|  |
| --- |
| Rapport du Projet Data Engineering |
| Année universitaire : 2022-2023 |

**Encadrant :**

Mr. Ferdjaoui Amine

**Réalisé par :**

* Fella MECHOUAR
* Ryad Mohamed CHERIFI
* Walid YAICI

Table des matières

[I. Eléments clés : 3](#_Toc124118693)

[II. Introduction 3](#_Toc124118694)

[III. Présentation de la solution adoptée : 3](#_Toc124118695)

[A. Architecture de la solution : 4](#_Toc124118696)

[B. Organisation du code : 4](#_Toc124118697)

[1. Règles de bonne hygiène de code 4](#_Toc124118698)

[2. Architecture du répertoire de code 5](#_Toc124118699)

[C. Répartition et gestion des taches de l’équipe : 6](#_Toc124118700)

[1. KPIs du projet : 7](#_Toc124118701)

[IV. Modèles d’IA 7](#_Toc124118702)

[A. Prédiction du diabète 8](#_Toc124118703)

[B. Prédiction du LungCancer 8](#_Toc124118704)

[C. Prédiction de la Pneumonie 8](#_Toc124118705)

[V. Conclusion 9](#_Toc124118706)

# Eléments clés :

**Vidéo de démonstration de l’outil :** [**https://www.youtube.com/watch?v=hny629SEGrU**](https://www.youtube.com/watch?v=hny629SEGrU)

**Lien git du projet :** [https://github.com/fellamch22/DE\_project\_M2](https://github.com/fellamch22/DE_project_M2.git)

**Lien de l’Image docker disponible sur docker hub :** <https://hub.docker.com/r/chevalsumo/de_project_mcy>

# Introduction

Dans le cadre du projet Data Engineering, nous allons créer une application web qui permettra à l'utilisateur de fournir des informations médicales sur lui-même et d'obtenir des résultats prédictifs à partir de trois modèles différents :

* Le premier modèle aura pour but de prédire si le patient est atteint du diabète à l’aide de plusieurs paramètres.
* Un deuxième modèle sera utilisé pour prédire si un patient est atteint d'un cancer du poumon à partir de différents paramètres.
* Dans le troisième modèle, l’utilisateur va fournir une radiographie du thorax afin que le modèle puisse prédire la une infection respiratoire aiguë du tissu pulmonaire (Pneumonie).

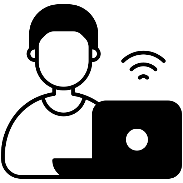
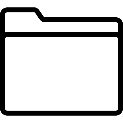
# Présentation de la solution adoptée :

Afin de répondre au mieux et d’une manière flexible aux besoins exprimés dans la problématique posée dans le projet, nous avons opté pour une solution qui peut se décrire dans les étapes suivantes :

* Création du projet sur github pour faciliter la gestion du code entre une équipe de trois personnes.
* Mettre en place des règles d’hygiène de code à respecter par les membres de l’équipe pour assurer la bonne tenance du repos git (règles détaillées dans la partie B. )
* Développements du produit sur plusieurs étapes :
  + Preprocessing des données
  + Construction et entrainement des modèles ML
  + Construction de l’interface web avec streamlit
* Déploiement du produit dans un conteneur docker

## Architecture de la solution :





Clone

App/

Localhost:8501

**Volume**





Pull image

## Organisation du code :

### Règles de bonne hygiène de code

Pour assurer la bonne maintenance du repository git , nous avons mis en place des règles d’hygiène de code comme suit :

* Utilisation de l’extension Flake8 qui permet de vérifier que l’ensemble des scripts python du projet respectes les règles PEP8 , ce qui permet d’avoir un code propre et plus lisible.
* Assurer et forcer le respect de formattage des fichiers avec Flake8 ainsi que quelques vérifications à l’aide de pre-commit qui empêche les commits de passer si l’une des règles mise n’est pas respectée.
* Assurer la création d’une issue pour chaque tache attribuée à un membre de l’équipe
* Création d’une branche par issue ouverte
* Toute création de nouvelle branche se fait
* Il est strictement interdit de pousser les changements d’une branche directement sur les branches dev et masters
* Création d’un pull request pour intégrer les changements de la nouvelle branche sur dev
* Suppression des branches après validation de la pull request.

### Architecture du répertoire de code

|  |  |  |
| --- | --- | --- |
| Dossier / Fichier | Description | |
| 📁 /app | Dossier contenant l’ensemble des scripts de toute l’application ( preprocessing , création de modèles ML , interface streamlit) |
| 📁 /notebook | Dossier contenant un ensemble de notebooks qui indique comment s’en servir de chacune des classes de l’application |
| 📁 /scripts | Ensemble de scripts bash qui facilitent l’utilisation de l’application et l’initialisation de l’environnement adéquate au répertoire git lors du premier clone |
| .dockerignore | Fichier permettant d’ignorer une liste de dossiers/ fichiers lors de la création de l’image docker |
| .gitignore | Fichier permettant d’ignorer le traçage de certains dossiers/fichiers par github |
| .pre-commit-config.yaml | Fichier permettant de définir l’ensemble des règles à vérifier lors d’un commit fait par un développeur et empêcher les commits de passer si l’une de ces règles n’a pas été respectée |
| Docker-compose.yaml | Fichier permettant de créer l’image docker souhaité à partir de l’image publiée sur Dockehub |
| Dockerfile | Fichier permettant de créer l’image docker |
| Makefile | Fichier permettant de raccourcir l’utilisation de l’application avec quelques petites commandes ex : «**$** make app » pour lancer l’application à la place d’écrire « **$** streamlit run app/interface/Home.py » |
| Requirements.txt | Fichier qui spécifie l’ensemble des packages à installer avec leurs versions respectives |
| Setup.cfg | Configurations liées à l’extension Flake8 |

Contenu du dossier de l’application «📁 app » :

|  |  |
| --- | --- |
| Dossier / Fichier | Description |
| 📁 configuration | Dossier contenant l’ensemble de configurations utilisées pour l’ensemble des scripts python de l’application |
| 📁 data | Dossier contenant l’ensemble des fichiers de données utilisées pour l’entrainement des modèles de machine Learning |
| 📁 interface | Dossier composé de l’ensemble des scripts de l’interface streamlit |
| 📁 models | Dossier contenant l’ensemble des scripts d’entrainements des modèles ML ainsi que les sauvegardes des modèles entrainées |
| 📁 preprocessing | Dossier contenant l’ensemble des scripts de preprocessing |

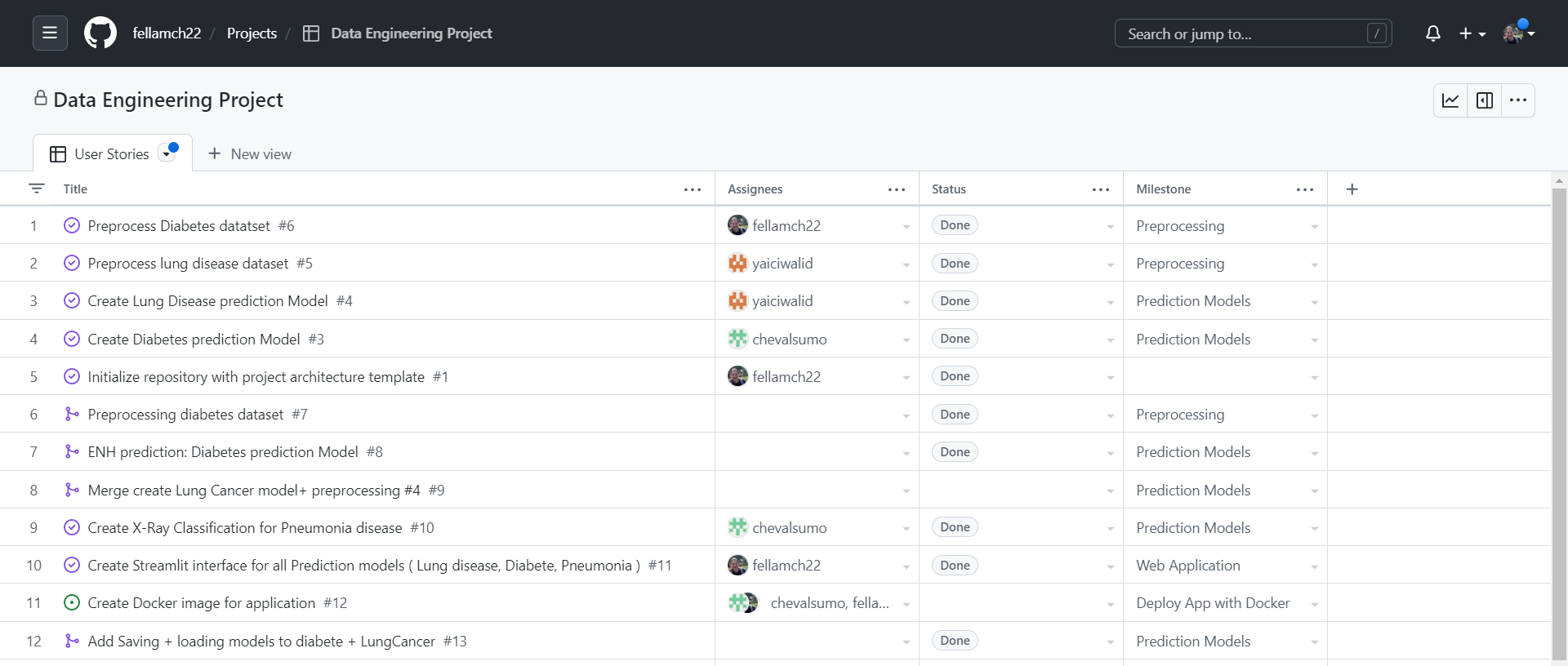
## Répartition et gestion des taches de l’équipe :

En vue de la petite taille du projet , la division des taches entre les membres de l’équipe n’était pas une tache simple.

Pour une meilleure gestion des taches entre les membres de l’équipe nous avons opté pour l’utilisation des outils de gestion de projet disponibles sur github.

Une image contenant texte

Description générée automatiquementGitHub Issues est une fonctionnalité de GitHub qui permet aux utilisateurs de suivre et de gérer les tâches et les bugs dans leur projet. Chaque tâche ou bug est représenté par une "issue" qui peut être assignée à un utilisateur, ajoutée à une liste de tâches à faire ou à une liste de bugs à résoudre. Les issues peuvent être discutées et commentées par les membres du projet, ce qui permet une communication transparente et une meilleure collaboration.

GitHub Projects est un autre outil de gestion de projet qui permet aux utilisateurs de créer des tableaux Kanban pour organiser leur travail. Les utilisateurs peuvent créer des colonnes pour chaque étape du processus de développement, comme "à faire", "en cours" et "terminé", et déplacer les issues entre ces colonnes au fur et à mesure que le travail avance. Cela permet une visualisation claire de l'avancement du projet et de l'état de chaque tâche.

En utilisant conjointement GitHub Issues et GitHub Projects, nous avons pu facilement suivre et gérer les tâches du projet, ce qui nous a permis de rester organisés et de travailler de manière efficace.

### KPIs du projet :

* + Trois développeurs
  + 8 issues créés
  + 7 pulls requests validées
  + 100 commits
  + 900 lignes de codes

# Modèles d’IA

Dans cette section, on va parler brièvement des modèles d’intelligence artificielle qu’on a utilisés pour chaque sous-problématique de notre travail.

## Prédiction du diabète

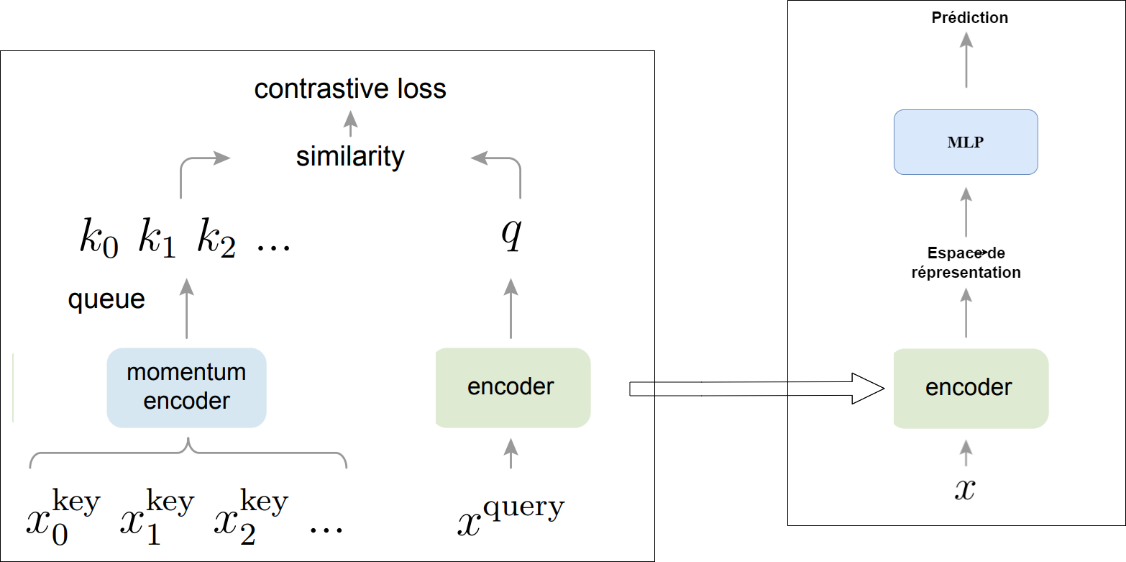
Pour cette problématique, la tâche à réaliser est relativement simple par rapport au Dataset utilisé. Dans le preprocessing de ces données, on a dû faire un encodage des colonnes données catégorielles avec la suppression des colonnes inutile, la normalisation des données n’a pas donnés des bons résultats, donc on a préféré ne pas l’utiliser. Pour le modèle de prédiction, on a abouté à un modèle de DeepLearning séquentiel sur 4 couches linéaires développé et entrainé à l’aide de Keras tensorflow, le modèle entrainé final était sauvegardé dans le dossier «models«.

## Prédiction du LungCancer

Dans la tâche du preprocessing, on s’est rendu compte qu’il y avait beaucoup de doublons dans nos données, l’élimination des doublons a réduit considérablement sa taille. On a d’abord exploré l’utilisation des modèles de DeepLearning séquentiels sur quelques couches avec ou sans l’élimination des doublons, mais le meilleur résultat obtenu sur l’ensemble des données (Train/test) a été obtenu avec un modèle de Linear Discriminant Analysis (LDA) qu’on a gardé et sauvegardé comme modèle final.

## Prédiction de la Pneumonie

Dans cette troisième et dernière problématique, on a voulu explorer une approche de pré-entraînement pour avoir un espace de représentation comme une base à notre modèle de classification. Pour cela on a voulu essayer l’approche proposée par Facebook AI Research (FAIR) nommée Momentum Contrast for Unsupervised Visual Representation Learning (MOCO). L’idée de cette approche est de construire un espace de représentation en utilisant des paires d'images similaires et dissimilaires. Pour ce faire, moco utilise un mécanisme de contraste de momentum, qui permet de mettre en évidence les similarités et les différences entre les images de la paire. Cela permet de créer un espace de représentation dans lequel les images similaires sont proches les unes des autres tandis que les images dissimilaires sont éloignées. Une fois ce modèle entraîné, on extrait l’encodeur, on gèle ces paramètres pour garder l’espace de représentation et on ajoute un MLP pour construire notre classifieur d’image X-Ray. La figure suivante, résume ce processus.



Pour l’architecture de l’encodeur utilisé dans MOCO, on a essayé les architectures connus dans le traitement d’images (ResNet18 et ResNet50) prés entrainés sur imageNet. On a obtenu les meilleurs résultats en termes de constrative loss en utilisant un ResNet18 suivi par un MLP de deux couches. Pour l’entraînement de MOCO, on a utilisé la plateforme Kaggle qui nous fournit deux GPU pour un entrainement parallèle qui est nécessaire pour une bonne implémentation de MOCO. Le notebook utilisé pour l’entraînement est présent dans le dossier notebook (Pneumonia\_pretrain\_distributed\_moco\_kaggle.ipynb). Une fois ce modèle entraîné, on a extrait l’encodeur pour construire notre classifieur final qui a été entraîné à son tour en utilisant la plateforme kaggle avec le notebook (Pneumonia\_classifier\_train\_kaggle.ipynb) avec une accuracy de 97% sur les données d’entraînement et 86% sur le jeu de test.

Le modèle final de chaque problématique a été sauvegardé dans le dossier (models/ trained\_models) et utilisé pour l’application web Streamlit.

# Conclusion

A travers ce projet, on a abouti à la réalisation d’une application web de prédiction de maladies développée avec Streamlit et combinant plusieurs modèles d'IA. Ce projet a été géré sur Github, ce qui a permis de suivre l'avancement du développement et de gérer les contributions de différents membres de l'équipe.

Les bonnes pratiques de développement sur Github ont été mises en œuvre avec l'utilisation de pull requests pour fusionner les changements apportés au code source et de issues pour suivre et résoudre les problèmes rencontrés au cours du développement.

Le déploiement de l'application a été effectué dans un conteneur Docker, ce qui a permis de s'assurer que l'application fonctionne de manière fiable et reproductible sur différents environnements. Un volume Docker a également été utilisé pour stocker l'application, ce qui a permis de la rendre disponible même lorsque le conteneur est arrêté ou redémarré.

En résumé, l'utilisation de Github et de Docker dans ce projet a permis de suivre et de gérer efficacement le développement de l'application, ainsi que de la déployer de manière fiable et reproductible.

