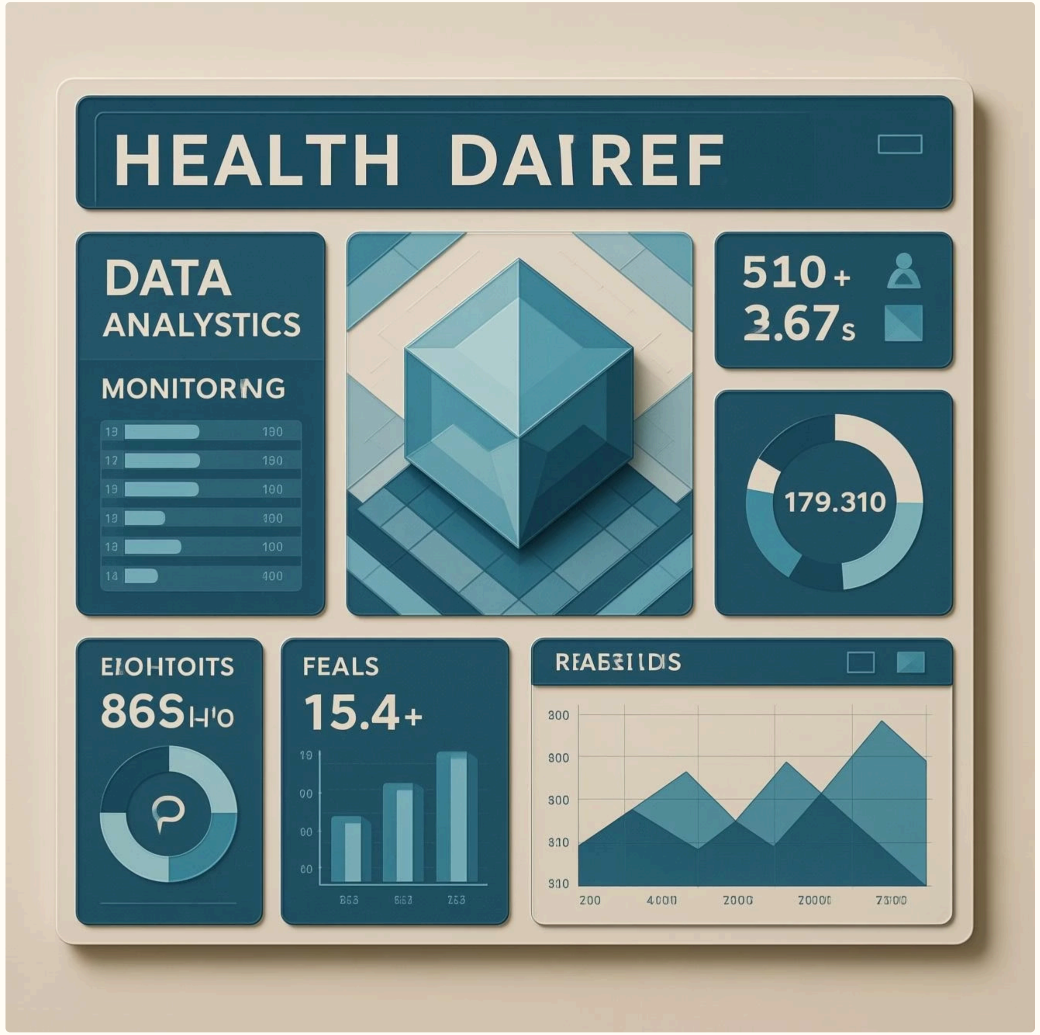
FitPulse Health

Intelligent Anomaly Detection System Using Python, Pandas, NumPy & Machine Learning to Detect Abnormal Heart-Beat Patterns

Introduction to FitPulse

FitPulse is an advanced health monitoring platform that continuously collects **Heart Beat Per Minute (HBPM)** data from customers to detect abnormal readings—both unusually high and low values.

Powered by **Pandas** for data manipulation, **NumPy** for numerical operations, and machine learning models including **Regression and K-Means clustering**, FitPulse provides early insights critical for health monitoring and personalized fitness coaching.



Why Anomaly Detection Matters

Spike Identification

Detects unexpected heart rate spikes that could indicate health issues or excessive workout intensity

Early Risk Monitoring

Monitors cardio-related risks before they become serious, enabling preventive healthcare measures

Safe Training Zones

Ensures workout intensity stays within safe, effective ranges for optimal fitness results

Hidden Pattern Discovery

Identifies subtle patterns in raw data that human analysis might overlook

Technical Stack & Tools

Python

Core programming language powering the entire analytics pipeline and machine learning workflows

Pandas

Advanced data cleaning, transformation, and processing framework for structured health data

NumPy

High-performance numerical operations and array processing for statistical computations

Matplotlib / Seaborn

Comprehensive visual analytics and data visualization for insights presentation

Scikit-Learn

Machine learning library providing regression and clustering algorithms for predictive analytics

Excel (openpyxl)

Data import, export, and handling for seamless integration with existing health records

Pandas: Data Processing Powerhouse

Key Capabilities

Pandas provides **DataFrames**—powerful data structures for organizing and analyzing structured health information efficiently.

Essential functions in FitPulse:

- `pd.read_excel()` – Import health data files
- `isnull().sum()` – Detect missing entries
- `astype() & to_numeric()` – Convert values to correct formats
- `groupby()` – Analyze heart rate by demographics
- `describe()` – Generate statistical summaries

FitPulse Implementation

```
data = pd.read_excel("fitpulse.xlsx")

data['Heart_Beat_Per_Minute'] = pd.to_numeric(
    data['Heart_Beat_Per_Minute'],
    errors='coerce'
)
```

This code loads the health dataset and converts heart rate values to numeric format, handling any conversion errors gracefully by coercing invalid values to NaN.

NumPy & Dimensionality Concepts

NumPy: Numerical Backbone

NumPy serves as the foundation for all numerical operations in FitPulse, providing:

- Fast mean heart rate calculations
- Efficient array handling and manipulation
- Optimized inputs for ML model training

```
import numpy as np

np.mean(data['Heart_Beat_Per_Minute'])
```

Understanding Dimensionality

Rows represent individual customers

Columns represent variables (Gender, HBPM, Status)

Impact on analysis:

- Learning accuracy and model performance
- Cluster separation quality
- Regression prediction precision

Low dimensions → Simple patterns

High dimensions → Deeper insights

Data Pipeline: From Excel to Analysis

1. Import Excel
2. Check Nulls
3. Convert Types
4. ML Preprocess

Reading Excel Files

```
data = pd.read_excel("fitpulse.xlsx")
print(data.head())
```

Key advantages:

- Supports multiple sheets
- Seamless Excel → DataFrame conversion
- Easy preprocessing for ML models

Checking Null Values

```
data.isnull().sum()
```

Why this matters:

- Missing values distort predictions
- Identifies sensor failures or data gaps
- Required step before ML training

Example output:

```
Gender 0
Heart_Beat_Per_Minute 2
```

Feature Engineering: Gender Encoding

Intelligent Gender Code Transformation

Machine learning models require numerical inputs. FitPulse implements a robust gender encoding system that handles various input formats:

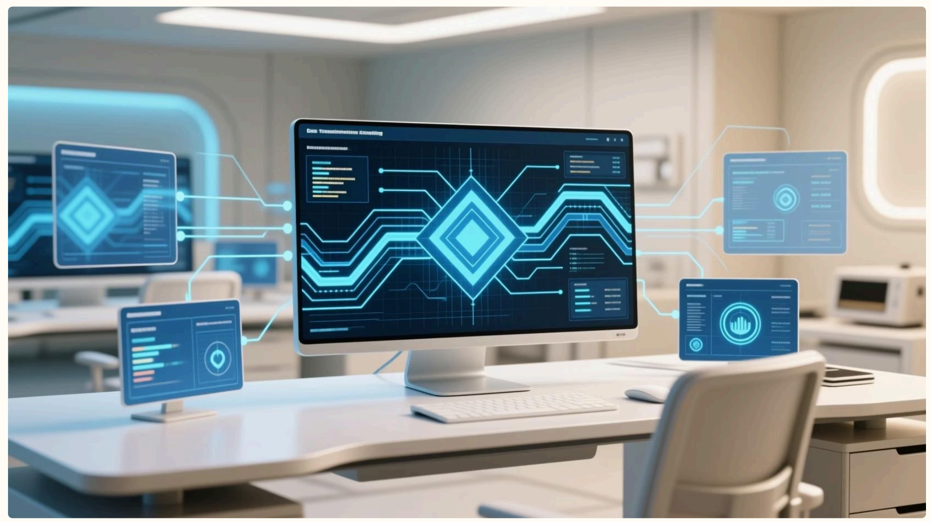
```
gender_map = {
    'male': 1, 'female': 0,
    'm': 1, 'f': 0
}

data['gender_code'] = (
    data['Gender']
    .astype(str)
    .str.strip()
    .str.lower()
    .map(gender_map)
    .fillna(2)
    .astype(int)
)
```

This transformation converts text gender values to numeric codes, handles variations in capitalization and spacing, and assigns unknown values to code 2.

Transformation Results

Gender	gender_code
male	1
Female	0
M	1
f	0
unknown	2



Rule-Based Health Status Classification

1

Low Heart Rate

< 60 HBPM

May indicate bradycardia or excellent cardiovascular fitness

2

Normal Range

60-100 HBPM

Healthy resting heart rate for most adults

3

High Heart Rate

> 100 HBPM

Potential tachycardia requiring medical attention

Implementation

```
def status(h):
    if h < 60:
        return "Low"
    elif h > 100:
        return "High"
    else:
        return "Normal"

data['Status'] = data['Heart_Beat_Per_Minute'].apply(status)
```

HBPM	Status
55	Low
92	Normal
145	High

Visualization & Analytics

Chart Types Used

- **Line chart** – Track HBPM trends over time
- **Bar chart** – Compare gender-wise average heart rates
- **Histogram** – Visualize distribution of heart rates
- **Scatter plot** – Display regression and K-Means clustering results

```
plt.plot(data['Heart_Beat_Per_Minute'])
plt.title("Heart Beat Trend")
plt.show()
```