

Milestone 2: Feature Extraction & Modeling

FitPulse: Health Anomaly Detection from Fitness Devices

By: Divya Sharma



Overview

This milestone focuses on analyzing time-series data collected from fitness devices, including heart rate, steps, and sleep patterns. The goal is to transform raw sensor data into meaningful insights that can power anomaly detection.

Key Objectives

- Extract meaningful statistical features from raw time-series data
- Model trends and patterns using advanced forecasting techniques
- Group behaviors using unsupervised clustering algorithms

This analytical foundation prepares the data for robust anomaly detection in Milestone 3.

Feature Extraction with TSFresh

TSFresh (Time Series Feature extraction based on scalable hypothesis tests) automatically extracts hundreds of statistical features from time-series windows, enabling machine learning on temporal data.

01

Window Creation

Created sliding windows of 12 data points, representing 1 hour of continuous monitoring

02

Feature Extraction

Applied MinimalFCParameters to extract mean, median, standard deviation, entropy, and autocorrelation

03

Feature Selection

Removed constant and non-informative features to reduce dimensionality

- ❑ Output: A clean feature matrix ready for clustering and anomaly detection modeling



TSFresh Implementation

Code Structure

The implementation follows a systematic pipeline: build windows, extract features, impute missing values, and filter constant features.

This approach ensures data quality while maintaining computational efficiency across large datasets.

```
dfp = build_windows(hr, 'heart_rate')
fc = MinimalFCParameters()

features = extract_features(
    dfp,
    column_id='window_id',
    column_sort='timestamp',
    default_fc_parameters=fc,
    n_jobs=0
)

features = impute(features)
features = features.loc[:, features.std()!=0]
```

Trend Modeling Approaches

Identifying baseline patterns is essential for detecting deviations that may indicate health anomalies.

Primary: Facebook Prophet

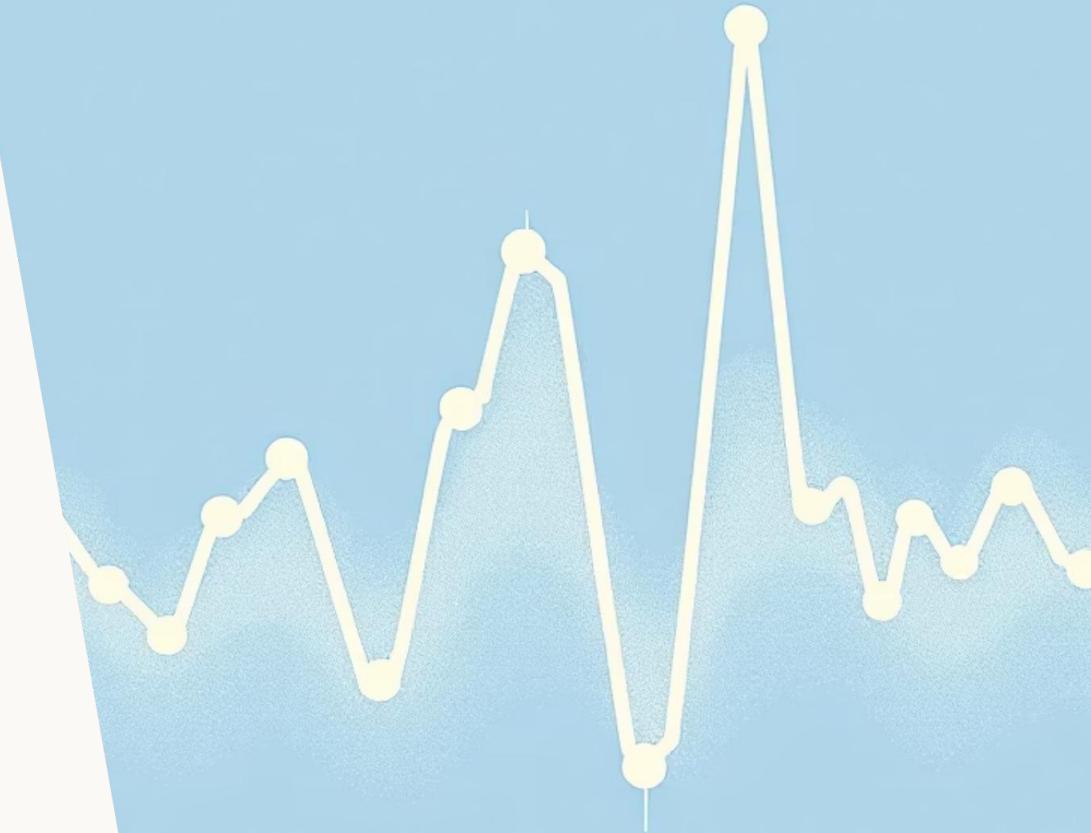
Advanced forecasting tool designed for time-series with strong seasonal patterns and multiple seasonality

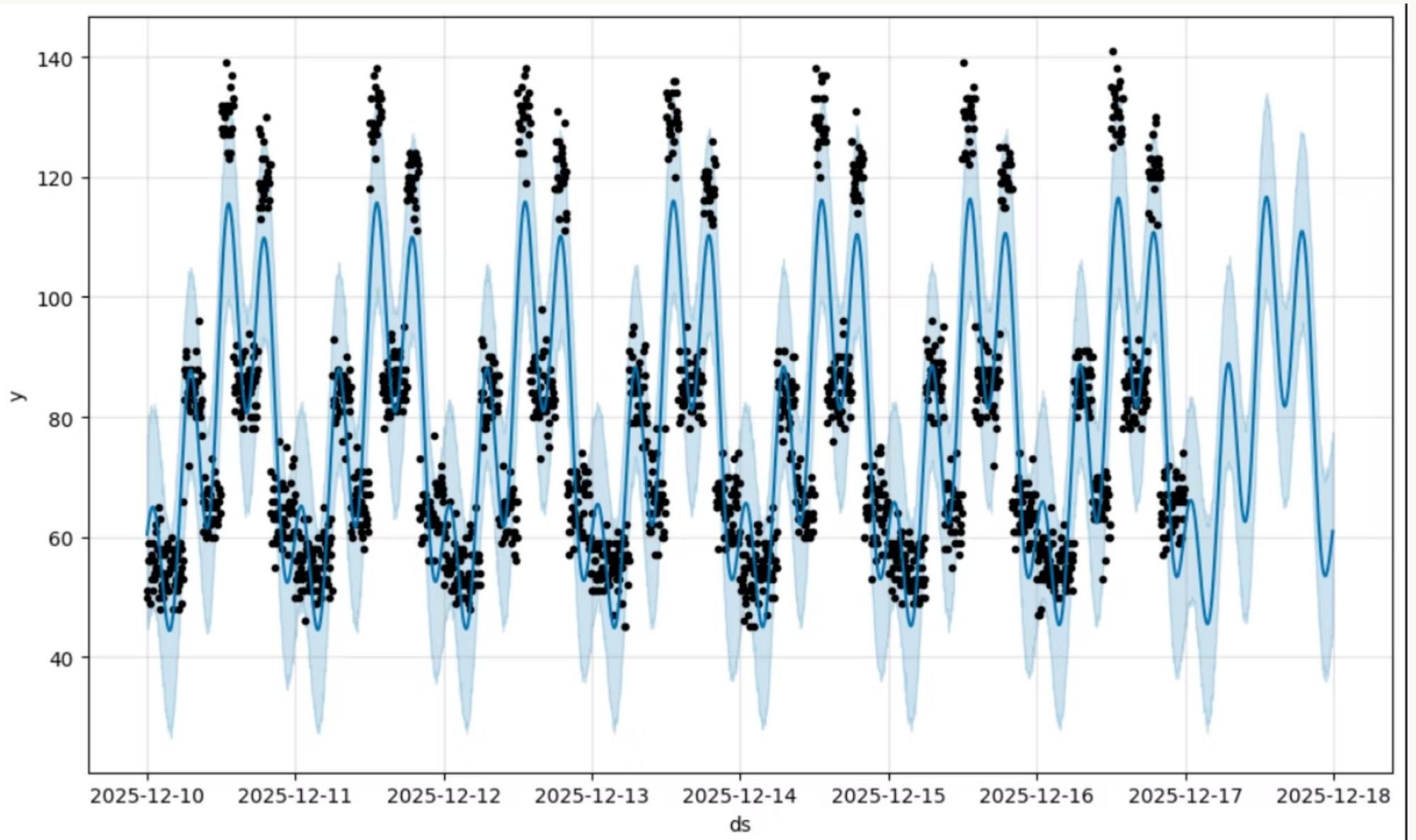
- Captures daily and weekly trends
- Handles missing data gracefully
- Provides uncertainty intervals

Backup: Rolling Median

When Prophet encountered environment constraints, rolling median provided a robust alternative

- Smooths noisy sensor data
- Captures typical behavior patterns
- Enables 3-sigma anomaly detection





Prophet Forecasting of Heart Rate (Seasonal Trend + Confidence Intervals)

The blue curve represents Prophet's predicted trend with daily seasonality, while the shaded region shows confidence intervals. Black dots represent actual heart-rate values aligned closely with the model's forecast.

Rolling Median Trend Detection

```
hr_series = hr.set_index('timestamp')['heart_rate']

baseline = hr_series.rolling(
    window=12,
    center=True
).median()

residual = hr_series - baseline
sigma = residual.std()

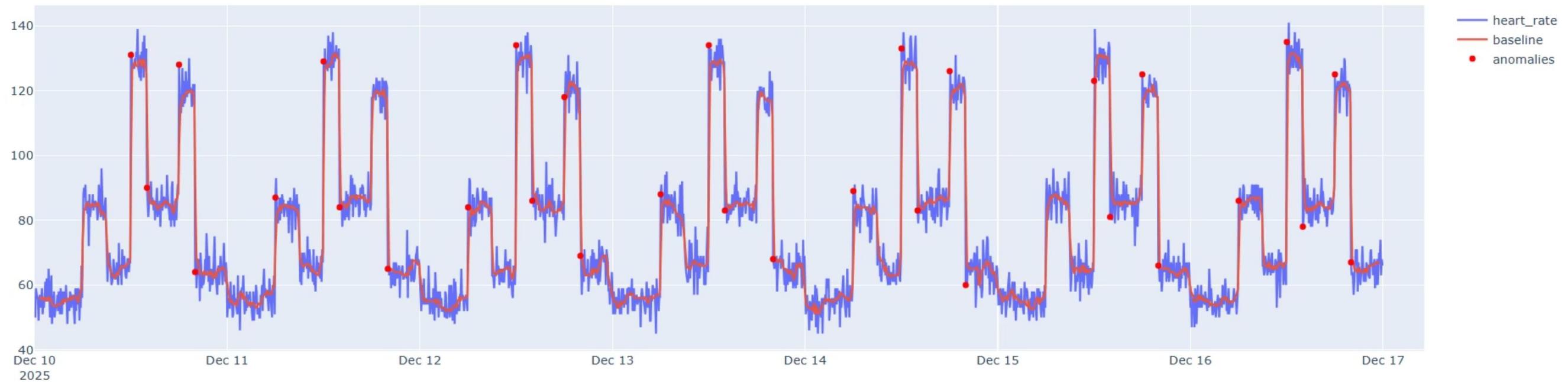
anomalies = residual[abs(residual) > 3*sigma]
```

Three-Sigma Rule

This statistical approach identifies data points that deviate more than 3 standard deviations from the rolling baseline.

In a normal distribution, 99.7% of data falls within 3σ , making outliers statistically significant.

Heart Rate Trend + Anomalies



Heart Rate Trend Modeling & Anomaly Detection (Rolling Median + 3-Sigma Rule)

This visualization shows the heart-rate signal over time, a rolling-median baseline that smooths the natural fluctuations, and anomalies detected using the 3-sigma method.

Red dots highlight points where heart rate deviates significantly from expected behavior.



Trend Modeling Results

Blue Line: Actual heart rate measurements from the fitness device, showing natural variability throughout the day

Orange Line: Rolling median baseline representing expected heart rate patterns based on historical trends

Red Dots: Detected anomalies where heart rate deviated significantly ($>3\sigma$) from the baseline

Clustering for Behavioral Patterns

Clustering algorithms group similar time windows together, revealing distinct behavioral patterns in fitness data without labeled training data.



KMeans Clustering

Partitions data into fixed number of clusters, ideal for identifying distinct activity levels

- Groups windows into 3 behavioral clusters
- Uses PCA for dimensionality reduction
- Visualizes patterns in 2D space

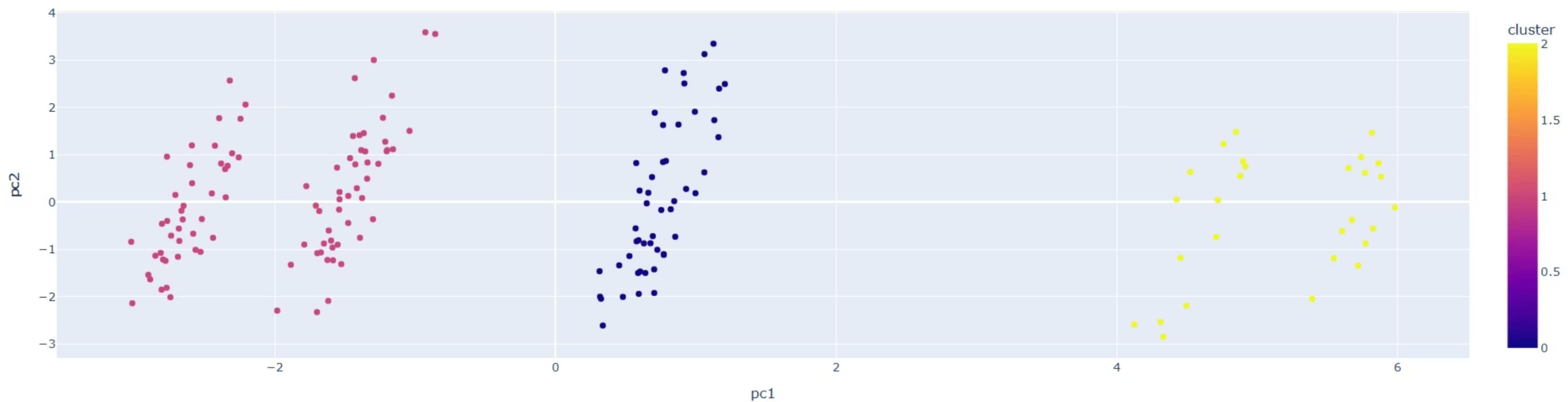


DBSCAN Clustering

Density-based approach that identifies outliers and noise points automatically

- No predefined cluster count required
- Detects noise and outlier windows
- Robust to irregular cluster shapes

KMeans Clustering (PCA)



KMeans Clustering (PCA Projection)

This plot shows 3 clusters formed from TSFresh-extracted features of heart-rate windows.

High-dimensional features were reduced to 2 components using PCA for visualization.

Each cluster represents a distinct pattern of heart-rate behavior, helping detect unusual activity patterns and potential anomalies

Clustering Implementation

KMeans Algorithm

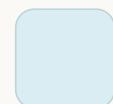
```
Xs = StandardScaler().fit_transform(  
    features.fillna(0)  
)  
  
kmeans = KMeans(  
    n_clusters=3,  
    random_state=42  
)  
labels = kmeans.fit_predict(Xs)  
  
pca = PCA(n_components=2)  
pcs = pca.fit_transform(Xs)
```

DBSCAN Algorithm

```
db = DBSCAN(  
    eps=0.6,  
    min_samples=5  
)  
  
db_labels = db.fit_predict(Xs)
```

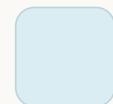
DBSCAN parameters: epsilon defines neighborhood radius, min_samples sets density threshold for core points.

Milestone 2 Summary & Next Steps



Accomplishments

Successfully extracted statistical time-series features using TSFresh, performed trend modeling with Prophet and rolling median fallback, applied KMeans and DBSCAN clustering, and identified anomalies using the 3-sigma rule



Looking Ahead

Milestone 3 will integrate these analytical components into a full-featured anomaly detection dashboard using Streamlit, enabling real-time health monitoring

GitHub Repository: FitPulse Health Anomaly Detection

[View Project Repository](#)



Thank you!