Applications du Deep Learning dans le suivi du kératocône

CHU de Toulouse

RAPPORT FINAL

JACOMI Nicolas, LEONE Julie, MARTY Morgan, MAZOYER Justin

INU Champollion – ECOLE D'INGENIEURS ISIS

ANNEE UNIVERSITAIRE 2021 – 2022

FIE5

Remerciements :

Dans le cadre de ce projet tuteuré, nous tenons à remercier toutes les personnes ayant contribué à la réalisation et au bon déroulement de notre projet.

Tout d'abord, nous aimerions adresser nos remerciements au service d'ophtalmologie du CHU de Toulouse, et plus particulièrement Pr Fournié et Drira Inès pour avoir proposé ce projet et nous avoir accompagné tout au long de celui-ci. Mais aussi pour la confiance et tout le temps qu'ils nous ont accordé pour nous aider en phase de développement.

Enfin, nous souhaitons remercier Mme Megdiche pour l'aide et l'encadrement apporté au cours de ce projet ainsi que pour sa disponibilité.

Sommaire :

Table des figures :

Introduction :

La médecine actuelle bien que très performante n'a de cesse de découvrir de nouvelles maladies avec leurs symptômes, leurs traitements et les complications impliquées. Les spécialistes sont compétents dans leurs domaines respectif, pour autant certaines maladie dont l'occurrence est très basse et demandant une attention toute particulière ne peut pas être traitée efficacement chez tous les spécialistes. En effet, certaines médecins spécialisés dans un domaine ont des maladies particulières dont ils sont plus ou moins coutumiers. C'est pourquoi des patients sont transférés d'un spécialiste à l'autre pour avoir un suivi plus efficace. Cependant, avant que ce changement soit effectué, il faut que le spécialiste en question ai compris les symptômes du patient pour l'aiguiller correctement vers un confrère expérimenté. Pour quelques cas, les maladies à diagnostiquer sont parfois asymptomatiques avant d'atteindre un stade avancé, c'est par exemple le cas avec la maladie du kératocône qui va tout particulièrement nous intéresser. Des critères bien précis doivent être mesurés et calculés pour que le diagnostic soit efficace, ce qui implique de connaitre les critères à chercher mais également de savoir qu'elles sont les valeurs limites à surveiller pour prendre en charge la maladie et connaitre son stade.

C'est pourquoi nous avons travaillé pendant notre projet tuteuré sur la mise en place d'un système expert pour diagnostiquer le kératocône efficacement. Nous avons donc collaboré avec le service d'ophtalmologie du Centre Hospitalier de Toulouse qui fait partie du centre de référence du kératocône (CRNK). Le premier problème majeur de cette maladie est son occurrence, ne se déclarant que pour 1 habitant sur 2000 en France, elle ne peut donc pas être qualifiée de commune. Sa deuxième particularité est son manque de symptômes claires. En effet, les symptômes majeurs de cette maladie sont la déformation de la cornée qui s'amincit, mais au niveau visuel, elle se déclare comme une myopie ou un astigmatisme irrégulier. Il est donc compliqué d'en déduire au premier abord un kératocône. Le diagnostic ne peut se faire qu'au bout d'un certain temps quand l'ophtalmologue se rend compte que le patient ne s'adapte pas au lunettes et que la topographie de sa cornée est anormale. Même si tous les ophtalmologues ont connaissance de cette maladie, très peu savent la prendre en charge et offrir un suivi efficace, c'est pourquoi ces patients sont transférés dans des services experts.

Cependant pour que ceux-ci ne soient pas débordés pour de fausses déclarations de kératocône, la solution proposée est de mettre au point un système expert basé sur une intelligence artificielle. Elle permettra de prendre en entrée les différents paramètres mesurés sur le patient et d'en ressortir la présence ou non de maladie ainsi que son stade.

L'interne en ophtalmologie Inès Drira nous a donc confié la mission de faire naitre ce système expert à partir des critères qu'elle a rassemblé pour déterminer l'expression du kératocône. La problématique qui nous a été proposée nous demandait de mettre au point l'application du Deep Learning pour le suivi du kératocône.

Nous exposerons notre travail selon trois axes principaux. Dans un premier temps, nous présenterons le contexte du sujet, puis nous continuerons avec le projet pour finir avec la conduite de projet.

I. Contexte

1. Présentation projet

a. Présentation générale

Le projet mené par l'interne en ophtalmologie Inès Drira au CHU de Toulouse concerne la création d'un système expert. Le but de celui-ci sera d'assister le diagnostic de dépistage du kératocône. En effet, cette maladie n'ayant une prévalence que de 1.38 / 1000 sur la population mondiale, les ophtalmologues ne peuvent pas avoir un niveau d'expertise assez élevé pour déterminer avec certitude si le patient est atteint de kératocône ou même définir le stade de cette maladie. En effet, de trop nombreux critères sont présents pour que la décision clinique soit claire et précise. Ces critères n'étant pas tous très bien définis, il reste des zones d'ombre même pour les ophtalmologues qui sont des experts sur cette maladie. Il est donc facilement compréhensible qu'une personne peut expérimentée puisse ne pas trouver les sources de la gêne visuelle du patient. Dans ce cas, certains ophtalmologistes libéraux préfèrent dans le doute envoyer leurs patients réaliser leur suivi au centre de référence. Le problème avec cette méthode est que ce centre peut donc facilement être surchargé à cause du nombre de consultations. Du point de vue du patient, il n'est également pas toujours évident de se déplacer jusqu'à ce centre qui ne se trouve pas obligatoirement dans la ville la plus proche.

De ce fait, il serait opportun de créer une solution qui permettrait à n'importe quel ophtalmologue de pouvoir définir la présence ou non de kératocône en fonction de quelques critères simples. C'est dans cet objectif que nous avons été contacté pour continuer le projet de création d'un système expert de détection du kératocône.

Notre ligne directrice a donc été de développer une solution qui combinerait le deep-learning et le système expert pour que les données de suivi de la maladie du patient soient intégrées et mises en relation avec les décisions clinique déjà prises. Ce dispositif permettra donc au final aux ophtalmologistes non spécialisés sur le kératocône d'avoir la possibilité de suivre la pathologie et d'être alerté en cas de dégénérescence trop brutale de la maladie pour contacter un centre expert et l'y transférer.

Le CHU de Toulouse propose donc des consultations spécialisées en tant que centre de référence du kératocône. Ces consultations spécialisées se composent d'un suivi du kératocône ainsi que des consultations privées, mais également un travail de contactologie avec la possibilité de créer des lentilles adaptées au patient souffrant de kératocône. Ce service d'ophtalmologie est par le Pr Fournié qui est le chef de service et assure en plus de sa spécialisation du kératocône des chirurgies réfractives, de la cataracte ou encore des greffes de cornée.

b. Objectifs

L'objectif principal de notre projet a été d'appliquer le Deep Learning au suivi du kératocône avec la création d'un système expert pour effectuer des tests pratiques sur la base de données du CHU de Toulouse. L'aboutissement de ce projet est de pouvoir mener des campagnes de test pour voir en action l'efficacité des modèles développés et de ce fait vérifier l'absence d'erreur dans l'analyse des données.

Pour ce faire, nous avons procédé selon différentes étapes chacune nous permettant d 'atteindre des objectifs intermédiaires.

Dans un premier temps, après présentation du sujet par notre commanditaire, nous avons reçu les différents critères de diagnostic du kératocône et sa classification en stade en fonction de la valeur des données recueillies (annexe 1). Nous avons dans un premier temps travaillé sur la compréhension de ces données, mais également d'écrire un premier code pour comprendre et mettre un forme une première ébauche d'un arbre décisionnel. Ce code a été écrit en python et validé par la commanditaire qui nous a alors donné plus de détails sur ses attentes et les améliorations que nous pouvions apporter à ce code.

Pour donner suite à ce début de travail, le groupe a été divisé en deux pour qu'une partie composée de Justin et Nicolas puisse travailler sur le détail du code. Le deuxième groupe composé de Morgan et Julie de leur côté a travaillé sur les différentes possibilités existantes de Deep learning et réseaux de neurones.

Ces recherches ont fourni des pistes intéressantes de travail, mais le temps a joué en la défaveur de notre groupe car la partie code n'a pas eu le temps de mettre en place ces idées compliquées à implémenter. Nous nous sommes basés sur les méthodes fournies par la commanditaire pour avoir un code qui fonctionne et qui permet d'assister un médecin qui ne connaitrait pas les critères de caractérisation des stades avancée du kératocône.

2. Vocabulaire technique

a. Kératocône

Le kératocône est une maladie asymptotique et dégénérative, qui crée des lésions irréversibles sur la cornée. Ces lésions sont caractérisées par un amincissement localisé de la cornée, ce qui va entrainer une baisse qualitative et quantitative de la vision. La cornée va donc passer d'une forme sphérique à une forme très irrégulière et amincie d'allure conique.

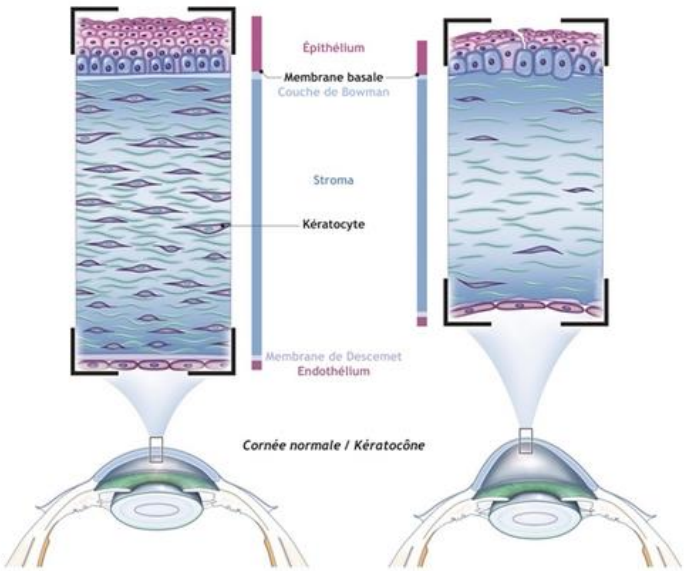


Figure 1 : Représentation de la déformation de la cornée dû au kératocône

Ce qui rend cette maladie encore plus compliquée à prendre en charge est sa prévalence qui est évaluée à 1.38/1000. Ceci ajouté à sa caractéristique d'être asymptotique au commencement de la maladie la rend très compliquée à prendre en charge rapidement. De plus, les ophtalmologues peuvent diagnostiquer cette maladie mais que grâce à de nombreux critères. En effet, si cette maladie est prise en charge assez tôt, il existe des traitements et des opérations permettant une correction rapide de la maladie si elle n'est pas à un stade trop avancé. Notons également, que plus le kératocône commence tôt et plus il est à risque de progresser rapidement et d'évoluer vers une forme sévère.

Les différents stades sont les suivants :

Stade I (léger)	Myopie/astigmatisme <5D (équivalent sphérique) Kmean<48D
Stade II (modéré)	Myopie/astigmatisme ≥5D et <8D Kmean <53D Pas d'opacité Pachymétrie min ≥400
Stade III (modéré)	Myopie/astigmatisme ≥8D et <10D Kmean ≥53D Pas d'opacité Pachymétrie min <400 mais ≥300
Stade IV (sévère)	Réfraction impossible Kmean ≥55D Opacité centrale Pachymétrie min <300

Figure 2 : Tableau des stades du kératocône en fonction des valeurs des paramètres

Les différents paramètres utilisés sont la myopie ou l'astigmatisme en fonction de comment se manifeste la maladie chez le patient.

Les ophtalmologues utilisent des équipements sophistiqués comme des cabinets de réfraction, topographes cornéens Orbscan et Pentacam, biomètre optique IOL Master, champ visuel Humphrey, tomographe en cohérence optique (OCT), aberromètre (Topcon KR1W), rétinographe, microscope spéculaire, laser YAG.

b. Système expert

Un système expert est un dérivé de l'intelligence artificielle qui va permettre d'assister à la prise de décision médicale. De manière générale, ces systèmes utilisent des bases de faits qu'on peut aussi appeler des règles sur lesquelles les différents critères vont se baser. Mais aussi un moteur d'interférence qui sera pour nous, notre système de deep-learning qui va donc chercher dans tous les cas connus par la machine, lequel correspond au mieux au patient pour lequel la requête est envoyée. Grâce à l'expérience, le système va donc essayer d'apprendre et de gagner de l'expérience pour guider au mieux le praticien dans sa prise de décision. Dans cette optique, il convient de rappeler que ce n'est pas le système expert qui va prendre la décision médicale finale, il ne va être qu'un support pour assister le médecin dans sa décision. Il ne doit en aucun cas avoir le dernier mot sur le cas du patient. En effet, le système sera dans une incapacité totale de détecter une maladie ou une phase de la maladie qu'il n'a jamais connu. Dans ce cas il risque de faire des recoupements trop rapides qui mèneraient à un diagnostic erroné, c'est pourquoi la décision ne doit pas reposer entièrement sur le système expert.

Un système expert est composé de la manière suivante : une base de connaissance, un moteur d'inférence, et une interface avec l'utilisateur. La base de connaissance permet de centraliser les connaissances et de les ordonnées pour que le système puisse fonctionner et donner un avis sur la maladie du patient en se basant sur des critères qu'il connait et qu'il a déjà rencontré. Le moteur d'inférence va quant à lui permettre la réflexion du système et représente donc le cœur du système, c'est lui qui va construire dynamiquement le raisonnement en se basant sur la base de données. Il va déclencher les différentes règles en les ordonnant pour qu'elles soient appliquées au nouvelles données qui vont lui être fournies pour le patient en question.

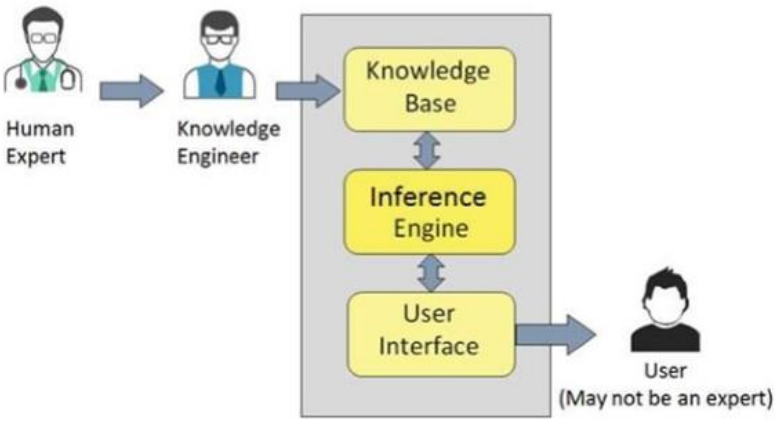


Figure 3 : Schéma représentatif d'un système expert

Ce moteur d'inférence comprend quatre phases : la sélection, le filtrage, le choix des règles et enfin l'exécution. La première phase permet de sélectionner les sous-ensembles des règles qui devront être utilisées en fonction des données fournies en entrée. La phase suivante concerne le filtrage qui détermine dans ce sous-ensemble les règles qui restent applicables. La phase de choix ensuite, confirme la ou les règles qui seront appliquées. Enfin, la dernière phase consiste à l'exécution des règles choisies, ce qui aura comme conséquence sur la base de connaissance initiale de lui en ajouter une nouvelle avec celle qui aura tout juste été traitée.

Pour le bon fonctionnement de ce système, il existe deux type de modes de raisonnement : le chainage avant et arrière. Notre explication sera basée uniquement sur le chainage avant car c'est celui que nous utilisons pour notre projet.

Le chainage avant est le plus simple des deux modes car son moteur d'inférence repose sur des faits pour arriver à son but. Il ne va donc que sélectionner les règles qui sont vérifiées pour ce qu'on lui demande puis il va l'appliquer. Ce principe va se répéter jusqu'à ce que le but soit atteint ou qu'il n'y ait plus de règles applicables. Le principe est donc simple car il ne va reposer que sur une base de connaissance qui associe à un symptôme une maladie et qui va recouper les différents symptômes pour obtenir une maladie finale. On travaille ici avec une stratégie de l'irrévocable.

Le dernier élément indispensable pour que le raisonnement se fasse sont les variables d'entrée. Le système nécessite différentes variables qui peuvent être classées selon trois catégories : les variables qualitatives non ordonnées, les variables quantitatives ordonnées et les variables quantitatives. Pour notre étude, nous avons principalement travaillé sur des variables quantitatives qui sont les mesures effectuées par les appareils de l'ophtalmologue (annexe 2).

II. Le projet

1. Machine learning

Dans cette partie, nous allons aborder le travail effectué par l'équipe de recherche sur le machine learning et les réseaux de neurone.

Dans un premier temps, nous allons rappeler la définition d'un algorithme de machine learning. Le machine learning est une technique de programmation informatique qui utilise des probabilités statistiques pour donner aux ordinateurs la capacité d'apprendre par eux-mêmes sans programmation explicite. Le Deep learning est un sous domaine du machine learning, son objectif est de donner une réponse à une question par le biais d'un algorithme entrainé sur un jeu de données spécifiques à cette question. L'algorithme est dans la majorité des cas basé sur une réseau de neurones.

Nous avons effectué des premières recherches sur le fonctionnement du système expert que nous avons expliqué dans la première partie, puis nous avons porté notre attention sur les différentes techniques de deep learning dans le domaine médical. Les recherches se sont étendues sur plusieurs articles, mais nous porterons une attention toute particulière à trois d'entre eux qui nous ont parus les plus pertinents.

Dans un premier temps, notre synthèse va se porter sur l'article « A comparison of Rule-based and deep learning models for patient phenotyping » [1]. Ce papier porte principalement sur les convolutionnal Neural Network (CNN) qui permettent l'apprentissage et l'identification des phrases d'un texte pour en faire une classification finale. Au cours de cette étude, une importance est portée à l'interprétabilité en étant évaluée par la comparaison de phrases déduites des phénotypes à un dictionnaire développé par des cliniciens. Ce papier propose plusieurs algorithmes avec des fonctionnements et des particularités propres. Il est par exemple possible d'y trouver deux algorithmes distincts de règles pour la lecture des documents de cliniques et pour les documents pathologiques. Ils sont combinables pour identifier la récurrence du cancer du sein. Pour autant des limites ont été notées, comme le coût et le temps de programmation qui l'empêchent de pouvoir appliquer ces méthodes sur des tâches plus conséquentes et répétitives. Il présente également une approche semi-supervisée où le clinicien devra initialement définir des « ancres » pour entrainer le modèle sur des données structurées ainsi que des notes de prédiction. Ces « ancres » sont des phrases types qui lui sont propres qui permettent d'identifier le concept avec une haute valeur positive de prédiction (PPV). Le clinicien écrivant souvent les mêmes diagnostics ou du moins avec le même vocabulaire, il aura donc moins d'efforts à faire pour entrainer le modèle, car il ne lui demandera d'apprendre que des phrases qu'il sait types. Cette méthode permet également de réduire la complexité, tout en ayant un haut PPV et une haute sensibilité. Pour que ce processus se fasse, les auteurs ont proposé d'utiliser la méthode cTAKES pour extraire le concept de toutes les notes. Ce principe repose sur la division de la phrase en mots. Ces mots sont ensuite normalisés (on leur retire le pluriel, le féminin pour qu'ils soient sous la forme la plus brute), tagués (nom, verbe…) et un arbre d'analyse superficielle est créé pour représenter la structure grammaticale d'une phrase. Pour connaitre le plan type d'une phrase et donc pouvoir comprendre les mots précédemment transformés. Au final, un algorithme de reconnaissance de nom permet de relier les différentes entités à un concept grâce au sens qui leur aura été préalablement donné.

En ce qui concerne les réseaux de neurone, nous avons lu deux articles sur ce sujet. Le premier intitulé « Introduction to Learning Rules in Neural Network »[2] décrit plusieurs méthodes de règles permettant de comprendre le fonctionnement des réseaux de neurone. Nous aborderons les quatre principales expliquées dans l'article. La première se nomme « Hebbian learning rule » et suggère que lorsque deux neurones sont excités conjointement, un lien les unissant est créé ou renforcé. Cette théorie tente d'expliquer l'apprentissage associatif, dans lequel une association est faite par la répétition de deux stimuli. La répétition d'un stimulus seul entraine le rappel de l'autre ensuite. La

seconde règle explique le processus d'apprentissage d'un perceptron. En effet, le perceptron est un algorithme d'apprentissage supervisé de classifieur binaire. Il s'agit d'un algorithme qui va calculer seul les poids synaptiques et qui va ensuite réaliser plusieurs tests sur les nœuds du réseau afin de vérifier si les poids sont corrects. Il va également pouvoir ajuster si dans le cas contraire les valeurs calculées ne sont pas tout à fait cohérentes avec celles qui auraient dû être trouvées. La règle suivante est le « delta learning rule » qui compare la sortie attendue avec la sortie obtenue. Cette valeur sert ensuite pour réduire l'écart qui correspond à l'erreur. La dernière règle est nommée « outstar learning rule » et porte sur l'organisation des neurones en couches. Tous les poids connectés à un certain neurone doivent égaler la sortie de tous les neurones connectés à ces poids.

Le dernier article que nous avons lu porte sur les problèmes majeurs des réseaux de neurone qui sont le manque d'interprétabilité, ainsi que le besoin important de gros dataset initiaux. Ce papier propose une solution, le « Rule-embedded Neural Network »[3] (ReNN). Son principe repose sur une stratégie d'optimisation en deux étapes : la séparation d'une première déduction neuronale locale, pour ensuite assembler ces modèles locaux pour former des modèles globaux. Pour une explication plus claire, nous allons prendre l'exemple de la photo d'un visage. Le système va dans un premier temps s'atteler à créer des connexions locales pour reconnaitre un œil, une bouche, un nez. De cette manière, chaque partie du visage sera connue par le système qui aura fait des modèles de détection pour chaque petite partie locale. Puis dans un second temps, le système va remettre tous ces modèles locaux ensemble pour reconnaitre un visage dans son intégralité. Le réseau de neurone n'aura donc pas à travailler sur l'ensemble des données, mais sur des petits parties qui seront localisées et beaucoup plus simples et rapides à traiter pour au final rassembler tous les éléments pour ne faire qu'un visage.

Nous avons tout particulièrement sélectionné ces trois articles car ils nous paraissaient d'une grande utilité pour notre projet. En effet, le premier article qui traite sur la reconnaissance de texte nous aurait été d'une grande aide car les données utilisées par le système expert auraient dû être basées sur une photo de compte rendu de tests ophtalmologiques. Ce qui nous aurait donc permis de lire les données et de les transmettre directement au système expert pour qu'il puisse les analyser. Le deuxième article quant à lui nous donnait des règles primordiales dans la compréhension du fonctionnement d'un réseau de neurone, ce qui aurait été très utile pour la mise en place de la méthodologie du troisième article. En effet, la segmentation du travail du réseau de neurone est très utilisée pour fractionner sonn fonctionnement. En effet, nous aurions demandé à chaque modèle local de travailler sur un paramètre pour ensuite remettre tous ces paramètres locaux ensemble pour définir le stade de la maladie du kératocône chez le patient.

Nous allons maintenant aborder la partie beaucoup plus technique effectuée principalement par Justin et Nicolas.

2. Les technologies utilisées

Durant ce projet nous avons utilisé différentes technologies :

Environnement de développement :

- **Visual Studio Code :** comme environnement de travail car il prend en charge les principaux langages de programmation
- **Sublime Text :** Éditeur de texte pour la partie Tkinter

Pour le développement du backend :

- **Python** : langage de programmation utilisé pour développer cette application car il a été utilisé par d'autres applications au CHU et d'autre part car ce langage est largement reconnu pour sa lisibilité et sa facilité d'utilisation.
- **Django**: framework python open source. Il est généralement utilisé pour faire des sites web en python (cf Annexe)

Pour la mise au point du backend :

- **Postman** : logiciel qui permet d'envoyer des requêtes http au serveur permettant de tester la bonne intégration au backend des nouvelles fonctions.

Pour le développement du frontend:

- **RectJS :** framework javascript front-end utilisé pour les applications Web

Pour la gestion du code et de la documentation:

- **Github :** Pour ce projet un repository GIT a été créé sur lequel le code était partagé.

3. Application

a. Mise en place Système Expert sous python

Pour mettre en place ce système expert nous avons découpé notre travail en plusieurs sous-objectifs :

- Implémenter chaque méthode indépendamment les unes des autres. Ici toutes les données sont rentrées à la main.
- Grouper toutes ces méthodes de manière à avoir le système complet.
- Faire une première interface graphique.
- Automatiser l'entrée des données pour éviter au médecin d'avoir à les rentrer à la main.

Ces méthodes ont été implémentées sous python. Pour ce faire nous avons préalablement travaillé sur la détermination des données dont nous allons avoir besoin (annexe 2). Cela fait, nous avons implémenté de manière isolée chaque méthode. Il faut savoir que nous avons implémenté ces méthodes avec un enchaînement de boucles « if » ou « for ».

Ce premier travail permettait à l'utilisateur de rentrer les valeurs des données sur le panneau de commande pour obtenir le résultat affiché selon la méthode demandée.

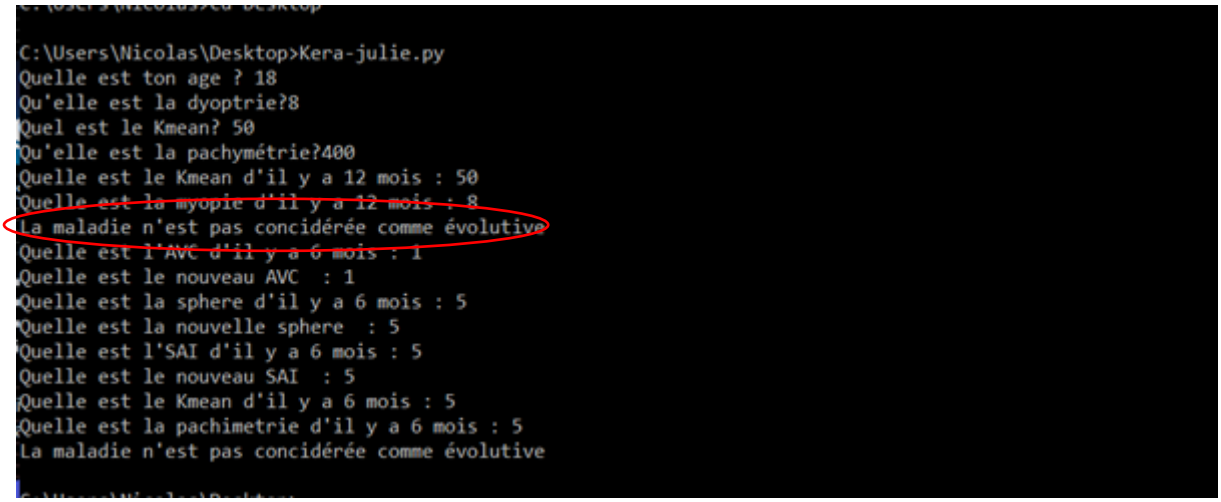


Figure 4 : Résultat (en rouge le diagnostic d'une méthode)

Ensuite nous avons regroupé certaines de ces méthodes, qui étaient complémentaires et permettaient d'avoir le diagnostic finalisé.

b. Tkinter

Organisation du code

Après avoir réalisé les différentes méthodes en langage python, nous voulions pouvoir les visualiser ainsi que les faire fonctionner en même temps sur un même fichier. De ce fait, nous avons commencé à nous intéresser à la création d'une interface graphique regroupant toutes les méthodes codées précédemment. Pour cela nous avons décidé de faire notre première application avec la bibliothèque graphique Tkinter. Nous avons choisi de prendre cette bibliothèque car elle est simple d'utilisation. De plus, elle est déjà présente dans la bibliothèque de Python standard. De nombreux cours sont disponibles sur internet et permettent un apprentissage simple et rapide de ces fonctionnalités. Elle permet d'avoir une application fonctionnant sans une architecture de code imposant. Tout le code était contenu dans un seul fichier .py possédant une seule classe. Cette application avait pour objectif de permettre à un médecin de connaître l'état de la maladie du Kératocône d'un patient et de déterminer si la maladie était évolutive ou non. Le tout en rentrant les données du patient directement dans l'application. 5 méthodes ont ainsi été implémentées dans l'application.

Figure 5 : Ecran d'accueil de l'application

Tout d'abord « Le suivi du Patient », cette méthode permettait de connaître le KC du malade ainsi que la date du prochain rendez-vous qu'il devait prendre. Pour ce faire le médecin devait rentrer des données telle que l'âge du patient, la taille de la dyoptrie, le Kmean, l'opacité, etc... Ensuite, il y a deux méthodes qui permettent de savoir si la maladie peut être considérée comme évolutive ou non. La méthode « Mazzotta » et la méthode « Wittig-Silva ». Ces deux méthodes comparent les données actuelles avec les données datant de 4 à 6 mois. Les deux dernières méthodes sont des fonctions plus importantes. En effet, elles permettent de savoir quelles sont les propositions thérapeutiques dont le patient peut bénéficier. Chacune des deux méthodes permet de faire un diagnostic en fonction de l'évolution ou non de la maladie. A la fin de l'utilisation de chacune de ces méthodes, une fenêtre s'ouvre. Cette fenêtre contient le résultat du diagnostic effectué par la méthode choisie ainsi qu'un bouton « Okay » qui permet de fermer cette fenêtre. Lorsque cette fenêtre disparait, on retourne sur l'écran d'accueil où il est possible d'effectuer une des cinq autres méthodes.

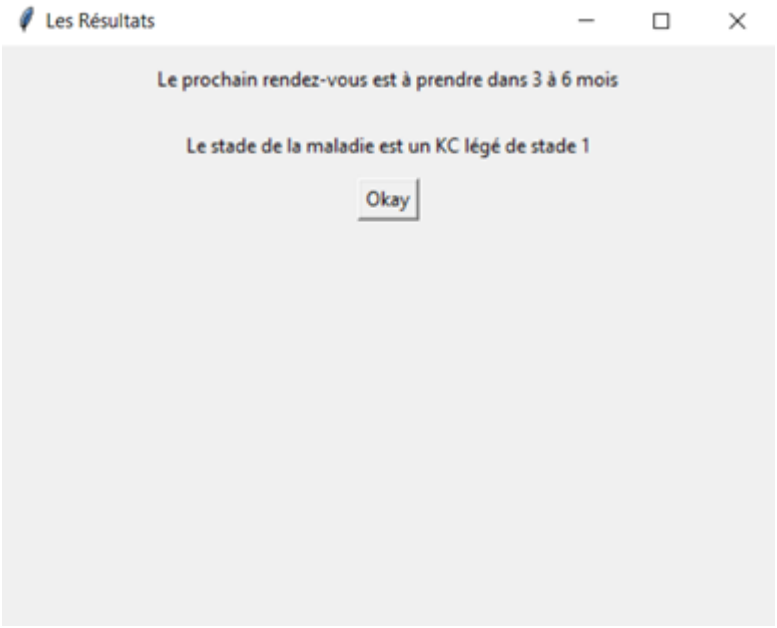


Figure 6 : Fenêtre affichant le résultat de la méthode

Pour rendre l'application plus ergonomique, nous avons mis en place un bouton « Quitter » sur la page d'accueil permettant de fermer l'application proprement. De plus, étant donné que le médecin doit rentrer les données à la main, nous avons décidé que chaque champ doit être validé pour passer au suivant. Mais aussi que le médecin puisse retourner à une valeur rentrée précédemment. Cette fonctionnalité de retour permet d'éviter de devoir recommencer entièrement la méthode en cas de faute de frappe sur le clavier. Ensuite nous avons intégré la mise en place de valeur par défaut pour chaque champ renseigné par le médecin. Avec cela, même si le médecin oublie de renseigner un champ lors de l'utilisation d'une méthode, l'application prendra cette valeur par défaut pour éviter des erreurs côté utilisateur.

Figure 7 : Interface de complétion de champ de données

Limite de Tkinter

Malgré le fait que Tkinter est simple d'utilisation et ne nécessite pas une architecture complexe, il possède de nombreux défauts. En effet, Tkinter est une bibliothèque ancienne, avec une API ancienne et qui manque considérablement de souplesse. La bibliothèque Tkinter ne permet d'obtenir que des interfaces utilisateur simples avec des widgets élémentaires. Pour passer outre ces difficultés, nous avons téléchargé de nombreuses ressources supplémentaires. De plus, il est parfois difficile à déboguer dans la mesure où les widgets Tkinter ne sont pas à la base des objets Python. Parfois l'obtention de message d'erreur non explicite pouvait engendrer des pertes de temps conséquentes sur la partie programmation. Le fait que Tkinter se base sur un modèle de programmation simple est un avantage mais devient problématique et encombrant lors de la création d'interface plus complexe.

Changement de langage

Avec l'évolution du projet, le commanditaire voulait éviter au maximum que le médecin rentre les données à la main dans l'application. Avec ceci, l'application devenait beaucoup plus complexe. En effet, le médecin devait simplement soumettre à l'application une image de la topographie de l'œil ou un fichier contenant les données du patient. Dans un premier temps, pour répondre à cette problématique de donnée, nous avons décidé d'utiliser la méthode de lecture de donnée avec un fichier CSV contenant toutes les données nécessaires à une bonne utilisation de l'application. Or cela n'est pas possible avec la bibliothèque Tkinter. En effet, on ne pouvait récupérer des fichiers depuis un ordinateur. Avec toutes les contraintes que nous engendrait l'utilisation de Tkinter nous avons décidé de le mettre de côté. A partir de là nous avons décidé de rechercher de meilleures solutions tant au niveau de l'architecture de codage que des frameworks à utiliser. Cette nouvelle solution devait permettre de répondre à toutes les demandes du commanditaire dans le temps imparti.

c. Architecture du code

Architecture du code

À la suite des demandes de notre commanditaire nous avons décidé de revoir notre travail afin que ce dernier réponde aux attentes et permette une saisie automatique des données. Nous avons aussi décidé de mettre en place une architecture REST API afin de rendre notre code modulable et interopérable pour qu'ensuite l'intégration au CHU puisse se faire plus facilement. Nous allons commencer par détailler rapidement ce qu'est une architecture REST API.

Rest-Api :

Une Api Rest est un style d'architecture qui permet à des logiciels de communiquer entre eux. Cela se fait au moyen de requêtes HTTP. Ce type d'architecture permet d'avoir une interopérabilité entre les différents logiciels car grâce aux API Rest même si ces derniers ont des architectures différentes, des OS différents, ou autres, ils peuvent tout de même communiquer.

Cela se base sur quatre règles :

Une URI qui va permettre d'identifier les différentes ressources. L'URI sera toujours construite de la même façon. Par exemple : *http://mywebsite/site*

Ensuite on a un verbe HTTP. Voici les 4 principaux :

- **POST**: permet de créer une information
- **GET**: permet de lire une information
- **PUT**: permet de mettre à jour une information
- **DELETE**: permet de supprimer une information

D'autres verbes existent mais ces quatre sont les plus utilisés. Pour notre application nous avons principalement utilisé la méthode POST.

Ce verbe sert de méthode lors de l'appel de l'URI. Par exemple, si le verbe HTTP est un DELETE lors de l'appel de l'URI, la ressource appelée sera supprimée.
Nous avons ensuite une réponse HTTP. Il s'agit d'une représentation de la ressource dans un format donné, par exemple JSON, XML, etc.

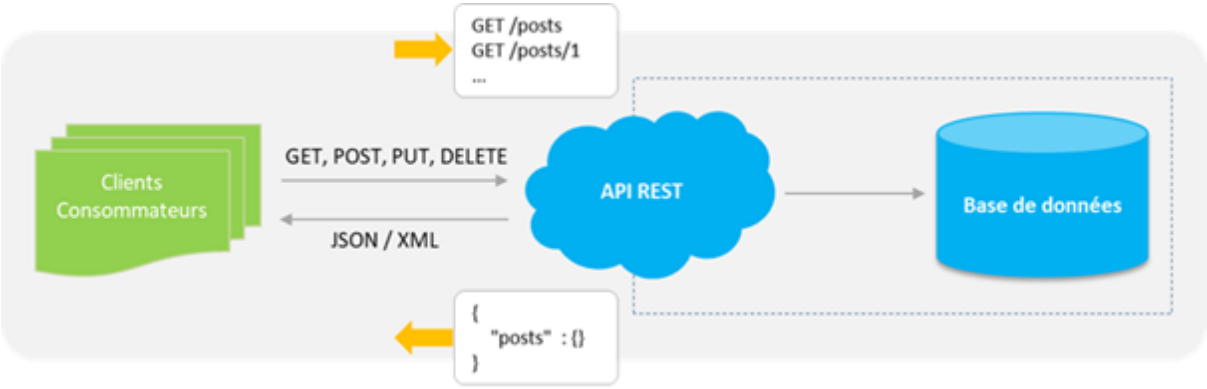


Figure 8 : Schéma API REST

Organisation du code

Nous allons maintenant aborder notre organisation du code.
L'infrastructure de l'application a été découpée en deux serveurs:

- Un serveur front-end permettant l'affichage à l'utilisateur.
- Un serveur back-end permettant de traiter les différentes requêtes soumises par le client.

Ces deux serveurs communiquent entre eux via REST API.

Back-end

Notre backend a été codé entièrement en python. Il reçoit les informations via le front, les traite et ensuite renvoie un résultat. Nous l'avons divisé en plusieurs APIs. L'une d'elle va permettre de traiter tout ce qui est relatif au patient. L'autre tout ce qui concerne le diagnostic de la maladie on a ensuite les deux algorithmes permettant de déterminer la thérapie à suivre selon si le KC est évolutif ou non. Chaque API est appelée via son URI.

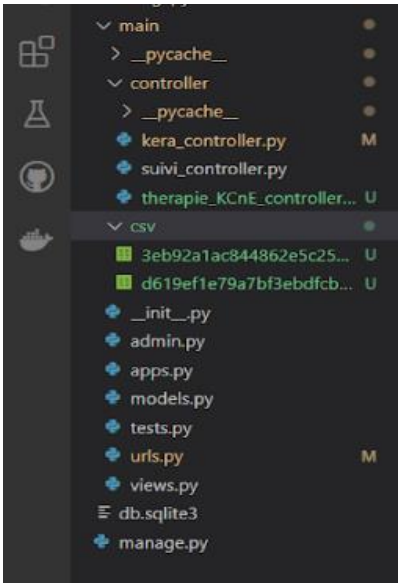


Figure 9 : Architecture

Le backend est construit de la manière suivante : un dossier controller contenant tous nos controllers se sont ces derniers qui sont appelés via le front. Le dossier CSV nous permet de stocker momentanément le fichier CSV envoyé depuis le front. Ce dernier est vidé à la suite du traitement du fichier. Le fichier URLS.py contient les chemins relatifs permettant d'appeler nos différents controllers. Le fichier test.py permet comme son nom l'indique d'écrire des classes de test. Les autres fichiers et dossiers ne nous sont pas utiles mais restent car ils sont nécessaires au bon fonctionnement du serveur.

Son fonctionnement est le suivant. Il reçoit via le frontend la ressource appelée ainsi que le fichier CSV contenant les données. Nous allons détailler par la suite le fonctionnement de l'API concernant le diagnostic de la maladie car il s'agit, à notre sens, de la partie la plus intéressante. Dans un premier temps nous extrayons les différentes données de ce fichier CSV que nous stockons dans ces variables. Ensuite via l'URL nous recevons la méthode à appliquer (sélectionnée via le front par l'utilisateur). L'algorithme détermine ensuite un résultat selon les données que nous avons récupérées puis ce résultat est envoyé au front.

Front-end

En ce qui concerne le Front-end, nous avons utilisé le Framework ReactJS pour notre application. Ce dernier utilise du JSX une extension syntaxique de JavaScript. Afin de ne pas avoir à créer l'ensemble des éléments (boutons, menus, feuilles de style, etc.) de zéro, nous avons utilisé la librairie Bootstrap. Il s'agit d'une librairie de composants pour l'interface utilisateur.

La communication entre le frontend et le backend se fait via les fichiers services.

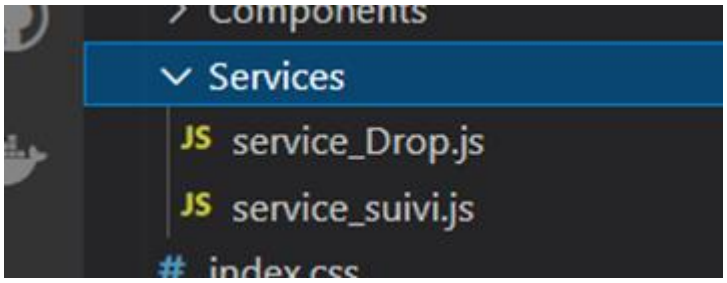


Figure 10 : Services

Ces services contiennent les méthodes d'appel du backend. Nous avons tout d'abord l'URL qui permet d'appeler la bonne API. On constate que cet url contient la ressource demandée par l'utilisateur (ici il s'agit de la méthode que souhaite utiliser l'utilisateur stojanovic, mazzotta, etc). Nous avons ensuite un body (ou payload) via lequel nous transmettons notre fichier CSV contenant les données au back-end. Ensuite nous transformons au format JSON la réponse obtenue que nous affichons ensuite au niveau de l'interface utilisateur.

```
const sendfiles={
    get_data(csv,methode){
        return fetch('/kera/'+ methode, {
            credentials: 'include',
            method: 'POST',
            headers: {
              Accept: 'application/vnd.ms-excel',
              'Content-Type': 'application/vnd.ms-excel'
            }
            ,
            body: csv
          }).then( (answer) => {
            return (answer.json())
          }).catch((error) => { throw error })
        },
}
export default  sendfiles
```

Figure 11 : Exemple de méthode d'appel du backend

Voici ensuite à quoi ressemble l'application finale.

Sur la page d'accueil on peut voir plusieurs onglets. Le premier concerne tout ce qui est relatif au diagnostic de la maladie. Le second concerne le suivi d'un patient si la maladie est avérée. Enfin les deux derniers concernent la thérapie selon si le KC est évolutif ou non.

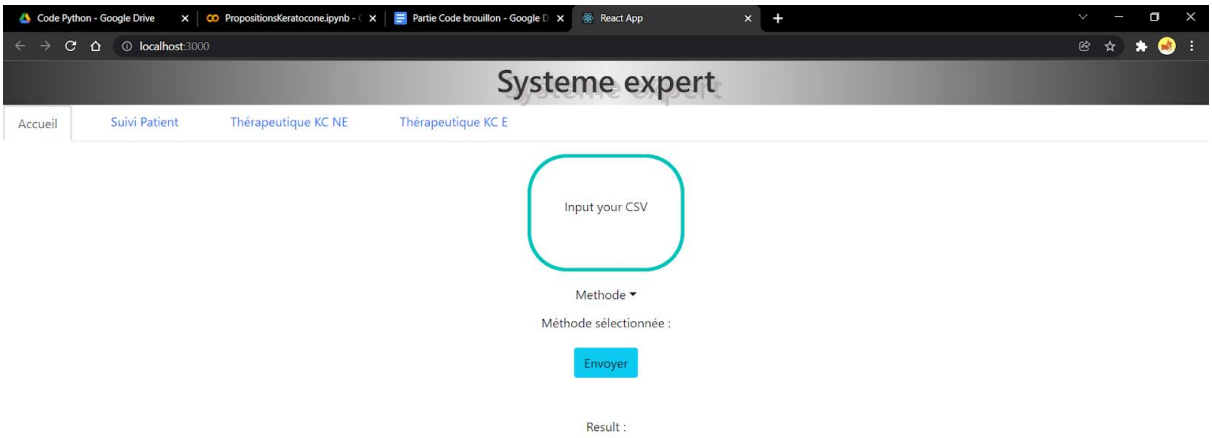


Figure 12 : Ecran d'accueil de l'application

Sur l'onglet concernant le diagnostic, nous pouvons voir une zone de dépôt. Cette zone de dépôt permet de glisser/déposer un fichier .csv contenant les informations relatives à un patient. Ensuite nous avons un bouton de type dropbox qui permet à l'utilisateur de sélectionner la méthode qu'il souhaite utiliser. Cette dernière s'affiche en bleu une fois sélectionnée.

Figure 13 : Demande de la méthode

Le résultat de la requête s'affiche ensuite dans la partie "result".

Figure 14 : Demande de données sous format CSV

Sur l'onglet "Suivi" le principe est similaire. Nous avons là aussi des dropbox grâce auxquels le médecin peut sélectionner les champs qui l'intéresse.

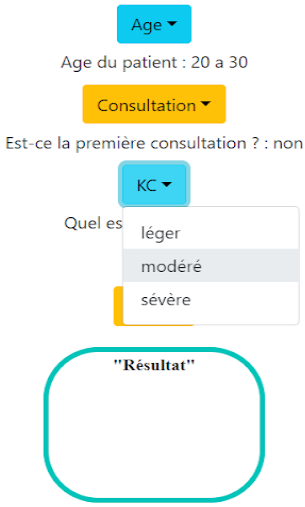


Figure 15 : Choix des champs

Ensuite le résultat s'affiche là aussi dans la partie "result".

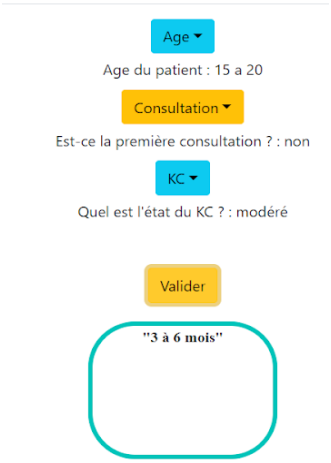


Figure 16 : Résultat

Nous avons aussi un onglet thérapie KC non évolutif qui permet au médecin de savoir quelle thérapie doit suivre son patient en fonction de paramètre tel que son AV, des opérations qu'il a suivies ou de type de KC (central, etc…)

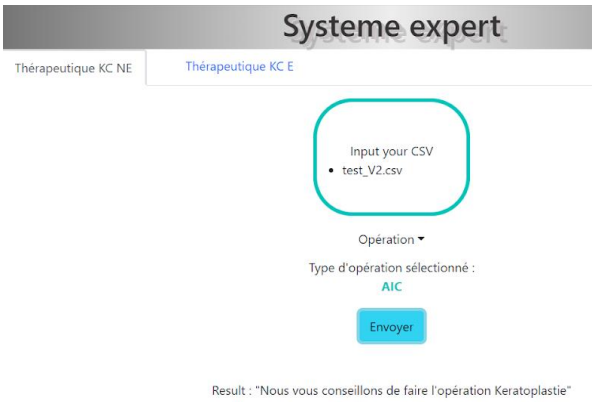


Figure 17 : KC non évolutif

4. Piste d'amélioration

a. Tests, validation et pistes d'amélioration

Nous n'avons pas réalisé de tests à proprement parler, mais nous avons assuré la bonne application de notre code au fur et à mesure du projet en lançant directement sur un navigateur les fonctionnalités développées. Pour ce faire, nous avons utilisé le navigateur Google Chrome. Ce dernier permet d'afficher à l'écran les sorties console via le module d'inspection. Nous avons aussi testé l'application sur Firefox ainsi que Microsoft Edge et nous n'avons pas rencontré de problèmes spécifiques.
Afin d'améliorer ces tests, surtout au niveau du backend, le framework Django permet de créer plutôt facilement des classes de test. Il serait donc envisageable d'utiliser cette fonctionnalité pour réaliser des tests unitaires et d'intégration afin que le code soit couvert dans sa globalité par des tests.

b. Authentification

Une des pistes d'amélioration au sein de l'application serait d'installer un système d'authentification. En effet, cette application ayant pour finalité d'être utilisée par des médecins et amenée à utiliser des données patients, il est utile de mettre en place un tel système afin d'avoir une première couche de sécurité. Nous avons choisi de ne pas l'implémenter afin de passer plus de temps sur le fonctionnement global de l'application.

c. Design

Afin d'avoir un visuel plus pratique, ergonomique et esthétique de l'application, nous pourrions voir avec les bénéficiaires comment améliorer son design avec pour objectif une utilisation facile et optimale pour ces derniers.

d. Thérapie KC évolutif

Concernant cette partie nous n'avons malheureusement pas eu le temps de terminer son implémentation au sein de l'application finale. Il s'agirait donc, dans l'idée ou le projet aurait une continuité, de l'implémenter.

e. Documentation du code

Nous n'avons, à l'heure actuelle, écrit aucune documentation concernant notre code. L'utilisation de la bibliothèque Sphinx serait judicieuse car cette dernière via les commentaires interne au code permet de générer automatiquement une documentation propre et claire.

5. Tesseract

Nous avons voulu mettre en place un système capable de récupérer les données contenues sur les images des outils ophtalmologiques tels que le topographe Pentacam et le topographe TMS. Voici par exemple une image test du topographe Pentacam :

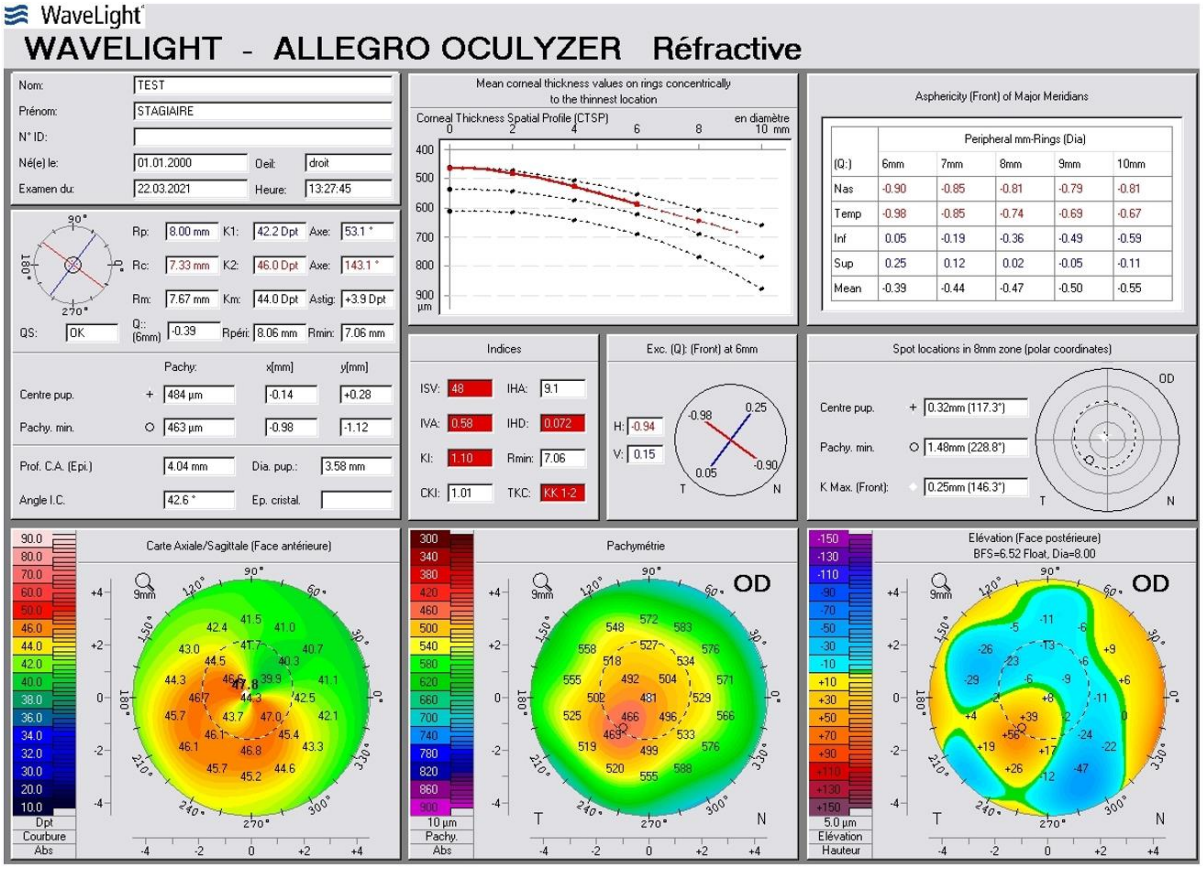


Figure 18 : Capture d'écran du test de topographe Pentacam

Pour réaliser l'extraction des données, nous avons utilisé Tesseract-OCR, un moteur de reconnaissance optique de caractères, qui permet d'extraire toutes les données sur une zone définie. Pour faciliter l'extraction et la précision nous avons fait le choix de limiter les zones d'extraction au libellé de la case ainsi que la case, comme cet exemple le montre :

Figure 19 : Libellé d'une des cases

Malheureusement, même avec des paramètres d'optimisation conçus par tesseract, la reconnaissance n'est pas parfaite et il y a des erreurs, voyons cela avec un exemple : voici le résultat dans le terminal du script python exécuté

```
PS C:\Users\mjp81\Documents> python .\tesseract1.py
Texte:  Kt: [A22Der
? <PIL.Image.Image image mode=RGB size=80x25 at 0x1EB3ACC0A30>
```

Figure 20 : Résultat du terminal

On peut voir que K1 devient Kt et que 42.2 Dpt devient A22Der donc le résultat ne correspond pas à la donnée de l'image.

Figure 21 : : libellé erroné de la case

```
PS C:\Users\mjp81\Documents> python .\tesseract1.py
Texte:  K2; [46.0Dpt
♀ <PIL.Image.Image image mode=RGB size=80x25 at 0x29B3C790A30>
```

Figure 22 : Résultat erroné du terminal

III. Conduite de projet

1. Parties prenantes

Fonction	Prénom / NOM	Adresse mail
Interne en ophtalmologie au CHU de Toulouse	Ines DRIRA	drira.ines@gmail.com
Professeur / Chef de service	Pr. Fournié	
Tuteur académique	Imen MEGDICHE	imen.megdiche@univ-jfc.fr
Elève ingénieur (Chef de projet)	Morgan MARTY	morgan.marty@etud.univ-jfc.fr
Elève ingénieur	Nicolas JACOMI	nicolas.jacomi@etud.univ-jfc.fr
Elève ingénieur	Justin MAZOYER	justin.mazoyer@etud.univ-jfc.fr
Elève ingénieur	Julie LEONE	Julie.leone@etud.univ-jfc.fr

Figure 23 : Tableau des parties prenantes

Le chef de projet prend en charge l'animation des réunions et leurs planifications, ainsi que la consolidation des documents relatifs à l'évolution du projet. Il est la pierre angulaire du projet car il maintient un lien fort entre le commanditaire et l'équipe projet.

2. Organisation au sein du groupe

L'organisation pour un projet tuteuré est primordiale pour le bon déroulement de celui-ci. En effet, le contact avec le commanditaire et la planification sont les deux axes principaux à privilégier. Nous avons donc mis un point d'honneur à garder contact régulièrement avec notre commanditaire. Nous communiquions facilement par mail si nous rencontrions une difficulté. Inès Drira était également très présente pour faire une réunion de dernière minute pour que nous puissions nous mettre d'accord sur la direction à prendre en cas de doute.

Pour autant, nous avons établi un programme de réunions que nous avons organisé toutes les deux semaines sauf problème extérieur. Durant ces réunions toutes les parties prenantes étaient présentes sauf le Pr Fournié qui n'a pas réussis à être présent à chaque fois.

De notre côté, nous avons tenu un document recensant les tenants et aboutissants des réunions pour garder une trace de ce qui était dit et constater l'évolution entre les différentes réunions.

De plus, même si nous avons toujours fait nos réunion avec le groupe entier, nous avons travaillé en binôme comme expliqué précédemment car le travail a été réparti selon un axe plus technique avec le code sur python et des recherches plus théoriques sur le fonctionnement d'un système expert et sur les différentes méthodes de deep learning utilisables pour ce projet.

Pour ce qu'il en est de la communication entre les membres de l'équipe et les commanditaires, nous avons créé un serveur discord pour la communication générale avec les commanditaire, mais aussi au sein même de l'équipe. Pour les réunions, nous avons mis en place un lien permanent Jitsi qui est une application web permettant de faire des réunions.

3. Planning prévisionnel

Le planning prévisionnel est une estimation de la durée et de la réussite des différentes tâches prévues pendant le projet. Pour que celui-ci voit le jour, nous avons listé les différents objectifs à atteindre et notre estimation de leur durée. Nous avons réalisé ce planning prévisionnel sur L*ucidchart* ci-dessous.

Comme expliqué précédemment, nous avons divisé le groupe en deux pour une partie code et une partie recherche, ce que nous avons représenté sur le diagramme de gant.

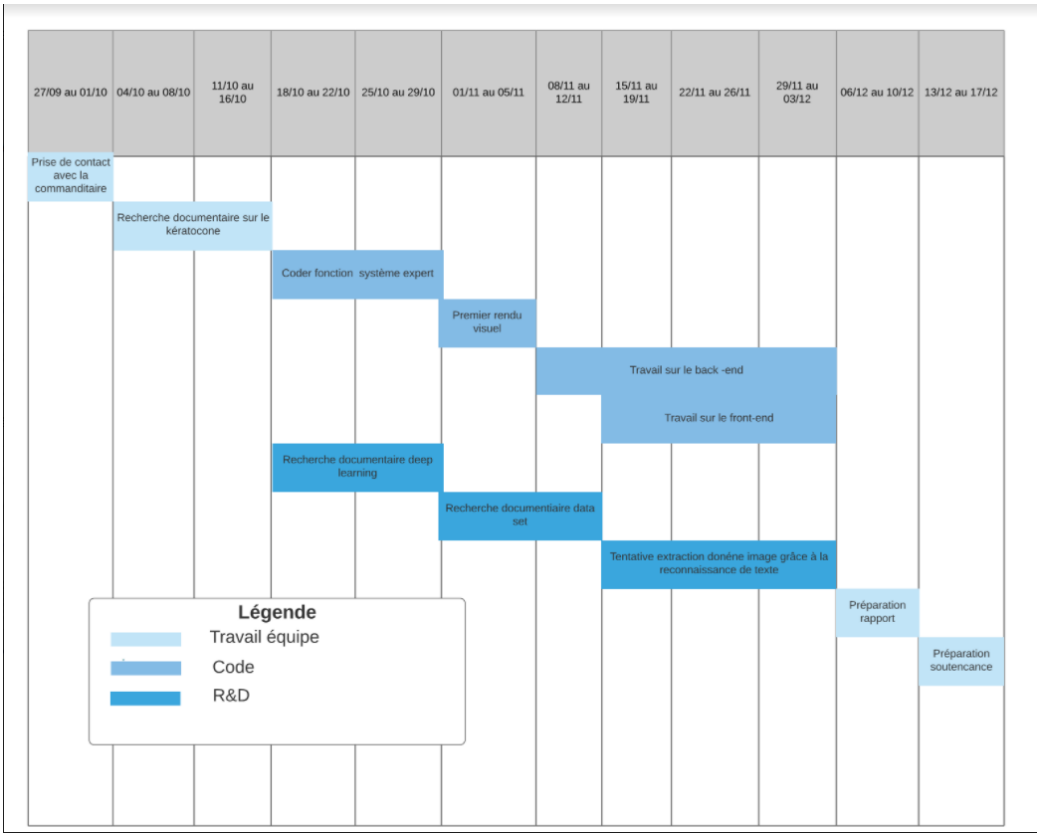


Figure 24 : Diagramme de Gant

4. Difficultés rencontrées

Le projet a également reposé sur différentes contraintes et difficultés comme la distance. En effet, notre système expert devait être initialement testé sur la base de données d'ophtalmologie du CHU de Toulouse, sauf qu'une fois le système effectif, il n'a pas été possible de nous déplacer au CHU. En effet, pour que l'entrainement du système soit aussi efficace que possible, nous aurions dû l'entrainer sur un nombre considérable de données, ce qui était trop compliqué d'anonymisée en un temps aussi réduit par la commanditaire. Le test final de fonctionnalité n'a donc pas eu la possibilité de se faire. Nous avons donc dû baser l'entièreté de notre système expert sur un fichier CSV de quelques lignes pour tester le fonctionnement du code.

Une autre difficulté que nous aurions eu la capacité d'éviter a été le travail sur le deep learning. En effet, nous pensons nous être concentré trop tôt sur cette partie alors que le code aurait dû être notre priorité. En effet, il nous a été demandé de travailler sur cette partie dans le but de nous en servir plus tard. Pour autant, en vue du temps qui nous était alloué pour le projet nous aurions dû nous douter que cette partie aurait été trop chronophage pour que nous puissions finir cette partie du projet.

Au niveau organisationnel, des heures allouées au projet tuteurées se sont vues déplacées pour être supprimées à cause de l'ajout de nouveaux cours à la place.

De la même manière, nous n'avons pas réussis à intégrer la lecture d'image de bilan de résultats pour que l'analyse se fasse sur cette image et non sur un CSV. En effet, cette méthode impliquerait moins de contraintes pour le spécialiste qui n'aurait seulement besoin que de donner en variable d'entrée l'image du bilan des mesures faites sur le patient.

Conclusion

Nous avons réalisé un projet tuteuré de fin septembre à mi-décembre pour le service d'ophtalmologie du CHU de Toulouse. Celui-ci fait partie du Centre de Référence du Kératocône (CNRK) et est donc un pilier dans le domaine médical pour le diagnostic de cette maladie. Pour autant, cette maladie même si connue de tous les ophtalmologues est très difficile à diagnostiquer étant donné son caractère asymptotique. C'est pourquoi dans le doute, de nombreux ophtalmologues peuvent soient passer à côté de la maladie dans un diagnostic par manque de connaissance sur les critères de diagnostic. Mais dans le cas contraire, de nombreux patients sont transférés inutilement dans un centre de référence par mauvais diagnostic de l'ophtalmologue général. Ce qui entraine un engorgement du service qui est déjà beaucoup sollicité car trop peu présents sur le territoire français. C'est pourquoi la création d'un système expert serait une opportunité sans précédent pour aider les ophtalmologues non spécialisés à diagnostiquer correctement un patient atteint de kératocône.

Notre projet s'est donc déroulé sur trois grands axes. Nous avons dans un premier temps pris connaissance des différents paramètres pour écrire un premier code de système expert, nous avons continué par un recherche des possibilités de machine learning et enfin nous avons finalisé le code avec l'aide et les conseils de notre commanditaire.

Nous nous sommes également rendu compte de l'importance de la présence des données pour que notre travail soit aussi efficace que possible. Mais aussi de la communication avec la commanditaire qui en tant que professionnelle a su nous aiguiller pour cibler aussi bien que possible les besoins des ophtalmologistes pour détecter une telle maladie que le kératocône. En effet, comme elle nous l'a expliqué, cette maladie peut être traitée si prise à temps en charge, il est donc primordial que les médecins spécialisés soient dans la capacité de la détecter pour transmettre les patients vers des services plus adaptés.

Ce projet nous aura appris à travailler sur un lapse de temps assez réduit, tout en s'adaptant avec l'impossibilité de tester son code en temps réel. De ce fait, il est tout à fait envisageable de le continuer sous forme de stage ou de projet tuteuré avec le travail déjà effectué pour fournir un vrai système expert à la commanditaire.

Bibliographie :

[1] : A Comparison of Rule-Based and Deep Learning Models for Patient Phenotyping Sebastian Gehrmann0,1* , Franck Dernoncourt0,2, Yeran Li0,3, Eric T Carlson0,4 , Joy T Wu0,5, Jonathan Welt0,6, John Foote Jr.0,7, Edward Moseley0,8, David W Grant0,9 , Patrick D Tyler0,5, Leo Anthony Celi0,2 0MIT Critical Data, Laboratory for Computational Physiology 1Harvard John A. Paulson School of Engineering and Applied Sciences 2Massachusetts Institute of Technology 3Harvard T.H. Chan School of Public Health 4Philips Research North America 5Beth Israel Deaconess Medical Center 6Massachusetts General Hospital 7Tufts University School of Medicine 8University of Massachusetts 9Washington University School of Medicine

[2] : Harnessing Deep Neural Networks with Logic Rules; Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, Eric P. Xing

[3] : ReNN: Rule-embedded Neural Networks Hu Wang Data Analysis and Algorithm Department LOHAS Technology (Beijing) Corporation Limited Beijing, China

Annexes :

Annexe 1 : Règles décisionnelles

Concernant le suivi :
Guidelines Bordeaux :

		KC léger	KC modéré	KC sévère
< 15 ans	**1ère consultation après le diagnostic**	1 à 3 mois ou CXL d'emblée	1 à 3 mois ou CXL d'emblée	CXL d'emblée si non contre-indiqué ou surveillance à 1 mois
	Suivi si KC stable	3 à 6 mois	3 mois	3 mois
15 à 20 ans	**1ère consultation après le diagnostic**	3 mois ou CXL d'emblée	3 mois ou CXL d'emblée	CXL d'emblée si non contre-indiqué ou surveillance à 1 mois
	Suivi si KC stable	6 mois	3 à 6 mois	3 mois
20 à 30 ans	**1ère consultation après le diagnostic**	3 à 6 mois	3 à 6 mois	3 mois
	Suivi si KC stable	6 mois	6 mois	3 à 6 mois
30 à 40 ans	**1ère consultation après le diagnostic**	6 mois	6 mois	3 à 6 mois
	Suivi si KC stable	6 mois	6 mois	6 mois
> 40 ans	**1ère consultation après le diagnostic**	6 mois	6 mois	6 mois
	Suivi si KC stable	1 an	1 an	6 mois à 1 an

Guidelines Toulouse

		Quel que soit le stade du KC
< 15 ans	**1ère consultation après le diagnostic**	3 mois
	Suivi si KC stable 4 examens	6 mois
15 à 20 ans	**1ère consultation après le diagnostic**	6 mois
	Suivi si KC stable	6 mois
20 à 30 ans	**1ère consultation après le diagnostic**	6 mois
	Suivi si KC stable (3 examens)	1 an
30 à 40 ans	**1ère consultation après le diagnostic**	1 an
	Suivi si KC stable à 1 an	2 ans
> 40 ans	**Dès la 1ère consultation**	2 ans

Concernant la classification :

Classification du KC par le score d'Amsler-Krumeich (AK) (en prenant en compte données pentacam dans un premier temps puis en ajoutant Kmean TMS => si données discordantes prendre en compte stade le + avancé mais préciser que les données sont discordantes):

Stade I léger	Courbure excentrée (en pratique on va considérer que si le diagnostic de KC a été posé ce critère est rempli) Myopie/astigmatisme <5D (équivalent sphérique) Kmean<48D
Stade II modéré	Myopie/astigmatisme ≥5D et <8D Kmean <53D Pas d'opacité Pachymétrie min ≥400
Stade III modéré	Myopie/astigmatisme ≥8D et <10D Kmean ≥53D Pas d'opacité Pachymétrie min <400 mais ≥300
Stade IV sévère	Réfraction impossible Kmean ≥55D Opacité centrale Pachymétrie min <300

Concernant la classification stable/progression :

Intégrer les différentes études précédentes et classer le KC en stable si aucun critère en faveur ou en progression et selon quelle(s) étude(s) (la plus importante est celle de Gomes mais concrètement + difficile à coder ; en 2eme plus importante celle de Wittig-Silva)

Gomes 2015 : Changement cohérent/concordant d'au moins 2 des paramètres suivants (1D):

- Augmentation de la kératométrie antérieure ou postérieure (Kmax ou K1, K2, Kmean)
- Amincissement cornéen ou majoration de la différence de pachymétrie entre la périphérie et la pachymétrie la plus fine

Wittig-Silva 2008 : Au moins un critère parmi :

- Augmentation ≥1D du Kmax
- Augmentation ≥1D cylindre subjectif
- Augmentation ≥1D équivalent sphérique
- Diminution ≥0,1mm rayon de la lentille de contact la mieux adaptée

Dresden – Raiskup 2008 : Un critère parmi les 3 :

- Augmentation du Kmax ≥1,00D
- Sensation de BAV d'après le patient

- Besoin de plus d'une réadaptation des lentilles de contact en 2 ans

Vinciguerra 2009 : Au moins un critère parmi :

- Changement de la myopie et/ou astigmatisme ≥3D au cours des 6 derniers mois
- Modification Kmean ≥1,5D sur 3 topographies consécutives au cours des 6 derniers mois
- Diminution de la pachymétrie centrale ≥5% sur 3 topographies consécutives réalisées au cours des 6 derniers mois

O'Brart 2011 : Au moins un des deux critères suivants au cours des 18 derniers mois :

- Diminution AVSC ou AVC de plus d'une ligne
- Augmentation du cylindre objectif ou subjectif, de la kératométrie ou du Kmax≥0,75D

Hersh 2011 : Au moins un des critères suivants sur une période de 24 mois :

- Augmentation ≥1D du Kmax
- Augmentation ≥1D cylindre subjectif
- Augmentation ≥0.5D équivalent sphérique subjectif

Chatzis Hafezi 2012 : Augmentation ≥1D du Kmax sur une période de maximum 12 mois

Hashemi 2013 : Au moins un des critères suivants sur une période de 12 mois :

- Augmentation Kmax ou cylindre subjectif ou équivalent sphérique subjectif≥1D
- Perte d'au moins 2 lignes d'AVC attribuée uniquement à la progression du KC

Mazzotta 2014 :
Variation d'au moins 3 des paramètres suivants (un clinique (donc AVC/AVSC) + 2 examens paracliniques) sur une période de 4 mois pour les patients de moins de 18 ans et 6 mois pour les adultes en se basant sur valeurs kératométrie et pentacam:

- Aggravation de l'AVC/AVSC >1 ligne d'acuité visuelle
- Augmentation de la sphère ou cylindre >0,5D
- Augmentation sur topographies de l'index SAI/SI>1D
- Augmentation du Kmean >1D
- Diminution du point de pachymétrie le plus fin (sur OCT en théorie mais ici nous utiliserons le pentacam) ≥10microns

Stojanovic 2014 :
KC évolutif si durant une période de 12 mois :

- Augmentation de l'astigmatisme ou myopie (réfraction subjective) ≥1D ou augmentation Kmean ≥1,5D

Shetty 2015 :

- Augmentation du Kmax + changement équivalent sphérique ≥1D ou diminution pachymétrie min ≥5% au cours des 6 derniers mois

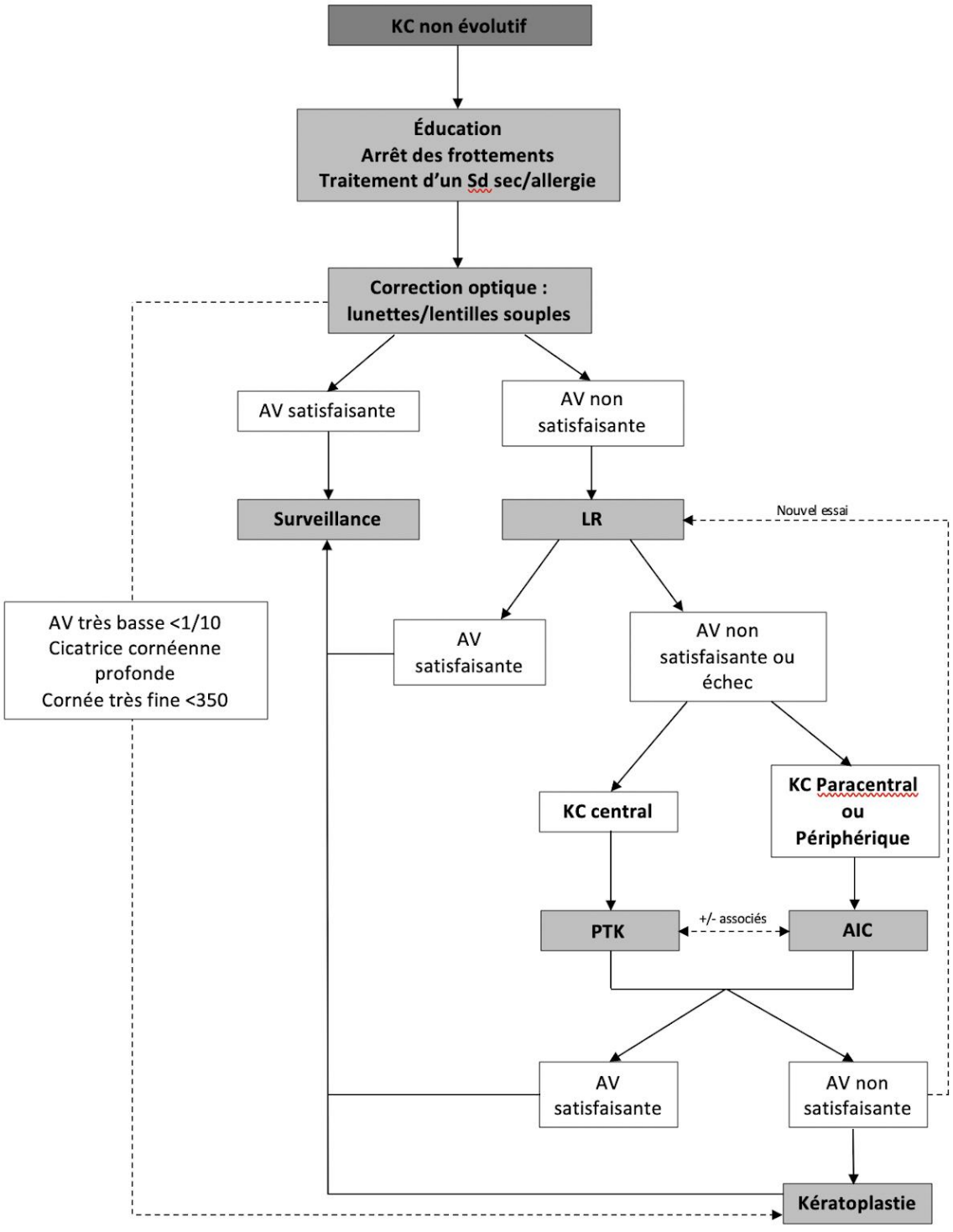
Poli 2015 : Au moins 1 critère au cours des 6 derniers mois parmi :

- Diminution de l'AVSC et/ou AVC >1 ligne Snellen,
- Diminution de l'équivalent sphérique
- Modification Kératométrie de l'apex (Kmax) >0.75D

diminution pachymétrie min >10microns

Godefrooji 2016 : Majoration ≥1D du Kmax en 6 à 12 mois

Concernant les thérapeutiques proposées :



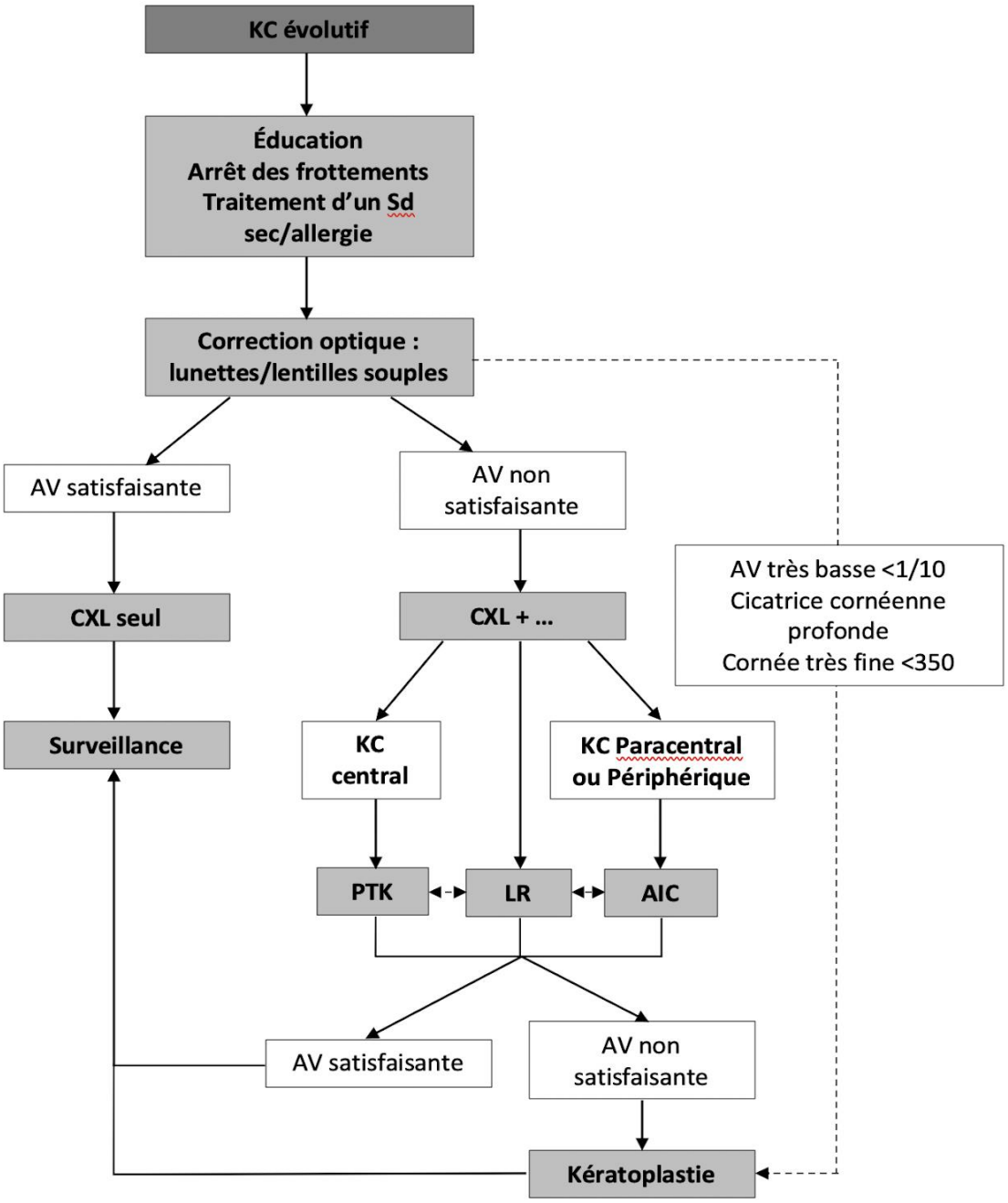
AV satisfaisante si ≥0.8
Différence KC central ou periphérique : périphérique si point le plus fin à plus de 2 mm du centre (sur image pentacam)

Contre-indications (= si donnée présente le patient ne peut pas bénéficier du traitement) du CXL (crosslinking) :

- Grossesse/Allaitement
- Antécédent d'Herpès cornéen
- Pachymétrie < 400 ⌈m
- Opacité cornéenne centrale
- Syndrome sec sévère, Kératopathie neurotrophique

Keratoplastie :
Proposer KLAP (Keratoplastie Lamellaire Antérieure Profonde) dans tous les cas sauf si :

- cicatrice stromale ou neovascularisation (données dans examen lampe à fente)
- antécédent hydrops (soit dans antécédents soit en conclusion lors suivi)
- proposer alors KT = kératoplastie transfixiante


KC évolutif
Éducation
Arrêt des frottements
Traitement d'un Sd sec/allergie
Correction optique : lunettes/lentilles souples
AV satisfaisante
AV non satisfaisante
AV très basse <1/10
Cicatrice cornéenne profonde
Cornée très fine <350
CXL seul
CXL + ...
Surveillance
KC central
KC Paracentral ou Périphérique
PTK
LR
AIC
AV satisfaisante
AV non satisfaisante
Kératoplastie

Annexe 2 : Liste des variables

Image CSV

age => DateNaiss ==> Né(e) le
Date Examen => Date ==> Examen du
consultation Date ==> Examen du
Kmean ==> Km
Pachy ==> Centre pup ou Pachy min
Kmax ==> K Max
K1 ==> K1
K2 ==> K2

Patient CSV

AVC ==> viszcr ou vsrr
Sphérique ==> obj1sphr
myopie = équivalent sphérique
Cylindrique ==> obj1cylr
Cicatrice ==> anamnese
Cornée ==> anamnèse

Listes des variables nécessaires :

Age ■
Date examen ■
Consultation ■
Myopie + Myopie 12 mois ⇒ sphérique négatif
Kmean + Kmean 12 mois + Kmean 4 mois ■
Opacité => lampes à fentes
Pachy + Pachy 4 mois ■
AVC + AVC 4 mois + AVC 6 Mois base de donnée ⇒ subjectif
sphère + sphère 4 mois + sphère 6 Mois => correction portée sur les lunettes
SAI + SAI 4 mois + SAI 6 Mois => TMS(pas récupérable pour l'instant)
Kmax + Kmax 4 mois ■
cylindre + cylindre 4 mois
sphérique + sphérique 4 mois
rayon lentille + rayon lentille 4 mois
K1 + K1 4 mois ■
K2 + K2 4 mois ■

Méthode de KC non évolutif (voir comment on fait avec ses variables)

opération
AV
type_operation

Méthode de KC évolutif (voir comment on fait avec ses variables)

AV
Cicatrice
Cornée
opération
symptômes
type_operation
type_KC
kératoplastie

type_operation pas même valeur pour les deux méthodes

Stovanovic prendre kmean et le dernier kmean

Diviser par une valeur si chiffre supérieur à 100