TRAORE Makam

Projet Titanic Survival Prediction

Rapport de projet – BUT VCOD 2025-2026 – IUT Paris Cité

Février 2026

Étape 1 – Refactorisation du code

Le point de départ du projet était un notebook Jupyter monolithique issu du tutoriel Kaggle Titanic. L'objectif de cette étape était de transformer ce code en une architecture modulaire et professionnelle. J'ai découpé le notebook en cinq modules Python distincts : config.py pour centraliser tous les paramètres, data_preprocessing.py pour le chargement et le nettoyage des données, model_training.py pour l'entraînement du Random Forest, model_evaluation.py pour les prédictions et l'export, et main.py qui orchestre le pipeline complet. Chaque module respecte les bonnes pratiques PEP 8, avec des type hints, des docstrings détaillées et une gestion des erreurs. Le code a été vérifié avec flake8 (linting), formaté avec black et les imports organisés avec isort.

Étape 2 – Tests unitaires

J'ai mis en place une suite complète de tests unitaires avec pytest pour valider chaque module du projet. Au total, 40 tests couvrent l'ensemble des fonctions : chargement des données, prétraitement, entraînement du modèle, génération des prédictions et export des résultats. Les tests utilisent des fixtures pytest pour créer des données de test isolées et des fichiers temporaires pour vérifier l'export CSV. La couverture de code atteint 100 % sur les modules fonctionnels, mesurée avec pytest-cov. Un fichier conftest.py configure automatiquement le PYTHONPATH pour que les imports fonctionnent correctement.

Étape 3 – Docker et conteneurisation

Pour garantir la reproductibilité de l'environnement, j'ai conteneurisé le projet avec Docker. Le Dockerfile utilise l'image Python 3.9-slim pour rester léger, installe les dépendances via requirements.txt et copie le code source. Un fichier docker-compose.yml permet de lancer le conteneur avec un simple docker-compose up --build, en montant un volume pour récupérer les résultats dans le dossier output/. L'image est publiée automatiquement sur Docker Hub.

Étape 4 – Git et GitHub

J'ai mis en place un workflow Git suivant une stratégie Git Flow simplifiée avec deux branches principales : master (code stable) et develop (développement). Chaque fonctionnalité est développée dans une feature branch (ex : feature/refactoring-notebook) puis mergée dans develop via des commits explicites. Le dépôt GitHub héberge l'ensemble du projet avec un .gitignore adapté au Data Science Python.

Étape 5 – CI/CD avec GitHub Actions

Deux pipelines CI/CD sont configurés avec GitHub Actions. Le premier (python-ci.yml) s'exécute à chaque push request sur master et develop : il installe les dépendances, lance le linting avec flake8, vérifie le formatage avec black, puis exécute les tests avec pytest et génère un rapport de couverture. Le second pipeline (docker-publish.yml) construit et publie automatiquement l'image Docker sur Docker Hub à chaque push sur master.

Liens du projet

Dépôt GitHub : https://github.com/Makamtrr/Dvt_Logiciel

Docker Hub : https://hub.docker.com/r/makamtrr/titanic-survival-prediction/tags

Note sur la méthodologie

En toute transparence, je me suis fait aider par l'assistant IA Claude (Anthropic) tout au long de ce projet, que ce soit pour la structuration du code, la rédaction des tests ou la mise en place de Docker et du CI/CD. Cependant, je tiens à préciser que je comprends l'ensemble de ce que j'ai réalisé : chaque module, chaque test, chaque configuration Docker et chaque pipeline CI/CD. L'utilisation de Claude m'a permis d'accélérer le développement et d'apprendre les bonnes pratiques, mais le travail de compréhension et d'intégration reste le mien. Je travaille ainsi en entreprise.