AI-Powered Smart Health Prediction System

A Machine Learning and RAG-Based Healthcare Analysis and Recommendation Platform

Prepared By: Kashan Baig (CT-24178), Hadiya Kashif (CT-22008), Hassan Shakil (CT-24185)

Department of Computer Science, NED University

Table Of Contents

[Problem Statement 3](#_Toc211975893)

[Objectives 3](#_Toc211975894)

[Importance 3](#_Toc211975895)

[Significance 3](#_Toc211975896)

[Tech Stack Used 3](#_Toc211975897)

[Limitations 4](#_Toc211975898)

[Front-End Interface 5](#_Toc211975899)

[Front-End Components: 5](#_Toc211975900)

[Data Files 9](#_Toc211975901)

[X\_train.csv & X\_test.csv 9](#_Toc211975902)

[y\_train.csv & y\_test.csv 9](#_Toc211975903)

[cleaned\_health\_data.csv 9](#_Toc211975904)

[enhanced\_health\_data.csv 9](#_Toc211975905)

[complete\_processed\_dataset.csv 9](#_Toc211975906)

[processed\_features.csv & processed\_target.csv 10](#_Toc211975907)

[Source Code Modules 11](#_Toc211975908)

[data\_ingestion.py 11](#_Toc211975909)

[data\_cleaning.py 11](#_Toc211975910)

[data\_transformation.py 11](#_Toc211975911)

[model\_training.py 11](#_Toc211975912)

[model\_evaluation.py 12](#_Toc211975913)

[rag\_integration.py 12](#_Toc211975914)

[Model Artifacts 13](#_Toc211975915)

[Deployment Components 13](#_Toc211975916)

[Jupyter Notebook Artifacts 13](#_Toc211975917)

[ML Pipeline Overview 14](#_Toc211975918)

[Requirements 15](#_Toc211975919)

[Conclusion 15](#_Toc211975920)

# Problem Statement

Build a transparent health risk service that predicts health risk from vitals and lifestyle data, explains results, suggests next actions, finds nearby doctors, and sends email alerts for high risk.

# Objectives

• Ingest, clean, and transform health data reproducibly.  
• Train multiple classifiers and compare their performance.  
• Serve predictions through a simple Flask API.  
• Generate RAG-based explanations and next steps.  
• Notify high-risk users via email.

# Importance

Early detection enables timely intervention. The project builds user trust through explanations and demonstrates practical use of AI in healthcare.

# Significance

A practical ML + RAG system for healthcare analytics with reproducibility and small footprint. Ideal for educational and prototype environments.

# Tech Stack Used

| **Category** | **Technologies Used** |
| --- | --- |
| **Data & ML** | pandas, numpy, scikit-learn, CatBoost, XGBoost, joblib |
| **API & Deployment** | Flask, flask-cors, python-dotenv, requests, SMTP |
| **RAG (Retrieval)** | LangChain, sentence-transformers, ChromaDB, FAISS CPU, transformers |
| **Visualization** | seaborn, matplotlib, BeautifulSoup (bs4), email-validator |

# Limitations

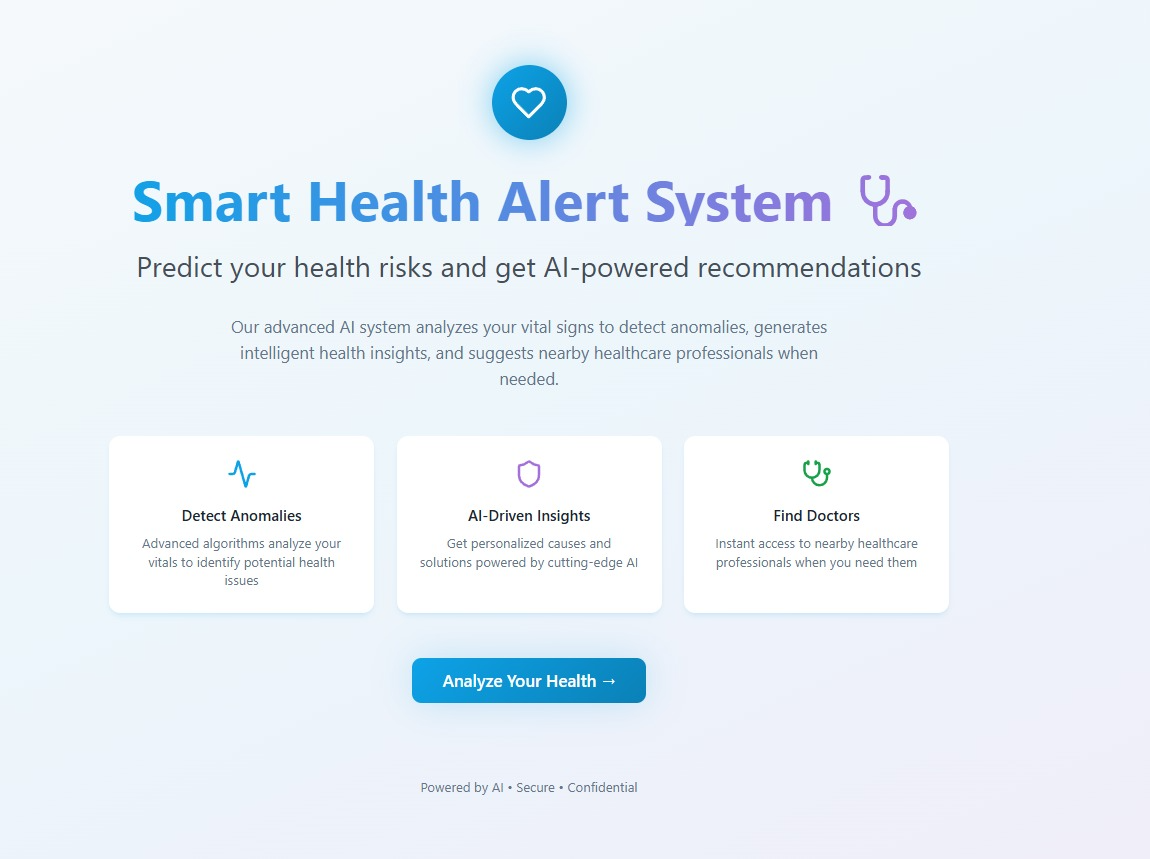
• **Prototype**; not validated for clinical use.  
• Overpass API may timeout.  
• Distance approximations only.  
• RAG quality depends on indexed content.  
• Some hard-coded paths.

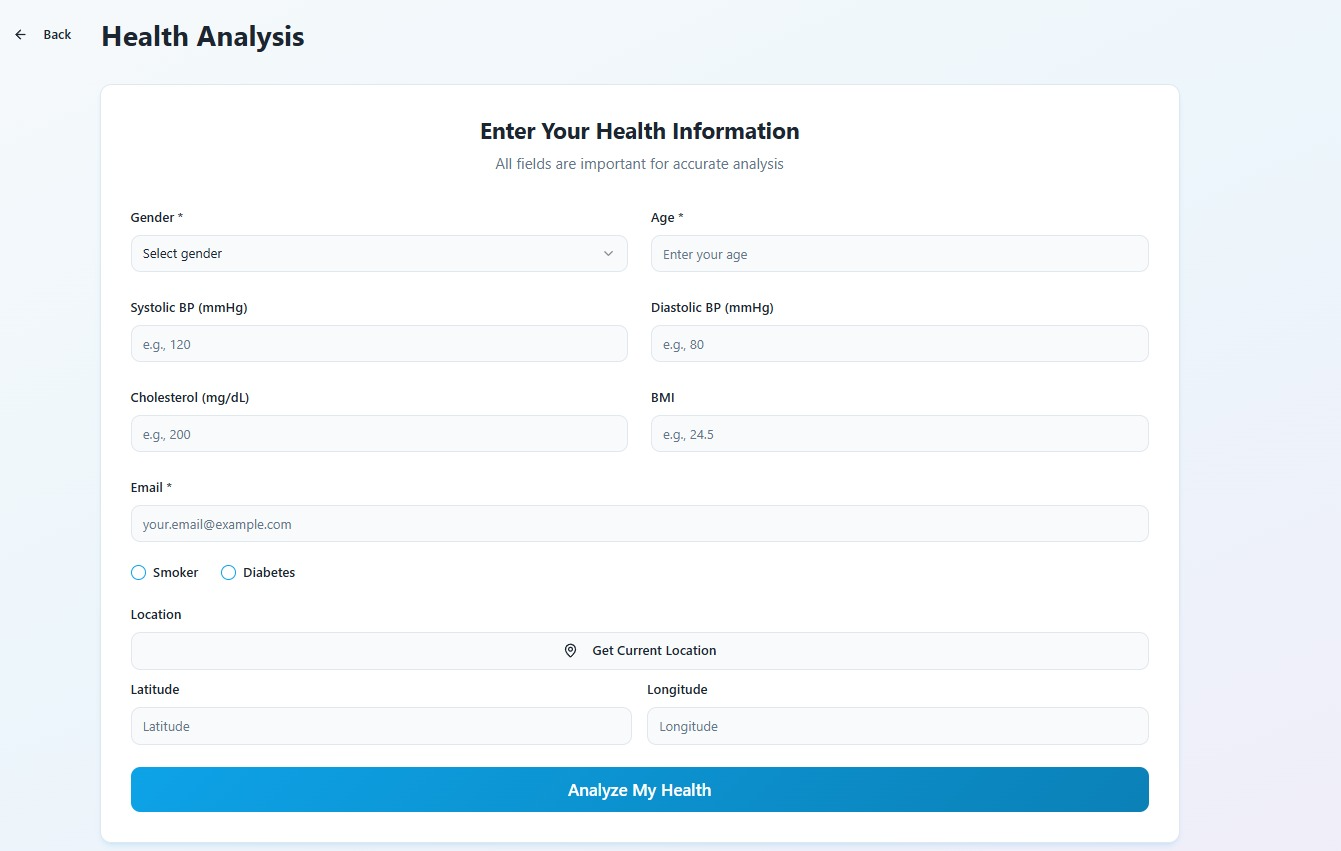
# Front-End Interface

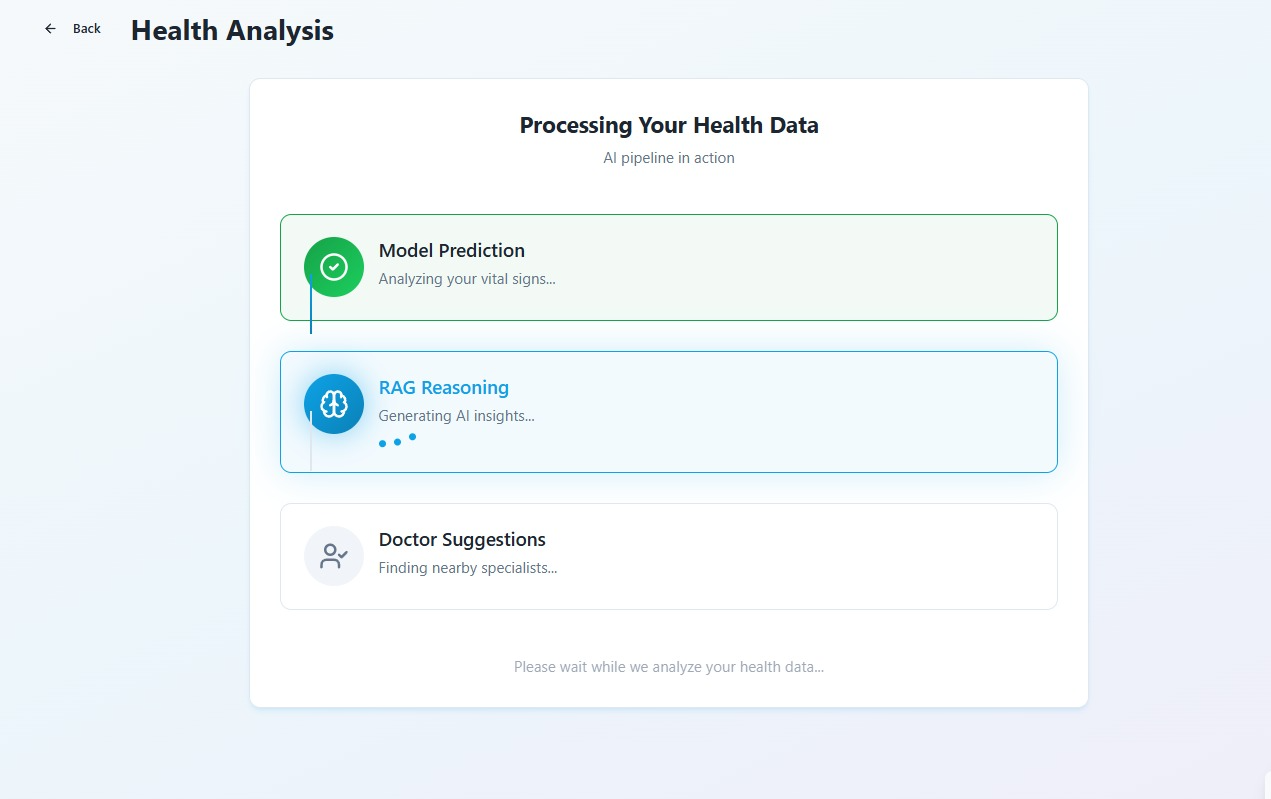
The Smart Health Prediction System provides an intuitive and responsive user interface that allows users to input   
their health-related information, view AI-generated predictions, and access recommendations through RAG integration.

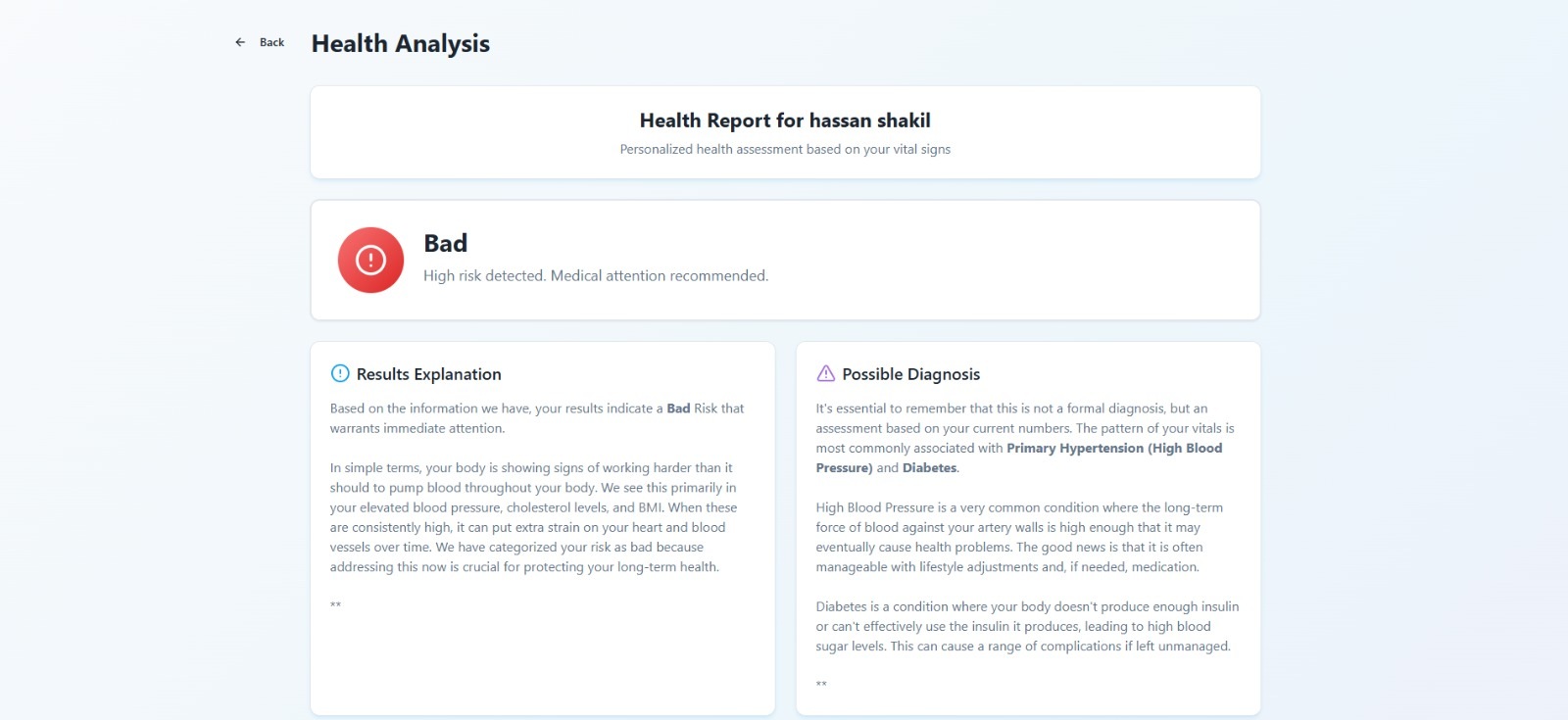
## Front-End Components:

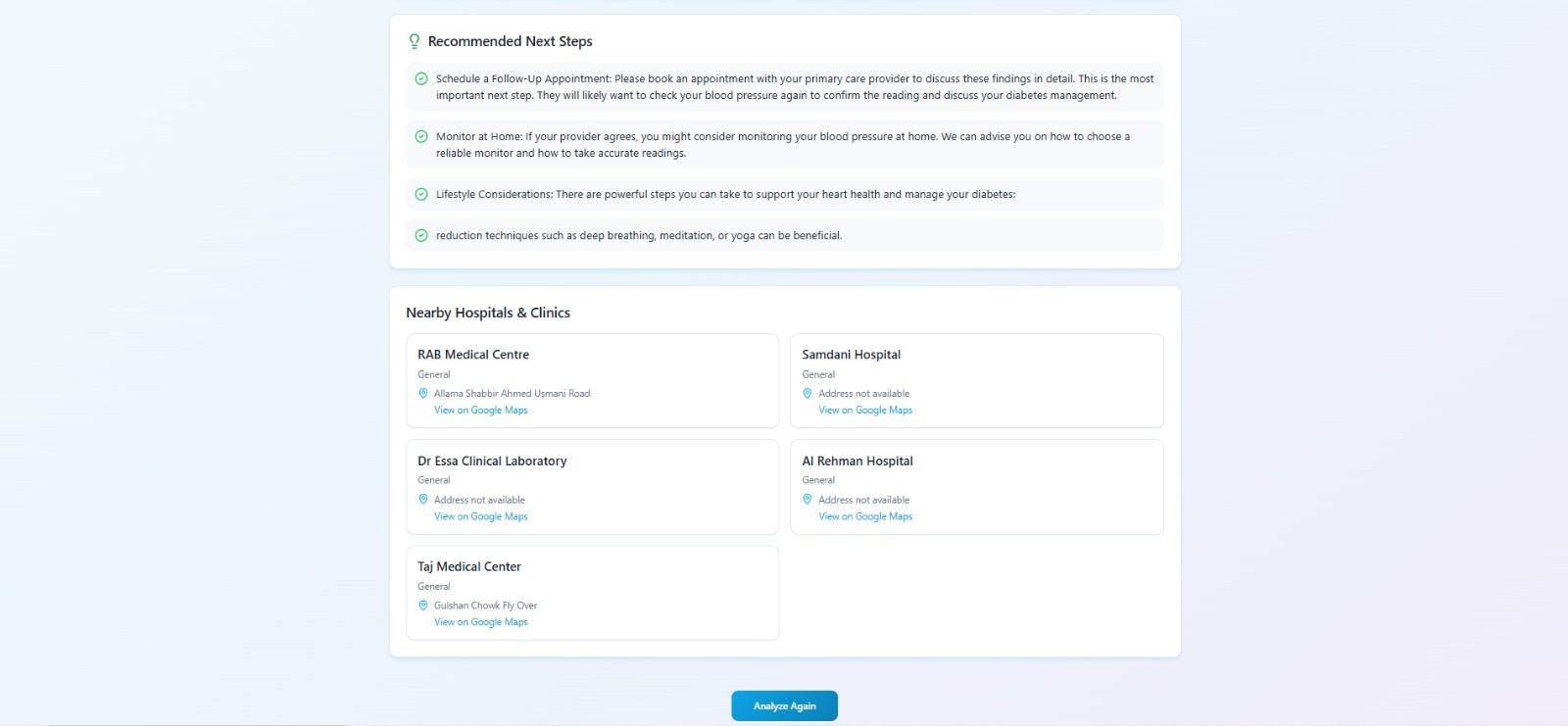
• Dashboard:



• Information Input: 

• Processing Page: 

• Results Page: 



# Data Files

The Data directory houses both raw and processed datasets in CSV format, serving as inputs and outputs throughout the ML pipeline.

## X\_train.csv & X\_test.csv

These files contain the feature matrices for model training and evaluation.  
• Generated by split\_data() in Src\_Code/data\_transformation.py.  
• Used as inputs by model\_training.py and model\_evaluation.py.  
• Columns correspond to encoded vitals and lifestyle factors (e.g., Gender, Age, BMI).

## y\_train.csv & y\_test.csv

These files store the target labels (Health risk categories) aligned with X\_train.csv and X\_test.csv.  
• Produced alongside feature splits by data\_transformation.py.  
• Loaded in model\_evaluation.py for reporting accuracy and per-class metrics.

## cleaned\_health\_data.csv

A cleaned version of the raw dataset, with missing values handled, unnecessary columns dropped, and outliers addressed.  
• Created by Src\_Code/data\_cleaning.py via DataCleaner methods.  
• Serves as input to data\_transformation.py.

## enhanced\_health\_data.csv

An enriched raw dataset containing engineered features (e.g., precomputed BMI) and original fields like patient coordinates and email.  
• Loaded by Src\_Code/data\_ingestion.py for quality checks.  
• Feeds into the cleaning pipeline.

## complete\_processed\_dataset.csv

A fully encoded dataset (all categorical → numeric) combining features and target.  
• Saved by save\_complete\_processed\_dataset() in data\_transformation.py.  
• Useful for external analyses or retraining.

## processed\_features.csv & processed\_target.csv

• processed\_features.csv: Contains numeric feature matrix after encoding Gender, Smoker, Diabetes.  
• processed\_target.csv: Column of raw target labels.  
• Produced by prepare\_features\_target() in data\_transformation.py.

# Source Code Modules

The Src\_Code folder implements the end-to-end pipeline: ingestion → cleaning → transformation → training → evaluation → RAG integration.

## data\_ingestion.py

Loads and inspects the raw enhanced dataset; reports missing values, data types, duplicates, and unique counts.  
• Functions: load\_and\_explore\_data(), check\_data\_quality()  
• Entry point: main()

## data\_cleaning.py

Encapsulates cleaning steps in DataCleaner:  
• remove\_unnecessary\_columns() (drops Name, redundant height/weight)  
• handle\_missing\_values() (fills numerics with mean, categoricals with mode)  
• standardize\_categorical\_data() (lowercases, strips whitespace)  
• detect\_and\_handle\_outliers() (clips extreme vitals and visualizes via boxplots)  
• validate\_data\_cleaning() returns final DataFrame.

## data\_transformation.py

Prepares data for modeling:  
1. load\_and\_preprocess\_data() reads cleaned CSV.  
2. save\_complete\_processed\_dataset() encodes all columns and writes full dataset.  
3. prepare\_features\_target() splits features/target and saves processed\_features.csv and processed\_target.csv.  
4. split\_data() stratifies and creates train/test CSVs.  
5. main() orchestrates the pipeline and returns splits plus feature names.

## model\_training.py

Defines ModelTrainer to train three classifiers:  
• Decision Tree (train\_decision\_tree())  
• Random Forest (train\_random\_forest())  
• CatBoost (train\_catboost())  
Also supports prediction (predict\_all()), evaluation per model (evaluate\_model()), and saving (save\_all\_models()).

## model\_evaluation.py

Implements ModelEvaluator for comprehensive assessment:  
• load\_model() and load\_test\_data()  
• comprehensive\_evaluation() computes accuracy, precision, recall, F1, confusion matrix, and classification report.  
• save\_metrics\_to\_json() writes a structured evaluation\_metrics.json.  
• print\_model\_comparison() outputs sorted accuracy table.

## rag\_integration.py

Builds or loads a Chroma vector database of health guidelines from WHO/CDC websites using LangChain.  
• init\_rag() fetches pages, embeds text, and prepares a retriever.  
• query\_rag() accepts vitals + predicted risk to return human-readable causes, diagnosis, and suggestions.

# Model Artifacts

Models folder contains serialized classifiers and evaluation metrics.  
• decision\_tree\_model.pkl: Pickled DecisionTreeClassifier.  
• random\_forest\_model.pkl: Pickled RandomForestClassifier.  
• catboost\_model.pkl: Pickled CatBoostClassifier (300 iterations, depth 8).  
• evaluation\_metrics.json: Per-model accuracy and ranking report.

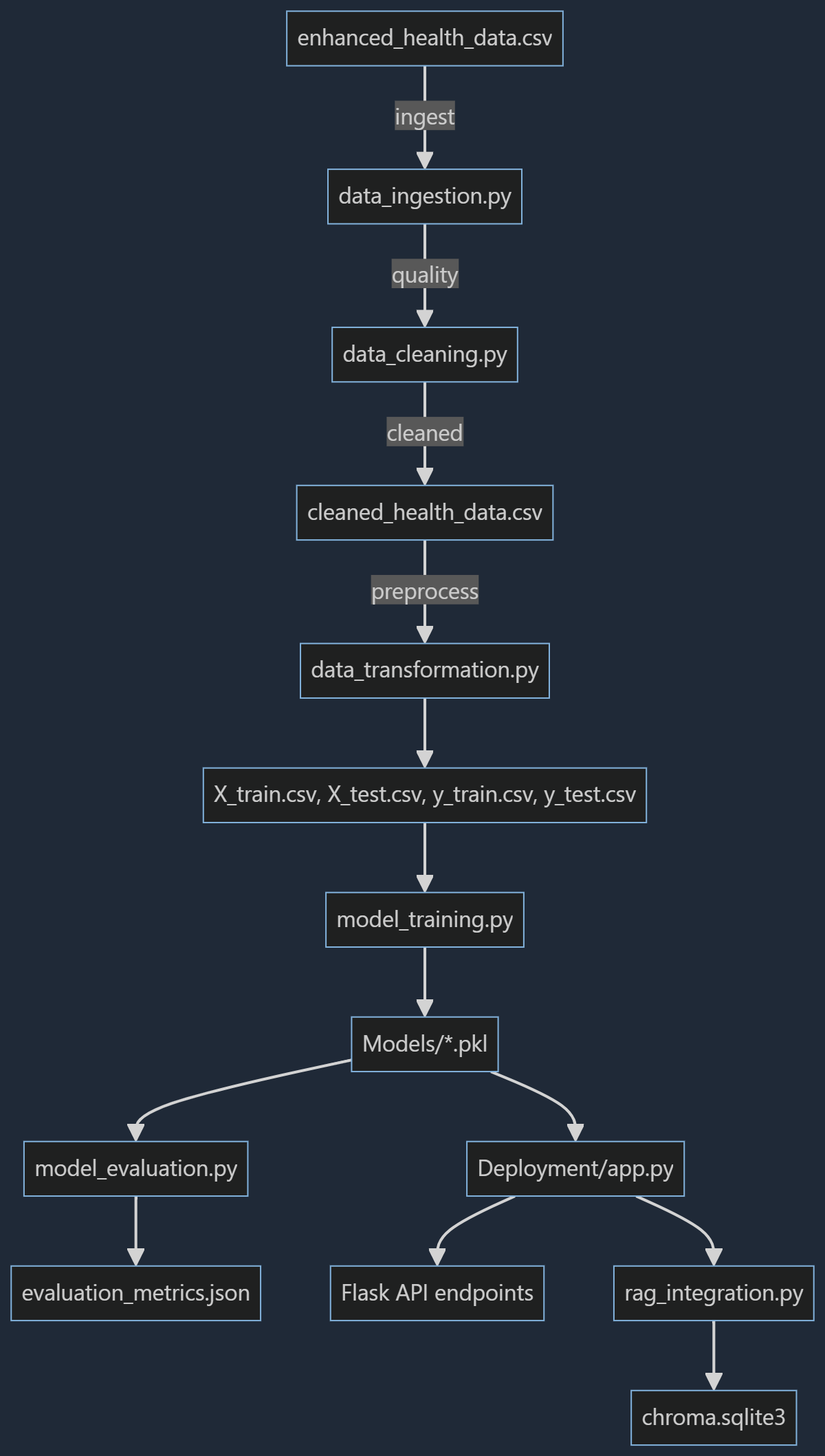
# Deployment Components

The Flask-based API enables deployment of the model as a RESTful service.  
Endpoints:  
• GET / : Health check.  
• POST /analyze : Accepts JSON input, predicts risk, triggers RAG explanations, fetches nearby doctors, and sends alert emails.

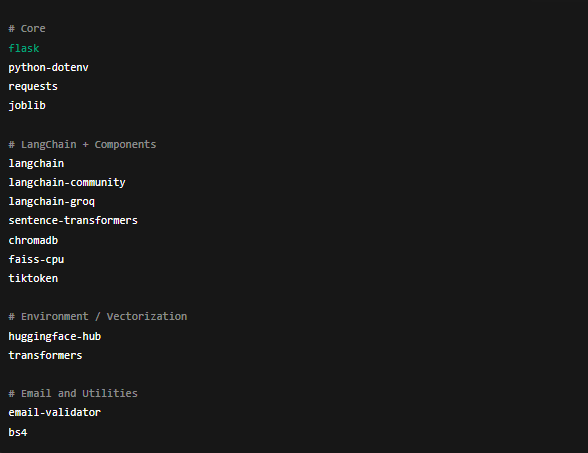
# Jupyter Notebook Artifacts

The notebook directory demonstrates exploratory analysis and interactive training.  
• eda.ipynb: Visual analysis of raw data.  
• model\_training\_evaluation.ipynb: Inline model training and comparison.

# ML Pipeline Overview



# Requirements



# Conclusion

The AI-Powered Smart Health Prediction System integrates machine learning and retrieval-augmented generation for interpretable, actionable health insights. It offers transparency, simplicity, and a base for future extensions such as fairness checks and calibration.