Rapport de projet 3A - Plateforme d’analyse et de machine learning

Réalisé par : ALDEANO Paul
Tutrice : BENMOUFFEK Dominique

Introduction :

Ce projet, proposé par Mme Benmouffek, fait suite au projet 3A de M. Damien Couillard réalisé sur l'année 2020-2021. L'objectif était de fournir un outil simple d'utilisation et adaptable à de nombreuses applications. A la demande de Mme Benmouffek, ce projet est réitéré lors de l'année 2021-2022. Les objectifs principaux énoncés en début d'année sont :

- de rendre la plateforme plus ergonomique et compréhensible pour un nouvel utilisateur
- d'améliorer les algorithmes de machine learning pour pouvoir préprocesser les données et appliquer les algorithmes à des jeux de données de plus grande taille.

Cependant, la plateforme réalisée par M. Couillard dépend de beaucoup de technologies différentes et nécessitant une maîtrise afin de comprendre comment la faire fonctionner. Mon travail de projet a donc consisté à essayer de faire fonctionner correctement la plateforme de M. Couillard, puis après avoir échoué à obtenir des résultats satisfaisants, j'ai développé une deuxième plateforme avec des dépendances techniques plus souples pour faciliter la continuation du projet.
Ce rapport sera donc divisé en deux parties :

- Prise en main de l'outil développé par M. Couillard et amélioration de l'ergonomie
- Création d'une plateforme autonome sur Python

Pour ce qui est de la première partie, mon but n'est pas de critiquer le travail de M. Couillard, que je n'ai pas réussi à m'approprier complètement, mais plutôt de faire état des difficultés que j'ai eu à prendre en main le projet et à le faire fonctionner correctement.

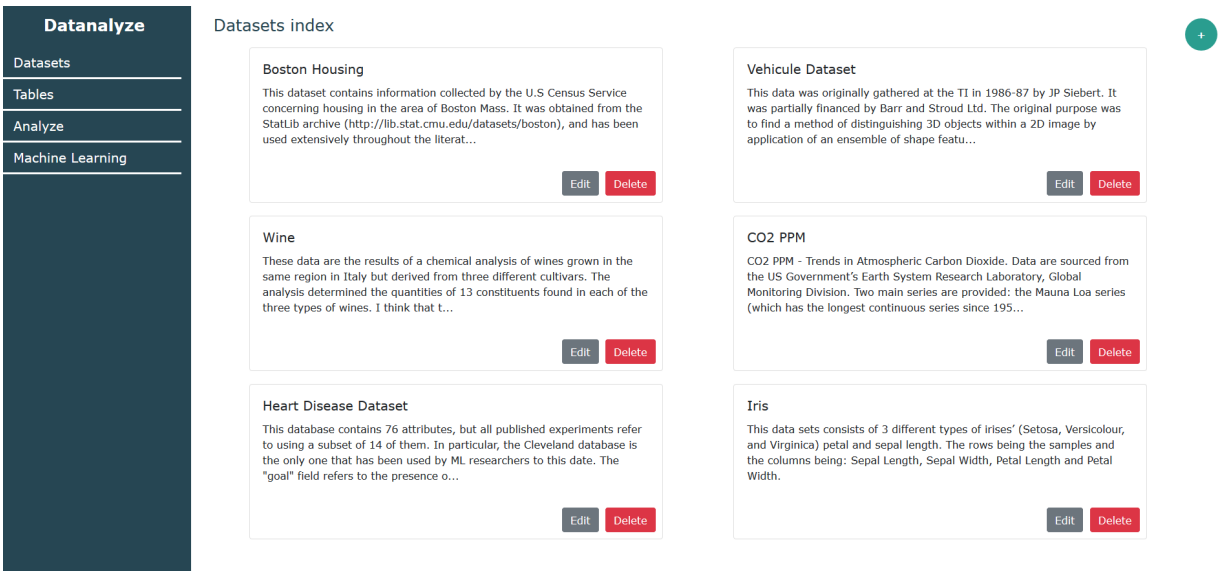
I - Prise en main de l'outil et amélioration de l'ergonomie.

I - A - Prise en main

Après avoir suivi les instructions de M. Couillard et installé les dépendances, je me suis heurté à un premier problème : toutes les dépendances (elasticsearch, kibana) démarraient correctement, mais pas la plateforme. En effet, les fichiers docker manquent d'une instruction concernant le composant react-js, en charge de la partie front de l'application. L'instruction est : "npm install react-js"

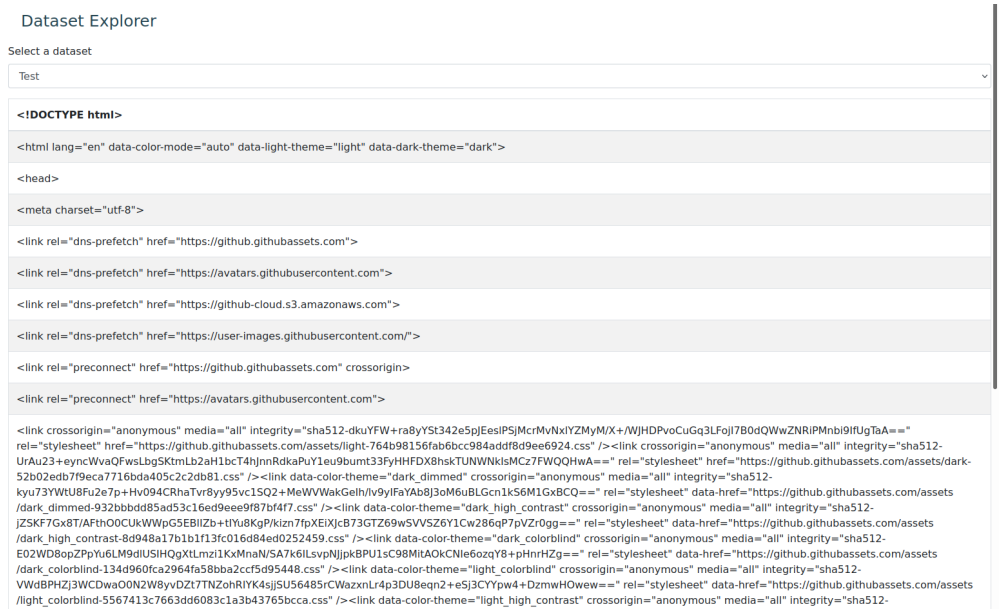
Par la suite, j'ai constaté que 2 fichiers de processus de machine learning, appelés Kmeans et SVM, manquaient. Or ces fichiers sont appelés de telle façon qu'ils sont indispensables au bon fonctionnement de la plateforme. Avec l'aide de M. Izart, qui avait également travaillé sur cette plateforme pour son projet 2A, j'ai pu recréer ces fichiers et les incorporer au code. Je les ai également mis à disposition dans le répertoire Git de mon propre projet, n'ayant pas accès à celui de M. Couillard.

A ce stade, la plateforme a pu se lancer correctement, sous cette forme :



Cependant, l'importation des datasets ne fonctionnait pas. Comprendre l'origine de ce problème a pris beaucoup de temps car c'était la version d'elasticsearch qui ne correspondait pas et empêchait le stockage des données que je rentrais dans la plateforme. En effet, elasticsearch-oss n'était plus supporté et il fallait passer à elasticsearch 7.9.1

J'ai donc essayé de rentrer un dataset sur la plateforme. L'éditeur de datasets ne fonctionne cependant qu'avec des URL de jeux de donnés, souvent déposés sur des liens Github. Le problème avec ce format est que les données sont souvent corrompues et illisibles : le téléchargement des données ou bien l'ouverture d'un fichier depuis Git avec la librairie Pandas propose un affichage illisible.



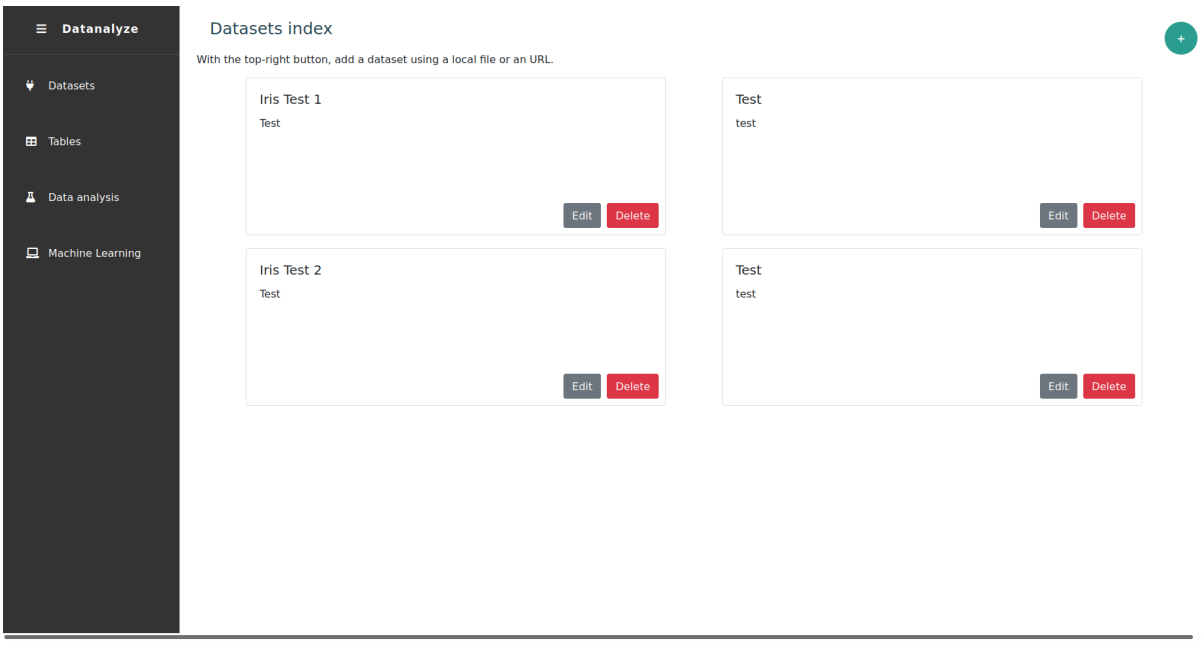
Pour remédier à ce problème, je souhaitais créer une fonction d'import locale. Mais la plateforme étant éditée sur Docker et donc lancée via un conteneur docker, elle agit comme une machine virtuelle, indépendante de la machine sur laquelle le programme est lancé. Par conséquent, les chemins d'accès de fichiers sont illisibles par le code, qui considère le conteneur docker comme seul environnement de développement.

Après ces multiples échecs, Mme Benmouffek m'a proposé de me concentrer sur le développement d'une deuxième plateforme afin de réussir à imiter le fonctionnement de celle-ci, mais en utilisant des technologies plus souples.

I - B - Ergonomie de la plateforme

L'objectif premier de mon projet étant de rendre la plateforme ergonomique et claire, j'ai tout de même réalisé un léger travail de refonte graphique. La technologie utilisée est CDB React, et m'a permis de rendre l'interface principale légèrement plus agréable visuellement parlant.

J'ai majoritairement travaillé sur une Sidebar plus élégante, avec des icônes et la capacité de s'étendre et se refermer :



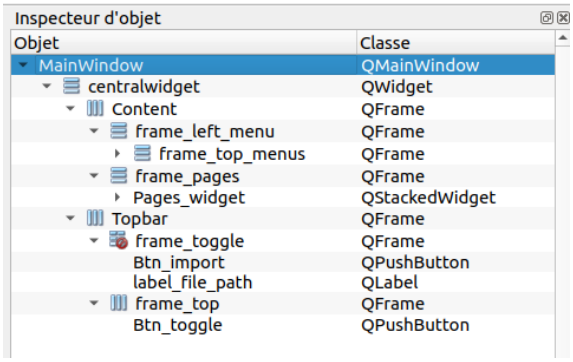
Cependant, les problèmes liés aux jeux de données ont occupé la plupart de mon temps de travail et je n'ai pas fourni d'autres améliorations.

II - Développement d'une deuxième plateforme sous python

Étant donné l'étendue des problèmes techniques auxquels j'avais fait face, j'ai souhaité réduire au maximum les dépendances technologiques afin de permettre au projet d'être renouvelable l'an prochain. Pour réaliser des algorithmes de machine learning simplement, python semble le langage le plus pratique. De plus, les algorithmes réalisés par M. Couillard étaient également en python.

II - A - Création d'une plateforme réactive

J'ai choisi d'utiliser la librairie PyQt5 et l'outil QTdesigner. Après discussion avec une autre élève du département informatique, il s'est avéré être l'outil le plus pratique pour réaliser une interface réactive en python. Dans un premier temps, je me suis familiarisé avec la librairie, en développant une interface primitive réactive. Une fois plus à l'aise, j'ai pu mettre en place les pages et les boutons qui renvoient à ces pages. L'architecture de l'application est décrite ici, avec à côté sa forme réelle :



Une Topbar permet d'importer ses datasets, peu importe la page consultée. Les pages sont décrites dans l'élément frame_pages et sont aux nombre de 4 :

- Learning : présentation de l'application et instruction générales
- Linear regression : effectuer une régression linéaire
- Polynomial regression : effectuer une régression polynômiale
- Clustering : chercher des clusters avec la méthode Kmeans

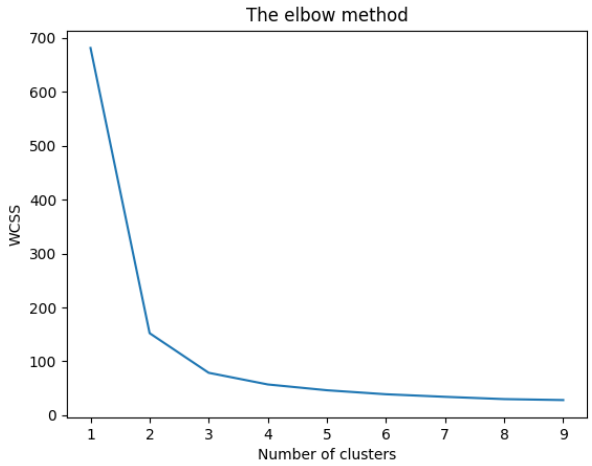
Le fonctionnement précis de ces pages est expliqué précisément dans la documentation technique jointe à ce rapport.

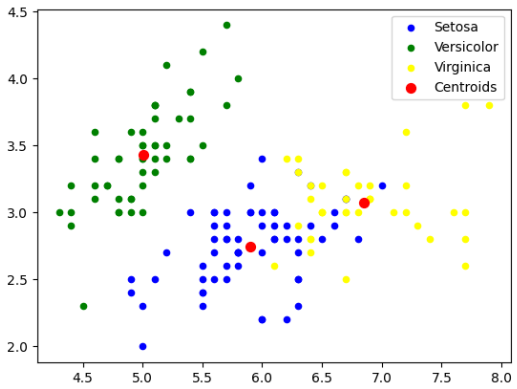
II - B - Fonctions d'analyse et machine learning.

Les fonctions implémentées sont des régressions et méthodes de clustering, créés avec l'aide de librairies python dédiées au machine learning. Tout d'abord, la régression linéaire. Cette fonction très basique mais courante permet de constater l'évolution d'un type de données par rapport à un autre. J'ai ensuite fait évoluer cette fonction pour pouvoir afficher des régressions polynômiales en en choisissant le degré. Avec ces deux outils, on peut affiner la régression pour fournir une meilleure analyse des données supervisées, et établir des prédictions (par exemple : déterminer la distance de freinage d'une voiture en fonction de la vitesse). Finalement, j'ai ajouté une méthode de clustering qui, si le jeu de données possède des classes différentes, va établir des clusters et les afficher à l'aide

de centroides en graphiques. Pour choisir le nombre de clusters à afficher, on utilise l' "elbow method" (qui consiste à simuler l'expérience de 1 à n clusters, et prendre la valeur correspondant au coude de la courbe).

En appliquant cet algorithme au bien connu dataset "iris.csv", on obtient :





III - Améliorations possibles de la plateforme :

En conclusion, le déroulement du projet n'a pas servi les objectifs établis en début de parcours. Cependant, la flexibilité de python permet d'envisager une amélioration plus aisée de la plateforme. Dans un premier temps, il serait intéressant d'adapter le code à d'autres types de fichiers (notamment xlsx), et de paramétrer des messages d'erreurs lorsque les données ne sont pas sous la forme attendue pour appliquer les algorithmes. Ensuite, ajouter un pré processing des données pour justement les rendre le plus traitables possible. Finalement, ajouter une page avec l'utilisation de réseaux de neurones, car c'est la partie la plus "attractive" du machine learning. Un algorithme similaire à celui de la Teachable Machine de Google serait une base solide, et est loin d'être irréalisable.