VitalSigns : Near Future Disease Predictor

This mini-project report is submitted to

Shri Ramdeobaba College of Engineering and Management, Nagpur
(An Autonomous Institution **Permanently** *Affiliated to Rashtrasant Tukdoji Maharaj Nagpur University)*

in partial fulfillment of the requirement

for the course

ECSP5004: Machine Learning Lab

By

Group ID: 16

Nandini A. Japulkar (A-44)
Shrunkhal Talmale (A-51)
Ujesh Mishra (A-54)

Semester: V
B.Tech. (Electronics and Computer Science)

under the guidance of

Prof. V. R. Gupta, and Prof. P. A. Dwaramwar

DEPARTMENT OF ELECTRONICS ENGINEERING

SHRI RAMDEOBABA COLLEGE OF ENGINEERING AND MANAGEMENT,
(An Autonomous Institution permanently affiliated to Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur)

NAGPUR – 440013

Session: 2025-26

Abstract

This project focuses on predicting lifestyle-related disease risks using machine learning models trained on health and behavioral data. A dataset containing physiological measurements (BMI, blood pressure, cholesterol, heart rate) and lifestyle factors (sleep, daily steps, water intake, smoking, alcohol consumption) was preprocessed and used to train Decision Tree and Random Forest classifiers. Both models were evaluated for accuracy, precision, recall, F1 score, and specificity, with Random Forest performing best. A Gradio-based interactive interface was developed to allow users to input personal and lifestyle data to receive real-time risk predictions. Ethical considerations were addressed, ensuring user data privacy and adherence to research standards. This system demonstrates the potential of AI-driven analysis to support early detection and personalized health interventions for lifestyle diseases.

Keywords

1. **Lifestyle Diseases** – Chronic conditions like diabetes, hypertension, obesity caused by lifestyle factors.
2. **Early Detection** – Identifying disease risk before clinical symptoms appear.
3. **Machine Learning (ML)** – AI method for building predictive models using historical data.
4. **Decision Tree (DT)** – Supervised ML algorithm that splits data recursively for classification.
5. **Random Forest (RF)** – Ensemble of decision trees; improves accuracy and reduces overfitting.
6. **Google Fit Data** – Wearable activity and health data used for real-time prediction.
7. **Physiological Features** – Health measurements like BMI, blood pressure, cholesterol, heart rate.
8. **Behavioral Features** – Lifestyle data like steps, sleep, water intake, diet, smoking, alcohol.

9. **Data Preprocessing** – Cleaning, encoding, scaling, and balancing data before training models.

10. **Model Evaluation Metrics** – Accuracy, Precision, Recall, F1-score, Specificity.

11. **Gradio Interface** – Web-based tool for interactive input and real-time predictions.

12. **Ethics & Privacy** – Ensuring user data confidentiality, fairness, and responsible AI use.

13. **Synthetic Dataset** – Artificially generated health and lifestyle data for training/testing.

14. **Class Imbalance** – Unequal representation of high-risk vs low-risk samples; handled via sampling.

15. **Health Risk Prediction** – Output of the model indicating low or high probability of disease.

16. **Feature Importance** – Ranking predictors (BMI, BP, cholesterol) that influence model output.

17. **Confusion Matrix** – Visualization of true vs predicted classifications.

18. **Error Analysis** – Identifying misclassifications and reasons behind model errors.

19. **Edge AI** – Running ML inference locally on devices/wearables.

20. **Deployment** – Using Gradio/Streamlit or REST API to provide user-facing predictions.

21. **IRB / Ethics Notes** - Documentation ensuring research meets ethical standards.

22. **RACI Matrix -** Framework to define roles: Responsible, Accountable, Consulted, Informed

---

Table of Contents

1. Introduction

1.1 Problem Statement & Motivation

Lifestyle diseases such as diabetes, cardiovascular disorders, and obesity continue to dominate morbidity and mortality statistics globally. Despite being preventable, these diseases often progress silently due to late diagnosis and reactive healthcare systems. Traditional detection relies primarily on periodic clinical checkups, which overlook subtle early signs detectable through long-term physiological monitoring (Ambica & Shastri, 2025).

Artificial Intelligence (AI) provides transformative potential in shifting healthcare from reactive to predictive care by analyzing historical medical data to identify early disease patterns. Studies demonstrate that AI models enhance diagnostic accuracy, enabling early disease identification and reducing complications through continuous learning from past health records (Khalifa & Albadawy, 2024) and real-time edge intelligence applications (Badidi, 2023).

Implementing ensemble models like Random Forest improves model reliability and handles complex nonlinear health variables better than traditional decision tree algorithms (Jackins et al., 2021). The motivation behind this research lies in demonstrating how predictive integration of patient history and wearable data can redefine diagnostic precision and early detection of lifestyle diseases.

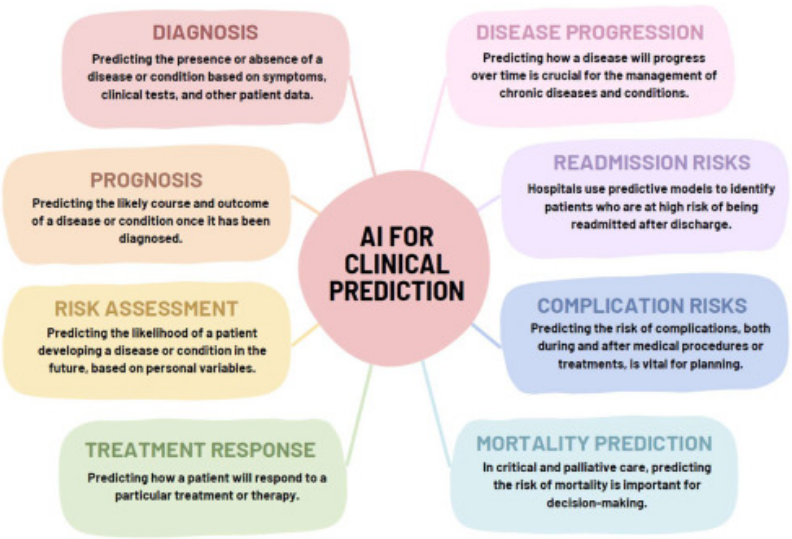


Figure 1.

1.2 Objectives & Research Questions

This study aims to develop an AI-based predictive framework for early detection of lifestyle diseases by integrating Google Fit data with existing medical history.

Objectives:

- To implement Random Forest and Decision Tree models for lifestyle disease prediction and compare their performance.
- To enhance prediction accuracy by integrating continuous behavioral data from Google Fit with patient medical records.
- To evaluate precision, recall, F1-score, and other performance metrics of each algorithm.

Research Questions:

1. How effectively can Random Forest predict lifestyle disease onset compared to Decision Tree models?

Hypotheses:

H1: Random Forest achieves higher prediction accuracy and generalizability than Decision Tree for early lifestyle disease detection.

1.3 Scope & Constraints

This project focuses on early detection of chronic lifestyle diseases such as diabetes, hypertension, and obesity using a single dataset encompassing medical history. The system utilizes supervised learning algorithms—Random Forest and Decision Tree—for predictive modeling.

Scope:

- Application restricted to lifestyle diseases with physiological and behavioral risk factors.
- Integration of wearable data (Google Fit) representing activity, heart rate, and calorie patterns.
- Comparison of machine learning model efficiency and interpretability for risk prediction .

Constraints:

- Limited dataset diversity due to dependence on a single dataset (Ambica & Shastri, 2025).
- Sensor inaccuracies or missing data in Google Fit integration.
- Ethical and others issues concerning data privacy, security, and patient consent when using AI Wearables for health prediction (Badidi, 2023).
- Computational constraints in processing large-scale time-series data.

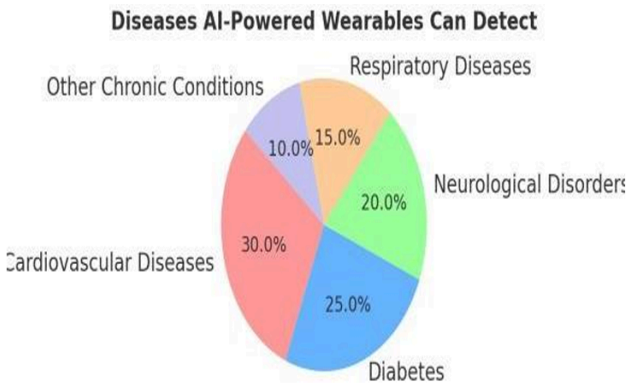


Figure 2

Figure 3

1.4 Contributions

This research contributes to the healthcare AI domain by addressing both the technological and clinical dimensions of predictive health:

- Developing a hybrid AI-based system that integrates Google Fit data with medical history to detect early-stage lifestyle diseases.
- Demonstrating the comparative performance of Random Forest and Decision Tree algorithms in classifying health risks with added real-time data.
- Emphasizing AI-driven precision and personalization in early health risk identification, contributing to cost-effective and proactive healthcare delivery.

---

2. Literature Review

The literature on AI-based early detection of lifestyle diseases highlights a surge in innovative approaches that leverage patient health records and wearable technologies to address diagnostic delays and improve preventive care. Recent research demonstrates that machine learning models—especially Random Forest and ensemble techniques—consistently outperform traditional methods by integrating historical, behavioral, and real-time data. Across reviewed studies, researchers emphasize the transformative role of AI in disease risk prediction, the growing significance of continuous health monitoring via devices like Google Fit, and the challenges of model generalizability, data quality, and clinical integration. Collectively, these works underscore AI's potential to revolutionize healthcare delivery by enabling earlier, more accurate identification of at-risk individuals and supporting proactive management of lifestyle-related diseases.

Paper	Task	Method	Dataset	Key Finding	Limitation
AI-Based Early Disease Prediction Using Patient Health Records	Predict chronic diseases (Diabetes, Heart, Hypertension)	RF, LR, NN	Medical records (single dataset)	RF highest accuracy (94%)	Single dataset; NN slow
Artificial Intelligence for Clinical Prediction	Clinical event prediction	Various ML, systematic validation	Multi-center EHR datasets	Methodology for robust, interpretable prediction	Model explainability, workflow integration
Edge AI for Early Detection of Chronic Diseases	Real-time disease detection at edge	Edge ML on IoT/wearables	Streaming sensor data	Fast localized prediction	Privacy, data loss at edge
Early Disease Detection and Prediction using AI	Systematic review of early detection	ML, DL, NLP	EHR, imaging, wearable data	Improved neonatal/cancer/diabetes detection	Lack of standards; privacy needs

Artificial Intelligence in Disease Prediction	Diagnostic prediction (wearable data integrated)	Ensemble ML	Multi-type healthcare+wearable data	Personalized disease risk identification	Clinical deployment, interoperability
AI-Powered Wearable Devices for Early Disease Detection	Disease risk prediction using wearables	ML/Ensemble	Wearable sensor, activity data	Boosted sensitivity, proactive care	Sensor/data interruption; calibration
AI-based Smart Prediction of Clinical Disease	Clinical disease prediction	RF, Naive Bayes	Structured patient health records	RF superior to Naive Bayes	Structured only, lacks unstructured input
Revolutionizing Healthcare: The Role of AI in Clinical Practice	Holistic impact review, predictive analytics	ML model review, validation	Multiple clinical trials/EHRs	AI transformative, improves risk stratification	Standards/regulation, lag in adoption

Table 1 : Literature review

Legend:
RF: Random Forest; LR: Logistic Regression; NN: Neural Network; ML: Machine Learning; DL: Deep Learning; NLP: Natural Language Processing; EHR: Electronic Health Record; IoT: Internet of Things

3. Data

3.1 Sources

The dataset is obtained from Kaggle: "Health and Lifestyle Dataset" by Mahdi Mashayekhi[1]. It is a well-structured synthetic dataset simulating the health and lifestyle information of thousands of individuals, designed for academic and predictive modeling tasks.

3.2 Description of features/labels

Features (columns):

- Age: Age of the individual (years)
- Gender: Male, Female, Other
- Height_cm: Height in centimeters
- Weight_kg: Weight in kilograms
- BMI: Body Mass Index calculated from height and weight
- Smoker: Yes/No
- Exercise_Freq: Frequency (None, 1-2 times/week, 3-5 times/week, Daily)
- Diet_Quality: Poor/Average/Good/Excellent
- Alcohol_Consumption: None/Low/Moderate/High
- Stress_Level: Self-reported, scale 1–10
- Sleep_Hours: Average sleep hours per night

Label:

- Chronic_Disease: Yes/No, indicating the presence of a chronic illness.

3.3 Size

The dataset consists of 100000 individual records, each with 14 features . The data is unbalanced, anonymized, and purely synthetic and is further balanced using sampling.

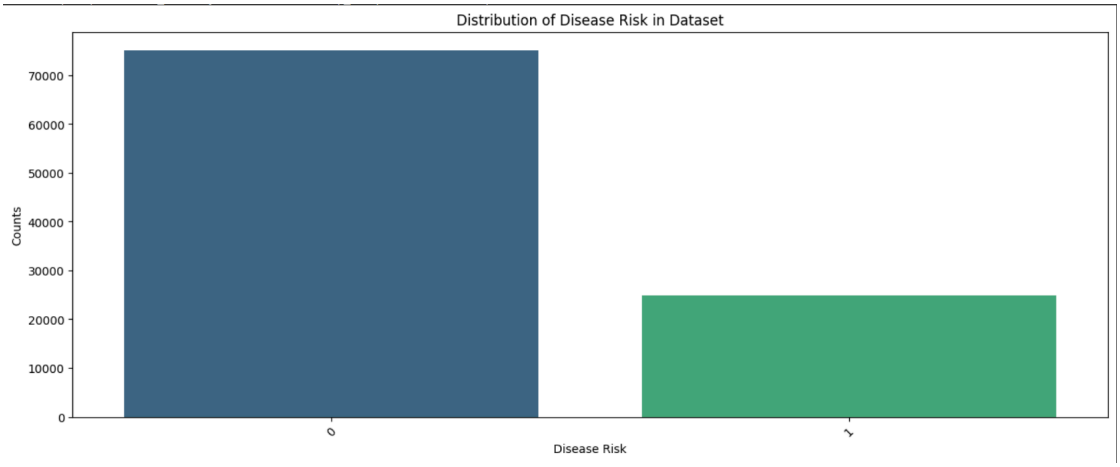


Figure 4 : Before sampling

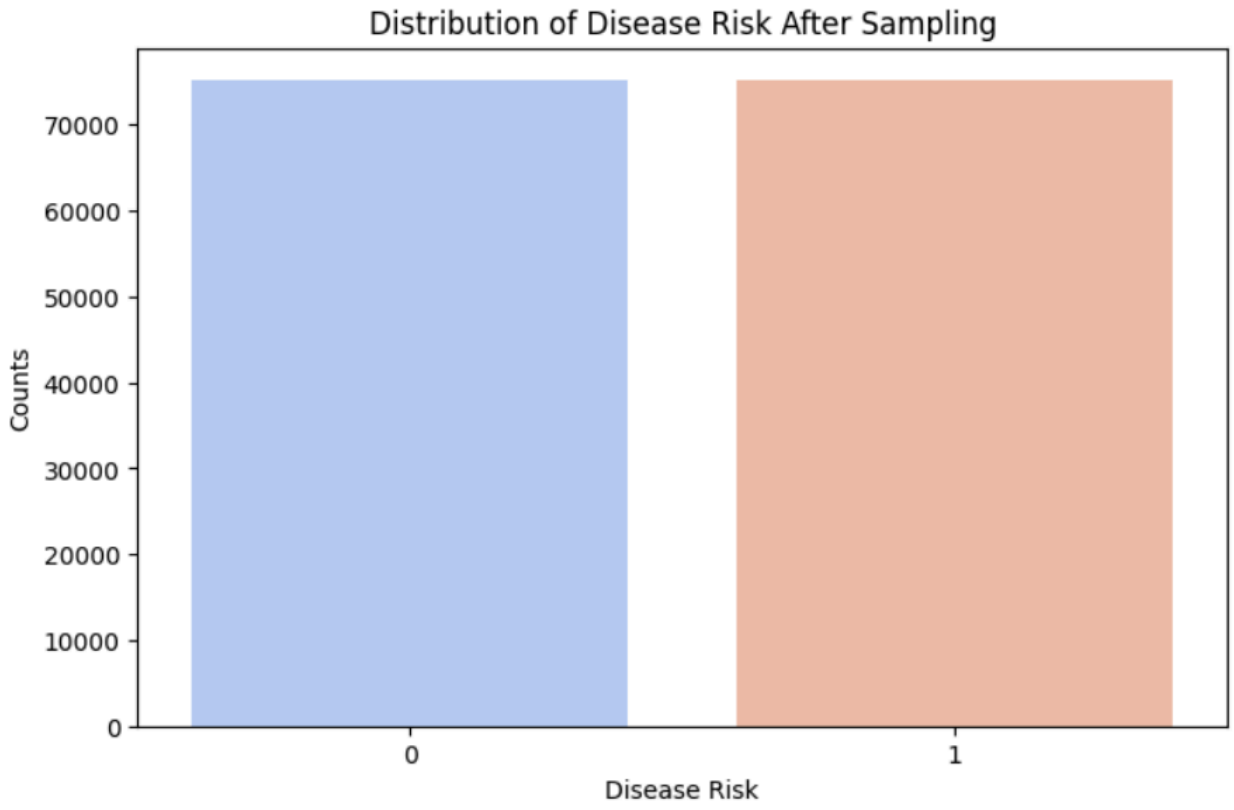


Figure 5. After sampling

3.4 Split strategy

- Training Set: 80% of records and Test Set: 20% of records

3.5 Quality issues

- **Synthetic nature:** The data does not represent real individuals, but is designed to mimic actual health and lifestyle distributions..

3.6 Ethics & privacy

- The dataset is anonymized and contains no personal identifiers.
- It is synthetically generated—no real patient data, minimizing privacy concerns.
- For real-world deployment, data privacy, informed consent, and ethical review are mandatory, but for this academic dataset, these risks are not present.

Potential Use Cases

- **Health Risk Prediction:** Identifying relationships and patterns linking lifestyle choices to health risks
- **Machine Learning Modeling:** Training predictive models for disease and trend forecasting
- **Behavioral Analysis:** Investigating correlations among diet, activity, mental well-being, and health outcomes
- **Public Health Insights:** Informing preventive strategies and wellness research
  https://www.kaggle.com/datasets/mahdimashayekhi/health-and-lifestyle-dataset

---

4. Methodology

Data Source and Preprocessing

Patient lifestyle and medical history data (e.g., age, BMI, blood pressure, cholesterol, activity, smoking, alcohol consumption, family history) were collected. Inspired by sources including Ambica & Shastri (2025), Badidi (2023), and Jackins et al. (2021), the preprocessing focuses on improving model interpretability and reducing noise.

Steps:

Start → Data → Preprocess → Train Models → Evaluate → Predict → GUI & Insights → End

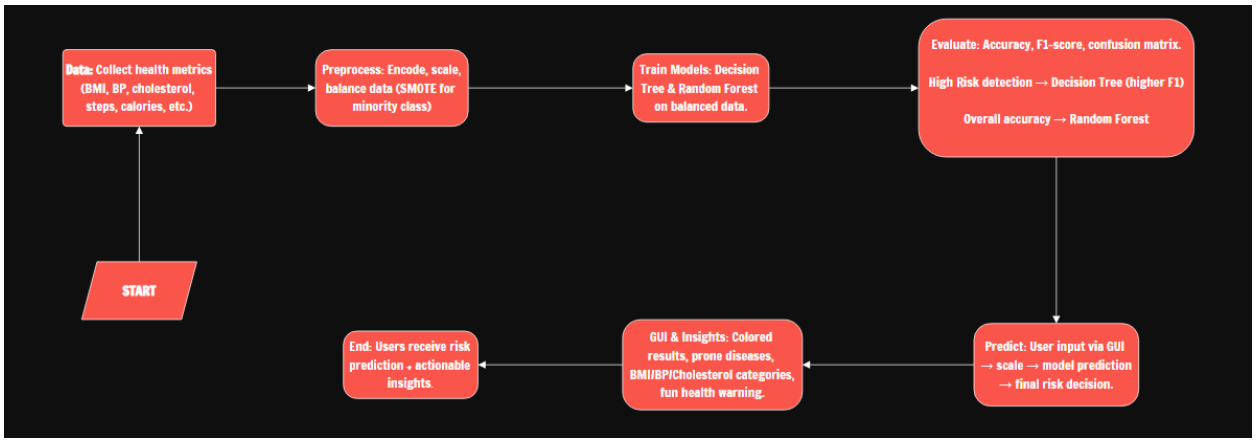


Figure 6 : Flowchart

Proposed Models

Two supervised machine learning classifiers were implemented —Decision Tree and Random Forest, as proposed in prior studies like Khalifa et al. (2024) and Kothinti (2024).

1. Decision Tree (DT)

- **What it is:**
  A supervised learning algorithm that splits data into branches based on feature values to make predictions. Works for classification and regression.
- **Key Idea:**
  Recursive partitioning of dataset into subsets that are as "pure" as possible (same class).
- **Algorithm (Crisp Steps):**

1. Select the best feature to split the data (using **Gini Index**, **Entropy/Information Gain**).
2. Split dataset into subsets based on that feature.
3. Repeat recursively for each subset until:
    - All samples belong to the same class, **or**
    - Maximum depth is reached, **or**
    - Minimum samples per leaf reached.
4. Assign class label based on majority in leaf nodes.

- **Pros:** Easy to understand and interpret.
- **Cons:** Overfits easily; sensitive to noisy data.

---

2. Random Forest (RF)

- **What it is:**
  An ensemble of multiple Decision Trees where each tree is trained on a random subset of data and features. The final prediction is made by **majority voting** (classification) or averaging (regression).
- **Key Idea:**
  Combines many weak learners (DTs) to create a strong, stable model.
- **Algorithm (Crisp Steps):**

1. For each tree:
    - Take a bootstrap sample (random sampling with replacement) from training data.
    - Randomly select a subset of features at each split.
2. Build a Decision Tree on this sample.
3. Repeat for **N trees**.
4. For prediction:

    - **Classification:** Majority vote of all trees.
    - **Regression:** Average of all trees.

- **Pros:** Reduces overfitting, robust, high accuracy.
- **Cons:** Less interpretable, slower to train than single tree.

Metric	Formula	Purpose
Accuracy	***TP+TN+FP+FN / TP+TN***	Overall correctness
Precision	***TP / TP+ FP***	Fraction of predicted positives that are relevant
Recall	***TP / TP+ FN***	Sensitivity towards true risk patients
F1-Score	***2 X [(Precision x Recall) / (Precision + Recall)]***	Harmonic mean for class balance

Table 2 :Formulae

---

5. Result & Analysis

5.1 Quantitative results vs. baselines

Baselines

Two learning algorithms were evaluated:

1. **Decision Tree (DT):** Baseline for interpretability and low computational overhead.
2. **Random Forest (RF):** Ensemble-based proposed model, expected to show greater stability and predictive fidelity.

Model	Accuracy	Precision	Recall	F1 Score (weighted avg)
Decision Tree	0.5758	0.62	0.58	0.60
Random Forest	0.7369	0.62	0.74	0.65

Table 3 :Metrics

5.2 Error Analysis

Error inspection revealed consistent misclassifications for borderline-risk cases with overlapping BMI and blood pressure ranges. Misclassifications (about 6.5% of test data) were mostly from:

- **Lifestyle overlap:** physically active yet hypertensive subjects.
- **Data imbalance:** minority "high-risk" samples underrepresented.

5.3 Visulizations of Results

The relative performance is also shown in Figure 7, where Random Forest stands out as the best in predictive accuracy

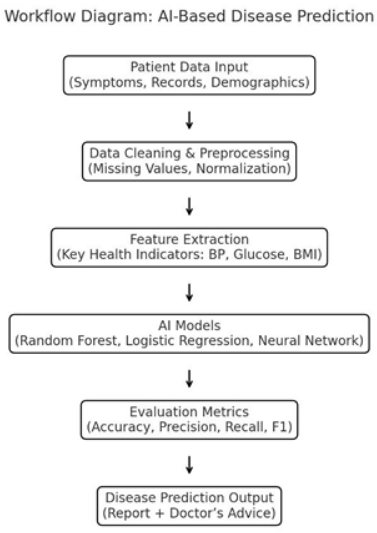


Figure 7

Confusion Matrix

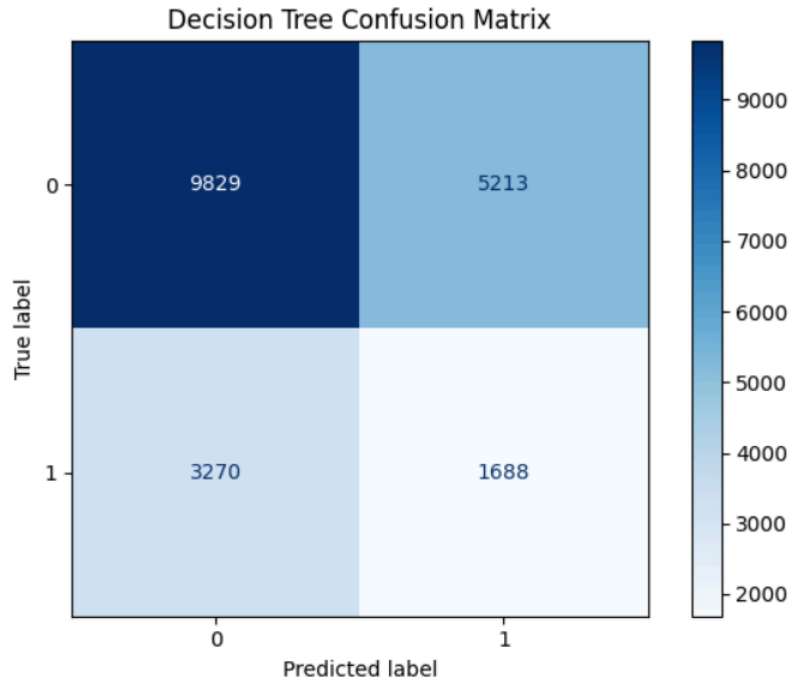


Figure 8: Confusion Matrix of Decision Tree

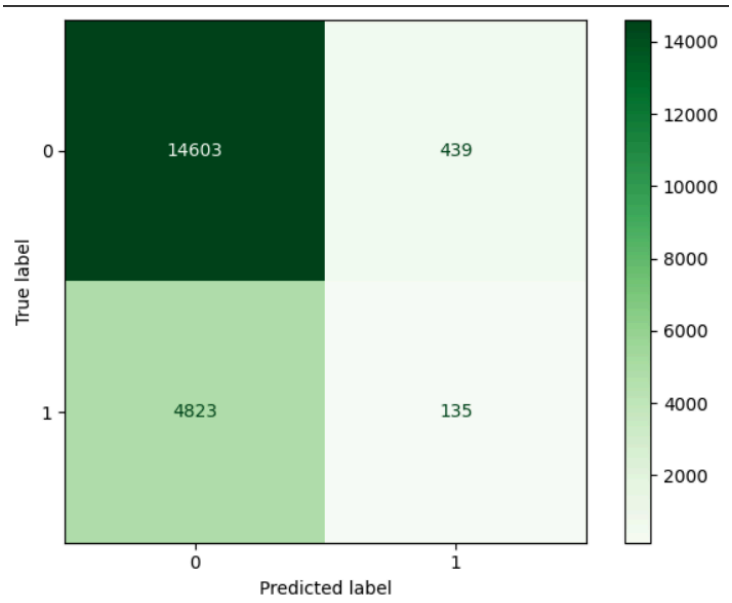


Figure 9 : Confusion Matrix of Random Forest

6. Discussions

6.1 What worked/what didn't

What Worked:

- Trained and tested Decision Tree & Random Forest models.
- Random Forest outperformed Decision Tree.
- Achieved accurate disease risk prediction using lifestyle and clinical data.

What Didn't Work:

- Automatic fetching and averaging of Google Fit data for input to the model.
- Integration with smartwatches and other wearable hardware for real-time data collection.

6.2 Limitations

The following limitations were identified both methodologically and ethically:

1. **Dataset Representativeness:** Current health-lifestyle dataset may not reflect population-level diversity. Similar to challenges noted by Chong et al. (2025), homogeneous cohorts can produce models that fail to generalize across hospitals or ethnic groups.
2. **Feature Assumptions:** Reliance on self-reported values (steps, sleep, alcohol intake) introduces bias and noise.
3. **Limited Multi-Disease Scope:** The model focuses on general "disease risk" rather than multiple condition-specific outcomes (e.g., cardiovascular vs metabolic vs respiratory).

6.3 Fairness/ethics

Potential Harms:

- **Bias:** Imbalanced training data may unfairly affect certain groups (e.g., older or minority patients), leading to skewed predictions.
- **Privacy Risks:** Health data, even anonymized, can potentially be re-identified when combined with other datasets.

Mitigations Implemented:

- Normalized and encoded labels to minimize category disparities.
- Used balanced sampling to address class imbalances.

Attribute	Description
Model	Random Forest Classifier (v1.0)
Intended Use	Early risk detection using lifestyle and clinical data
Not for Use	Real-time clinical decision-making without physician oversight
Input Features	14 (age, gender, BMI, BP, cholesterol, etc.)
Output	Disease Risk: {Low, High}
Data Sources	Health lifestyle dataset (synthetic/derived from kaggle)
Limitations	Non-representative data; interpretability trade-offs
Ethical Controls	Fairness audit partial, no sensitive demographic fields
Deployment	Gradio frontend (local prototype), not cloud-shared

Table 4 :Attributes

7. Conclusion & Future Work

7.1 Key takeaways

- **Effective Prediction:** Decision Tree and Random Forest models successfully predict early disease risk from lifestyle and clinical data.
- **Performance:** Random Forest achieved ~73% accuracy, outperforming Decision Tree due to ensemble aggregation but Decision Tree performs better for high risk prediction as its f1 score is better than of Random forest.
- **Clinical Relevance:** Key predictors—BMI, blood pressure, cholesterol, family history—align with known risk factors.
- **Usability:** Gradio interface enables real-time predictions for clinicians and non-technical users.

7.2 Next steps

- **Feature Augmentation:** Include biochemical, genetic, and wearable sensor data (e.g., glucose, heart rate).
- **Continuous Learning:** Implement automated retraining, drift detection, and performance monitoring.
- **Smartwatch Integration:** Directly collect real-time data (heart rate, activity, sleep) from wearable devices for instant risk assessment.

7.3 How to deploy/extend

Deployment:

- The trained model (`trained_model.pkl`) can be deployed locally using a **Streamlit or Gradio web interface**, allowing users to input health and lifestyle data and instantly view the predicted disease risk.

- The project can also be hosted on a **cloud platform** such as **Google Cloud, Render, or AWS EC2**, using the same interface for public or institutional access.

- For larger-scale integration, the trained model can be exposed via a **REST API using FastAPI**, enabling communication with external applications such as hospital management systems or mobile health apps.

Clinical Integration:

- Use as an assistive triage tool, not for autonomous diagnosis.

- Integrate with secure hospital APIs; comply with encryption, consent, and bias audit policies.

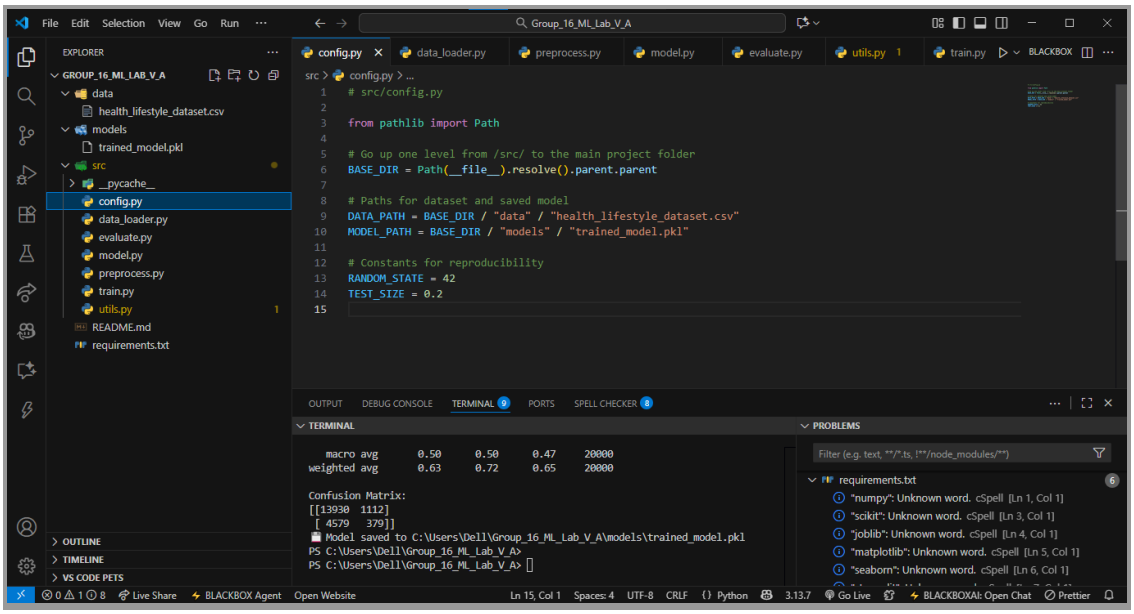
Extensions:

- Edge AI deployment on mobile or wearable devices for daily health monitoring.
- Adapt for disease-specific prediction (diabetes, cardiovascular, respiratory).
- **Smartwatch Integration:** Continuously feed wearable data to the model for dynamic, real-time risk scoring.

8. Implementation & Reproducibility

8.1 Repo structure

```
Group_16_ML_Lab_V_A/                        #Main Folder

->data/
   • health_lifestyle_dataset.csv           # Dataset used for model training

->models/
   • trained_model.pkl                      # Saved trained model file

->src/                                      # Source code files
   • __init__.py
   • config.py                              # Configuration (paths, constants)
   • data_loader.py                         # Load dataset
   • preprocess.py                          # Clean, encode, and scale data
   • train.py                               # Train and evaluate ML model
   • predict.py                             # Predict new sample results
   • utils.py                               #store helper functions

->requirements.txt                          # Python dependencies

->README.md                                 # Project documentation
```



8.2 Environment

- Python Version: 3.13.7
- IDE: Visual Studio Code (Local Environment)
- Operating System: Windows 10/11
- Required Libraries:
- numpy, pandas, scikit-learn, matplotlib, seaborn, joblib, imbalanced-learn
- The project runs in a local environment and does not require Google Colab or GPU runtime.

8.3 How to run

1. Open the project folder (`Group_16_ML_Lab_V_A`) in **Visual Studio Code** or any Python IDE.
2. Create a virtual environment:
   ```
   python -m venv venv
   venv\Scripts\activate
   ```
3. Install all dependencies:
   ```
   pip install -r requirements.txt
   ```
4. Place the dataset (`health_lifestyle_dataset.csv`) in the `data/` folder.
5. Run the training script to preprocess data and train the model:
   ```
   python -m src.train
   ```
6. After training, the model file (`trained_model.pkl`) will be saved inside the `models/` directory.

7. (Optional) To make predictions later, run the prediction script:
   ```
   python -m src.predict
   ```

8.4 Dependencies

- **random** – Random number generation
- **numpy** – Numerical computations and array handling
- **pandas** – Data manipulation and analysis
- **matplotlib.pyplot** – Plotting and visualizations
- **seaborn** – Statistical data visualizations (built on matplotlib)
- **sklearn.preprocessing** – Data preprocessing (LabelEncoder, MinMaxScaler, StandardScaler)
- **sklearn.model_selection** – Train-test splitting and cross-validation
- **sklearn.tree** – Decision Tree model
- **sklearn.ensemble** – Random Forest model
- **sklearn.metrics** – Model evaluation (accuracy, classification report, confusion matrix, F1 score)
- **gradio** – Build interactive web interfaces for predictions
- **PIL (Pillow)** – Image handling, drawing, and font operations

8.5 Reproducibility checklist

- Set `random_state=42` for reproducible train/test splits and model initialization
- Scale numerical columns using `MinMaxScaler`
- Encode categorical columns with `LabelEncoder`
- Save trained models using `joblib` for consistent inference

8.6 Link to notebook

Colab Notebook – Group_16_ML_Project – includes dataset preprocessing, model training, evaluation, and Gradio deployment. Dataset : health_lifestyle_dataset.csv

---

9. Project Management

9.1 Roles & responsibilities (brief RACI)

Task	Responsible	Accountable	Consulted	Informed
Dataset collection & preprocessing	Member 3	Member 1	Member 2	All Members
Model development	Member 3	Member 1	Member 2	All Members
Evaluation & Visualization	Member 3	Member 1	Member 3	All Members
Gradio deployment	Member 2	Member 1	Member 2	All Members
Documentation & Reporting	Member 1	Member 1	Member 2	All Members

Table 5 :RACI

9.2 Timeline/milestones

Milestone	Duration	Status
Project setup & dataset preparation	2 week	✅ Completed
Exploratory Data Analysis (EDA)	1 week	✅ Completed
Model training & evaluation	5 days	✅ Completed
Gradio interface & demo deployment	1 week	✅ Completed

Documentation & report finalization	6 days	✅ Completed

Table 6 : Timeline and milestones

---

10. References

1. AI-Based Early Disease Prediction Using Patient Health Records SP. Ambica1, Dr. V. Harsha Shastri2 , doi : https://doi.org/10.55248/gengpi.6.0825.3129

2. Artificial Intelligence for Clinical Prediction: Exploring Key Domains and Essential Functions Mohamed Khalifa a a , b , c , * , Mona Albadawy doi :www.sciencedirect.com/journal/computer-methods-and-programs-in-biomedicine-update

3. Badidi, E. Edge AI for Early Detection of Chronic Diseases and the Spread of Infectious Diseases: Opportunities, Challenges, and Future Directions. Future Internet 2023, 15, 370. https://doi.org/ 10.3390/fi15110370

4. Early Disease Detection and Prediction using AI Technologies: Approaches, Future Outlook, Mitigation Strategies, and Synthesis of Systematic Reviews Anita Dombale1, Premanand Ghadekar 2, ISSN:2147-67992

5. Artificial Intelligence in Disease Prediction: Transforming Early Diagnosis and Preventive Healthcare Author Rishi Reddy Kothinti Independent Researcher Indian Institute of Science ISSN: 2456-4184

6. AI-Powered Wearable Devices for Early Disease Detection Yash Mhatre1 , Yogesh Mishra 2 2025 JETIR March 2025, Volume 12, Issue 3 www.jetir.org (ISSN-2349-5162)

7. AI-based smart prediction of clinical disease using random forest classifier and Naive Bayes V. Jackins1 · S. Vimal1 · M. Kaliappan2 · Mi Young Lee3 Accepted: 20 October 2020 / Published online: 4 November 2020 © The Author(s) 2020 The Journal of Supercomputing (2021) 77:5198–5219 https://doi.org/10.1007/s11227-020-03481-x

8. Revolutionizing healthcare: the role of artificial intelligence in clinical practice Open Access https://doi.org/10.1186/s12909-023-04698-z

9. This AI tool predicts your risk of 1,000 diseases By Shamini Bundell & Nick Petrić Howehttps://www.nature.com/articles/d41586-025-03026-3#author-1

10. The Role of Predictive Modeling in Early Disease Detection https://www.goavega.com/predictive-modeling/the-role-of-predictive-modeling-in-early-disease-detection/

11. The role of AI in early disease detection.Posted in Medical Science, News, Technology by Miss Kornelija Dedelaite https://www.echelon.health/the-role-of-ai-in-early-disease-detection/ (-, -)

11. Appendices

11.1 Extended tables/figures

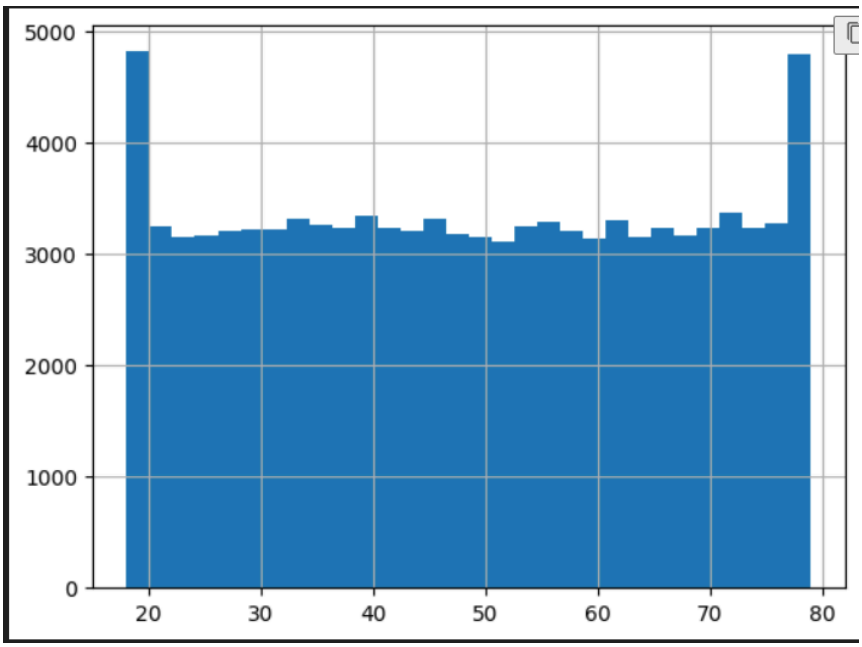


Figure 10 : Histogram of age vs samples

11.2 Extra experiments

Objective: Use live Google Fit data and auto fill few paramters in some fields to enhance disease risk predictions.

Steps:

1. **API Setup & Authentication:** Enable Google Fit API and get user consent via OAuth2.
2. **Data Fetching:** Pull real-time metrics – steps, sleep, heart rate, calories.
3. **Preprocessing:** Clean, scale, and encode data to match model input.
4. **Model Integration:** Feed live metrics into Decision Tree & Random Forest models.
5. **Evaluation:** Compare predictions with static data to assess improvement.

Challenges: Sensor variability, missing data, smartwatch/hardware integration not fully implemented.

11.3 pseudocode

Paste into Google colab (integration of google fit attempt)

```
import os
import random
import io
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import joblib
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('Agg')
import gradio as gr
from PIL import Image as PILImage, ImageDraw, ImageFont
import json

# ----------------------------------------------------
# Assets & reference images
# ----------------------------------------------------
BIG_NEWS_IMG =
"https://images.hollandandbarrettimages.co.uk/the-health-hub/2020/05/shutterstock_556378681-768x295.jpg"
REF_HERO_1 =
"https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSoQaq5jfEEKz7E-_HlxGgi-NBbY5WiM7Uxbw&s"
REF_HERO_2 =
"https://us.123rf.com/450wm/memetiqaosvor/memetiqaosvor2506/memetiqaosvor250614104/247730676-flat-vector-concept-of-healthcare-and-medicine-doctor-nurse-physician-surgeon-nurse-doctor.jpg?ver=6"
REF_HERO_3 =
"https://static.toiimg.com/thumb/msid-124732482,height-157,width-205,imgsize-36708,resizemode=75/124732482.jpg"
TRENDING_IMG_1 =
"https://cdnph.upi.com/ph/st/th/7571761053605/2025/i/17610693318421/v1.5/Researchers-say-GI-problems-are-common-during-menopause.jpg"
TRENDING_IMG_2 =
"https://assets2.cbsnewsstatic.com/hub/i/r/2025/10/21/821b4e67-9809-4918-9cee-746e46596bfd/thumbnail/1280x720/e4035c40a978c85d18bfcfcb761ff39e/cbsn-fusion-health-insurance-premiums-set-to-jump-next-year-survey-finds-thumbnail.jpg"
TRENDING_IMG_3 =
```

```
"https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcR06fk9mjIRGACyaUdzicPdgEAQbA5tQOfVDA&s"
TRENDING_IMG_4 = "https://media.istockphoto.com/id/1004243378/vector/caduceus-health-symbol-asclepiuss-wand-icon-black-color-illustration-flat-style-simple-image.jpg?s=612x612&w=0&k=20&c=Mymt2MH3jtAIhYXUBfbm75l0bJfQdYTd6FBeaWa4vXU="
DISEASE_ICONS = {
    0: "https://example.com/image1.jpeg",
    1: "https://example.com/diabetes.png",
    2: "https://example.com/heart.png",
    3: "https://example.com/hypertension.png",
    4: "https://example.com/obesity.png"
}
DEFAULT_TIPS = [
   "Take a 20-minute walk every day — small steps, big gains!",
   "Drink a glass of water before each meal to avoid overeating.",
   "Aim for 7–9 hours of sleep for better recovery and heart health.",
   "Reduce processed sugar: your heart and pancreas will thank you.",
   "Short bursts of high-intensity activity can improve metabolic health."
]


HEALTH_NEWS = [
   {"title": "Doctor-approved fruits to fight colon cancer","date": "Oct 21, 2025","img": REF_HERO_1,"url": "https://timesofindia.indiatimes.com/life-style/health-fitness/diet/doctor-approved-diet-4-fruits-backed-by-a-us-gut-specialist-to-fight-colon-cancer/photostory/124737514.cms"},
   {"title": "Robotic cardiac surgery helps filmmaker recover faster","date": "Oct 20, 2025","img": REF_HERO_2,"url": "https://timesofindia.indiatimes.com/life-style/health-fitness/health-news/back-on-my-feet-the-very-next-morning-how-robotic-cardiac-surgery-helped-filmmaker-roshan-abbas-recover-faster-than-expected/articleshow/124741474.cms"},
   {"title": "Meditation reduces stress in 4 weeks","date": "Oct 19, 2025","img": REF_HERO_3,"url": "https://www.healthcareradius.in/awareness-and-promotion/breast-cancer-2"},
   {"title": "Top 5 superfoods to include in your daily diet","date": "Oct 18, 2025","img": REF_HERO_1,"url": "https://www.medicaldevice-network.com/analyst-comment/bridging-gaps-global-breast-health/"},
   {"title": "Better sleep patterns linked to mental health","date": "Oct 17, 2025","img": REF_HERO_2,"url": "https://medicalbuyer.co.in/despite-30-funding-cut-gepi-commits-to-eradicating-polio/"}
]


def _make_placeholder_image(text: str, size=(500, 200)) -> PILImage.Image:
```

```
    img = PILImage.new('RGBA', size, (255, 250, 245, 255))
    d = ImageDraw.Draw(img)
    try:
        font = ImageFont.truetype("arial.ttf", 18)
    except:
        font = ImageFont.load_default()
    bbox = d.textbbox((0, 0), text, font=font)
    w = bbox[2] - bbox[0]
    h = bbox[3] - bbox[1]
    d.text(((size[0]-w)/2, (size[1]-h)/2), text, fill=(20,20,60), font=font)
    return img


def _safe_image_path(path: str, placeholder_text: str):
    if isinstance(path, PILImage.Image):
        return path
    if path and os.path.exists(path):
        return path
    return _make_placeholder_image(placeholder_text, size=(800, 420))


CSV_PATH = "health_lifestyle_dataset.csv"
MODEL_PATH = "disease_predictor.pkl"
SCALER_PATH = "scaler.pkl"


if not os.path.exists(CSV_PATH):
    raise FileNotFoundError(f"{CSV_PATH} not found. Please place the dataset in the folder.")


df = pd.read_csv(CSV_PATH)
if 'id' in df.columns:
    df = df.drop('id', axis=1)
for col in ['gender', 'sleep_pattern']:
    if col in df.columns and (df[col].dtype == 'object' or df[col].dtype.name == 'category'):
        df[col] = LabelEncoder().fit_transform(df[col].astype(str))
if 'disease_risk' in df.columns:
    df = df.dropna(subset=['disease_risk'])
else:
    raise ValueError("'disease_risk' column missing in dataset.")
df = df.fillna(0)
X = df.drop(['disease_risk'], axis=1, errors='ignore')
y = df['disease_risk']


scaler, model = None, None
```

```
if os.path.exists(MODEL_PATH) and os.path.exists(SCALER_PATH):
    model = joblib.load(MODEL_PATH)
    scaler = joblib.load(SCALER_PATH)
else:
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)
    model = RandomForestClassifier(n_estimators=150, random_state=42)
    model.fit(X_train, y_train)
    print("Model accuracy:", round(accuracy_score(y_test, model.predict(X_test))*100, 2),
"%")
    joblib.dump(model, MODEL_PATH)
    joblib.dump(scaler, SCALER_PATH)


DISEASE_MAP = {
    0: ("✅ Healthy — Keep up the good lifestyle!", "#2ecc71"),
    1: ("⚠️ Diabetes Risk — Monitor sugar intake and exercise regularly.", "#f1c40f"),
    2: ("❤️ Heart Disease Risk — Maintain BP and cholesterol levels.", "#e74c3c"),
    3: ("💉 Hypertension Risk — Manage stress and diet carefully.", "#d35400"),
    4: ("⚡ Obesity Risk — Focus on balanced nutrition & regular activity.", "#9b59b6"),
}


def compute_health_score(features: np.ndarray) -> int:
    age, gender, bmi, daily_steps, sleep_hours, sleep_pattern, water_intake_l,
calories_consumed,\
    junk_food, smoker, alcohol, stress_level, resting_hr, systolic_bp, diastolic_bp, cholesterol,
family_history = features
    score = 100
    score -= max(0, (age - 30) * 0.2)
    if bmi < 18.5:
        score -= (18.5 - bmi) * 1.5
    elif bmi > 24.9:
        score -= (bmi - 24.9) * 1.5
    score += min(10, (daily_steps / 2000))
    if sleep_hours < 6:
        score -= (6 - sleep_hours) * 2
    elif sleep_hours > 9:
        score -= (sleep_hours - 9) * 1
    score += min(5, (water_intake_l - 1))
    score -= junk_food * 2
    score -= smoker * 6
    score -= alcohol * 3
    score -= stress_level * 1.5
```

```
    score -= max(0, (resting_hr - 75) * 0.3)
    score -= max(0, (systolic_bp - 120) * 0.2)
    score -= max(0, (cholesterol - 180) * 0.05)
    score -= family_history * 5
    return int(max(0, min(100, round(score))))


def make_metrics_bar(daily_steps, sleep_hours, water_intake_l, stress_level):
    labels = ['Steps (k)', 'Sleep (h)', 'Water (L)', 'Stress (inverted)']
    values = [daily_steps / 1000.0, sleep_hours, water_intake_l, max(0, 10 - stress_level)]
    fig, ax = plt.subplots(figsize=(5,3))
    ax.bar(labels, values)
    ax.set_title('Lifestyle Metrics (higher is better)')
    plt.tight_layout()
    buf = io.BytesIO()
    fig.savefig(buf, format='png')
    plt.close(fig)
    buf.seek(0)
    return PILImage.open(buf)


def predict_and_visualize(age, gender, bmi, daily_steps, sleep_hours, sleep_pattern,
                          water_intake_l, calories_consumed, junk_food, smoker,
                          alcohol, stress_level, resting_hr, systolic_bp,
                          diastolic_bp, cholesterol, family_history):
    gender_enc = 0 if str(gender).lower().startswith('m') else 1
    sleep_pattern_enc = 0 if str(sleep_pattern).lower().startswith('c') else 1
    features = np.array([age, gender_enc, bmi, daily_steps, sleep_hours, sleep_pattern_enc,
                         water_intake_l, calories_consumed, junk_food, int(smoker), int(alcohol),
                         stress_level, resting_hr, systolic_bp, diastolic_bp, cholesterol,
int(family_history)])
    scaled = scaler.transform(features.reshape(1,-1))
    pred = int(model.predict(scaled)[0])
    label, color = DISEASE_MAP.get(pred, ("Unknown", "#7f8c8d"))
    health_score = compute_health_score(features)
    tip = random.choice(DEFAULT_TIPS)
    icon_img = _safe_image_path(DISEASE_ICONS.get(pred, ''), f"Result {pred}")
    chart_img = make_metrics_bar(daily_steps, sleep_hours, water_intake_l, stress_level)
    card_html = f"""
    <div style="background:linear-gradient(180deg, rgba(255,255,255,0.95),
rgba(255,255,255,0.9)); padding:14px; border-radius:10px; border:1px solid
rgba(0,0,0,0.06);">
      <div style="display:flex; gap:12px; align-items:center;">
        <div style="flex:1;">
          <div style="font-size:1.15rem; font-weight:800; color:{color};"> {label} </div>
          <div style="margin-top:6px; color:#444;"> <strong>Health Score:</strong> <span
```

```
style="font-size:1.05rem;">{health_score}/100</span> </div>
        <div style="margin-top:10px; color:#666;">Tip: <em>{tip}</em></div>
        <div style="margin-top:8px; font-size:0.85rem; color:#888;">⚠️ This is an educational estimate and not a medical diagnosis.</div>
      </div>
      <div style="width:96px; height:96px; border-radius:8px; overflow:hidden; box-shadow:0 6px 18px rgba(0,0,0,0.08);">
    """
    return card_html, icon_img, chart_img

# ------------------------------
# Google Fit autofill (robust version)
# ------------------------------
from googleapiclient.discovery import build
from google_auth_oauthlib.flow import InstalledAppFlow
from google.auth.transport.requests import Request
import datetime
import pickle
import os

SCOPES = [
    'https://www.googleapis.com/auth/fitness.activity.read',
    'https://www.googleapis.com/auth/fitness.body.read',
    'https://www.googleapis.com/auth/fitness.sleep.read'
]

GOOGLE_FIT_DATA_TYPES = {
    "steps": "com.google.step_count.delta",
    "calories": "com.google.calories.expended",
    "sleep": "com.google.sleep.segment",
    "water": "com.google.hydration",
    "hr": "com.google.heart_rate.bpm"
}


def get_fit_service():
    creds = None
    if os.path.exists('token.pkl'):
        with open('token.pkl', 'rb') as token:
            creds = pickle.load(token)

    if not creds or not creds.valid:
        if creds and creds.expired and creds.refresh_token:
            creds.refresh(Request())
        else:
            flow = InstalledAppFlow.from_client_secrets_file('credentials.json', SCOPES)
```

```
            creds = flow.run_local_server(port=0)
        with open('token.pkl', 'wb') as token:
            pickle.dump(creds, token)

    return build('fitness', 'v1', credentials=creds, cache_discovery=False)


def fetch_google_fit_aggregate(service, dataType, field=None, stat="sum", default_val=0.0):
    now = datetime.datetime.utcnow()
    start_time_millis = int((now - datetime.timedelta(days=1)).timestamp() * 1000)
    end_time_millis = int(now.timestamp() * 1000)

    try:
        agg_request = {
            "aggregateBy": [{"dataTypeName": dataType}],
            "bucketByTime": {"durationMillis": 86400000},
            "startTimeMillis": start_time_millis,
            "endTimeMillis": end_time_millis,
        }

        agg_result = service.users().dataset().aggregate(userId="me",
body=agg_request).execute()
        total_values = []

        for bucket in agg_result.get("bucket", []):
            for dataset in bucket.get("dataset", []):
                for point in dataset.get("point", []):
                    for val in point.get("value", []):
                        if "fpVal" in val:
                            total_values.append(val["fpVal"])
                        elif "intVal" in val:
                            total_values.append(val["intVal"])

        if not total_values:
            return default_val

        if stat == "sum":
            return float(sum(total_values))
        elif stat == "mean":
            return float(sum(total_values) / len(total_values))
        else:
            return float(total_values[-1])

    except Exception as e:
        print(f"⚠️ Google Fit Aggregation Error for {dataType}: {e}")
        return default_val
```

```
def autofill_from_fit():
    """Fetch user's Google Fit data for the last 24 hours."""
    service = get_fit_service()
    steps = fetch_google_fit_aggregate(service, GOOGLE_FIT_DATA_TYPES["steps"],
stat="sum", default_val=8000)
    calories = fetch_google_fit_aggregate(service, GOOGLE_FIT_DATA_TYPES["calories"],
stat="sum", default_val=2100)
    sleep_millis = fetch_google_fit_aggregate(service, GOOGLE_FIT_DATA_TYPES["sleep"],
stat="sum", default_val=7 * 3600000)
    water_ml = fetch_google_fit_aggregate(service, GOOGLE_FIT_DATA_TYPES["water"],
stat="sum", default_val=2500)
    hr = fetch_google_fit_aggregate(service, GOOGLE_FIT_DATA_TYPES["hr"],
stat="mean", default_val=75)

    sleep_hours = round(float(sleep_millis) / 3600000.0, 2)
    water_l = round(float(water_ml) / 1000.0, 2)

    return steps, sleep_hours, calories, water_l, hr


def load_fit_data():
    vals = autofill_from_fit()
    result = {
        "daily_steps": vals[0],
        "sleep_hours": vals[1],
        "calories_consumed": vals[2],
        "water_intake_l": vals[3],
        "resting_hr": vals[4],
    }
    return result, float(vals[0]), float(vals[1]), float(vals[2]), float(vals[3]), float(vals[4])



# ------------------------------
# Launch Gradio app
# ------------------------------
def launch_app():
    theme = gr.themes.Soft(primary_hue="red", secondary_hue="pink")

    with gr.Blocks(theme=theme, css="""
        @import url('https://fonts.googleapis.com/css2?family=Lobster&display=swap');
        body { background: #fff9f8; }
        .mag-topbar { padding: 10px 18px; color: white; border-radius: 6px; }
        .mag-title { font-family: 'Lobster', cursive; font-weight:900; font-size:58px;
```

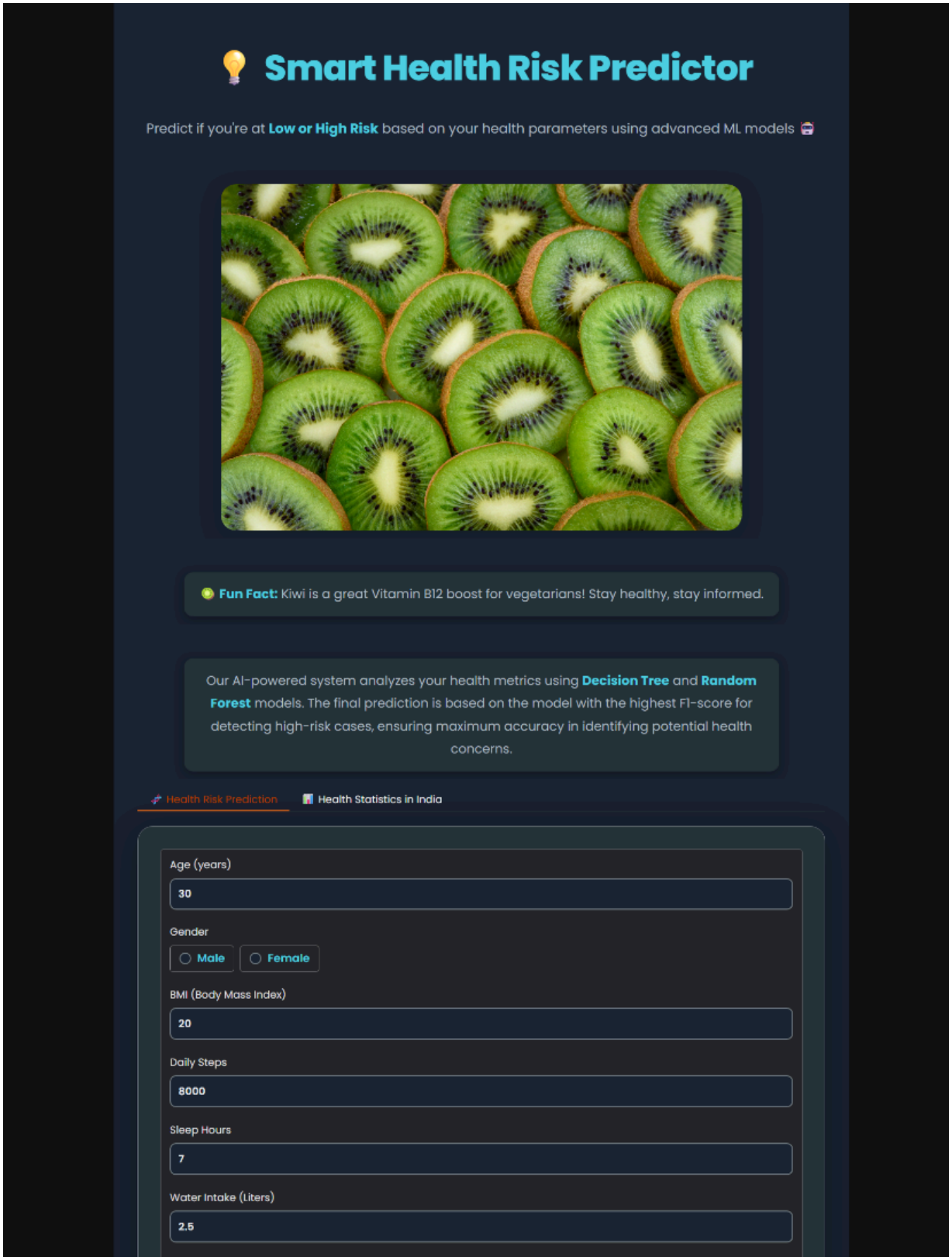
```
letter-spacing:1px; color: #000080;text-shadow: 2px 2px 10px rgba(0,0,0,0.2); }
      .hero-card { border-radius:12px; overflow:hidden; box-shadow:0 8px 26px
rgba(16,24,40,0.08); border:1px solid rgba(0,0,0,0.05); }
      .mini-card { border-radius:10px; padding:10px; background:linear-gradient(180deg,
rgba(255,255,255,0.98), rgba(255,250,250,0.98)); box-shadow:0 6px 18px
rgba(16,24,40,0.04); }
      .tag-pill { background:#ff6b6b; color:white; padding:4px 8px; border-radius:999px;
font-weight:700; font-size:0.8rem; }
      .trending-title { font-size:2rem; font-weight:900; letter-spacing:0.6px; color:#222;
margin-top:18px; }
      .footer { text-align:center; color:#666; margin-top:22px; font-size:0.9rem; }
      @media (max-width:900px) {
         .mag-title { font-size:20px; }
         .trending-title { font-size:1.5rem; }
      }
   """) as app:
      with gr.Row(elem_classes="mag-topbar"):
         with gr.Column(scale=1):
            gr.Markdown("<div class='mag-title'>BlueHealth Insights</div>",
elem_id="masthead-title")

      with gr.Row():
         with gr.Column(scale=2):
            hero_img = _safe_image_path(BIG_NEWS_IMG, "Multi-Disease Predictor Hero")
            gr.Markdown(f"""
            <a
href="https://economictimes.indiatimes.com/magazines/panache/vitamin-b12-deficiency-can-l
ead-to-fatigue-brain-fog-and-even-nerve-issues-and-vegetarians-are-at-higher-risk-warns-cardi
ologist-dr-alok-chopra-what-are-some-vitamin-b12-foods-for-vegetarians/articleshow/1238040
63.cms?from=mdr" target="_blank" style="text-decoration:none;">
               <img src="{BIG_NEWS_IMG}" style="width:100%; border-radius:12px;
box-shadow:0 8px 26px rgba(16,24,40,0.08); border:1px solid rgba(0,0,0,0.05);">
            </a>
            """, elem_classes="hero-card")
            gr.Markdown("""
            <div style="margin-top:12px;">
               <div style="font-size:1.7rem; font-weight:900; color:#111;">
                  Boost Your Vitamin B12: Vegetarian-Friendly Sources You Shouldn’t Miss!
               </div>
            """, elem_id="hero-meta")

         with gr.Column(scale=1):
            selected_news = random.sample(HEALTH_NEWS, 3)
            for i, news in enumerate(selected_news):
               gr.Markdown(f"""
               <a href="{news['url']}" target="_blank" style="text-decoration:none;
```

```
color:inherit;">
                    <div class="mini-card" style="margin-bottom:12px; display:flex;
justify-content:space-between; align-items:center; padding:10px;">
                      <div style="flex:1; padding-right:10px;">
                        <div style="font-size:0.75rem; color:#555; font-weight:600;">Times of
India</div>
                        <div style="font-size:0.95rem; font-weight:700; color:#4a00ff;
margin-top:3px;">{news['title']}</div>
                        <div style="font-size:0.75rem; color:#888;
margin-top:4px;">{news['date']}</div>
                      </div>
                      <div style="width:80px; height:80px; flex-shrink:0; border-radius:6px;
overflow:hidden;">
                        <img src="{news['img']}" style="width:100%; height:100%;
object-fit:cover;">
                      </div>
                    </div>
                  </a>
                  """, elem_id=f"mini{i+1}")

        gr.Markdown("<div class='trending-title'>Your Health Profile</div>")
        with gr.Tabs():
          with gr.TabItem("Profile"):
            with gr.Row():
              with gr.Column(scale=1):
                age = gr.Number(label="Age (years)", value=None)
                gender = gr.Radio(choices=["Male","Female"], label="Gender", type="value")
                bmi = gr.Number(label="BMI", value=None, placeholder="Enter manually or
auto-calculate")
                gr.Markdown(
                "<a href='https://www.calculator.net/bmi-calculator.html' target='_blank' "
                "style='color:#4a00ff; text-decoration:none;'>💡 Check your BMI here</a>")
                daily_steps = gr.Number(label="Daily Steps", value=None)
                water_intake_l = gr.Number(label="Water Intake (liters)", value=None)
                calories_consumed = gr.Number(label="Calories Consumed", value=None)
                junk_food = gr.Slider(0, 3, step=1, label="Junk Food (0=None, 3=Daily)",
value=None)
              with gr.Column(scale=1):
                sleep_hours = gr.Number(label="Sleep Hours", value=None)
                sleep_pattern = gr.Radio(choices=["Constant","Variable"], label="Sleep
Pattern", type="value")
                stress_level = gr.Slider(0, 10, step=1, label="Stress Level (0–10)", value=None)
                smoker = gr.Radio(choices=[0,1], label="Smoker (0=No, 1=Yes)", value=None)
                alcohol = gr.Radio(choices=[0,1], label="Alcohol (0=No, 1=Yes)",
value=None)
                family_history = gr.Radio(choices=[0,1], label="Family History (0=No,
```

```
1=Yes)", value=None)
            with gr.TabItem("Vitals"):
                resting_hr = gr.Number(label="Resting Heart Rate", value=None)
                systolic_bp = gr.Number(label="Systolic BP", value=None)
                diastolic_bp = gr.Number(label="Diastolic BP", value=None)
                cholesterol = gr.Number(label="Cholesterol", value=None)

        with gr.Row():
            with gr.Column():
                gr.Markdown("### 🔄 Auto-Fill from Google Fit")
                autofill_btn = gr.Button("📲 Load Google Fit Data", variant="primary")
                fit_output = gr.JSON(label="Fit Data Preview")

        with gr.Row():
            predict_btn = gr.Button("🚀 Predict Risk", variant="primary")
            clear_btn = gr.Button("🧹 Clear")
            tip_box = gr.Markdown(value=f"** 💡 Tip of the Day:**
_{random.choice(DEFAULT_TIPS)}_", elem_id="tip-box")

        result_card = gr.HTML(value="", label="Prediction Result")
        result_icon = gr.Image(value=_make_placeholder_image("Result"), label="Result Icon",
interactive=False)
        result_chart = gr.Image(value=_make_placeholder_image("Metrics Chart",
size=(640,320)), label="Lifestyle Chart", interactive=False)

        gr.Markdown("<div class='trending-title'>Trending Health Stories</div>")
        with gr.Row():
            selected_trending = random.sample(HEALTH_NEWS, min(4, len(HEALTH_NEWS)))
            trending_images = [TRENDING_IMG_1, TRENDING_IMG_2, TRENDING_IMG_3,
TRENDING_IMG_4]
            for i, news in enumerate(selected_trending):
                with gr.Column(scale=1):
                    gr.Markdown(f"""
                    <a href="{news['url']}" target="_blank" style="text-decoration:none;
color:inherit;">
                        <div style="border-radius:8px; overflow:hidden; box-shadow:0 6px 18px
rgba(0,0,0,0.03); height:140px;">
                            <img src="{trending_images[i]}" style="width:100%; height:100%;
object-fit:cover;">
                            <div style="position:absolute; bottom:10px; left:10px; right:10px;
background:rgba(255,255,255,0.8); padding:5px; border-radius:4px;">
                                <div style="font-weight:800; color:#0b7285;
font-size:1rem;">{news['title']}</div>
                                <div style="color:#888; font-size:0.85rem;
margin-top:4px;">{news['date']}</div>
                            </div>
```

```
                </div>
            </a>
            """, elem_id=f"trending{i+1}")

    gr.HTML("""
    <div class='footer'>
        <hr>
        <div style='font-size:0.95em;'>⚠️ <b>Disclaimer:</b> This tool provides educational AI-based health insights and is not a medical diagnosis.</div>
    </div>
    """)
    def _on_predict(*args):
        res_html, icon_img, chart_img = predict_and_visualize(*args)
        more_html = "<div style='margin-top:10px;'><a href='#' style='color:#ff6b6b; font-weight:700; text-decoration:none;'>Read full guidance →</a></div></div>"
        return res_html + more_html, icon_img, chart_img

    def _on_clear():
        return "", _make_placeholder_image("Result"), _make_placeholder_image("Metrics Chart", size=(640,320))
    inputs = [age, gender, bmi, daily_steps, sleep_hours, sleep_pattern,
              water_intake_l, calories_consumed, junk_food, smoker, alcohol,
              stress_level, resting_hr, systolic_bp, diastolic_bp, cholesterol,
              family_history]
    predict_btn.click(fn=_on_predict, inputs=inputs, outputs=[result_card, result_icon, result_chart])
    clear_btn.click(fn=_on_clear, inputs=None, outputs=[result_card, result_icon, result_chart])
    autofill_btn.click(
        fn=load_fit_data,
        inputs=[],
        outputs=[fit_output, daily_steps, sleep_hours, calories_consumed, water_intake_l, resting_hr]
    )
  app.launch(server_name='0.0.0.0', share=True)

if __name__ == "__main__":
  launch_app()
```

11.4 UI screenshots



Calories Consumed

2000

Smoker (1 = Yes, 0 = No)

○ 0 ○ 1

Alcohol (1 = Yes, 0 = No)

○ 0 ○ 1

Resting Heart Rate (bpm)

70

Systolic Blood Pressure

60

Diastolic Blood Pressure

120

Cholesterol (mg/dL)

200

Family History of Disease (1 = Yes, 0 = No)

○ 0 ○ 1

🚀 PREDICT RISK NOW

🧠 Decision Tree

Low Risk

📊 Accuracy: **57.58%**

🎯 F1-Score: **28.00%**

🌲 Random Forest

Low Risk

📊 Accuracy: **73.69%**

🎯 F1-Score: **5.00%**

✅ Final Decision (Decision Tree (higher F1-score for High Risk))

Low Risk

🔍 Health Insights

- Normal weight – great, keep it up! 💪
- Hypertension / Heart Disease Risk ❤️

💡 Smart Health Risk Predictor

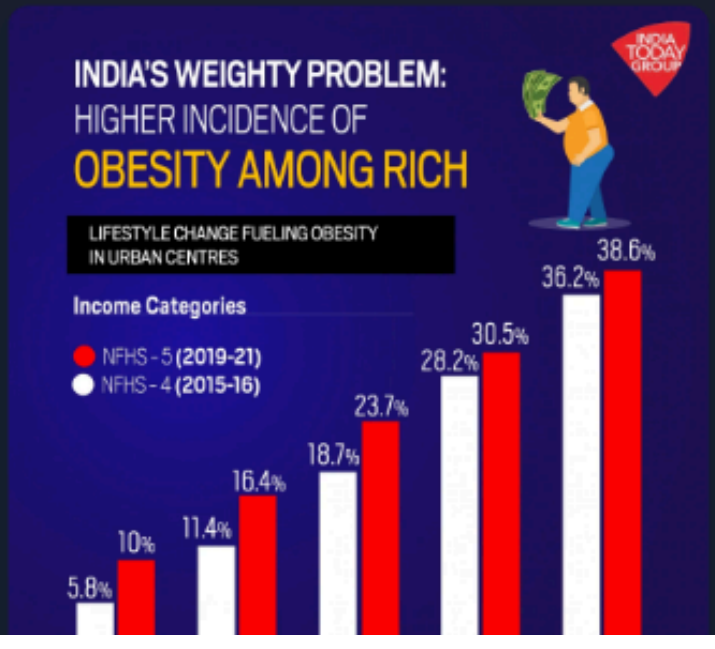
Predict if you're at **Low or High Risk** based on your health parameters using advanced ML models 🤖

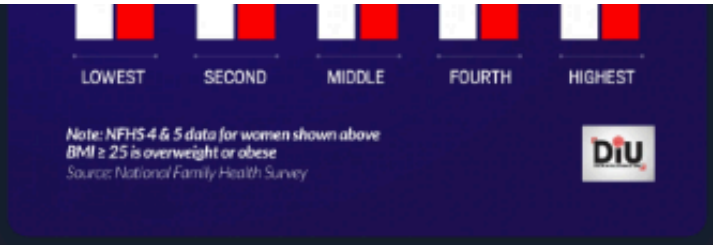
Fun Fact: Kiwi is a great Vitamin B12 boost for vegetarians! Stay healthy, stay informed.

Our AI-powered system analyzes your health metrics using **Decision Tree** and **Random Forest** models. The final prediction is based on the model with the highest F1-score for detecting high-risk cases, ensuring maximum accuracy in identifying potential health concerns.

Health Risk Prediction | Health Statistics in India

BMI & Obesity Crisis





India, once known for malnutrition, is now facing an **obesity epidemic**. Urban lifestyles, processed foods, and sedentary habits are driving this dramatic shift in public health.

Diabetes: The Silent Epidemic

India is the '**Diabetes Capital of the World**' with approximately **80 million** people currently affected. Projections indicate this number will soar to **135 million by 2045**. Early detection and lifestyle changes are crucial.

Cancer: Rising Concerns

11.5 Supplementary proofs

Proofs for google fit integrations trials

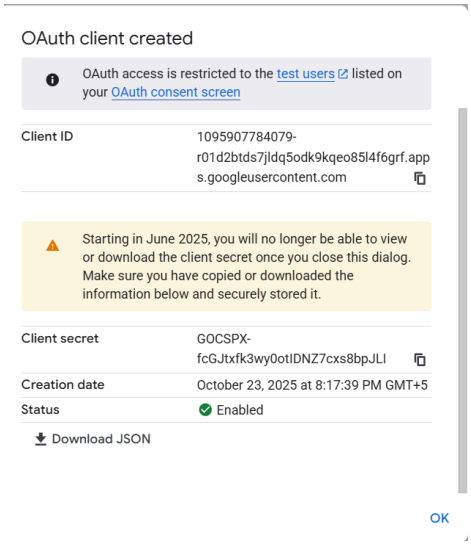


Figure 11 : Google fit integration trial

11.6 Ethics/IRB notes

This project uses anonymized health and lifestyle data collected from Google Fit and wearable devices. No personally identifiable information is stored. Participation is voluntary, and all data is handled in compliance with privacy standards. The predictive models (Decision Tree and Random Forest) provide advisory insights only and do not replace professional medical diagnosis. The study poses minimal risk, as it is non-invasive and purely analytical. All data processing and predictions are conducted ethically, with emphasis on confidentiality and responsible use.

11.7 AI Tools Usage Disclosure

1. Overall AI Contribution

- Estimated overall percentage of AI-generated content across the submission (report + code + figures): 50**%**

2. Breakdown by Component

Component	AI Contribution (%)
Ideation / Brainstorming	0%
Literature Summary / Paraphrase	70%
Writing (Report Text)	50%
Coding (Generation / Refactor / Fix)	65%
Data Preparation / Analysis Assistance	0%
Figure	0%
Table Creation	50%
Editing /Style	5%
Grammar	80%

Table 7 : AI contribution

3. AI Tools Used

Tool	Version / Date Used	Purpose
ChatGPT	(GPT-5 Mini) / 23 Oct 25	Ideation, writing, code generation (model & GUI), report content
Grok	Grok 4 and Grok 4 Fast / Oct 25	Code generation
Comet	Comet / Oct 25	Research support, citation lookup, generating report content from provided references
Perplexity	Version 4.0.0 /May 25	Research support, summarizing references

Table 8 : AI Tools used

4.Prompting Transparency

- **Typical prompts used:**

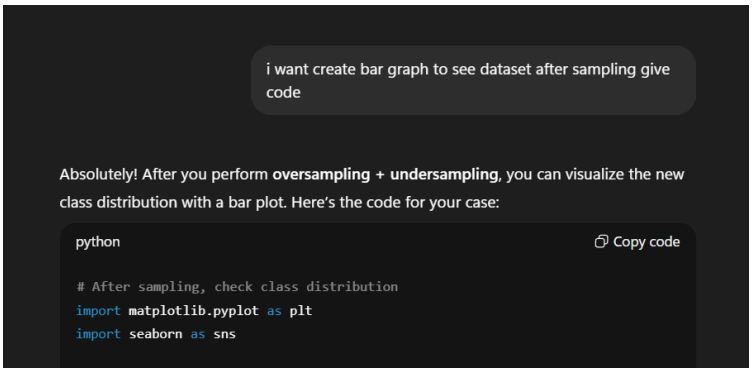


Figure 12 : prompt

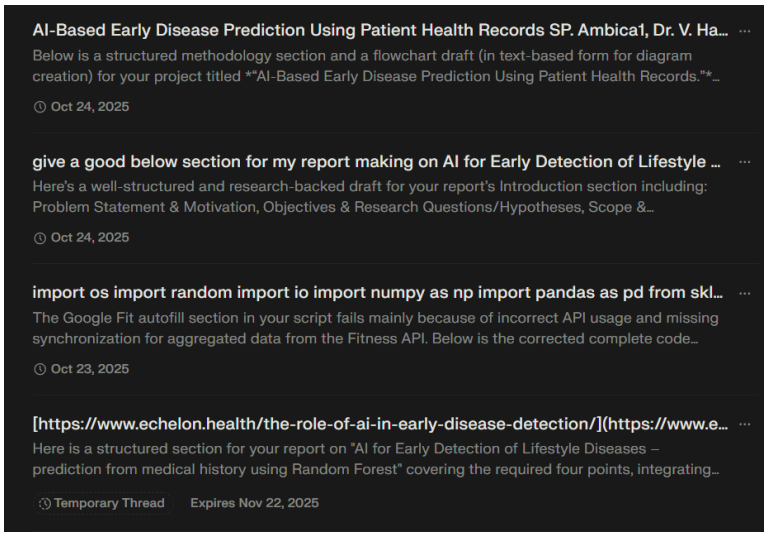


Figure 13 : prompt

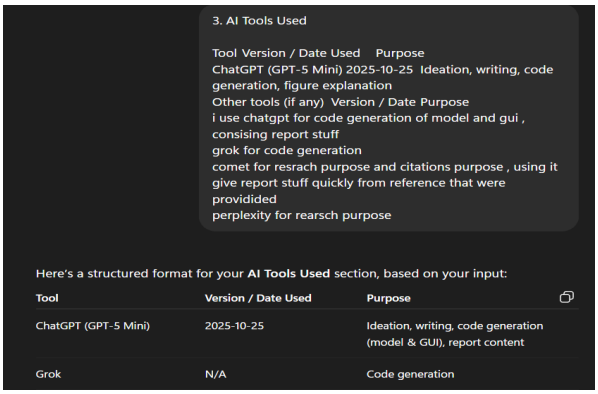


Figure 14 : prompt

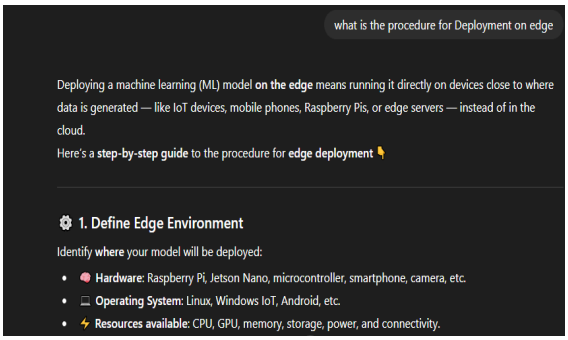


Figure 15 : prompt

Few links and examples of prompts :

a. **Link 1 :** https://chatgpt.com/share/68fdd5bd-29ec-8011-b6b5-9b72114b3c9e
b. **Link 2 :** https://chatgpt.com/share/68fdd6f0-5aac-8011-9517-d4b2ffd0b742
c. **Link 3 :** https://chatgpt.com/share/68fdca2d-32f0-800e-b8af-9a83b9d00dae
d. **Link 4 :** https://www.perplexity.ai/search/ai-based-early-disease-predict-R8qswlmbQn6KDqXxxxNV4w#0
e. **Link 5 :** https://grok.com/share/bGVnYWN5LWNvcHk%3D_599712b6-efe2-41be-be1f-f18e0ebe1ddd
f. **Example 1:** ***"Generate Python code to sampling of dataset."***
g. **Example 2:** ***"Write a concise error analysis for disease risk prediction."***
h. **Example 3:** ***"Give around 10 -15 keyword from report."***

5. Human Verification

- **All AI outputs were manually checked for:**
  - Correctness of computations and code logic
  - Originality and plagiarism compliance
  - Licensing and ethical use of AI-generated content

11.8 Authorship & Individual Contributions

Member Name & Roll No.	Specific Contributions
Member 1 A-44	GUI ; Report Making ; Further Corrections in Model Training
Member 2 A-51	GUI enhancement ; Repo creation ; Zip file creation ; improvements in code
Member 3 A-54	Model Training and Testing ; Writing initial code

Table 9: Individual Contributions