Projet Final

Début 9h, retour 17h30

(Aucun retard ne sera permis)

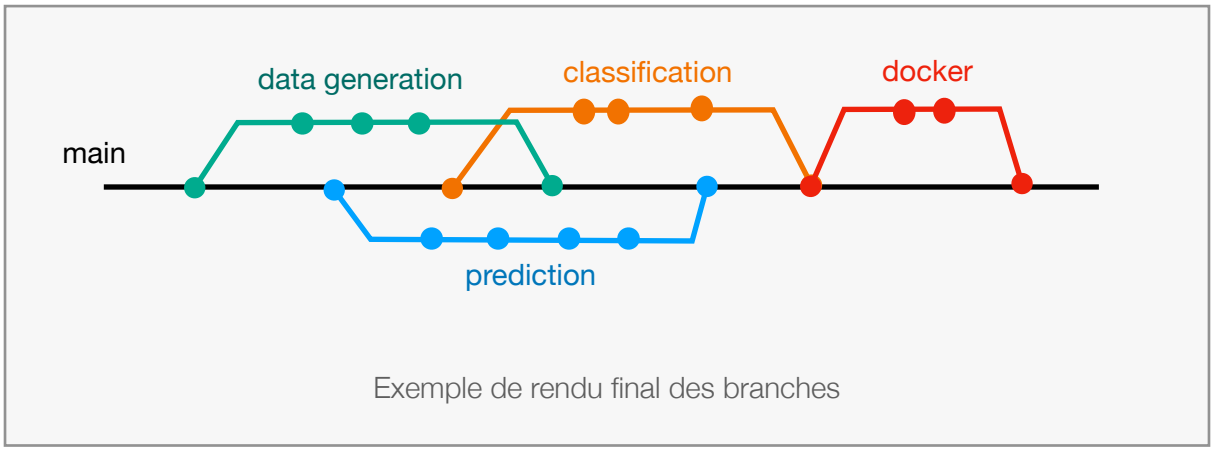
Objectif

- Simuler un projet d'entreprise réel où plusieurs développeurs travaillent sur le même projet et chacun développe une fonctionnalité dans une branche différente puis fusionne sa version finale avec la branche principale (main).
- Créer ensuite un docker qui se base sur le repository Github (branche main) en rajoutant le Dockerfile.
- Rajouter un docker-compose pour faciliter la mise le paramétrage et la mise à jour du docker.

Consignes

La première partie du projet concerne le travail collaboratif pour remplir un fichier **train_classifier.py** permettant de générer des données synthétiques et d'entrainer un modèle de classification (exemple : Random forest, SVM, GradientBoosting), et de créer ensuite un fichier **predict_classification.py** pour faire la prédiction sur des données de validation.

Chaque étudiant crée une branche pour développer une fonctionnalité : génération et analyse des données, classification et évaluation des performances, prédiction sur les données de test.



Exemple de rendu final des branches

I. Remplir le fichier *train_classifier.py*

1. Générer des données synthétiques à l'aide de la fonction make_classification de scikit-learn (voir la documentation).
2. Analyser et visualiser (à l'aide d'une méthode de réduction de dimensions) les données générées sur un jupyter notebook, faire varier les paramètres suivants : *n_informative, n_classes, n_clusters_per_class, class_sep*.
3. Séparer les données en 3 : 70% entrainement, 20% test, et 10% validation. Entrainer un modèle de classification sur les données d'entrainement, et évaluer les performances du modèle sur les données de test.
4. Une fois le modèle entrainé, exporter le modèle ainsi que les données de validation. Ignorer le modèle exporté ainsi que les données du repository github.

II. Remplir le fichier *predict_classification.py*

Importer le modèle entrainé et les données de validation, faire la prédiction, et évaluer les performances du modèle.

III. Préparation d'un docker

1. Une fois les deux fichiers testés en local, préparer un Dockerfile permettant de les executer sur un conteneur, càd :
    1. Générer des données et lancer l'entrainement d'un modèle, afficher les résultats de classification, et exporter le modèle et les données de validation à la fin (sur le conteneur).
    2. Réaliser la prédiction sur les données de validation et **afficher les résultats à la fin**.
2. Tester le docker en local avec docker build et docker run.
3. Préparer un docker-compose pour sauvegarder la configuration (ports, volumes, etc.).
4. Envoyer le modèle sur un repository dans docker-hub.

Rendu :

- Le lien vers Github et dockerHub dans le fichier : lien groupes.

Important : Ne plus faire de push après avoir rendu le projet.

Évaluation :

Le projet sera évalué en fonction de :

- Fonctionnement de la solution et la capacité à interpréter les résultats.
- Integration des fonctionnalités et pratiques vues en cours.
- Répondre aux besoins en travaillant en collaboration, et faciliter l'intégration de la solution finale.

Bonus :

- Améliorer la qualité du rendu final, par exemple : permettre à l'utilisateur de choisir le(s) modèle(s) qu'il souhaite tester, ou bien executer le modèle dans un backend. Toute autre proposition sera appréciée.
- Rédaction d'une bonne documentation (*README*) et des commentaires dans le code (notamment pour interpreter les résultats).
- Réaliser des cross validations.
- Optimiser les hyperparamètres à l'aide de grid search.
- Mettre à jour un des modèles après l'avoir dockerisé (à travers un volume).
- Développer le modèle en local avec docker et monter un volume sur le projet, ça vous permet de remplir le Dockerfile au fur et à mesure et de travailler ensemble sur la même version de python et des librairies.

Bon courage !