Mission - Déployez votre modèle de Machine Learning

Comment allez-vous procéder ?

Cette mission suit un scénario de projet professionnel.

Vous pouvez suivre les étapes pour vous aider à réaliser vos livrables.

Avant de démarrer, nous vous conseillons de :

- lire toute la mission et ses documents liés ;
- prendre des notes sur ce que vous avez compris ;
- consulter les étapes pour vous guider ;
- préparer une liste de questions pour votre première session de mentorat.

Prêt à mener la mission ?

Dans ce projet, votre mission sera de déployer un modèle de machine learning à l'aide d'outils modernes. Vous avez le choix de déployer soit le modèle du projet 3 (Anticipez les besoins en consommations de bâtiment) ou celui du projet 4 (Classifiez automatiquement des informations).

Vous êtes freelance spécialisé en machine learning et vous venez de recevoir une demande de la part de votre client Futurisys, une entreprise innovante qui souhaite rendre ses modèles de machine learning opérationnels et accessibles via une API performante. Vous êtes chargé de déployer un modèle de machine learning en production.

Le directeur technique de Futurisys, Aurélien, vous formule une demande impliquant de :

- créer une API avec FastAPI (ou équivalent) pour exposer le modèle ;
- écrire des tests unitaires avec Pytest pour garantir sa fiabilité ;
- et gérer la version du code avec Git pour une collaboration fluide.

L'objectif ? Rendre le modèle utilisable en production tout en respectant les meilleures pratiques de l'ingénierie logicielle. À la fin du projet, vous aurez un Proof of Concept (POC) fonctionnel dont vous pourrez être fier !

Tout en réfléchissant, vous dressez la liste des livrables que vous aurez à construire et présenter :

- **Un dépôt Git structuré contenant :**
  - L'ensemble du code source.
  - Un requirements.txt (ou équivalent).
  - Un historique de commits clair, avec des branches dédiées aux fonctionnalités et l'utilisation de tags pour la gestion des versions.
  - Un README complet présentant le projet, notamment les instructions d'installation, d'utilisation, de déploiement, d'authentification et de sécurisation.
- **Une API fonctionnelle et déployée** développée avec FastAPI (ou équivalent) exposant le modèle de machine learning accompagnée d'une documentation intégrée (par exemple via Swagger/OpenAPI) pour décrire les endpoints, les schémas de données et les exemples d'appels.
- **Des scripts de tests unitaires et fonctionnels associés à :**
  - Un ensemble de tests écrits en Pytest couvrant les cas critiques et les scénarios d'erreur.
  - Un rapport de couverture de tests (par exemple via pytest-cov) afin de démontrer la robustesse du code.
- **Une base de données PostgreSQL fonctionnelle:**
  - Un script SQL (.sql) ou Python (create_db.py) pour la création de la base de données et des tables.
  - Un modèle de données/de la documentation expliquant la structure des tables (ex: schéma UML).
  - Des exemples d'entrées en base (SQL ou CSV contenant des inputs et outputs du modèle de ML).
  - Des scripts pour interroger les données et interagir avec le modèle ML.
- **Une configuration du pipeline CI/CD** capable de gérer les environnements (dev test, prod) et intégrer la gestion des secrets.
  - Un fichier YAML (par exemple pour GitHub Actions) qui automatise les tests et le déploiement

> Vous planifiez dans votre agenda de présenter l'ensemble de votre projet à Aurélien, à l'aide d'un support de présentation, et vous vous mettez au travail.

Étapes

Cette mission est entièrement guidée.

> Suivez les étapes ci-dessous et consultez la fiche d'autoévaluation dans la dernière étape avant d'envoyer vos livrables afin d'être sûr de ne rien avoir oublié.

Étapes

Étape 1 - Mettez en place un système de gestion de version et collaboration

Cette étape initiale vise à mettre en place un système de gestion de versions robuste qui servira de fondation à tout le projet. L'objectif est de créer un environnement de travail structuré, permettant une collaboration fluide, un suivi précis des modifications et une traçabilité complète du développement.

Prérequis

- Connaissances de base de Git.
- Compréhension des workflows de collaboration.
- Notions de gestion de versions.
- Création d'un compte sur une plateforme de versionnage (GitHub, GitLab).

Résultats attendus

- Un dépôt Git bien structuré :
  - Un requirements.txt ou équivalent.
  - Un historique de commits clair et significatif.
  - Des branches pour les différentes fonctionnalités.
- Une documentation du projet sous forme de README complet présentant le projet, les instructions d’installation.

Recommandations

- Utiliser des commits descriptifs.
- Créer une structure de projet claire (en ne mettant pas tout à la racine par exemple...).
- Rédiger un README complet.
- Gérer les versions avec des tags.
- Définir les conventions de nommage des branches.

Point de vigilance

- Gestion des conflits : utilisez un outil de résolution de conflits intégré à votre environnement de développement.

Outils

- Git

- GitHub/GitLab
- Éditeur de code avec intégration Git

Ressources

- Documentation Git officielle
- Guide de bonnes pratiques de versionnage

> À ce stade de votre projet, prenez un moment pour vérifier votre travail et faire le point si nécessaire avec votre mentor pour vous assurer que vous êtes dans la bonne direction.

Étape 2 - Configurez la CI/CD

Cette étape se concentre sur la mise en place d'une infrastructure d'intégration continue et de déploiement continu sur la plateforme Hugging Face Spaces. L'objectif est de créer un pipeline automatisé qui garantira la qualité du code, facilitera les tests et permettra un déploiement rapide et fiable du modèle de machine learning.

Cette étape se concentre sur la mise en place d'une infrastructure d'intégration continue et de déploiement continu sur une plateforme cloud (telle que Hugging Face Spaces par exemple). L'objectif est de créer un pipeline automatisé qui garantira la qualité du code, facilitera les tests et permettra un déploiement rapide et fiable du modèle de machine learning.

Prérequis

- Compréhension des principes d'intégration continue.
- Compréhension des différents environnements (développement, test, production).

Résultats attendus

- Un pipeline CI/CD automatisé (autant que possible avec la solution choisie).
- Un fichier yaml configurant au moins une GitHub Action.
- Des tests automatiques à chaque commit ou push vers une branche.
- Une validation avant fusion de branches.
- La gestion des différents environnements.

Recommandations

- Définir les étapes du pipeline avant de les implémenter sur GitHub.
- Écrire vos standards de code et d'expérimentation ML dans un README dédié.

Points de vigilance

- Être attentif au temps d'exécution des pipelines (ex : plus de 10 min pourrait vous amener à vous interroger).
- Gérer les secrets.

Outils

- GitHub Actions (ou équivalent) pour effectuer automatiquement la partie test du pipeline CI/CD.
- Hugging Face Spaces ou équivalent.

Ressources

- Documentation GitHub Actions
- Démarrer avec Hugging Face Spaces
- cours Mettez en place l'intégration et la livraison continues avec la démarche devops

Étape 3 - Développement de l'API

Vous allez créer une API avec FastAPI (ou Gradio) pour exposer le modèle de machine learning, en utilisant Pydantic pour valider les données entrantes et garantir la qualité des prédictions. Vous configurerez les endpoints nécessaires pour retourner les prédictions du modèle et testerez chaque endpoint individuellement pour assurer leur bon fonctionnement.

Vous allez créer une API avec **FastAPI** (ou équivalent) pour exposer le modèle de machine learning, en utilisant **Pydantic** pour valider les données entrantes et garantir la qualité des prédictions. Vous configurerez les endpoints nécessaires pour retourner les prédictions du modèle et testerez chaque endpoint individuellement pour vous assurer de leur bon fonctionnement.

Prérequis

- Avoir une compréhension claire des modèles de machine learning développés dans les projets précédents de ce parcours.
- Avoir une expérience de base avec Python et les API REST.

Résultats attendus

- Une API fonctionnelle exposant le modèle de machine learning.
- Des endpoints bien documentés et testés.
- Une validation robuste des données d'entrée.

Recommandations

- Commencer par définir les endpoints nécessaires pour exposer le modèle.
- Utiliser Pydantic pour valider les données entrantes et garantir la qualité des prédictions.
- Tester chaque endpoint individuellement avant de passer à la suite.

Points de vigilance

- Valider la bonne conformité des données entrantes avec les attentes du modèle.
- Gérer correctement les erreurs de validation des données.

Outils

- FastAPI (ou équivalent)
- Pydantic
- Environnement Python configuré

Ressources

- Documentation FastAPI
- Documentation Gradio
- Documentation Pydantic
- Principes et bonnes pratiques de conception d'API

> À ce stade de votre projet, prenez un moment pour vérifier votre travail et faire le point si nécessaire avec votre mentor pour vous assurer que vous êtes dans la bonne direction.

Étape 4 - Insérez le dataset et gérez-le via PostgreSQL

Vous commencerez par créer une base de données (BDD) **PostgreSQL** via un script SQL (ou Python) afin de pouvoir insérer votre dataset complet dans cette DB.

> Pour des raisons de simplicité, l'interaction avec la DB pourra se

faire entièrement et uniquement en local.

Toutes les interactions avec le modèle de machine learning devront passer par la base de données : les inputs envoyés au modèle et les outputs générés seront enregistrés dans des tables dédiées.

Cette approche permet d'assurer une traçabilité complète des échanges et d'offrir une interface robuste entre le modèle et la gestion des données.

Prérequis

- Connaissance de base de PostgreSQL et des concepts relationnels.
- Familiarité avec un ORM (exemple : SQLAlchemy qui est présenté dans les ressources de cette étape) pour faciliter la gestion des opérations sur la base.
- Compréhension des interactions entre une API et une base de données.

Résultats attendus

- Un schéma de la BDD (UML par exemple).
- Un script SQL (.sql) ou script Python (create_db.py) pour la création de la base de données PostgreSQL et des tables.
- Un dataset inséré et correctement structuré dans une base de données PostgreSQL.
- L'enregistrement systématique des inputs et outputs du modèle dans des tables dédiées (via connecteur python).
- Une traçabilité complète des échanges entre l'API et la base de données.

Recommandations

- Utiliser un ORM tel que SQLAlchemy pour simplifier la gestion des données.
- Créer le schéma de la base dans le but de gérer efficacement le volume des données traitées.
- S'assurer que toutes les interactions avec le modèle passent obligatoirement par la base de données.
- Intégrer cette étape dans la documentation du projet pour expliquer les choix techniques et fournir des exemples d'interactions.

Points de vigilance

- Veiller à la sécurité des données et à la gestion fine des accès à la base.
- Garantir la cohérence entre les données enregistrées et les opérations du modèle.

Outils

- PostgreSQL
- SQLAlchemy (ou autre ORM adapté)

Ressources

- Documentation officielle de PostgreSQL
- Tutoriels et guides sur SQLAlchemy
- Alternative: Psycopg 3

> À ce stade de votre projet, prenez un moment pour vérifier votre travail et faire le point si nécessaire avec votre mentor pour vous assurer que vous êtes dans la bonne direction.

Étape 5 - Développez des tests unitaires et fonctionnels

Cette étape vise à garantir la fiabilité et la robustesse du modèle de machine learning en développant une suite complète de tests, à la fois unitaires et fonctionnels.

- Les tests unitaires se concentreront sur la validation des composants individuels.
- Les tests fonctionnels évalueront le modèle dans son intégralité, en le soumettant à des cas d'usage réels et en vérifiant sa capacité à produire les résultats attendus selon les spécifications.

Prérequis

- Connaissance des principes de tests unitaires.
- Compréhension approfondie du modèle de machine learning.
- Familiarité avec Pytest.
- Disposer de jeux de données représentatifs.

Résultats attendus

- Scripts de test unitaires et fonctionnels
- Suite de tests unitaires couvrant tous les cas critiques et les scénarios d’erreur.
- Rapport de couverture de tests (via pytest-cov par exemple) afin de démontrer la robustesse du code et de valider de la fiabilité du modèle.
- Identification et correction des points faibles.

Recommandations

- Utiliser le jeu de données qui vous a été donné dans le P3 ou P4 (selon votre choix) pour réfléchir à la pertinence de vos tests unitaires.
- Créer des jeux de données de test variés.
- Tester chaque fonction et endpoint individuellement.
- Vérifier les performances du modèle.
- Tester les cas limites et les erreurs.

Points de vigilance

- S'assurer :
- de la complétude de la couverture des scénarios de test.
- que les résultats doivent être reproductibles.
- que les rapports de tests sont bien tous présents.

Outils

- Pytest
- Pytest-cov (couverture de tests)
- Pydantic

Ressources

- Documentation pytest
- Documentation sur la couverture de tests

> À ce stade de votre projet, prenez un moment pour vérifier votre travail et faire le point si nécessaire avec votre mentor pour vous assurer que vous êtes dans la bonne direction.

Étape 6 - Documentez le modèle de machine learning

La dernière étape se concentre sur la création d'une documentation complète et accessible qui permettra aux utilisateurs et aux développeurs de comprendre, utiliser et maintenir le modèle de Machine Learning et son API.

Prérequis

- Compétences rédactionnelles techniques.
- Compréhension approfondie du projet.

Résultats attendus

- Une documentation de l'API.
- Une documentation technique du modèle, de ses performances et de sa maintenance.
- Un README informatif sur le repo git et son déploiement.

> Consultez le template README fourni dans les ressources de cette étape.

Recommandations

- Inclure des exemples d'utilisation.
- Documenter l'architecture.
- Justifier les choix techniques.
- Fournir des instructions d'installation et de configuration.
- Identifier un protocole de mise à jour régulière.

Points de vigilance

Outils

- Swagger/OpenAPI (intégré dans FastAPI)
- Markdown (Readme)
- MkDocs (documentation)
- Sphinx (documentation Python)

Ressources

- Template de README de qualité
- Documentation de MkDocs
- Documentation de Sphinx

Vérifiez votre travail et faites le point avec votre mentor

Avez-vous une suggestion pour nous ?