CAHIER DES CHARGES

HEALTHINSIGHT WEB

1. Présentation générale

1.1 Contexte

HealthInsight Web est une application web pédagogique qui démontre l'intégration d'un pipeline de Machine Learning dans une application web moderne. Le système prédit un risque de santé à partir de facteurs de risque individuels et permet d'explorer les statistiques du dataset utilisé pour l'entraînement du modèle.

1.2 Objectifs

- Fournir une application web permettant à un utilisateur de saisir ses informations de santé et d'obtenir un score de risque.
- Mettre en place un pipeline complet : ETL des données, entraînement du modèle, exposition via API, consommation par une interface web.
- Offrir un module d'exploration des données (statistiques et visualisations).
- Documenter et afficher les performances du modèle (accuracy, precision, recall, F1-score, ROC-AUC).
- Valoriser l'utilisation de technologies modernes : FastAPI, scikit-learn, Next.js, React, TypeScript.

2. Périmètre fonctionnel

2.1 Acteurs

- Utilisateur : toute personne utilisant l'application pour obtenir une estimation de risque et explorer les données.
- Équipe projet : responsables du développement, du déploiement et de la maintenance.

2.2 Fonctionnalités principales côté utilisateur

F1 – Page d'accueil

- Afficher le nom de l'application et une description courte.
- Afficher un indicateur d'état du système (ex. API opérationnelle, modèle chargé).
- Proposer des boutons d'accès rapide : Prédiction de Risque, Exploration des Données.
- Présenter une section "À propos du projet" (PFE, technologies, avertissement médical).

F2 – Formulaire de prédiction

- Permettre la saisie des informations suivantes :
- Âge (entier)
- Sexe (Masculin / Féminin)
- IMC (nombre réel)
- Fumeur (Oui / Non)
- Niveau d'activité physique (Faible / Modéré / Élevé)
- Hypertension (Oui / Non)
- Niveau de cholestérol (Normal / Élevé)
- Antécédents familiaux (Oui / Non)
- Valider les valeurs saisies (types, plages de valeurs).
- Envoyer la requête de prédiction à l'API FastAPI.

F3 – Affichage du résultat de prédiction

- Recevoir la réponse de l'API (score de risque, niveau de risque, confiance estimée).
- Afficher :
- Score de risque en pourcentage (0–100 %).
- Catégorie de risque : Faible, Modéré ou Élevé, avec un code couleur.
- Jauge ou barre de progression illustrant le niveau de risque.
- Message explicatif adapté au niveau de risque.
- Confiance du modèle (en %).

- Liste de recommandations générales (hygiène de vie, suivi médical, etc.)

- Afficher un avertissement : “Ce modèle est à usage pédagogique uniquement. Ne pas utiliser pour des décisions médicales réelles.”

F4 – Exploration des données
- Afficher des indicateurs globaux :
- Nombre d’échantillons utilisés.
- Taux de maladie dans le dataset.
- Âge moyen.
- IMC moyen.
- Afficher des graphiques :
- Répartition par sexe (camembert).
- Répartition sain/malade.
- Facteurs de risque (ex. fumeurs, hypertension).
- Distributions d’âge et d’IMC (moyenne, min, max, écart-type).
- Afficher une “Note méthodologique” décrivant :
- Taille et source du dataset.
- Variables utilisées.
- Proportion train/test.

F5 – Informations sur le projet
- Présenter le projet de fin d’études (contexte académique, objectifs).
- Lister les technologies Backend, Frontend et ML.
- Rappeler les limites et l’avertissement médical.

2.3 Fonctionnalités techniques

FT1 – Pipeline ETL et entraînement
- Charger le dataset brut (format CSV).
- Effectuer le nettoyage (valeurs manquantes, types, doublons).
- Encodage des variables catégorielles.
- Séparer les données en ensembles d’entraînement et de test (ex. 80 % / 20 %).
- Entraîner un modèle de régression logistique (Logistic Regression).
- Calculer les métriques : accuracy, precision, recall, F1-score, ROC-AUC.
- Sauvegarder le modèle entraîné au format joblib.

FT2 – API REST (FastAPI)
- Endpoint POST /predict :
- Entrée : JSON contenant les features (âge, sexe, IMC, fumeur, activité physique, hypertension, cholestérol, antécédents).
- Sortie : score de risque, niveau de risque, probabilité brute, confiance estimée.
- Endpoint GET /health :
- Vérifier que l’API répond et que le modèle est correctement chargé.
- (Optionnel) Endpoint GET /stats :
- Fournir les statistiques globales nécessaires à la page Exploration des données.

FT3 – Frontend (Next.js / React)
- Créer les pages : Accueil, Prédiction, Exploration.
- Consommer l’API via Axios.
- Gérer les états de chargement et d’erreur.
- Intégrer des graphiques interactifs via Recharts.
- Assurer une navigation fluide entre les différentes pages.

3. **Exigences non fonctionnelles**

3.1 Performance
- Temps de réponse moyen de l’API pour une requête de prédiction : inférieur à 500 ms.
- Temps de chargement initial de l’interface : inférieur à 3 s sur une connexion standard.

3.2 Sécurité
- Aucune donnée nominative (nom, prénom, email, etc.) ne doit être collectée ni stockée.
- Validation des entrées côté frontend et backend.
- Utilisation de HTTPS pour la communication en production.

3.3 Fiabilité
- Gestion des erreurs et messages explicites en cas d'échec de l'API ou d'input invalide.
- Mise en place de tests unitaires de base pour le prétraitement des données et la fonction de prédiction.

3.4 Ergonomie et UX
- Interface responsive (desktop prioritaire, lisible sur tablette).
- Utilisation d'une palette cohérente (fonds clairs, couleurs distinctes pour les niveaux de risque).
- Messages compréhensibles par un public non expert.

3.5 Maintenabilité
- Séparation claire des responsabilités (ETL/ML, API, Frontend).
- Code commenté et structuré en modules.
- Fichier de configuration pour les paramètres variables (chemins de fichiers, hyperparamètres).

3.6 Portabilité
- Applicatif déployable sur un environnement standard (Python 3.12 et Node.js).
- Compatibilité avec un déploiement sur serveur ou plateforme cloud.

4. **Architecture technique**

4.1 Vue globale
- Couche données : dataset de santé (fichier CSV).
- Couche Machine Learning : scripts d'ETL, d'entraînement et d'évaluation utilisant scikit-learn, pandas, numpy.
- Couche API : FastAPI exposant les endpoints /predict, /health et éventuellement /stats.
- Couche Frontend : application Next.js/React en TypeScript consommant l'API et affichant formulaires, résultats et graphiques.

4.2 Technologies
Backend :
- Python 3.12
- FastAPI, Uvicorn
- scikit-learn, pandas, numpy
- joblib
- Pydantic

Frontend :
- Next.js 14
- React 18
- TypeScript
- Axios
- Recharts

Outils :
- Git, GitHub
- IDE : VS Code / autre