Healthcare Data Pipeline

Generate, Clean & Validate Patient Datasets

Version	Date	Status
1.0.0	February 2025	Active

Project Overview

An end-to-end Python toolkit for generating, cleaning, and validating realistic healthcare patient datasets. Designed for data analysts and researchers, it addresses common issues with messy clinical data.

1

Generate Datasets

Create synthetic, realistic healthcare datasets with 12 clinically meaningful fields.

2

Detect & Fix Issues

Identify and correct common data quality problems like duplicates, missing values, and outliers.

3

Validate & Score

Validate cleaned data against medical rules and produce a quality health score.

4

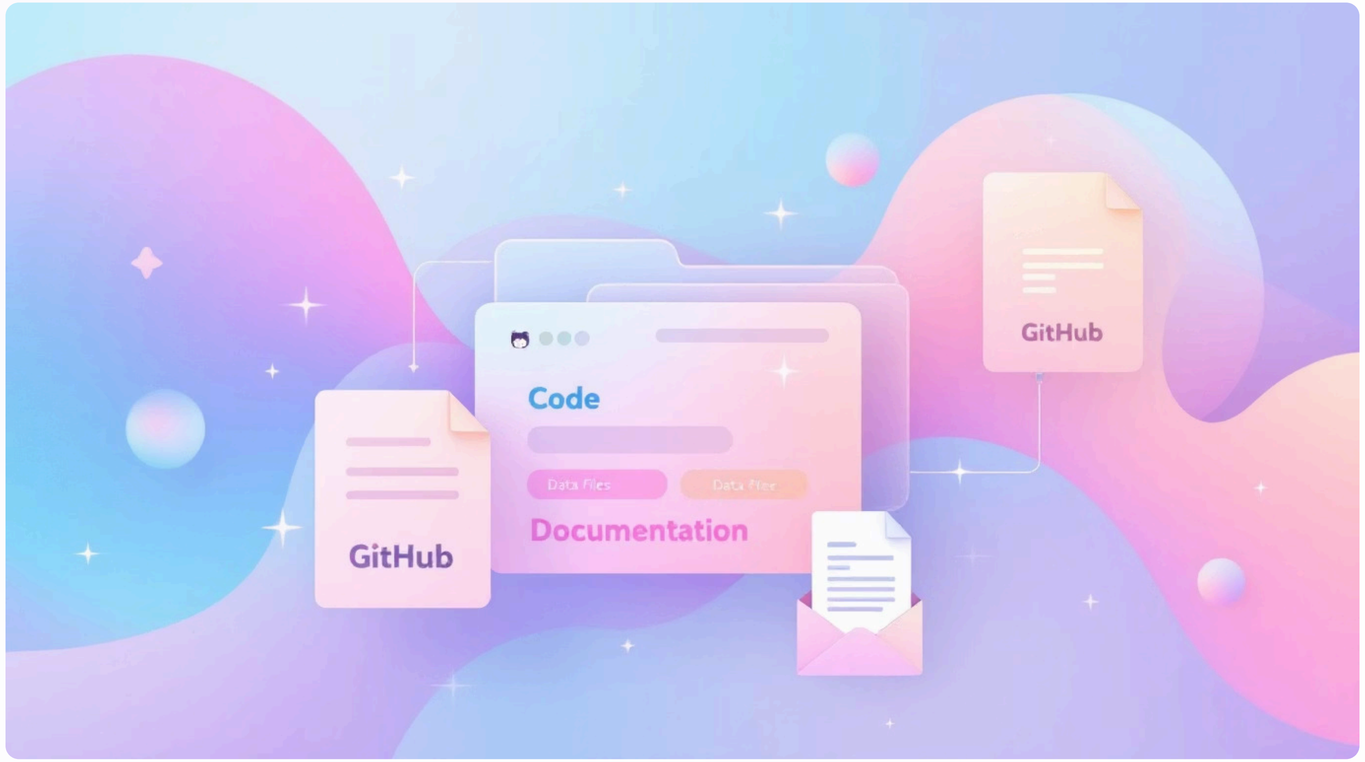
Provide Scripts

Deliver clean, well-documented Python scripts ready for GitHub.

Project Structure

A clear, organized repository for easy navigation and collaboration.

- **scripts/**: Core Python scripts for generation, cleaning, and reporting.
- **data/**: Sample messy and cleaned datasets.
- **docs/**: Detailed usage guides.
- **requirements.txt**: Python dependencies.



Dataset Fields & Ranges

The dataset includes 12 fields covering patient demographics, vital signs, and clinical data. Each numeric field has a defined valid medical range for cleaning and validation.

patient_id	int	1000+	Unique patient identifier
Age	int	0 – 90 years	Patient age in years
gender	str	M / F / U	Standardised to M, F, or Unknown
height_cm	float	100 – 200 cm	Patient height in centimetres
weight_kg	float	20 – 150 kg	Patient weight in kilograms

Step 1: Generate Dataset

Produces a synthetic patient dataset with 200 rows and 12 columns, using statistically realistic distributions for each field.

Generation Logic

- **Age**: Uniform random integer (18-89)
- **Gender**: Random choice (49% Male / 51% Female)
- **Height/Weight**: Normal distribution, clipped to realistic ranges
- **Temperature**: Uniform (97.5–99.5 °F for 92%; fever 99.6–103 for 8%)
- **Blood Pressure**: Random systolic/diastolic strings

Step 2: Clean Messy Data

The cleaning pipeline addresses intentionally injected data quality issues to simulate real-world clinical data problems.

01

Load Data

Read CSV with pandas, log row/column counts.

02

Remove Duplicates

Identifies and removes 15 exact duplicate rows.

03

Fix Data Types

Corrects types, splits blood pressure, handles 'nan' strings.

04

Fill Missing Values

Numeric columns filled with median, categorical/date with mode.

05

Clip Outliers

Numeric columns clipped to valid medical ranges.

06

Standardise Text

Cleans and standardizes text fields like gender and insurance type.

Common Data Issues Addressed

The cleaning script tackles various real-world data problems:

Duplicate Rows

15 exact copies removed.

Missing Values

84 cells filled across various columns.

Outlier Values

~31 rows with extreme values (e.g., temperature=307.7).

Wrong Data Types

~40 cells corrected (e.g., age='forty-two').

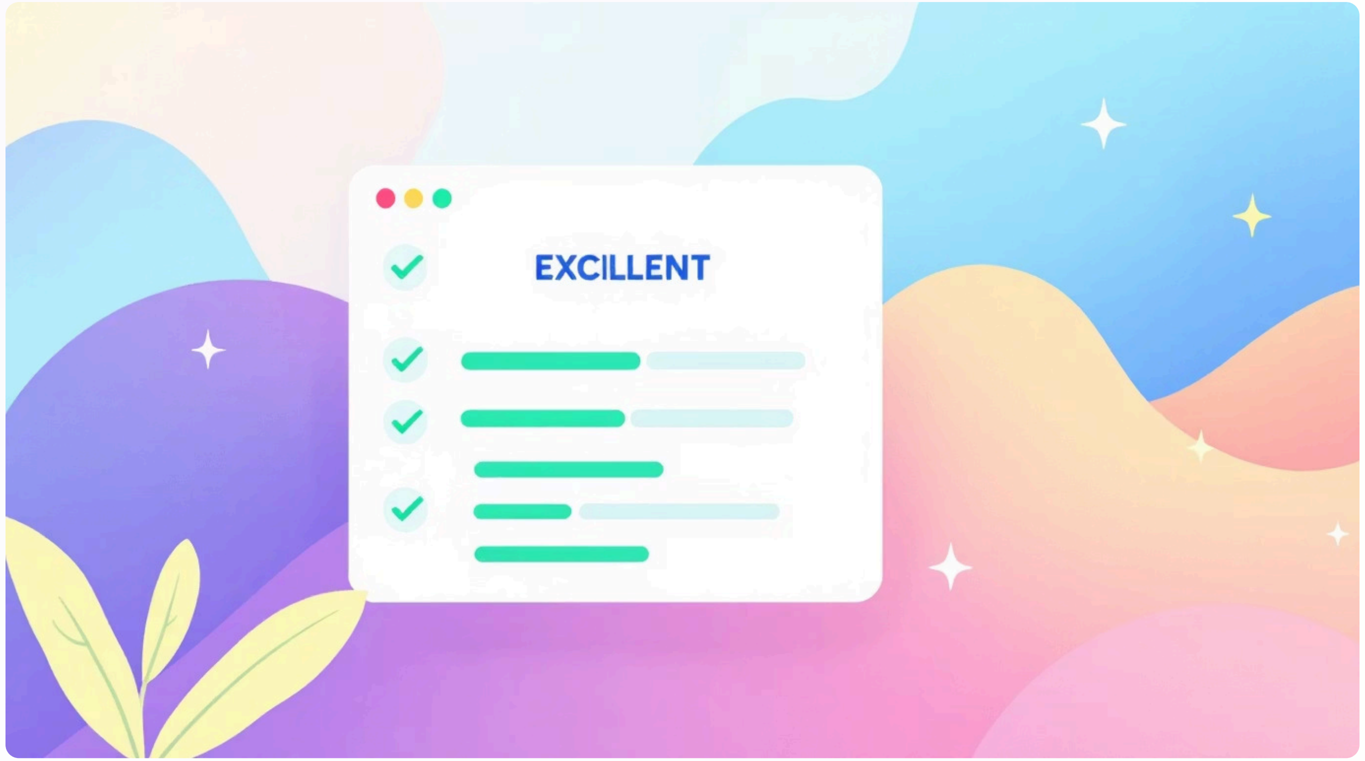
Inconsistent Text

~30 cells standardized (e.g., gender='male').

Step 3: Data Health Report

The health report runs four independent checks against the cleaned dataset, producing a qualitative quality score.

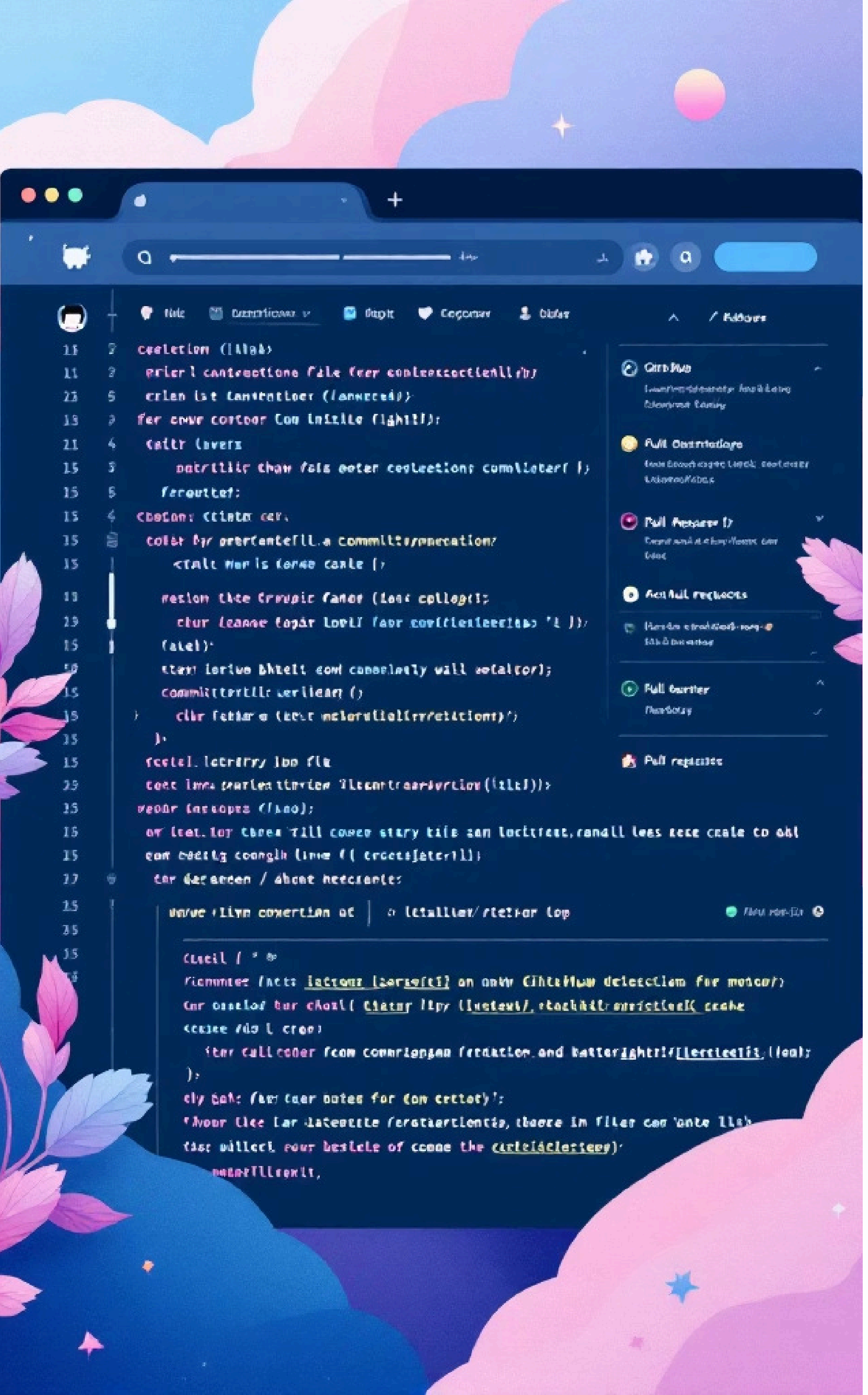
- **Missing Values**: Flags any column with > 0 missing.
- **Duplicate Rows**: Reports exact number of duplicate records.
- **Numeric Range Check**: Tests numeric columns against medical min/max bounds.
- **Categorical Consistency**: Validates gender and insurance_type against expected values.



Results: Before vs. After Cleaning

The cleaning pipeline transformed a messy dataset into a fully validated, analysis-ready one, achieving an EXCELLENT data health score.

Total rows	⚠️ 215	✅ 200
Duplicate rows	⚠️ 15	✅ 0
Missing values	⚠️ 84 cells	✅ 0
Outlier values	⚠️ ~31	✅ 0
Data health score	🔴 POOR	✅ EXCELLENT



GitHub Repository & Tech Stack

The project is open-source, well-documented, and built with robust Python libraries.

Tech Stack

- **Python 3.8+**: Core programming language.
- **Pandas 2.0+**: DataFrame operations, CSV I/O, type coercion.
- **NumPy 1.24+**: Numeric distributions, random seeds, array clipping.
- **argparse**: Command-line argument parsing.

GitHub Documents

- **README.md**: Project overview, quick start, requirements.
- **CONTRIBUTING.md**: Contribution guidelines.
- **CHANGELOG.md**: Version history and planned features.
- **LICENSE**: MIT open-source license.