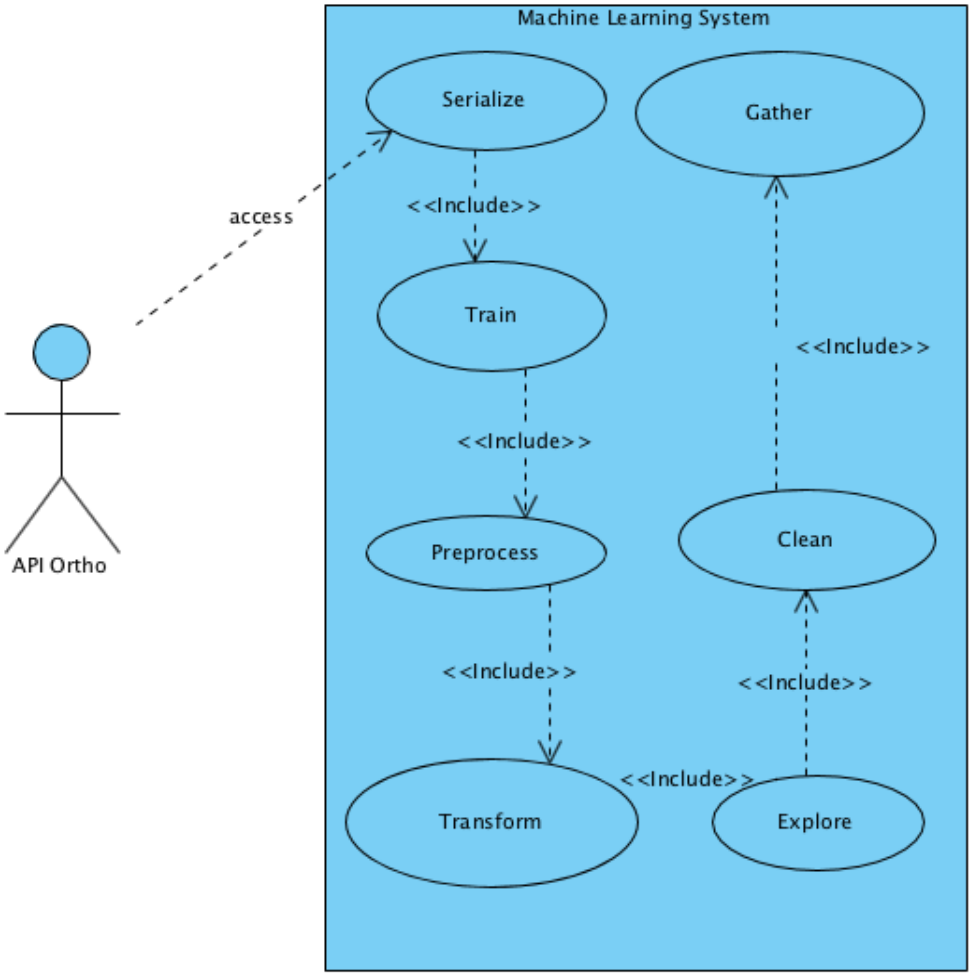

Fig. 1.2 – A use-case diagram for describing our Machine Learning System

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 ces genres 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 nous mettons l'accent sur un sous-ensemble de tous les concepts jugés pertinents pour la réalisation de notre projet. Par exemple, le machine learning est un champ de recherche informatique de plusieurs décennies de travaux de recherche, il est impossible de d'aborder dans un seul chapitre tout ce qui a trait au ML, par contre nous nous concentrons 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 de machine learning 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 le machine learning 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 traite du problème de classification en ML ; puisque notre projet était un problème de classification, nous décidons de le consacrer une section indépendante. La cinquième et dernière se porte sur les classes déséquilibrées car notre ensemble de données était déséquilibrée.

2.2 Sciences des données

On ne peut pas parler de machine learning 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 ce objectifs.

Les sciences de données peuvent être considérées comme un jonction de l'informatique et les statistiques. Du coté de l'informatique, c'est la modélisation et tandis que du coté des statistiques, c'est l'exploration. La modélisation est l'utilisation d'un algorithme d'apprentissage sur un ensemble de données et l

L'objectif du « data scientist » (expert en données massives) est de produire des méthodes (automatisées, autant que possible) de tri et d'analyse de données de masse et de sources plus ou moins complexes ou disjointes de données, afin d'en extraire des informations utiles ou potentiellement utiles. Le métier de data scientist est apparu pour trois raisons principales [**?**] :

- l'explosion de la quantité de données produites et collectées par les humains ;
- l'amélioration et l'accessibilité plus grande des algorithmes de machine learning ;
- l'augmentation exponentielle des capacités de calcul des ordinateurs.

Le cycle de travail du data scientist comprend notamment :

- la récupération des données utiles à l'étude ;
- le nettoyage des données pour les rendre exploitables ;
- une longue phase d'exploration des données afin de comprendre en profondeur l'articulation des données ;
- la modélisation des données ;
- l'évaluation et interprétation des résultats ;
- la conclusion de l'étude : prise de décision ou déploiement en production du modèle.

La figure 2.1 nous donne une vue générale du travail d'un data scientist. Étant stagiaire en science de données, nous avons suivi minutieusement, dans le cadre de notre projet, le cycle de travail complet du data scientist. 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.

Deux composantes sont nécessaires pour pouvoir commencer à se demander si

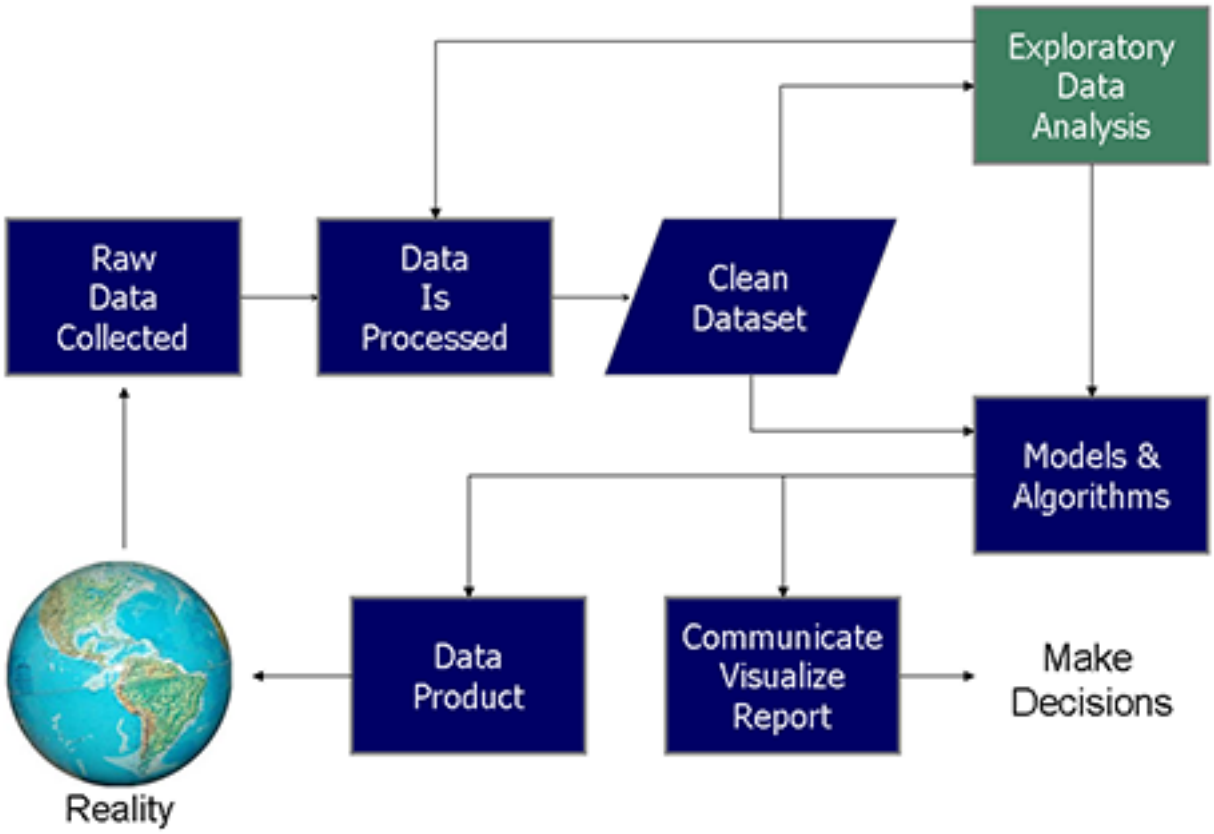

Fig. 2.1 – Work cycle of data scientist [**?**]

la data science peut, oui ou non, apporter de la valeur et aider à la résolution d'un problème : des **données** et une **problématique** bien définie [**?**].

2.2.1 Données.

Les données constituent la ressource principale pour qu'un data scientist puisse effectuer son travail correctement. Nos données proviennent du département orthopédique de l'université de Seattle, Washington, États-Unis. Elles sont au nombre de 2718 files et 92 colonnes. Nous les utilisons comme données historiques (*historical data, en anglais*) dans le cadre du développement de notre modèle de machine learning.

2.2.2 Problématique.

La problématique de notre projet, comme nous l'avons vu dans l'introduction, est d'aider à décider si un patient doit rester à l'hôpital ou rentrer chez lui après une arthroplastie.

Au sein de ce cycle, le machine learning désigne l'ensemble des méthodes de modélisation statistique à partir des données, et se situe bien au coeur du travail du data scientist. Dans la sous-section qui suit nous allons aborder le machine learning, car le gros de notre travail est de construire un model machine learning d'aide à la décision. Nous verrons une brève introduction sur ce quoi le machine learning, 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 ils peuvent visiter wikipedia ou n'importe quelle autre page pour avoir des idées générales sur ce point.

2.3 Vue d'ensemble sur Machine Learning

Dans la section précédente, nous avons pu y voir plus clair sur le cycle global de travail du data scientist. Nous allons maintenant parler du machine learning dans cette section, c'est à dire la modélisation des données. Nous utilisons le machine learning 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 c'est parce que leur logiciel de machine learning a trouvé comment classer les pages. Lorsque Facebook ou l'application photo d'Apple reconnaît nos amis dans nos images, c'est aussi du machine learning. Chaque fois que nous lisons notre courrier électronique 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 courrier non-spam. Donc, c'est du machine learning. C'est la science qui consiste à

apprendre les ordinateurs sans être explicitement programmés. La figure 2.2 nous montre en grosso modo les techniques du machine learning, à un très haut d'abstraction.

Le machine learning 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 [**?**].

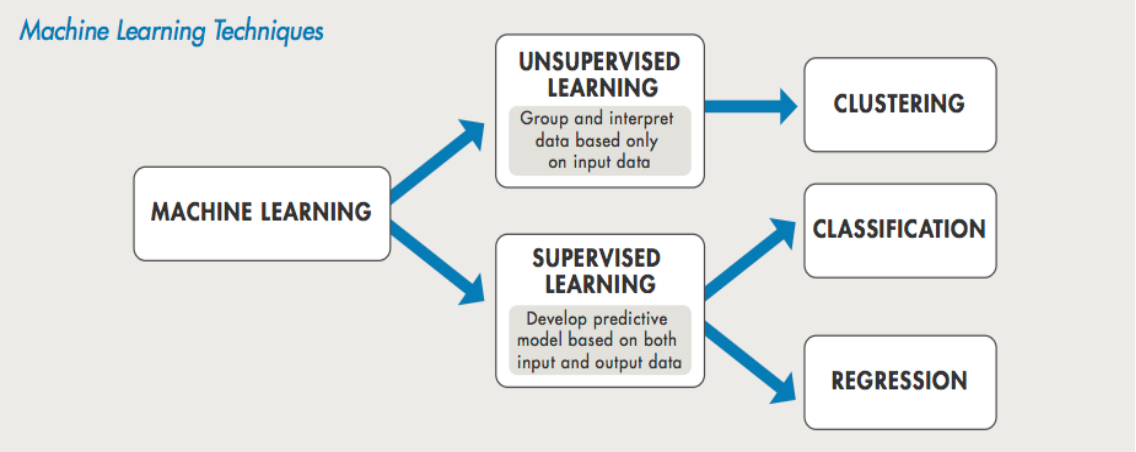

Fig. 2.2 – Machine learning representation [**?**]

Les algorithmes d'apprentissage peuvent se catégoriser selon le mode d'apprentissage qu'ils emploient :

2.3.1 Apprentissage supervisé

L'apprentissage supervisé (supervised learning en anglais) est une technique d'apprentissage automatique où l'on cherche à produire automatiquement des règles à partir d'une base de données d'apprentissage contenant des « exemples » (en général des cas déjà traités et validés).

Une base de données d'apprentissage (ou ensemble d'apprentissage) est un ensemble de couples entrée-sortie $(x_n, y_n)_{1 \leq n \leq N}$ avec $x_n \in X$ et $y_n \in Y$, que l'on considère être tirées selon une loi sur $X \times Y$ fixe et inconnue, par exemple x_n suit une loi uniforme et $y_n = f(x_n) + w_n$ où w_n est un bruit centré.

On distingue trois types de problèmes solubles avec une méthode d'apprentissage automatique supervisée [**?**] :

- $Y \subset \mathbb{R}$: lorsque la sortie que l'on cherche à estimer est une valeur dans un ensemble continu de réels, on parle d'un problème de régression. La fonction de prédiction est alors appelée un régresseur.
- $Y = \{1, \cdots, I\}$ lorsque l'ensemble des valeurs de sortie est fini, on parle d'un problème de classification, qui revient à attribuer une étiquette à chaque entrée. La fonction de prédiction est alors appelée un classificateur (ou classificateur).
- Lorsque Y est un ensemble de données structurées, on parle d'un problème de prédiction structurée, qui revient à attribuer une sortie complexe à chaque entrée. Par exemple, en bio-informatique le problème de prédiction de réseaux d'interactions entre gènes peut être considéré comme un problème de prédiction structurée dans laquelle l'ensemble possible des sorties structurées est l'ensemble de tous les graphes modélisant les interactions possibles [**?**].

2.3.2 Apprentissage non supervisé

Quand le système ou l'opérateur ne disposent que d'exemples, mais non d'étiquettes, et que le nombre de classes et leur nature n'ont pas été prédéterminés, on parle d'apprentissage non supervisé ou clustering. Aucun expert n'est requis. L'algorithme doit découvrir par lui-même la structure plus ou moins cachée des données. Le partitionnement de données, data clustering en anglais, est un algorithme d'apprentissage non supervisé.
Le système doit ici — dans l'espace de description (la somme des données) — cibler les données selon leurs attributs disponibles, pour les classer en groupe homogènes d'exemples. La similarité est généralement calculée selon une fonction de distance entre paires d'exemples. C'est ensuite à l'opérateur d'associer ou déduire du sens pour chaque groupe et pour les motifs (patterns en anglais) d'apparition de groupes, ou de groupes de groupes, dans leur « espace ». Divers outils mathématiques et logiciels peuvent l'aider. On parle aussi d'analyse des données en régression (ajustement d'un modèle par une procédure de type moindres carrés ou autre optimisation d'une fonction de coût). Si l'approche est probabiliste (c'est-à-dire que chaque exemple, au lieu d'être classé dans une seule classe, est caractérisé par un jeu de probabilités d'appartenance à chacune des classes), on parle alors de

« soft clustering » (par opposition au « hard clustering »).
Cette méthode est souvent source de sérendipité(le fait de réaliser une découverte scientifique ou une invention technique de façon inattendue à la suite d'un concours de circonstances fortuit)
ex. : Pour un épidémiologiste qui voudrait dans un ensemble assez large de victimes de cancer du foie tenter de faire émerger des hypothèses explicatives, l'ordinateur pourrait différencier différents groupes, que l'épidémiologiste chercherait ensuite à associer à divers facteurs explicatifs, origines géographique, génétique, habitudes ou pratiques de consommation, expositions à divers agents potentiellement ou effectivement toxiques (métaux lourds, toxines telle que l'aflatoxine, etc.) [**?**].

2.3.3 Apprentissage semi-supervisé

Effectué de manière probabiliste ou non, il vise à faire apparaître la distribution sous-jacente des exemples dans leur espace de description. Il est mis en œuvre quand des données (ou « étiquettes ») manquent... Le modèle doit utiliser des exemples non étiquetés pouvant néanmoins renseigner. Ex. : En médecine, il peut constituer une aide au diagnostic ou au choix des moyens les moins onéreux de tests de diagnostic. Apprentissage partiellement supervisé probabiliste ou non, quand l'étiquetage des données est partiel3. C'est le cas quand un modèle énonce qu'une donnée n'appartient pas à une classe A, mais peut-être à une classe B ou C (A, B et C étant 3 maladies par exemple évoquées dans le cadre d'un diagnostic différentiel) [**?**].

2.3.4 Apprentissage par renforcement

l'algorithme apprend un comportement étant donné une observation. L'action de l'algorithme sur l'environnement produit une valeur de retour qui guide l'algorithme d'apprentissage [**?**]. ex. : L'algorithme de Q-learning est un exemple classique.

2.3.5 Apprentissage par transfert

L'apprentissage par transfert peut être vu comme la capacité d'un système à reconnaître et appliquer des connaissances et des compétences, apprises à partir de tâches antérieures, sur de nouvelles tâches ou domaines partageant des similitudes. La question qui se pose est : comment identifier les similitudes entre la ou les tâche(s) cible(s) et la ou les tâche(s) source(s), puis comment transférer la connaissance de la ou des tâche(s) source(s) vers la ou les tâche(s) cible(s) ? [**?**]

2.3.6 Notion de dataset

Le dataset est l'ensemble des données utilisé pour entraîner un modèle. Il existe deux types de jeux de données (dataset) généraux. Jeux étiqueté et jeux sans étiquette [**?**].

2.3.7 Training Set and Test Set

Dans l'apprentissage automatique, un jeu de données universel inconnu est supposé exister, qui contient toutes les paires de données possibles ainsi que leur distribution de probabilité d'apparition dans le monde réel. Alors que dans les applications réelles, ce que nous avons observé est seulement un sous-ensemble de l'ensemble de données universel en raison du manque de mémoire ou d'un autre inévitable les raisons. Cet ensemble de données acquises s'appelle l'ensemble d'apprentissage **(Training Set)** et est utilisé pour apprendre les propriétés et la connaissance de l'ensemble de données universel. Afin d'examiner la performance de l'apprentissage, un autre ensemble de données peut être réservé pour le test, appelé ensemble de test ou données de test **(Test Set)** [**?**].

Le machine learning est un champ d'études vaste, c'est un domaine de recherche. Dans ce rapport nous ne pouvons pas aborder tous les concepts découlant du machine learning mais nous allons considérer un sous-ensemble de concepts que nous avons utilisés pour le développement de notre projet. Dans notre cas, c'est un problème de classification qui fait partie de l'apprentissage supervisé. C'est comme l'exemple des courriers spam et anti-spam que nous avons pris dans l'introduction de cette section. Le problème est de décider est-ce que le patient doit retourner chez lui ou s'il doit rester à l'hôpital après une arthoplastie. Si la réponse est vraie le patient reste à l'hôpital sinon il retourne chez lui. Il faut toujours garder en esprit que cette décision est importante pour un centre orthopédique car elle peut réduire ou augmenter le coût budgétaire.

2.4 Problème de Classification

Dans l'apprentissage automatique et les statistiques, 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 [**?**].

Voici quelques exemples de problèmes de classification. Nous avons déjà parlé de la classification du spam par courrier électronique comme exemple de problème

de classification. Un autre exemple serait le classement des transactions en ligne. Donc, si vous avez un site Web qui vend des trucs et si vous voulez savoir si une transaction particulière est frauduleuse ou non, si quelqu'un utilise une carte de crédit volée ou a volé le mot de passe de l'utilisateur. Si la transaction est frauduleuse, on retourne 1 comme valeur sinon l'algorithme retourne 0.
0 est aussi appelé la classe négative, et 1 la classe positive, et ils sont parfois aussi désignés par les symboles "-" et "+".

Dans notre projet, le problème est un problème de classification 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).

2.4.1 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 de chaque algorithme d'apprentissage. Différentes mesures de performance sont utilisées pour évaluer différents algorithmes d'apprentissage automatique. Pour l'instant, nous allons nous concentrer sur ceux utilisés pour les problèmes de classification. Quelques metrics que nous pouvons utiliser pour évaluer la performance des problèmes de classification sont Log-Loss, Accuracy, AUC(Area under Curve) etc. Ils font partie des plus communs mais nous pouvons créer des metrics personnalisés en fonction de nos besoins [**?**].

Les metrics que vous choisissez pour évaluer votre modèle d'apprentissage automatique sont très importantes. Le choix des paramètres influence la façon dont la performance des algorithmes d'apprentissage automatique est mesurée et comparée. Pour ne pas perdre plus de temps, voyons ce que sont quelques metrics que nous avons utilisé dans notre projet.

2.4.2 Matrice de confusion

La matrice de confusion **(Confusion Matrix en anglais)** est l'une des mesures les plus intuitives et les plus faciles (à moins bien sûr, vous n'êtes pas confus) utilisées pour trouver l'exactitude et la précision du modèle. Il est utilisé pour les problèmes de classification où la sortie peut être de deux types ou plus de classes [**?**].

La figure 2.3 est une représentation de la matrice de confusion. La valeur actuelle (Actual value) représente la vraie valeur de l'étiquette tandis que la valeur prédite (Predicted value) représente la valeur retournée par le modèle machine learning.

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

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

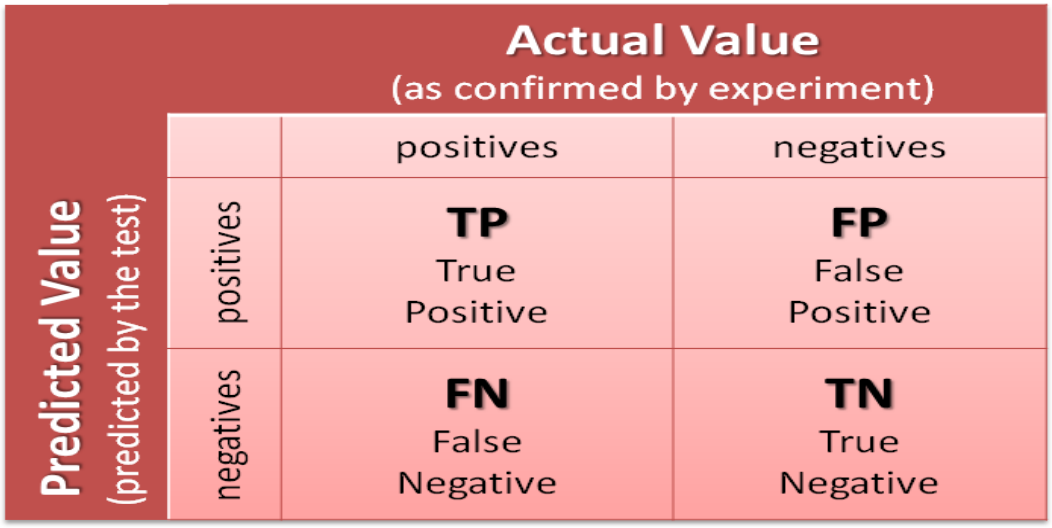

Fig. 2.3 – Confusion Matrix representation [**?**]

False Positives (FP) : Les faux positifs 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) : Les faux négatifs 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)

2.4.3 Calcul de certains metrics de classification à partir de la matrice confusion

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. Nous allons voir comment utiliser la matrice de confusion pour calculer quelques metrics de classification [**?**].

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 = $\frac{2}{\frac{1}{\text{precision}}+\frac{1}{\text{recall}}} = 2 \times \frac{\text{precision}\times\text{recall}}{\text{precision}+\text{recall}} = \frac{TP}{TP+\frac{FN+FP}{2}}$
5. **ROC-AUC :** Une courbe ROC (receiver operating characteristic) est un graphique représentant les performances d'un modèle de classification pour tous les seuils de classification. Cette courbe trace le taux de vrais positifs **(True Positifs en anglais)** en fonction du taux de faux positifs **(False Positifs en anglais)** [**?**].

2.5 Classes déséquilibrées

Nous terminerons le chapitre sur la classification en abordant un sujet très important en machine learning, surtout dans le problème de la classification qui est "*Jeux de données déséquilibré*" **(Imbalanced datasets or Imbalanced Classification or Imbalanced Classes, ses appellations en anglais)**. C'est un problème très commun en machine learning, 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 [**?**].

Plusieurs techniques ont été proposées pour manipuler les classes déséquilibrées. Voyons un bref résumé de quelques-unes d'entre elles [**?**] :

2.5.1 Random Under-Sampling :

Le Random Under-Sampling 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.

2.5.2 Random Over-Sampling :

Le Random Over-Sampling 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.

2.5.3 Cluster-Based Over Sampling :

Dans ce cas, 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.

2.5.4 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.

2.5.5 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.

Dans notre projet nous avons opté pour la technique de SMOTE parce qu'elle permet d'atténuer le problème de sur-adaptation **(overfitting en anglais)** causé par le sur-échantillonnage aléatoire **(random oversampling, en anglais)**

2.6 Conclusion

Le chapitre 2 a présenté une vue générale sur les différents concepts utilisés en sciences de données et en machine learning. On a 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 ces concepts car 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 machine learning, pour cela nous avons dédié une section à l'étude brève de classification. Aussi notre dataset est déséquilibré, par conséquent nous avons eu une section qui montre comment manipuler les classes déséquilibrées et les différentes techniques de manipulation. 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 machine learning

3.1 Introduction

Le développement du modèle est la première grande phase de l'implémentation de notre solution. Le modèle estimateur (meilleur modèle entrainé en fonction de notre ensemble de données) va être utilisé pour introduire la deuxième phase, celle de la création de l'API Ortho. Comme nous avons déjà vu dans l'introduction le diagramme de cas d'utilisation du système machine learning. Nous proposons maintenant un diagramme de classe (figure 3.1) qui répond aux différentes exigences que nous avons décrites. 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 les sections qui suivent nous allons voir les différentes étapes d'implémentation de notre système machine learning. Nous commençons d'abord par l'obtention des données, pour ensuite les explorer, ensuite les nettoyer, les transformer et enfin les modéliser.

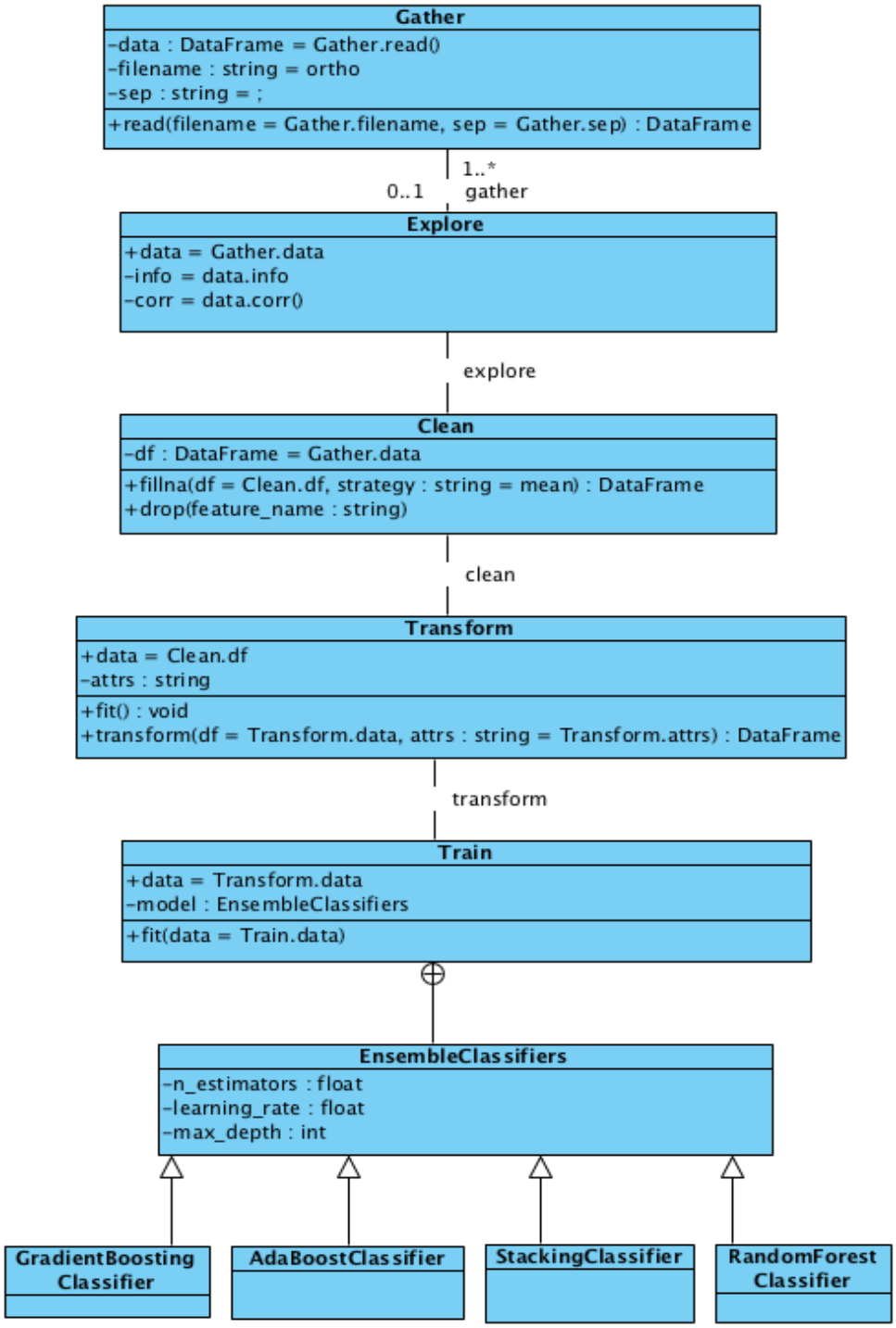

Fig. 3.1 – Class diagram of our machine learning system

3.2 Obtention des données

Une fois qu'on décide à attaquer un problème de machine learning, 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 de machine learning afin qu'il puisse apprendre et devenir plus performant.

Les données que nous utilisons dans ce projet proviennent de la compagnie pour laquelle nous faisons le stage. Cette dernière les avait obtenues de l'université Seattle, Washington, USA. Les données sont disponibles dans un fichier d'extension csv et hébergées sur le serveur de la compagnie. Par conséquent, nous avons un accès direct aux données sans avoir besoin de les télécharger à partir d'un dépôt de Machine Learning.

Comme vous pouvez remarquer dans la classe diagramme du machine learning (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 un 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 les données, il faut 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 à la classe Exploration de données.

3.3 Exploration des données

Le but de l'analyse exploratoire est de «connaître» l'ensemble de données. Si vous le faites à l'avance, le reste du projet sera beaucoup plus fluide, de trois façons principales [**?**] :

1. Vous obtiendrez des conseils précieux pour le nettoyage des données **(Data Cleaning en anglais)** qui peuvent faire ou défaire vos modèles.
2. Vous penserez à des idées pour l'ingénierie des fonctionnalités **(Feature Engineering en anglais)** qui peuvent prendre vos modèles de bonne à grande.
3. Vous obtiendrez une «idée» de l'ensemble de données, ce qui vous aidera à communiquer les résultats et à avoir un impact plus important.

Pour l'analyse exploratoire de nos données, Tout d'abord, nous commençons par une analyse basique de l'ensemble de nos données **(dataset in english)**. La

table 3.1 donne une description de notre dataset 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 du dataset après une analyse basique de notre exploration :

Number of Features : Représente le nombre totale de features ou colonnes de notre dataset. Les features sont importants pour un modèle de machine learning puisque le modèle les utilise pour prédire des événements. Dans l'annexe A, La table A.1 donne une description de tous les features de notre dataset. Le nombre total des features 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.

Number of Features	92
Number of Observations	2718
Number of Numerical Features	87
Number of Categorical Features	5
Target value	Discharge

TABLE 3.1 – Gives a description of our dataset

	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 – An overview of raw data from our dataset

Number of Observations : Représente le nombre d'échantillons **(samples or rows en anglais)** de notre ensemble de données.

Numerical Features : Ce sont les caractéristiques numériques de notre dataset. Les valeurs continues de notre dataset.

Categorical Features : Ce sont les features qui viennent sous forme de texte. Plus loin dans la section de la transformation, nous convertirons ces features en données numériques car la plupart des algorithmes de machine learning ne reconnaissent pas les données discrètes ou textuelles.

Target value : C'est l'étiquette (**label or target en anglais)** de notre ensemble de données. L'étiquette est une valeur très importante pour les algorithmes de classification car elle permet d'évaluer la valeur booléenne d'une décision. Dans notre cas c'est le *Discharge* qui permet de dire si oui ou non le patient doit rester à l'hôpital pour une réadaptation.

Target's name	Discharge
Total positive classes	2421
Percentage positive classes	89.072848 %
Negative classes	297
Percentage negative classes	10.927152 %
Total classes	2718

TABLE 3.2 – Important details about target value

La table 3.2, quant à elle, c'est la représentation de notre variable target value. Dans le dataset, 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 du dataset. Nous avons déjà discuté dans la section *"Classes déséquilibrées"*, les différentes techniques pour équilibrer les classes déséquilibrées.

3.3.1 Correlation entre les données

Nous terminons la section d'exploration de données avec les corrélations. Les corrélations nous permettent d'examiner les relations entre les entités numériques et d'autres entités numériques **(Numerical features en anglais)**. La corrélation est une valeur comprise entre -1 et 1 qui représente à quel point deux entités se déplacent à l'unisson [**?**].

Une corrélation positive signifie que lorsqu'une caractéristique augmente, l'autre augmente. Par exemple. Le BMI d'un patient et son poids **(Weight dans le dataset)**.

La corrélation négative signifie que lorsqu'une caractéristique augmente, l'autre diminue. Par exemple. heures consacrées à l'étude et nombre de participants. Les corrélations proches de -1 ou 1 indiquent une relation forte. Les plus proches de 0 indiquent une relation faible. 0 indique aucune relation.

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 – General view of the correlation between some features of our dataset

La figure 3.3 nous donne une idée comment nos variables sont corrélées entre elles. Par exemple, le poids **(Weight dans le dataset)** et le **BMI** ont une correlation de 0.81 très proche de 1. On peut déduire que les patients qui ont leur BMI plus élevé sont les plus pesants. L'objectif de la correlation est de gagner en intuition sur les données, ce qui nous aidera tout au long du flux de travail.

Pour calculer la correlation, nous avons utilisé la méthode **corr** de la classe **Explore**. Nous avons passé un peu de style comme paramètre pour avoir les couleurs.

Après avoir exploré les données et acquérir suffisamment de connaissance sur notre ensemble de données, c'est le temps maintenant de les nettoyer. La section qui suit montre comment nous avons procédé pour nettoyer notre dataset.

3.4 Nettoyage des données

Le nettoyage des données **(Data cleaning en anglais)** est l'une de ces choses que tout le monde fait mais dont personne ne parle vraiment. Bien sûr, ce n'est pas la partie la plus intéressante de l'apprentissage automatique. Et non, il n'y a pas de trucs cachés et de secrets à découvrir.

Cependant, un nettoyage correct des données peut faire ou défaire notre projet. Les spécialistes des données professionnelles consacrent généralement une très grande partie de leur temps à cette étape. En fait, si vous avez un jeu de données correctement nettoyé, même des algorithmes simples peuvent apprendre des informations impressionnantes à partir des données ! [**?**].

3.4.1 Observations indésirables

La première étape du nettoyage des données consiste à supprimer les observations indésirables de votre jeu de données. Cela inclut des observations en double **(Duplicate observations)** ou non pertinentes **(Irrelevant observations)** [**?**].

3.4.2 Duplicate observations :

Les observations en double surviennent le plus souvent lors de la collecte de données, notamment lorsque vous combinez des ensembles de données provenant de plusieurs endroits, capturez les données et recevez des données de clients / autres départements [**?**].

3.4.3 Irrelevant observations :

Les observations non pertinentes sont celles qui ne correspondent pas vraiment au problème spécifique que vous essayez de résoudre [**?**].

Dans notre projet, nous avons trouvé beaucoup de données impertinentes. Par exemple. pendant la phase d'exploration nous avons trouvé que le "**Weight**" est une donnée impertinente car elle très corrélée avec le "**BMI**". On a décidé de maintenir le **BMI** et de laisser tomber le **"Weight"**. De la même manière nous nous rendons compte que le **"RawDX"** et **"GHOA"** étaient les mêmes données sauvegardées sous 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.

3.4.4 Données manquantes

Les données manquantes **(Missing data en anglais)** sont un problème trompeur dans l'apprentissage automatique appliqué. Tout d'abord, juste pour être clair, vous ne pouvez pas simplement ignorer les valeurs manquantes dans un ensemble de données. Vous devez les gérer d'une manière ou d'une autre pour la raison très pratique que la plupart des algorithmes n'acceptent pas les valeurs manquantes.

Les deux stratégies les plus couramment recommandées pour traiter les données manquantes sont les suivantes [**?**] :

1. Suppression des observations qui ont des valeurs manquantes **(Dropping en anglais)**
2. Imputation des valeurs manquantes en fonction d'autres observations **(Imputing en anglais)**

Après avoir essayé les deux recommandations, nous avons constaté que le **"imputing"** nous donne de meilleurs résultats. Pour arriver à cette affirmation, nous avons essayé plusieurs modèles avec leurs différentes évaluations. Il faut souligner que notre dataset a de beaucoup de valeurs nulles. C'est la deuxième difficulté du projet après le déséquilibre au niveau du dataset.

Il est important de préciser que dans le **"imputing"**, il y a plusieurs méthodes de remplissage. On peut remplir avec la valeur "**zéro (0)**", la valeur moyenne "**mean**" ou la valeur moyenne "**median**". Encore, nous avons opté de remplir avec la valeur moyenne "**mean**", car elle nous a donné de meilleurs résultats. La méthode **fiilna** de la classe **Clean** est utilisée pour remplir les données manquantes. Elle prend un jeu de données et la stratégie de remplissage en paramètres et elle retourne un jeu données sans données manquantes.

La prochaine section va être l'étape de transformation ou le feature engineering. Une fois que nos données sont nettoyées et il n'y a pas de données vides, on peut les transformer pour en avoir de nouvelles beaucoup plus pertinentes.

3.5 Feature Engineering

L'ingénierie des caractéristiques **(Feature Engineering en anglais)** consiste à créer de nouvelles entités en entrée à partir de celles qui existent déjà. En général, vous pouvez considérer le nettoyage des données comme un processus de soustraction et l'ingénierie des fonctionnalités comme un processus d'ajout. C'est

souvent l'une des tâches les plus précieuses qu'un data scientist puisse faire pour améliorer les performances du modèle, et ce, pour trois grandes raisons [**?**] :

1. Vous pouvez isoler et mettre en évidence les informations clés, ce qui aide vos algorithmes à se concentrer sur ce qui est important.
2. Vous pouvez apporter votre propre expertise de domaine.
3. Plus important encore, une fois que vous comprenez le «vocabulaire» de l'ingénierie des caractéristiques, vous pouvez apporter l'expertise de domaine d'autres personnes !

Le feature engineering est l'étape qui précède la modélisation. Il prend en entrée le jeux de données brutes et produit en sortie un jeux de données préparé pour l'étape de la modélisation. Le modèle prend en entrée le jeux de données produit par le processus du feature engineering.

Nous avons fait notre feature engineering en deux étapes :

1. Premièrement, les features **Dx1, Dx2, Dx3** qui existent déjà dans notre dataset doivent être combinés entre eux pour créer un nouveau feature appelé **degree_dx** qui est le niveau général diagnostique du patient. Nous soulignons que **Dx1, Dx2, Dx3** représentent respectivement : premier niveau diagnostique, deuxième niveau diagnostique et troisième niveau diagnostique du patient. En combinant ces trois diagnostiques, nous créons un niveau général diagnostique du patient.
   Logique de création : On regroupe ces colonnes (**Dx1, Dx2, Dx3**) suivants la logique si les 3 sont presents (c'est-à-dire si les 3 ont valeurs non-null), on donne la valeur de 3 ; si uniquement DX2 et DX1 sont présents, on donne la valeur de 2 ; si uniquement DX1 on donne la valeur de 1 si aucune valeur n'est présente, on donne la valeur 0. Le feature résultant se nomme **degree_dx**
2. Deuxièment, nous créons un nouvel autre feature appelé **medcond** qui détermine les conditions médicales du patient. Nous nous basons sur l'ensemble de caractéristiques pré-opératoires du patient. Par exemple son niveau de glucose, les quantités d'hémoglobine blanches et d'hémoglobines rouges. Ces features sont au nombre de 46 au total.
   Logique de création : On prend toutes les colonnes de conditions médicales (**PreOpHgb** à **Depression**, ils sont au nombre de 46 dans le dataset) : On donne la valeur 1 lorsqu'on trouve une valeur non-nulle à chacune des conditions sauf pour les colonnes suivantes :
   - On donne la valeur 2 pour les colonnes suivantes : [**PreOpHgb, PreOpGlucose, pulm circ, other neuro, chronic pulm**]

- On donne la valeur 3 pour les colonnes suivantes [**PreOpC, Paralysis, renal failure, liver failure**]

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 de l'étape suivante sur la modélisation.

3.6 Modélisation des données

La modélisation est un processus d'entrainement de modèles. Il consiste à 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.

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.

Puisque notre ensemble de données est déséquilibré, nous ne pouvons pas utiliser les algorithmes de classification simples tel que LogisticRegression, TreeDecisionClassifier, LinearClassifier etc. Nous avons utilisé l'approche des méthodes d'ensemble qui consiste à combiner plusieurs classificateurs simples pour donner un classificateur fort. L'image 3.4 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.

Dans le chapitre sur *"Contexte théorique"* nous avons vu comment manipuler les données déséquilibrées en ré-échantillonnant **(resampling en anglais)** les données d'origine pour fournir des classes équilibrées. Nous avons vu toutes les avantages à utiliser les techniques SMOTE que les autres. Dans la sous-section qui suit sur la modélisation, nous allons brièvement examiner une autre approche, à savoir la modification des algorithmes de classification existants pour les rendre appropriés aux ensembles de données déséquilibrés.

3.6.1 Méthodes d'ensemble

L'objectif principal de la méthodologie d'ensemble **(Ensemble methods en anglais)** est d'améliorer la performance des classificateurs uniques. L'approche consiste à construire plusieurs classificateurs à deux étapes à partir des données originales, puis à agréger leurs prédictions [**?**]. Les méthodes d'ensemble sont nombreuses, mais nous allons brièvement définis les trois plus importantes : Bagging, Boosting et Random Forest.

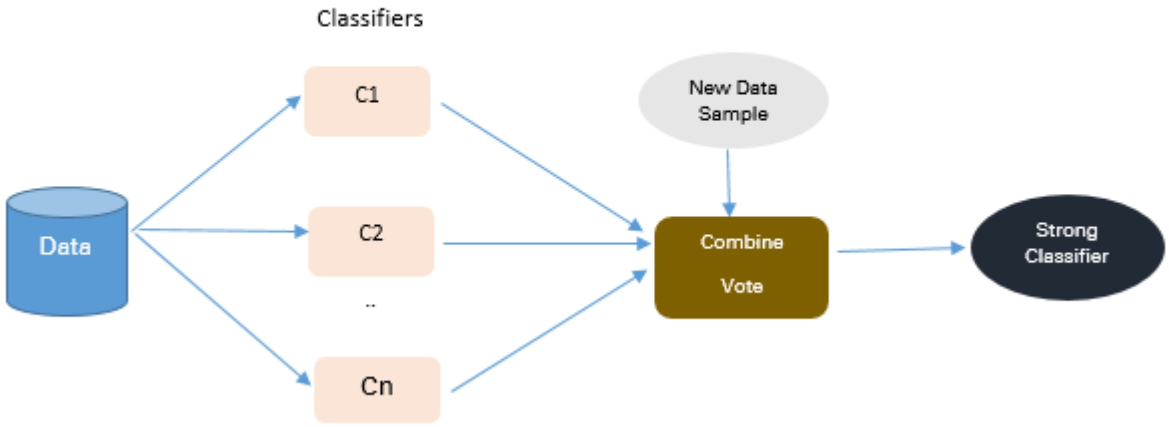

Fig. 3.4 – Approach to Ensemble based Methodologies [**?**]

3.6.1.1 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 [**?**].

3.6.1.2 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 » [**?**].

3.6.1.3 Stacking

Le stacking est similaire au boosting : vous appliquez également plusieurs modèles à vos données d'origine. La différence ici est, cependant, que vous n'avez pas juste une formule empirique pour votre fonction de poids, plutôt que vous introduisez un méta-niveau et utilisez un autre modèle / approche pour estimer l'entrée avec les sorties de chaque modèle pour estimer les poids ou , en d'autres termes, pour déterminer quels modèles fonctionnent bien et ce qui est mal donné ces données d'entrée [**?**].

3.6.1.4 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. [**?**]

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 le random forest classification. 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 processus automatiques en nous basant sur les scores de chaque modèle. Le meilleur modèle va être celui du meilleur score obtenu.

Algorithm	**Metrics**		
	Accurency 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 – Comparison of ensemble methods algorithms used in our project

Nous tenons compte aussi des metrics d'évaluation comme "Precision and Recall".Ces derniers nous permettent de déterminer la performance de notre modèle à classifier les classes positives et celles qui sont négatives. Le but de notre projet était d'obtenir la précision en dessus de 90%. Généralement, dans la pratique une telle précision est acceptable pour pouvoir fier le modèle.

Vous pouvez constater à la table 3.3 que nous obtenons des résultats différents pour chaque algorithme. Pour faire le choix du bon algorithme, notre API machine learning est capable d'évaluer chaque algorithme de façon indépendante et itérativement pour déterminer le meilleur modèle, tout en combinant les différents résultats de chaque metric. Dans ce cas, c'est l'**AdaBoostClassifier** qui serait le modèle estimé meilleur.

Après avoir déterminé le meilleur modèle à partir des processus automatiques. 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 machine learning.

4 | Implémentation de l'outil Ortho

4.1 Introduction

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 d'un autre système se passe généralement aussi bien. Cependant, d'étranges interactions peuvent commencer à apparaître d'une manière qui serait impossible du point de vue de l'ingénierie logicielle. 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, basés sur des micro-services. Les systèmes complexes d'apprentissage profond peuvent évoluer pour montrer des comportements pathologiques «changer quoi que ce soit, tout changer», même s'ils apparaissent superficiellement comme des micro-services bien conçus avec de hauts degrés d'isolation [**?**].

Compte tenu de cette complexité qui existe au niveau des applications de machine learning, nous avons décidé de construire un API Ortho pour faire le dé-

ploiement 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.

4.2 Construction de l'API Ortho

Les API (Application Program Interfaces) sont des méthodes de communication logicielle développées sur une norme particulière. De nombreuses entreprises ont leurs propres API publics qui résolvent des problèmes spécifiques pour les développeurs. 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 permettent une fonctionnalité multi-plateforme, rendue possible par une norme agnostique de plate-forme, telle que la spécification REST. La beauté des API est qu'elles sont super accessibles - que vous construisiez une application mobile, des appareils IoT, un serveur ou que vous souhaitiez simplement trouver un moyen de communiquer entre vos propres micro-services, les API vous y aident [**?**].

Pour communiquer avec notre modèle, nous avons construit un API en Django framework web en suivant l'architecture présentée dans la figure 4.1. Voyons un peu cette 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 autorisés 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 machine learning que nous avons construit.

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

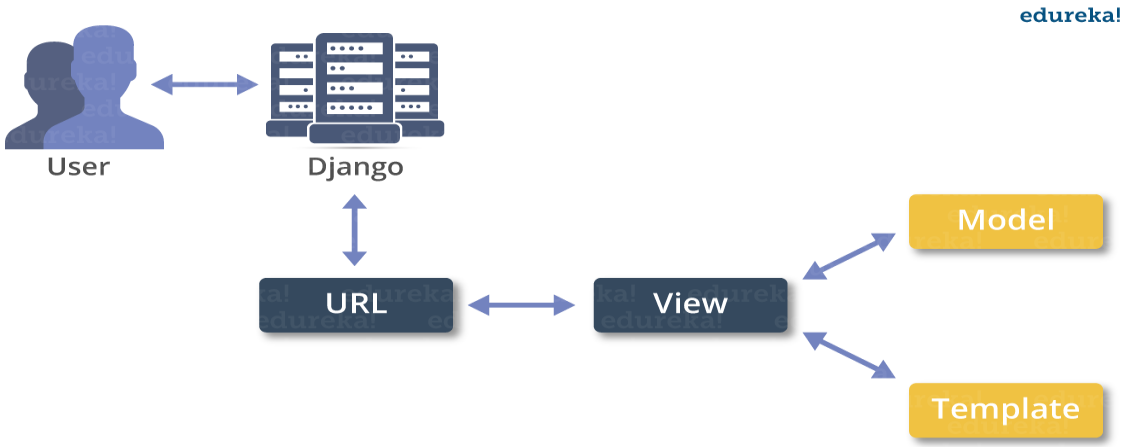

Fig. 4.1 – Django Framework web architecture [**?**]

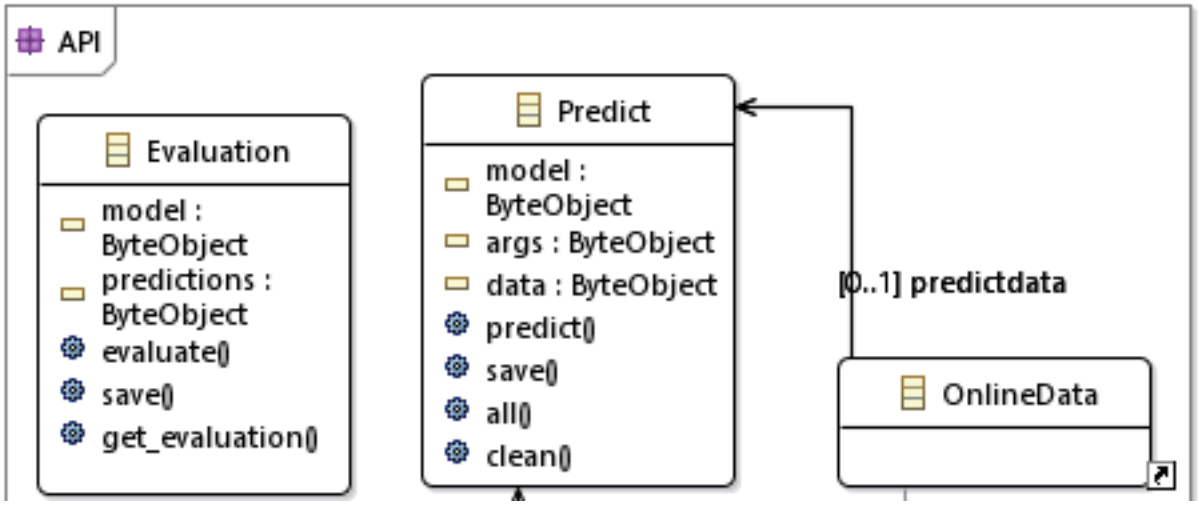

Fig. 4.2 – Class diagram of our Ortho API

Notre API prend un formulaire web de Django en entrée et retourne un message de décision. Le framework django fait 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 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. Mais nous allons voir brièvement le fonctionnement de ce diagramme dans l'API :

Predict. La classe qui répond à l'exigence **Predict** défini à l'introduction. Elle trois attributs et quatre méthodes. L'attribut **model** est le modèle machine learning 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 du machine learning 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 (**Machine learning et Software Engineering**) décrit dans la classe digramme présenté à la figure 4.3. Il faut relater que les modèles de machine learning ont besoin d'une couche d'ingénierie de logiciels pour leur rendre accessible aux utilisateurs qui ne comprennent rien en machine learning. Dans la vraie vie, ceux qui utilisent les modèles ne connaissent rien ni en machine learning ni en software engineering. C'est pourquoi dans tout lieu où il y a une équipe de data scientist qui travaillent, il y a aussi une équipe de développement de logiciels.

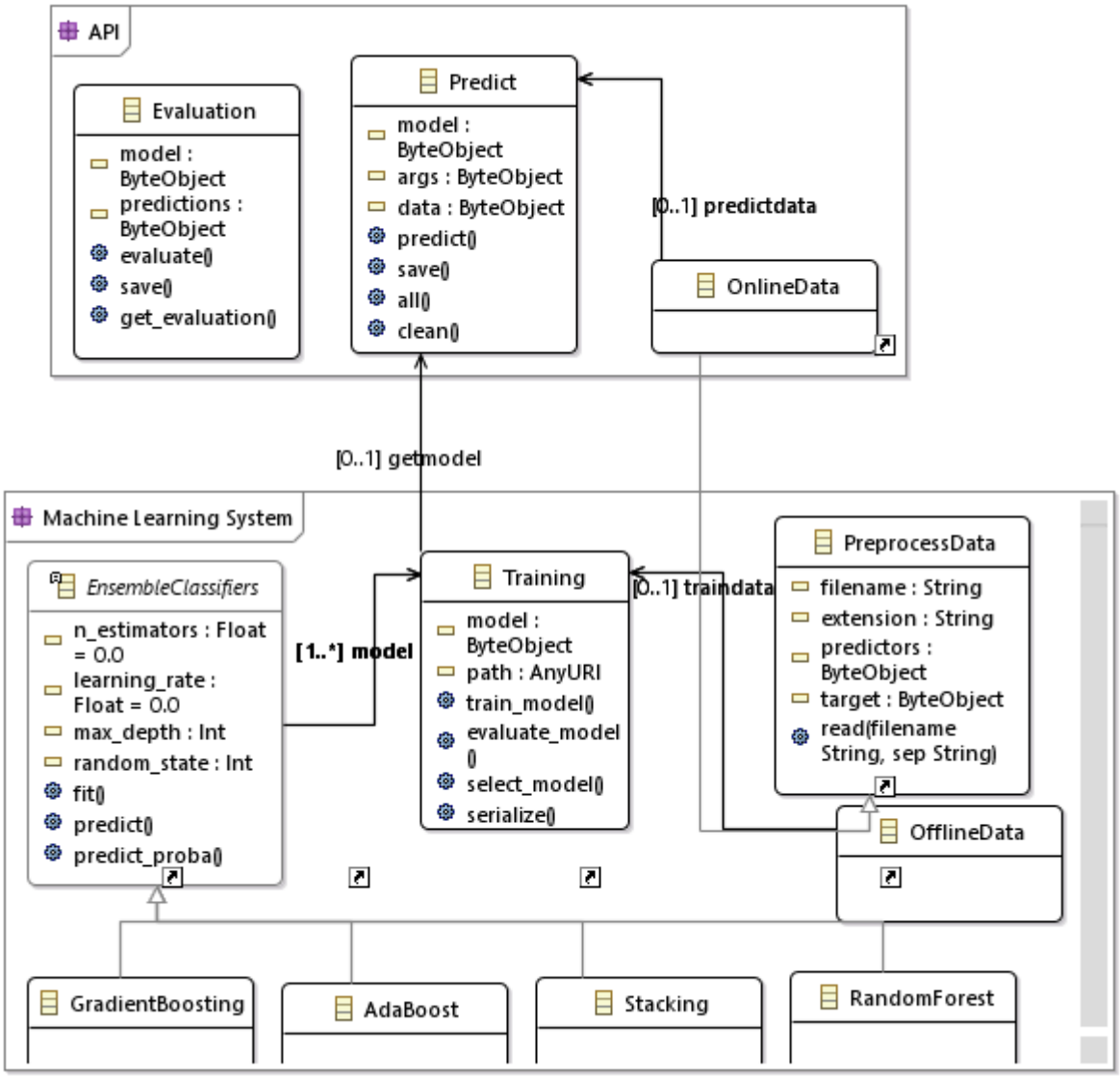

Fig. 4.3 – Connexion between our API and machine learning system

4.3 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 machine learning.

Deployment System. Le système de construction de notre API Ortho et l'environnement Django.

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

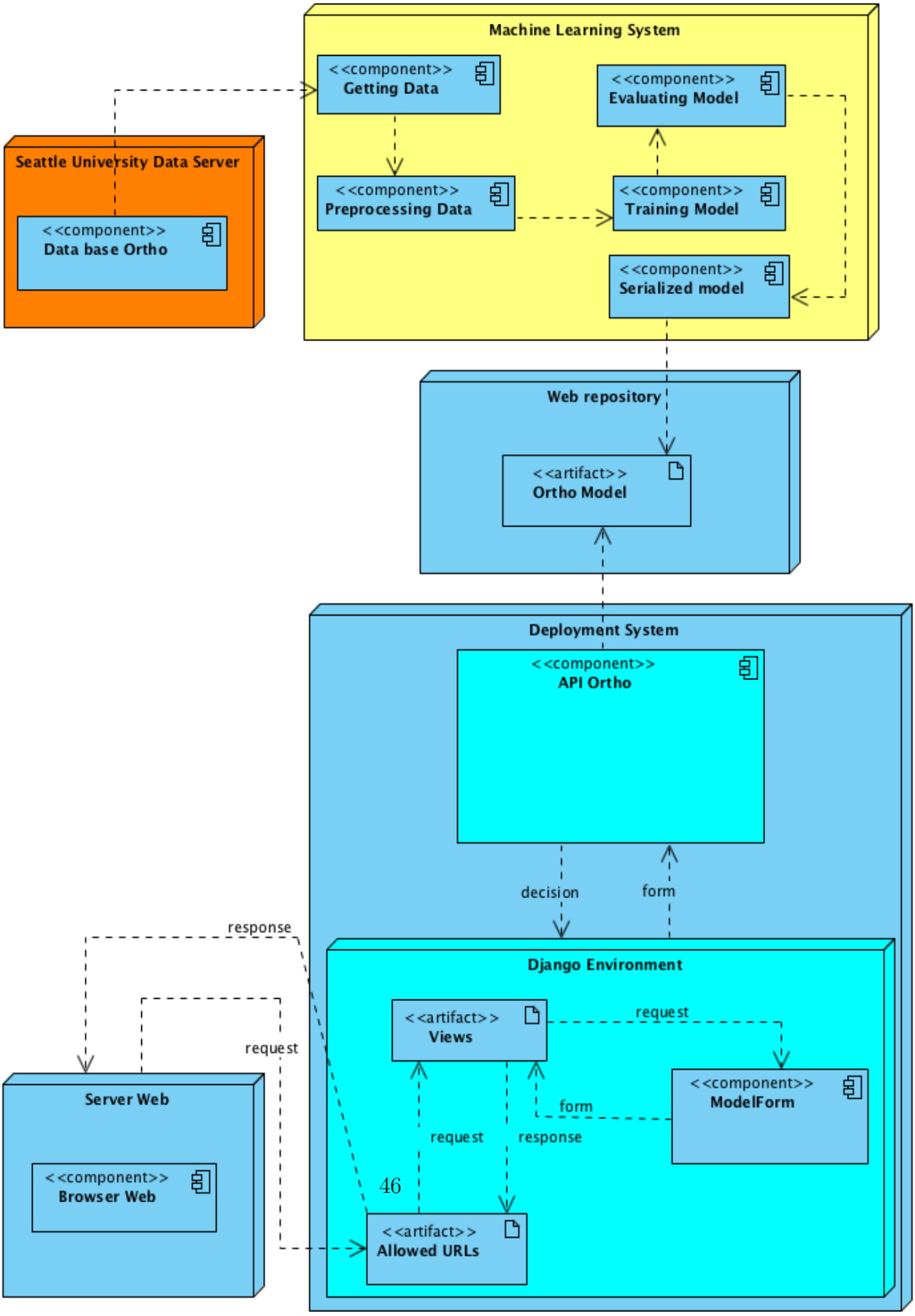

Fig. 4.4 – A deployment diagram of our tool

5 | Évaluation

5.1 Évaluation du modèle machine learning

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 **logarithmic_loss** ou toute autre mesure de ce type. Évaluer un projet de machine learning 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 de machine learning permettant de faire des évaluations sur des modèles. Ce que nous avons déjà dans la sous-section 2.3.1 portant sur la "**Mesure de performance des algorithmes de classification**" au chapitre "**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 se 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 de comparaison des différents algorithmes entrainés dans le projet (table 3.3). 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 interprétation sommaire de cette évaluation est que 8% des prédictions faites par le modèle est erroné et 92% sont correctement classées. Pour faire cette interpre-

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

TABLE 5.1 – Gives different results of our model evaluation

tation, nous nous tenons compte du " **Precision score** " de la table 5.1.

5.2 Evaluation de notre API Ortho

Pour évaluer notre API, nous avons construit un diagramme de séquence qui montre comment les orthopédistes peuvent utiliser l'API pour prédire un retour. Et l'API à son tour se connecte au repository du machine learning pour charger le modèle sérialisé par le système machine learning. La figure 5.1 est la classe diagramme 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 de machine learning. 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 tout le monde peut avoir accès au module de prédictions mais les modules monitoring et management sont réservés aux administrateurs du système.

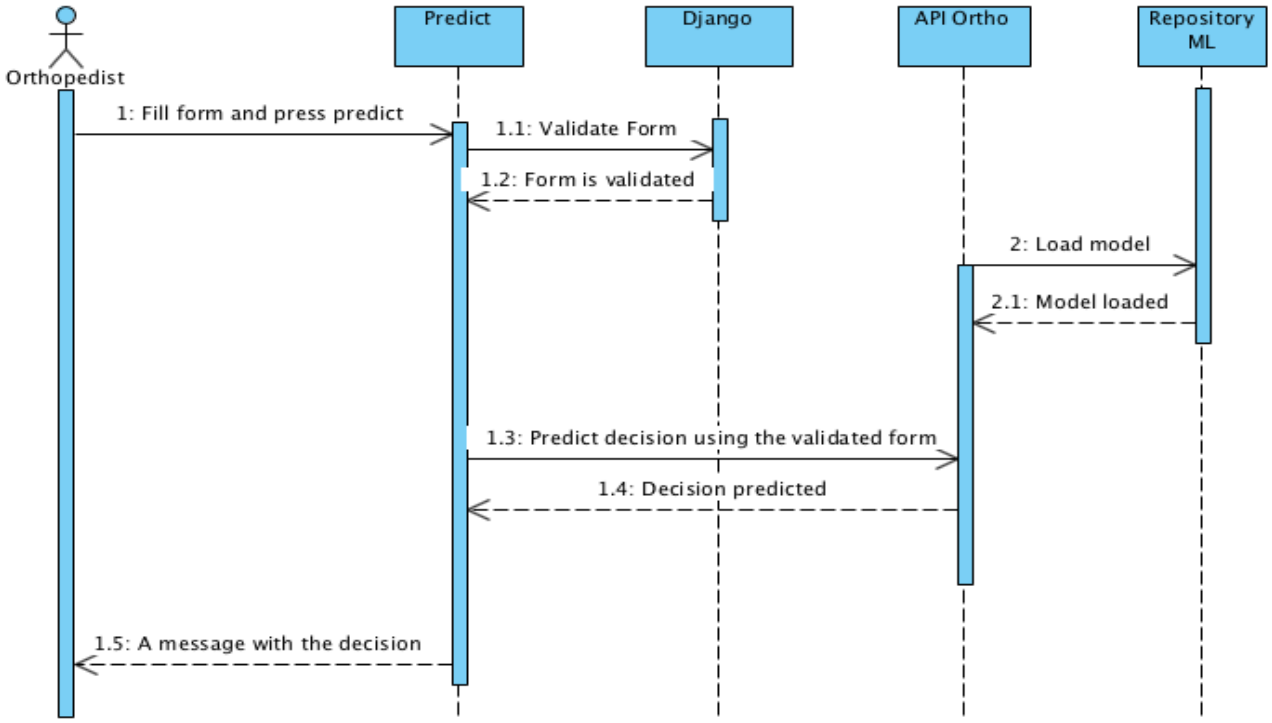

Fig. 5.1 – Sequence diagram for evaluating our API tool

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. [**?**]. 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 sys-

tèmes 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. [**?**]. Nous avons préféré de développer notre propre API en fonction de nos besoins et notre ensemble de données. En machine learning, 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. [**?**] 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 de machine learning 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. [**?**] 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. [**?**] 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 [**?**] 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 [**?**] 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 vous utilisez SMOTE dans vos projets, vous ne pouvez pas utiliser les algorithmes classiques de machine learning parce que les résultas peuvent être biaisés facilement. Il faut toujours utilisé les classificateurs fort comme les 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 reste 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 le 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 de machine learning et celles du software engineering 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 data scientist depuis l'obtention des données jusqu'à la modélisation (étape qui consiste à entrainer différents modèles machine learning 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 machine learning 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 diagramme de cas d'utilisation expliquant les différentes exigences que notre système doit satisfaire et nous avons donné une brève explication de chaque cas d'utilisation 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 de machine learning 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 le machine learning et le software engineering. Puisqu'au début du stage je n'ai pas eu de connaissances en machine learning ;
- 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 machine learning 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.

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 que toutes les spécialités informatiques peuvent se combiner les unes avec les autres.

Par exemple un API développé en Software engineering peut être utilisé pour connecter à des modèles développés en machine learning. On peut aussi utiliser les diagrammes UML pour décrire un système de machine learning 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 début de ce projet, nous ne savions rien ni en machine learning ni en python. Maintenant nous savons utiliser toutes les librairies de machine learning de python telles que sckit-learn, mlxtend, pandas, matplotlib, numpy, scipy etc. Nous avons appris à développer en Django un framework web de python dont nous avons utilisé pour le développement de notre API.

Notre outil va être déployé sur le serveur web de la compagnie pour laquelle je fais 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 veulent évaluer, suivre et gérer un modèle doivent être d'abord administrateur du système. Pour être administrateur, il faut contacter 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

Feature name	**datatype**	**non-null**	**null**	**Comments**
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)
Procedure	Boolean	2718	0	On veut savoir si oui ou non le patient a eu une procédure dans le passé
RawDx	Categorical	2718	0	Diagnostique brute du patient
PriorContralateral	Boolean	2703	15	On veut savoir si oui ou non le patient est atteint de *PriorContralateral*
TSA	Boolean	2718	0	On veut savoir si oui ou non le patient est atteint de *TSA*
RTSA	Boolean	2718	0	On veut savoir si oui ou non le patient est atteint de *RTSA*
Tendon Transfer	Boolean	2718	0	On veut savoir si oui ou non le patient est atteint de *Tendon Transfer*
Dx1	Categorical	2718	0	Représente diagnostique 1
Dx2	Categorical	176	2542	Représente diagnostique 2
Dx3	Categorical	2	2716	Représente diagnostique 3
GHOA	Boolean	2718	0	On veut savoir si oui ou non le patient est atteint de *GHOA*
AVN	Boolean	2718	0	On veut savoir si oui ou non le patient est atteint de *AVN*
RCT	Boolean	2718	0	On veut savoir si oui ou non le patient est atteint de *RCT*

FailedRCR	Boolean	2718	0	On veut savoir si oui ou non le patient est atteint de *FailedRCR*
ComboRCT	Boolean	2718	0	On veut savoir si oui ou non le patient est atteint de *ComboRCT*
RCTA	Boolean	2718	0	On veut savoir si oui ou non le patient est atteint de *RCTA*
IA	Boolean	2718	0	On veut savoir si oui ou non le patient est atteint de *IA*
PTA	Boolean	2718	0	On veut savoir si oui ou non le patient est atteint de *PTA*
PHFx	Boolean	2718	0	On veut savoir si oui ou non le patient est atteint de *PHFx*
PHFxSequelae	Boolean	2718	0	On veut savoir si oui ou non le patient est atteint de *PHFxSequelae*
Other	Boolean	2718	0	On veut savoir si oui ou non le patient est atteint de *Other*
AGE_AT_ADMIT	Numerical	2718	0	L'âge du patient
ASA_SCORE	Numerical	2716	2	Les commentaires sont à venir
Gender	Boolean	2718	0	On veut savoir si oui ou non le patient est un *Gender*
Female	Boolean	2718	0	On veut savoir si oui ou non le patient est un *Female*
Discharge	Boolean	2718	0	On veut savoir si oui ou non le patient doit retourner chez lui
Height	Numerical	2293	425	La hauteur du patient
Weight	Numerical	2301	417	Le poids du patient
BMI	Numerical	2283	435	L'indice de masse corporelle (IMC)- BMI en anglais
PreOpHgb	Numerical	272	2446	Hémoglobines blanches du patient
PreOpCr	Numerical	255	2463	Conditions pré-opératoires du patient
PreOpALC	Numerical	157	2561	Conditions pré-opératoires du patient
PreOpGlucose	Numerical	253	2465	Glucose du patient

arrhythmias	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *arrhythmias*
valvular	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *valvular*
pulm circ	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *pulm circ*
PVD	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *PVD*
HTNw/oCx	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *HTNw/oCx*
HTNw/Cx	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *HTNw/Cx*
Paralysis	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *Paralysis*
other neuro	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *other neuro*
chronic pulm	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *chronic pulm*
DMw/oCx	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *DMw/oCx*
DMw/Cx	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *DMw/Cx*
hypothyroid	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *hypothyroid*
renal failure	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *renal failure*
liver failure	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *liver failure*
PUD	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *PUD*
AIDS	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *AIDS*
L0oma	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *L0oma*

Mets	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *Mets*
solidCaw/oMets	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *solidCaw/oMets*
RA/CVD	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *RA/CVD*
coagulopathy	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *coagulopathy*
obesity	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *obesity*
weightLoss	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *weightLoss*
fluidElectrolyte	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *fluidElectrolyte*
bloodLossAnemia	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *bloodLossAnemia*
DeficiencyAnemia	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *DeficiencyAnemia*
EtOH	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *EtOH*
Drugs	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *Drugs*
Psychosis	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *Psychosis*
Depression	Boolean	2092	626	On veut savoir si oui ou non le patient est atteint de *Depression*
Plavix	Numerical	2482	236	Les commentaires sont à venir
Dabigatran	Boolean	2482	236	On veut savoir si oui ou non le patient est atteint de *Dabigatran*
Enoxaparin	Numerical	2482	236	Les commentaires sont à venir
Rivaroxaban	Numerical	2482	236	Les commentaires sont à venir
Warfarin	Numerical	2482	236	Les commentaires sont à venir

PreOpNarcotic	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *PreOpNarcotic*
PreOpBloodThinner	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *PreOpBloodThinner*
PreOpSteroids	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *PreOpSteroids*
PreOpInsulin	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *PreOpInsulin*
PreOpDMMeds	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *PreOpDMMeds*
Oxycodone	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *Oxycodone*
Hydrocodone	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *Hydrocodone*
Vicodin	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *Vicodin*
percocet	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *percocet*
Oxycontin	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *Oxycontin*
Dilaudid	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *Dilaudid*
Aspirin	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *Aspirin*
Warfarin.1	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *Warfarin.1*
Coumadin	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *Coumadin*
Plavix.1	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *Plavix.1*
Clopidogrel	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *Clopidogrel*

Prednisone	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *Prednisone*
insulin	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *insulin*
humulin	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *humulin*
lantus	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *lantus*
lispro	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *lispro*
aspart	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *aspart*
metformin	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *metformin*
glipizide	Boolean	900	1818	On veut savoir si oui ou non le patient est atteint de *glipizide*

TABLE A.1 – Our whole dataset

Bibliographie