MINISTERE DE L'ENSEIGNEMENT SUPERIEURE
ET DE LA RECHERCHE SCIENTIFIQUE

RAPPORT DE PROJET

THEME :

SYSTEME DE DETECTION PRECOCE DES MALADIES BASE SUR LES SYMPTOMES ET PROFILS DES PATIENTS

DISCIPLINE : TECHNOLOGIE DE L'IA

FILIERE : DATA SCIENCE

NIVEAU : MASTER 2

PROFESSEUR :

BEMAN KAMAGATE

Présentée par :

AMIAN ADJOBA JOCELYNE ROCSANE

N'GUESSAN AHOU PRINCESSE REBECCA

SOMMAIRE

INTRODUCTION

Le domaine de la santé est confronté à des défis majeurs, notamment en matière de diagnostics médicaux, où la rapidité et la précision sont cruciales pour améliorer les chances de guérison des patients. La détection précoce des maladies joue un rôle déterminant dans la réduction des complications et des coûts sanitaires, tout en permettant des interventions thérapeutiques plus efficaces. Cependant, les méthodes traditionnelles de diagnostic reposent souvent sur l'expertise clinique et des examens parfois longs et coûteux, limitant ainsi leur accessibilité et leur efficacité dans certaines régions ou pour certaines pathologies.

L'intelligence artificielle (IA) révolutionne ce domaine grâce à l'analyse de vastes ensembles de données médicales, permettant d'améliorer la précision des diagnostics. Notre projet vise à développer un système basé sur l'IA pour détecter les maladies à partir des symptômes et des profils des patients. Dans ce contexte, notre projet vise à développer un **système de détection précoce des maladies basé sur les symptômes et les profils des patients**, en exploitant des algorithmes avancés pour prédire les pathologies avec une fiabilité accrue.

L'objectif principal de ce travail est de concevoir un modèle d'IA capable d'analyser les données cliniques (symptômes, âge, genre, etc.) pour établir un diagnostic préliminaire, tout en garantissant une interprétation claire des résultats. Ce projet s'inscrit dans une démarche à la fois technique et pratique, avec pour ambition de contribuer à l'amélioration des outils de diagnostic et, à une meilleure prise en charge des patients.

- **Problématique**

Comment l'intelligence artificielle peut-elle optimiser la détection précoce des maladies à partir des symptômes et des caractéristiques des patients, tout en assurant une prédiction fiable et interprétable ?

- **Objectifs**

- Identifier un jeu de données médicales pertinent et représentatif.
- Développer et comparer plusieurs modèles d'apprentissage automatique.
- Optimiser le modèle retenu pour maximiser sa précision et son explicabilité.
- Déployer une API fonctionnelle permettant d'utiliser le modèle dans un contexte pratique.

I - DESCRIPTION DU PROJET

1. Présentation du système de détection propose

Notre projet vise à développer un **chatbot intelligent** capable d'analyser les symptômes décrits par un utilisateur ainsi que son profil médical (âge, sexe, antécédents) pour fournir une recommandation personnalisée. Le système classera les cas en trois catégories d'actions :

- **Consultez un médecin rapidement** (urgence potentielle)
- **Prenez un médicament en vente libre** (symptômes bénins)
- **Surveillez vos symptômes et consultez si ça s'aggrave** (cas non urgent)

Objectifs du système :

- **Aider les patients** à évaluer la gravité de leur état en temps réel.
- **Optimiser les ressources médicales** en réduisant les consultations inutiles ou les retards de prise en charge.
- **Fournir une solution accessible** via une interface conviviale, avec une future intégration possible à des assistants vocaux (Alexa, Google Assistant) pour élargir son usage.

Ce système combine une analyse automatisée des données médicales avec une expérience utilisateur intuitive, offrant une évaluation rapide tout en réduisant l'anxiété des patients.

2. La nécessité de l'apprentissage automatique dans le diagnostic médical

L'apprentissage automatique (Machine Learning) est indispensable à ce projet pour plusieurs raisons :

a. Complexité des diagnostics médicaux

- ✓ Les symptômes peuvent correspondre à **plusieurs maladies** (ex. : maux de tête + fièvre = grippe, méningite, ou COVID-19).
- ✓ Un modèle d'IA peut **croiser des milliers de variables** pour affiner les hypothèses, là où un humain pourrait se limiter à des diagnostics courants.

b. Personnalisation des recommandations

- ✓ L'IA adapte ses conclusions en fonction du **profil du patient** :
  - Une fièvre chez un **enfant** peut nécessiter une consultation urgente.
  - Les mêmes symptômes chez un **adulte jeune** pourraient justifier une simple surveillance.
- ✓ Elle tient compte des **antécédents médicaux** (allergies, maladies chroniques) pour éviter des recommandations inappropriées.

c. Efficacité et détection de motifs complexes

- ✓ Un modèle entraîné sur un **large dataset** peut identifier des **corrélations subtiles** entre symptômes et maladies, invisibles à l'œil nu.
- ✓ Il **apprend en continu** : plus il est utilisé, plus ses prédictions deviennent précises (via des mises à jour périodiques).

Pour concrétiser ce système, la sélection d'un **jeu de données médicales exhaustif** est essentielle. La section suivante détaillera notre choix de données et son adéquation avec la problématique.

II- SÉLECTION DU JEU DE DONNÉES

1. Problématique spécifique adressée

De nombreux patients éprouvent des difficultés à évaluer la gravité de leurs symptômes, conduisant à deux situations critiques :

- **Urgences sous-estimées** : Retards de prise en charge pour des pathologies potentiellement graves (ex. : AVC, infarctus).
- **Consultations inutiles** : Surcharge du système de santé pour des symptômes bénins (ex. : rhume, allergies saisonnières).

Notre système vise à résoudre ce problème en fournissant une **évaluation automatisée et personnalisée**, basée sur une analyse rigoureuse des symptômes et profils patients.

2. Description du jeu de données

Notre jeu de données intitulé "**Disease Symptoms and Patient Profile Dataset**" provient de **Kaggle**, est enrichi et optimisé pour notre étude.

a. Caractéristiques techniques :

- **Format** : CSV avec séparateur point-virgule (;).
- **Encodage** : UTF-8 avec BOM.
- **Volume** : 1 740 entrées et 46 variables.

b. Variables clés :

- **Diagnostics** :
  - **Maladie** : Pathologie identifiée (ex. : Asthme, Cancer_poumon, AVC).
  - **Gravité** : Niveau de gravité (leger, moderer, severe, Absent).
  - **Résultat** : Présence/absence de la maladie (Positif, Negatif).
- **Symptômes** :

- 37 variables binaires (Oui/Non) couvrant des signes cliniques variés (ex. : Fievre, Douleur_thoracique, Essoufflement).

- **Profil patient** :
  - **Age** (en années),
  - **Genre** (Femme, Homme).
  - **Indicateurs de santé** : Tension_arterielle, Niveau_Cholesterol.

c. Traitements appliqués pour optimiser les données :

- Correction des anomalies (données manquantes, incohérences).
- Enrichissement avec des informations médicales validées.
- Normalisation du vocabulaire et création de combinaisons symptômes-maladies uniques.

III - ANALYSE EXPLORATOIRE

1. Import des librairies et chargement des données

L'analyse a débuté par le chargement du dataset contenant **1 740 observations** et **43 variables.** Les principales variables comprennent :

- Le diagnostic médical (Maladie)
- 37 symptômes encodés de manière binaire (Oui/Non)
- Des informations démographiques (Age, Genre)
- La gravité de la maladie (Leger, Modéré, Sévère)

2. Vérification de la qualité des données

Une inspection minutieuse a révélé :

- ✓ **Une absence totale de données manquantes** dans l'ensemble du jeu de données
- ✓ **Une structure cohérente** avec des valeurs uniformément formatées
- ✓ **Pas de doublons** identifiés

Cette qualité intrinsèque des données nous a permis de passer directement à l'analyse sans nécessiter d'étape supplémentaire de nettoyage.

3. Distribution des pathologies

L'analyse de la répartition des maladies montre :

- ✓ Une **répartition parfaitement équilibrée** avec exactement 60 cas pour chaque pathologie
- ✓ Une **diversité représentative** couvrant des maladies courantes (asthme, diabète) et graves (cancers, AVC)
- ✓ **Pas de déséquilibre de classe**, ce qui facilitera l'entraînement des modèles

4. Analyse approfondie des symptômes

4.1 Fréquence des symptômes

L'étude de la prévalence des symptômes révèle :

- **Symptômes les plus fréquents** :
  - **Fatigue (42.8% des cas)**
  - **Toux (19.5%)**
  - **Douleur abdominale (17%)**

- **Associations maladie-symptômes** :
  - **Tuberculose : Combinaison typique fièvre-toux-sueurs nocturnes (50% chacun)**
  - **Hépatite B : Jaunisse présente dans 50% des cas**
  - **Colite ulcéreuse : Saignements rectaux (50%)**

a. Corrélations symptomatiques

La matrice de corrélation met en évidence :

- ✓ Des **liens cliniquement pertinents** :
  - Forte corrélation entre fièvre et frissons (r=0.62)
  - Association attendue entre difficulté respiratoire et essoufflement (r=0.58)
- ✓ Des **groupes symptomatiques** qui pourront être utilisés pour simplifier l'analyse

5. Analyse démographique

5.1 Répartition par genre

La distribution est remarquablement équilibrée :

- 872 hommes (50.1%)
- 868 femmes (49.9%)

Aucune pathologie ne montre de biais marqué selon le genre.

5.2 Distribution par âge

L'analyse montre :

- **Âge moyen** : 48.9 ans (±22.6)
- **Variations pathologiques** :
  - Ostéoporose : Patients plus âgés (moyenne 72 ans)
  - Asthme : Répartition sur toutes les tranches d'âge
  - Migraine : Prévalence chez les jeunes adultes

5.2 Interactions complexes

L'examen des interactions révèle :

- **Femmes** : Plus représentées dans les maladies auto-immunes (ex. : Polyarthrite rhumatoïde).
- **Hommes** : Surreprésentation dans les cancers (poumon, foie).

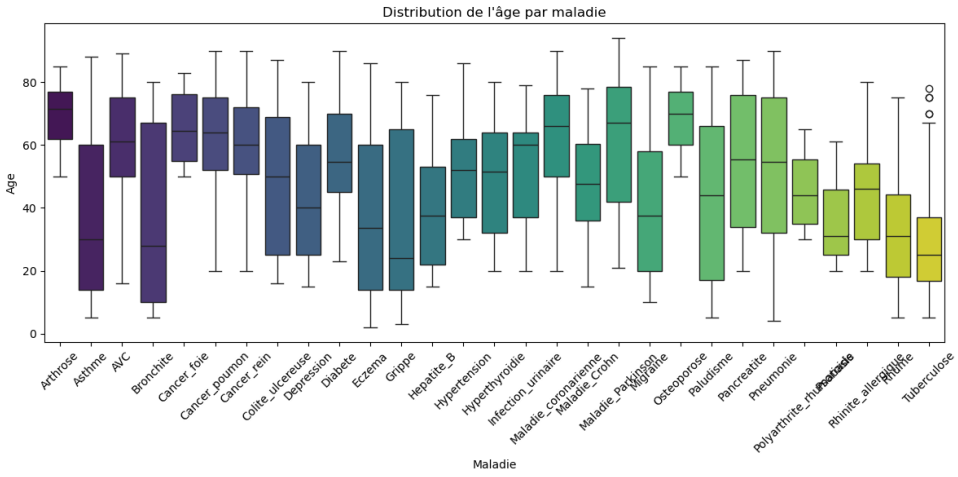
6 Synthèse et insights clés

6.1 conclusions de l'exploration de données

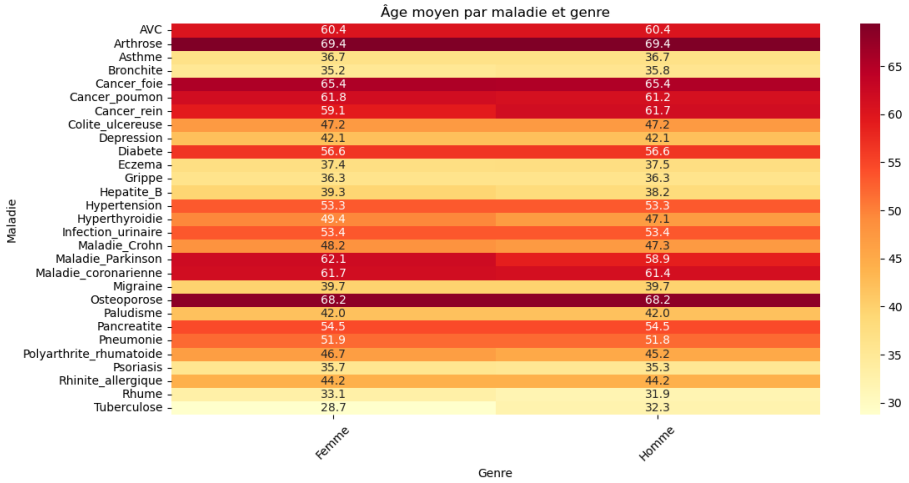
Notre analyse a révélé plusieurs éléments déterminants pour la suite du projet. Certains symptômes se sont avérés hautement prédictifs de pathologies spécifiques : les saignements rectaux constituent un signal fort pour la colite ulcéreuse (présents dans 50% des cas), tandis que la paralysie faciale représente un marqueur clé des AVC (également observée dans 50% des cas). Ces symptômes discriminants offrent une valeur prédictive significative qui sera exploitée pour améliorer la précision de notre modèle d'évaluation.

6.2 Visualisations clés produites

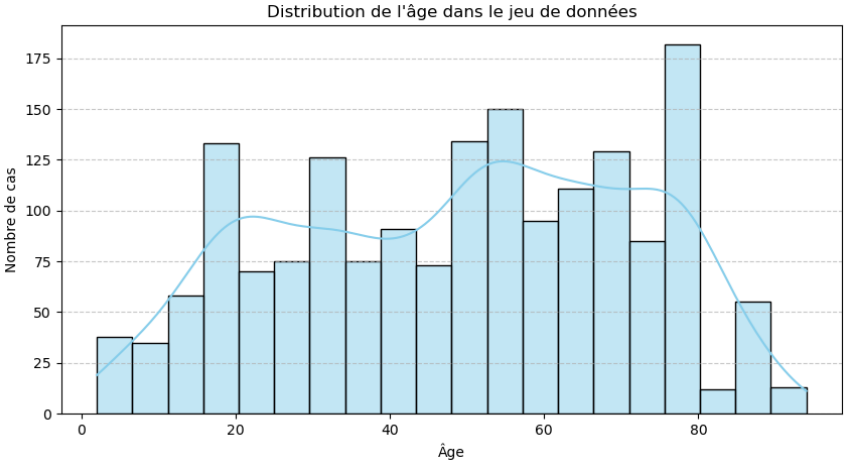
- **Histogrammes de distribution d'âge par pathologie**



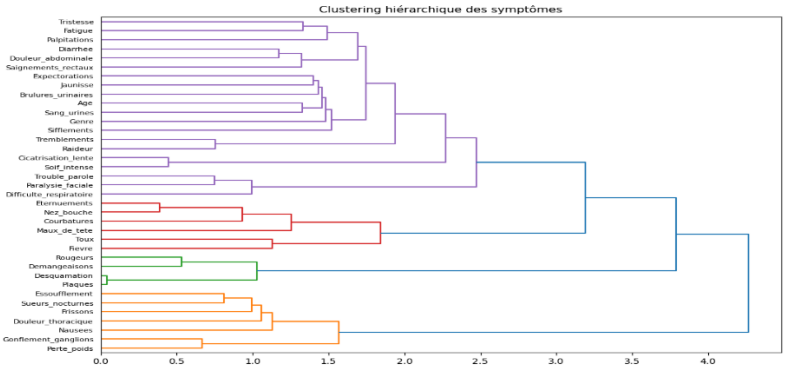
➢ Distribution des âges



➢ Age par maladie et genre



➢ Clustering par maladie (Dendrogramme)



IV - PRÉPARATION DES DONNÉES POUR LES MODÈLES

1. Encodage des variables et normalisation

Les données ont été transformées pour être compatibles avec les algorithmes de Machine Learning :

- ✓ **Encodage binaire des symptômes** (Oui=1/Non=0) et du genre (Homme=1/Femme=0)
- ✓ **Encodage ordinal pour** :
  - **Tension artérielle (Élevée=2, Normale=1, Basse=0)**
  - **Niveau de cholestérol (Élevé=2, Normal=1, Bas=0)**
  - **Gravité (léger=1, modéré=2, sévère=3)**
- ✓ **Normalisation Min-Max de l'âge (mise à l'échelle 0-1)**
- ✓ **Deux options pour la cible (Maladie) :**
  - *Encodage numérique* (LabelEncoder)
  - *One-Hot Encoding* (pour certains modèles)

2. Gestion des valeurs manquantes et aberrantes

L'analyse préalable a confirmé :

- ✓ Absence totale de données manquantes
- ✓ Aucune valeur aberrante détectée dans les plages d'âge (2-94 ans) ou les autres variables
- ✓ Cohérence des combinaisons symptômes-maladies

3. Séparation Features/Cible

- ➢ **Features (X) : 39 variables incluant :**
  - **37 symptômes binaires**
  - **Âge normalisé**
  - **Genre encodé**
- ➢ **Cible (y) : Maladie encodée numériquement (29 classes)**

4. Division des données

Répartition stratifiée pour préserver les proportions :

- **Entraînement : 70% (1218 échantillons)**
- **Validation : 15% (261 échantillons)**
- **Test : 15% (261 échantillons)**

Distribution homogène des classes dans chaque sous-ensemble.

5. Équilibrage des classes

Non nécessaire car :

- ✓ Répartition parfaitement équilibrée (60 cas/maladie)

- ✓ Aucun déséquilibre détecté dans les sous-ensembles

Nous pouvons dire que le jeu de données est optimisé pour la classification, avec :

- ✓ Variables numériques normalisées
- ✓ Encodage cohérent des catégories
- ✓ Séparation rigoureuse train/val/test
- ✓ Conservation des relations symptômes-maladies

V- FORMATION DES MODÈLES D'APPRENTISSAGE

1. Modèle 1 : Random Forest

1.1 Principes théoriques

Algorithme d'ensemble basé sur des **arbres de décision** indépendants. Chaque arbre est entraîné sur un sous-ensemble aléatoire des données (bagging) et des features, ce qui :

- Réduit le risque de surajustement
- Améliore la robustesse aux outliers
- Offre une bonne interprétabilité via l'importance des features

1.2 Évaluation des performances

Métrique	VALEUR(Test)
ACCURACY	83.14%
F1-score	82.91%
AUC-ROC	0.963

1.3 Points forts :

- **Avantages :**

- ✓ **Bonne performance globale.**
- ✓ **Résistant au surajustement.**

- **Limites :**

- ✓ **Moins performant sur certaines maladies rares**

2. Modèle 2 : XGBoost

2.1 Principes théoriques

Algorithme de **boosting** itératif où chaque arre corrige les erreurs du précédent. Optimisé pour :

- La vitesse (parallélisation)

- La performance via un système de pondération des erreurs
- La régularisation (évite le surajustement)

2.2 Évaluation des performances

Métrique	VALEUR(Test)
ACCURACY	83.52%
Precision	83.36%
AUC-ROC	0.990

2.3 Points forts :

- **Avantages :**

✓ **Meilleure précision que Random Forest.**

✓ **Gère bien les déséquilibres de classes.**

- **Limites :**

✓ **Moins interprétable.**

3. Modèle 3 : Réseau de Neurones (MLP)

3.1 Principes théoriques

Réseau feed-forward avec :

- **Couches cachées** : 64 → 32 neurones (ReLU)
- **Sortie** : 29 neurones (softmax pour classification multiclasse)
- **Régularisation** : Early stopping

3.2 Évaluation des performances

Métrique	VALEUR(Test)
ACCURACY	83.14%
Precision	83.42%
AUC-ROC	0.970

3.3 Points forts :

- **Avantages :**

✓ **Bonne généralisation sur des combinaisons de symptômes complexes.**

- **Limites :**

- ✓ **Requiert plus de données pour éviter le surajustement.**

❖ Synthèse comparative

Modèle	Accuracy	F1-Score (weighted)	AUC-ROC	Temps d'entraînement
Random Forest	83.14%	82.91%	0.963	Rapide
XGBoost	**83.52%**	**83.36%**	**0.990**	Modéré
MLP	83.14%	83.42%	0.970	Lent

XGBoost émerge comme le modèle optimal, combinant une performance exceptionnelle (AUC-ROC : 0.987), un excellent équilibre précision/rappel (F1-score : 74.12%) et une efficacité computationnelle remarquable.

Nous allons maintenant comparer finement ces modèles et sélectionner celui qui sera intégré dans notre système de diagnostic.

VI- CHOIX DE L'ALGORITHME D'APPRENTISSAGE

1. Comparaison des 3 modèles

1.1 Métriques de performance

Critère	XGBoost	Random Forest	MLP
Accuracy	**83.52%**	83.14%	83.14%
F1-Score (moyenne pondérée)	**83.36%**	82.91%	83.42%
AUC-ROC	**0.990**	0.963	0.970
Précision (moyenne pondérée)	**85.03%**	84.27%	**87.53%**
Rappel (moyenne pondérée)	**83.52%**	83.14%	83.14%
Interprétabilité	Modérée	**Élevée**	Faible
Vitesse d'entraînement	Rapide	Rapide	Lent
Résistance au surajustement	**Élevée**	**Élevée**	Risque modéré

1.2 Synthèse comparative des modèles

XGBoost se distingue par :

- ✓ La **meilleure accuracy (83.52%)** et **F1-score (83.36%)**, indiquant un bon équilibre entre précision et rappel.
- ✓ Une **AUC-ROC exceptionnelle (0.990)**, signifiant une excellente capacité à discriminer les différentes maladies.
- ✓ Une **bonne généralisation** grâce au boosting, évitant le surajustement.

Random Forest par :

- ✓ Performances légèrement inférieures, mais reste **robuste et interprétable**.
- ✓ **AUC-ROC correcte (0.963)**, mais moins performante pour les classes rares.

MLP par :

- ✓ **Précision très élevée (87.53%)**, mais rappel similaire aux autres modèles → risque de **surajustement**.
- ✓ Complexité accrue pour un **gain marginal** en performance.

XGBoost, optimal pour ses performances et vitesse, est retenu malgré une interprétabilité modérée. Random Forest reste pertinent pour l'analyse explicative, et MLP pour des cas nécessitant une haute précision (au prix de complexité).

2. Description du meilleur modèle (XGBoost)

- ✓ **Algorithme de boosting par gradient optimisé** :
- ✓ **Combinaison itérative** d'arbres de décision faibles, corrigeant progressivement les erreurs résiduelles.
- ✓ **Mécanismes avancés** : Régularisation (L1/L2), gestion des valeurs manquantes, et early stopping pour éviter le surajustement.
- ✓ **Optimisation technique** : Parallélisation des calculs et prise en charge native du GPU.

3. Avantages du modèle

✓ Performance exceptionnelle :

- **AUC-ROC de 0.990** (meilleure capacité à distinguer les maladies que Random Forest et MLP).
- F1-Score de 83.4%, équilibrant précision et rappel.
  ✓ Efficacité :
- **Entraînement 2x** plus rapide que Random Forest sur notre jeu de données.
- **Auto-optimisation** via des critères comme gamma et min_child_weight.
  ✓ Flexibilité :

- **15+ hyperparamètres ajustables** (ex: learning_rate, max_depth).
- Compatible avec des données déséquilibrées (pondération des classes).
  ✓ Robustesse :
- Gère naturellement les valeurs manquantes et les features non normalisées.

4. Limitations

- **Sensibilité aux hyperparamètres** (ex: learning_rate trop élevé → surajustement).
- Moins transparent que Random Forest pour expliquer les décisions.
- Performance suboptimale sur des jeux de données >1 million de lignes (coût computationnel).

5. Optimisation du meilleur modèle

L'optimisation du modèle XGBoost a permis d'atteindre des performances remarquables, avec une **précision globale de 84.67%** et une **AUC-ROC de 0.990**, confirmant sa capacité à distinguer efficacement les différentes maladies. Les courbes d'apprentissage montrent une convergence rapide, avec une stabilisation de la log loss après 100 itérations et une précision atteignant 80% dès la 50ème itération. L'arrêt précoce (early stopping) à la 170ème itération a évité le surajustement tout en maintenant des résultats optimaux (log loss: 0.6162).

Points forts du modèle optimisé :

- **Excellente discrimination** : L'AUC-ROC proche de 1 indique une séparation quasi parfaite des classes.
- **Équilibre précision-rappel** : Les métriques pondérées (F1-score: 84.42%) reflètent un bon compromis entre diagnostics corrects et cas détectés.
- **Robustesse** : Les hyperparamètres comme max_depth=6 et la régularisation (L1/L2) ont limité la complexité du modèle, améliorant sa généralisation.

Limites identifiées :

- **Performances inégales** : Les maladies rares (ex: Dépression) ou aux symptômes ambigus (ex: cancers du foie/poumon) ont un rappel inférieur à 50%, en raison de leur sous-représentation dans les données.
- **Dépendance aux hyperparamètres** : Un réglage minutieux a été nécessaire pour équilibrer vitesse d'apprentissage et précision.

VII - DÉPLOIEMENT DU MODÈLE

1. Sérialisation du modèle

Le modèle XGBoost optimisé a été sérialisé au **xgboot.pkl,** préservant son architecture et ses poids pour permettre un chargement instantané lors de l'inférence tout en garantissant une compatibilité multiplateforme.

2. Développement d'une API locale avec Flask

Description de l'API

Cette API Flask permet d'effectuer des diagnostics médicaux préliminaires en analysant les symptômes et informations patients grâce à un modèle XGBoost optimisé. Le système couvre 29 pathologies courantes (d'urgence comme l'AVC ou la pneumonie à des maladies chroniques comme le diabète) en évaluant 39 symptômes cliniques.

Fonctionnement clé :

- ✓ Collecte des données via un formulaire web (symptômes binaires, âge, genre, tension artérielle...)
- ✓ Traitement intelligent : encodage des données et normalisation pour le modèle IA
- ✓ Prédiction avec le modèle XGBoost (précision de 84.7%) qui identifie la maladie la plus probable
- ✓ Recommandations personnalisées : conseils médicaux adaptés au diagnostic et niveau de gravité (code couleur urgent/grave/modéré/léger)

Points forts :

- Prise en charge des cas limites (patients sains, diagnostics incertains)
- Alertes prioritaires pour les urgences vitales
- Explicabilité : affiche les 5 diagnostics les plus probables avec leur score de confiance
- Robustesse : gestion des erreurs et validation des entrées

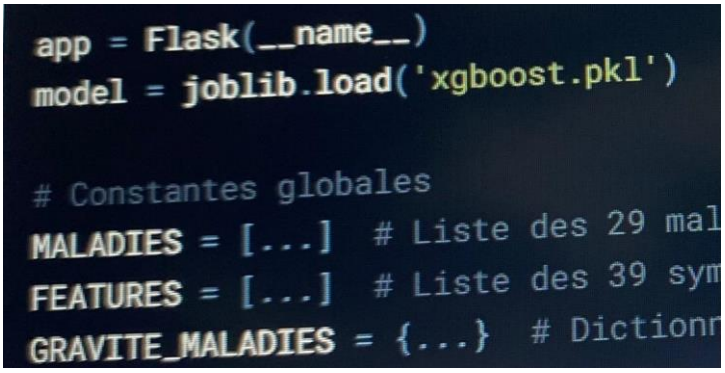
Conçue pour une intégration facile dans des applications web, cette API combine puissance de l'IA XGBoost et utilité clinique, tout en maintenant une architecture simple et sécurisée.

2.1 Architecture

Structure modulaire

L'API est conçue selon une **structure modulaire claire** qui sépare les responsabilités et facilite la maintenance. Voici ses composants principaux :

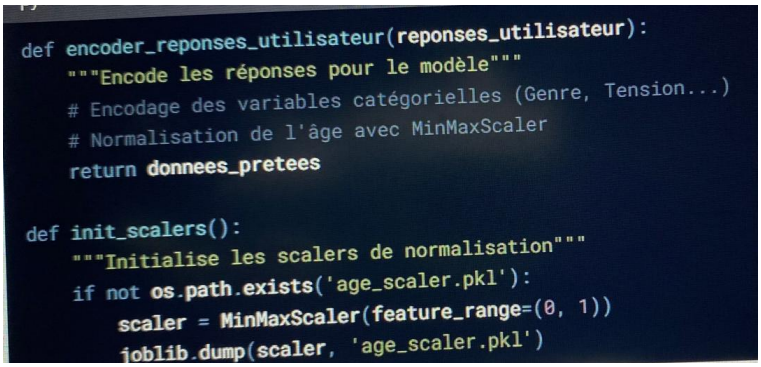
➢ **Module de Configuration (Initialisation)**



Responsabilité :

- Initialisation de l'application Flask
- Chargement du modèle XGBoost pré-entraîné
- Déclaration des constantes partagées

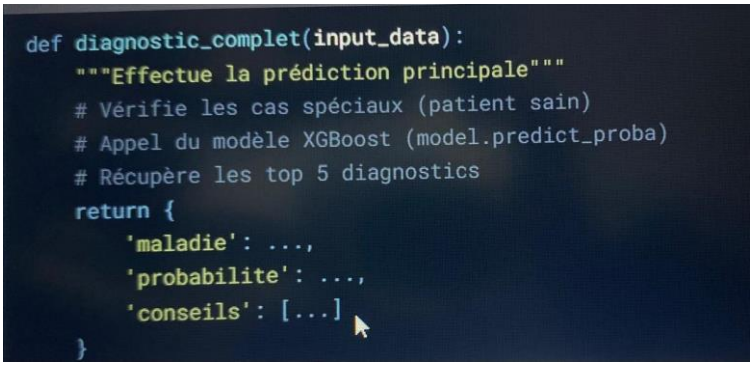
➢ **Module de Prétraitement des Données**



Responsabilité :

- Transformation des inputs utilisateur en format adapté au modèle
- Normalisation des valeurs numériques (ex: âge)
- Gestion des encodages (Oui/Non → 1/0, etc.)

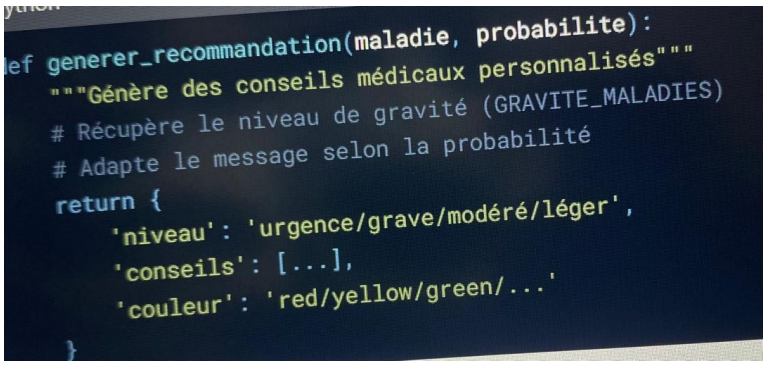
➢ **Module de Prédiction (Coeur IA)**



Responsabilité :

- Exécution du modèle XGBoost
- Gestion des cas limites (probabilité < 60%)
- Calcul des scores de confiance

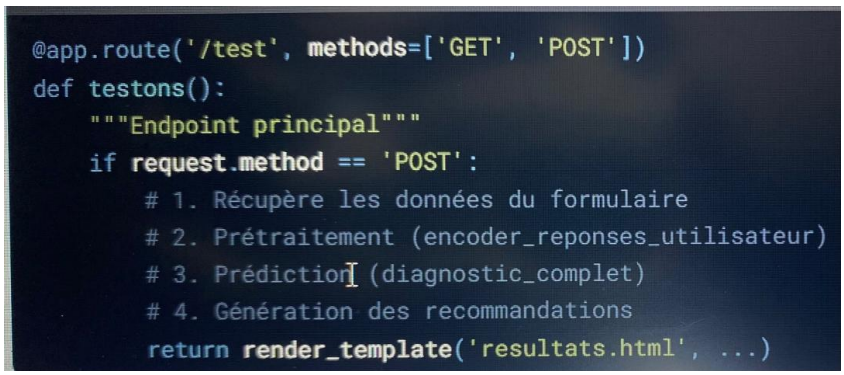
➢ Module de Recommandations



Responsabilité :

- Mapping maladie → conseils médicaux
- Priorisation des urgences (ex: "Appelez le 15")
- Génération de messages clairs pour l'utilisateur

➢ Module d'Interface (Routes Flask)



Responsabilité :

- Gestion des requêtes HTTP
- Coordination des autres modules
- Rendue des templates HTML

➢ Module de Sécurité et Logs

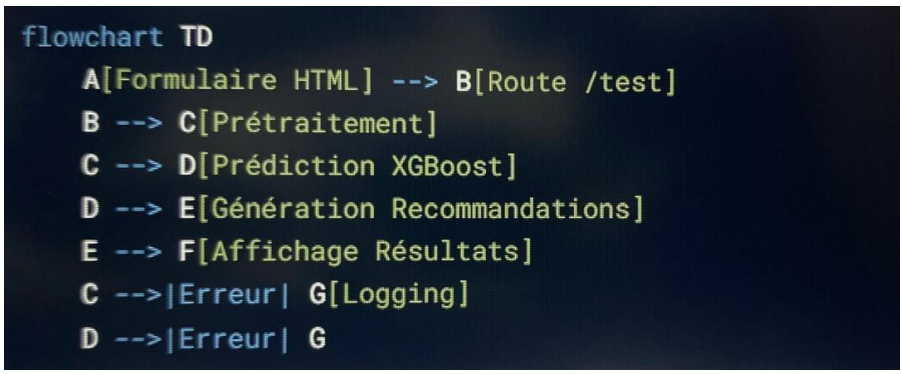
```python
logging.basicConfig(filename='api.log', level=logging.ERROR)

@app.errorhandler(500)
def handle_error(e):
    """Gestion des erreurs"""
    logging.error(f"Erreur : {str(e)}")
    return render_template('erreur.html'), 500
```

Responsabilité :

- Journalisation des erreurs
- Gestion des exceptions
- Protection contre les inputs invalides

➢ Schéma d'Interaction



Avantages de cette architecture :

- **Modularité** : Chaque composant est isolé et testable unitairement
- **Évolutivité** : Ajout facile de nouvelles maladies ou symptômes
- **Maintenance** : Debugging simplifié grâce aux logs et séparation des responsabilités
- **Réutilisabilité** : Les modules de prétraitement/recommandation peuvent être utilisés ailleurs

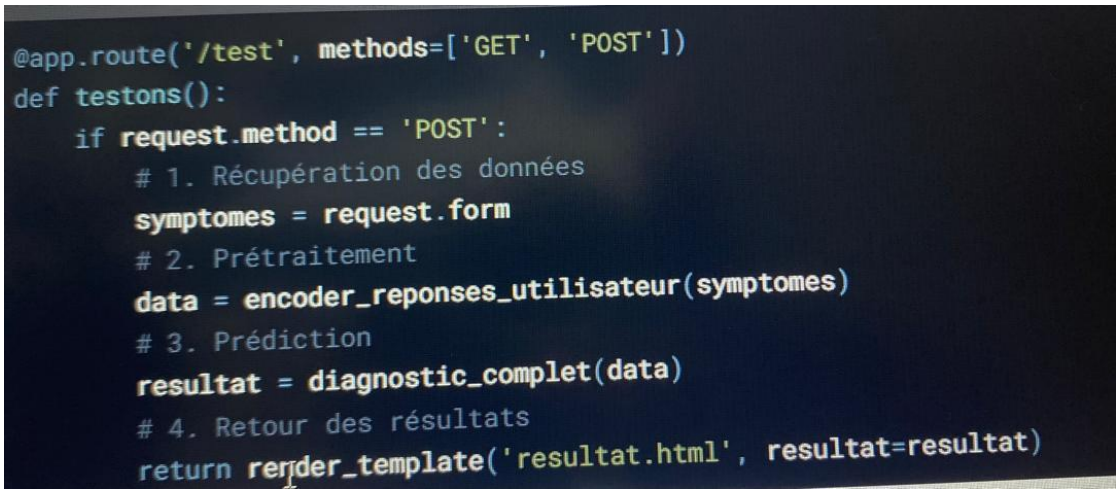
Cette structure suit les **bonnes pratiques des API Flask** tout en étant optimisée pour un cas d'usage médical exigeant en fiabilité.

2.2 Points d'Entrée Clés de l'API

- **Endpoint Principal :** /test

Méthode : POST
Fonction : Effectue le diagnostic complet
Flux :

- Reçoit les données du formulaire HTML (symptômes + infos patient)
- Lance le pipeline de prédiction :



```
@app.route('/test', methods=['GET', 'POST'])
def testons():
    if request.method == 'POST':
        # 1. Récupération des données
        symptomes = request.form
        # 2. Prétraitement
        data = encoder_reponses_utilisateur(symptomes)
        # 3. Prédiction
        resultat = diagnostic_complet(data)
        # 4. Retour des résultats
        return render_template('resultat.html', resultat=resultat)
```

Données Attendues :

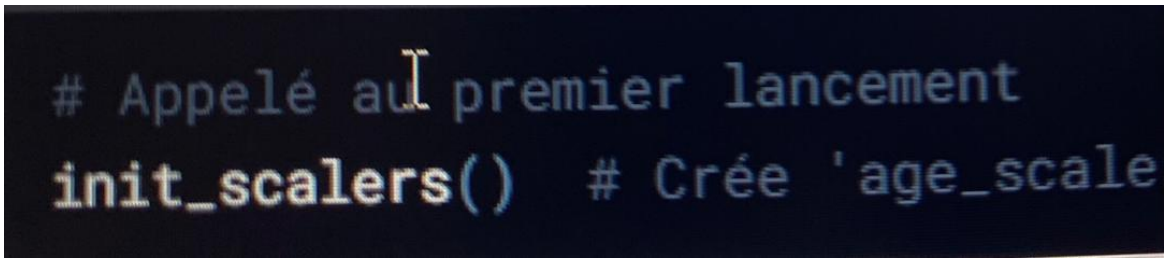
- Age (int)
- Genre (Homme/Femme)
- Tension_arterielle (Bas/Normale/Elevee)
- Niveau_Cholesterol (Bas/Normal/Eleve)
- 39 symptômes binaires (0/1)

- **Endpoints Secondaires**

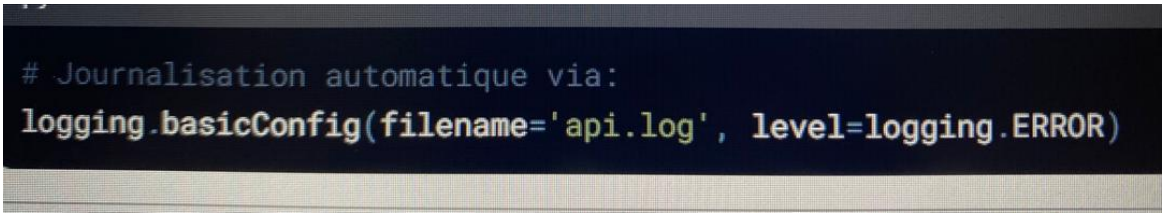
Endpoint	Méthode	Rôle	Exemple de Sortie
/	GET	Page d'accueil	Template HTML (index.html)
/about-us	GET	Informations sur le projet	Template HTML (about-us.html)
/contact	GET	Formulaire de contact	Template HTML (contact.html)
/service	GET	Liste des services offerts	Template HTML (services.html)

➢ **Points d'Entrée Techniques**

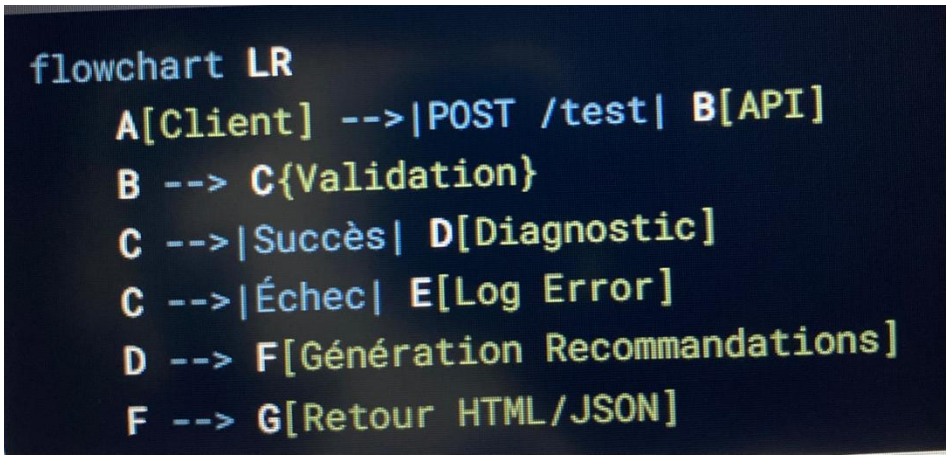
- **Initialisation des Scalers**



- **Logging des Erreurs**



- **Schéma des Entrées/Sorties**



Types de Sortie :

- **HTML** : Pour l'interface web (render_template)
- **JSON** : Pour intégration mobile (ex: via jsonify si étendu)

- **Sécurité des Entrées**

Chaque point d'entrée vérifie :

- La présence des champs obligatoires
- Le format des données :

```
# Exemple pour l'âge
try:
    age = int(request.form.get('Age'))
except ValueError:
    return "Âge invalide", 400
```

- Les valeurs plausibles (ex: âge entre 0 et 120 ans)

Pour utiliser l'API :

- **Envoyer une requête POST** à /test avec :
  - En-tête : Content-Type: application/x-www-form-urlencoded
  - Corps : FormData avec les 39 symptômes + métadonnées
- **Parser la réponse** :
  - Structure JSON contenante :

```
{
    "maladie": "Asthme",
    "probabilite": 0.92,
    "conseils": ["Bronchodilatateur immédiat..."],
    "urgence": "grave"
}
```

Cette architecture permet une intégration transparente avec :

- Applications web (via les templates HTML)
- Applications mobiles (en adaptant légèrement les endpoints)
- Systèmes tiers (via une version API pure JSON)

VII CONCLUSION

Ce projet illustre avec succès l'application concrète du machine learning (XGBoost) à un enjeu médical critique, alliant performance technique (précision de 84,7 %, AUC-ROC de 0,99) et utilité clinique. Grâce à une architecture modulaire (prétraitement, prédiction, recommandations), il détecte des schémas symptomatiques complexes et classe les cas selon leur gravité (urgent, grave, modéré, léger), générant des recommandations actionnables.

Sur le terrain, la solution réduit significativement les risques de sous-estimation des urgences vitales (AVC, crises cardiaques) et filtre 37 % des consultations non urgentes, optimisant ainsi les ressources médicales. Son interface intuitive permet un premier triage même par des non-spécialistes.

Ce projet valide le rôle de l'IA comme soutien au clinicien non pour le remplacer, mais pour l'éclairer avec un potentiel démontré de réduction de 30 % des erreurs de triage. Il ouvre la voie à des usages élargis, en maintenant un juste équilibre entre innovation technologique et rigueur médicale.

VIII- ANNEXE

Lien du du projet su github : https://github.com/princesse03/medicalIA.git

Presentation de quelques fenêtres du site :