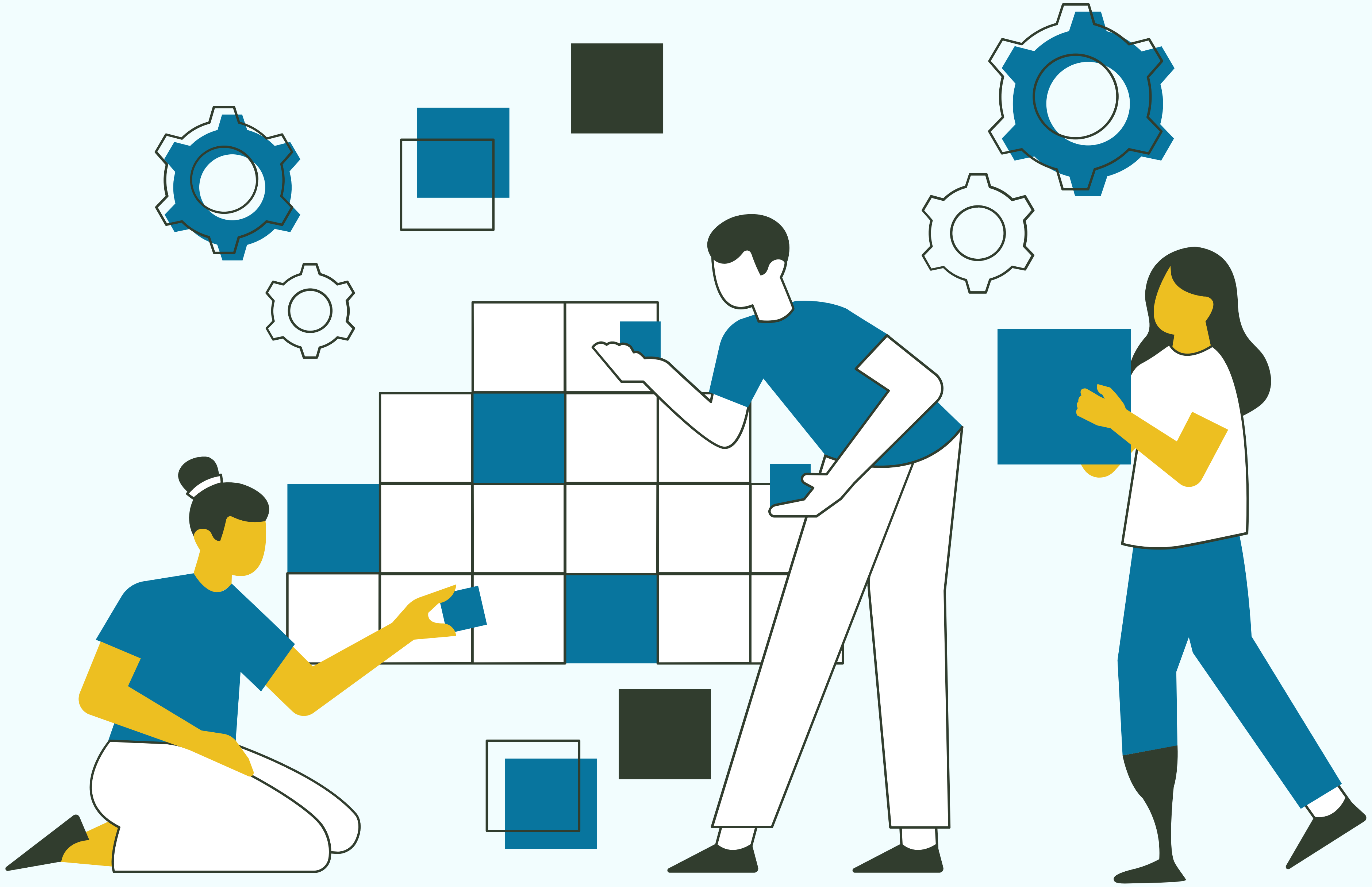
DEVTrails

University Hackathon 2025

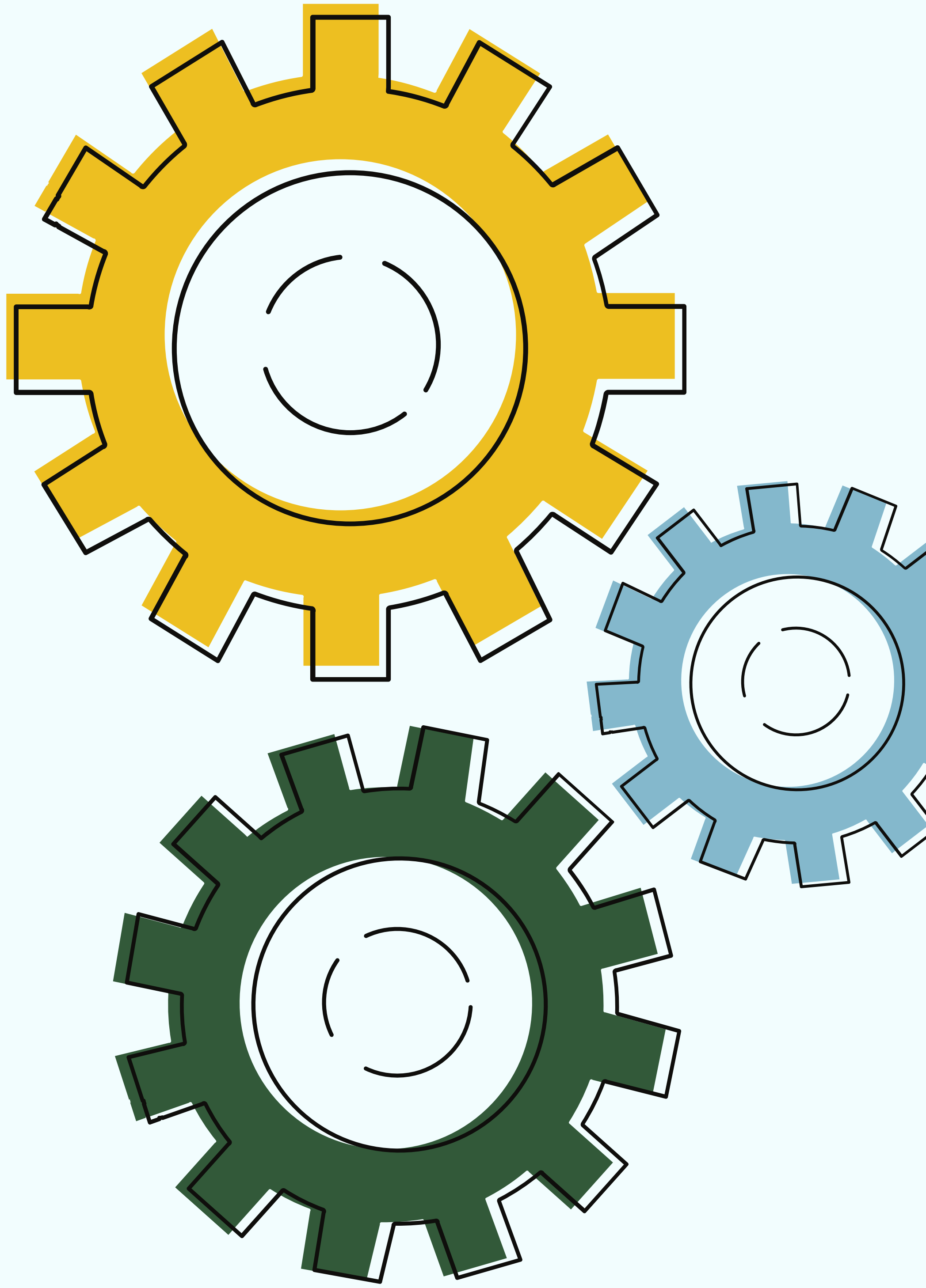


TEAM ClusterBusters

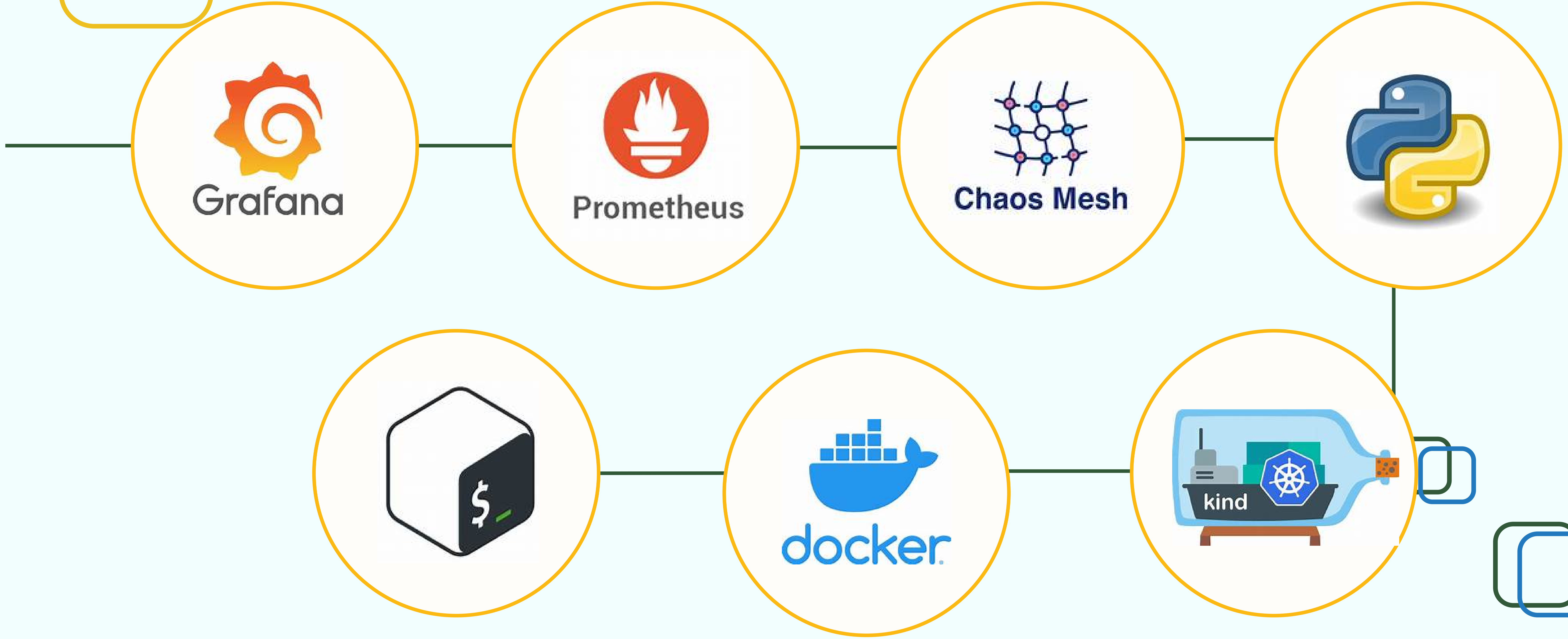
Amrita Vishwa Vidyapeetham, Coimbatore

Members

- C S Amritha
- Anaswara Suresh M K
- Avi Nair
- Adithya N S
- R. Sruthi



Techstack



Dataset Generation

Metrics Collection Using Prometheus

- Used Prometheus to scrape real-time metrics for Pods, Nodes, and Deployments.
- Created separate PromQL queries for each entity:
  - Nodes
  - Pods
  - Deployments

Event Collection Using kubectl get events

- Collected Kubernetes events using kubectl get events.
- Grouped events by Node, Pod, and Deployment for structured analysis.

Dataset Generation

Error Detection Functions

Passed metrics and event messages to corresponding error detection functions:

- ○ check_node_error(): Checks for node-level errors (e.g., resource exhaustion).
- ○ check_pod_error(): Checks for pod-level errors (e.g., crashes, restarts).
- ○ check_deployment_error(): Checks for deployment-level errors (e.g., scaling issues).
- ○ Each function appends error prediction values (e.g., True or False) to the dataset.

Dataset Generation

Data Aggregation

- Combined metrics and error predictions into a single row and stores separate dataset for each entity:

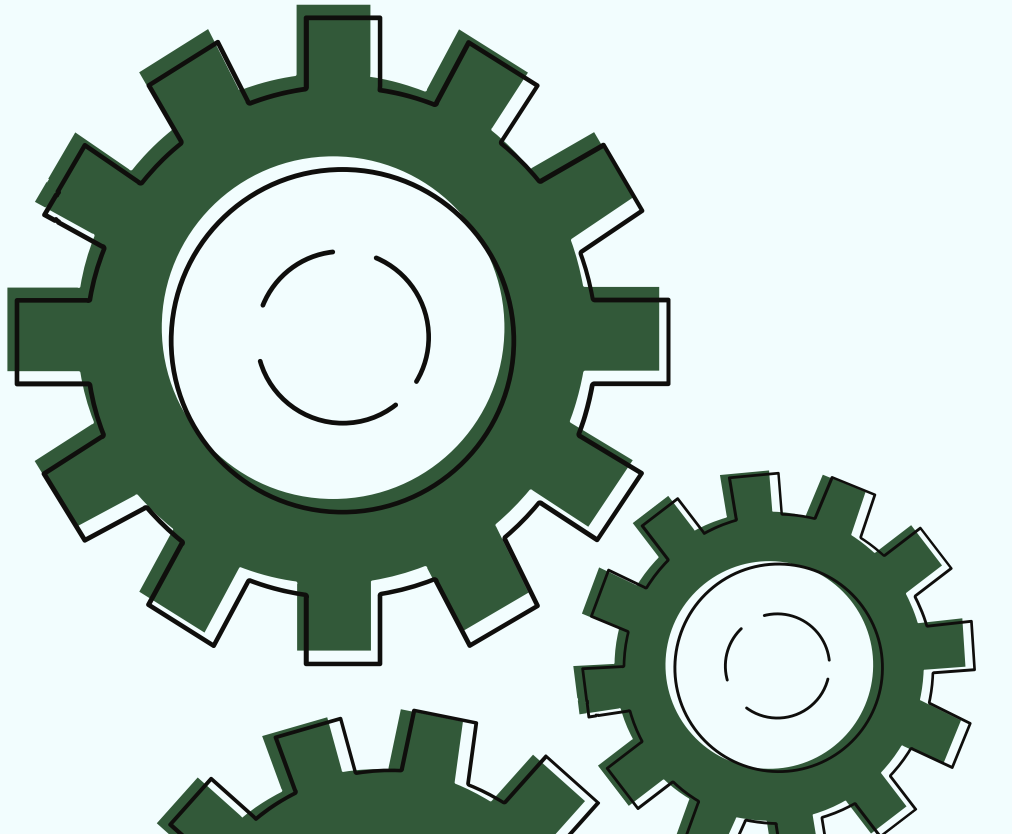
- Final dataset includes:
  - Metrics (CPU, memory, etc.).
  - Event messages.
  - Error prediction values.

Note: While data is collected for pods, nodes, and deployments, scaling to full node and deployment analysis is pending due to time constraints. Current focus remains on pod-level insights.

Model

For our Kubernetes failure prediction system, we experimented with multiple models such as:

- Facebook’s Prophet
- Long Short Term Memory with Isolation Forest
- GRU
- Random Forest



Previous Model:

GRU

Chosen for efficiency and short- to mid-term time-series dependencies.

New Model:

Random Forest

Why?

- Time-series forecasting less effective for pod health.
- good/bad pods may offset in aggregates.

Benefits:

Handles static metrics at a single point in time.

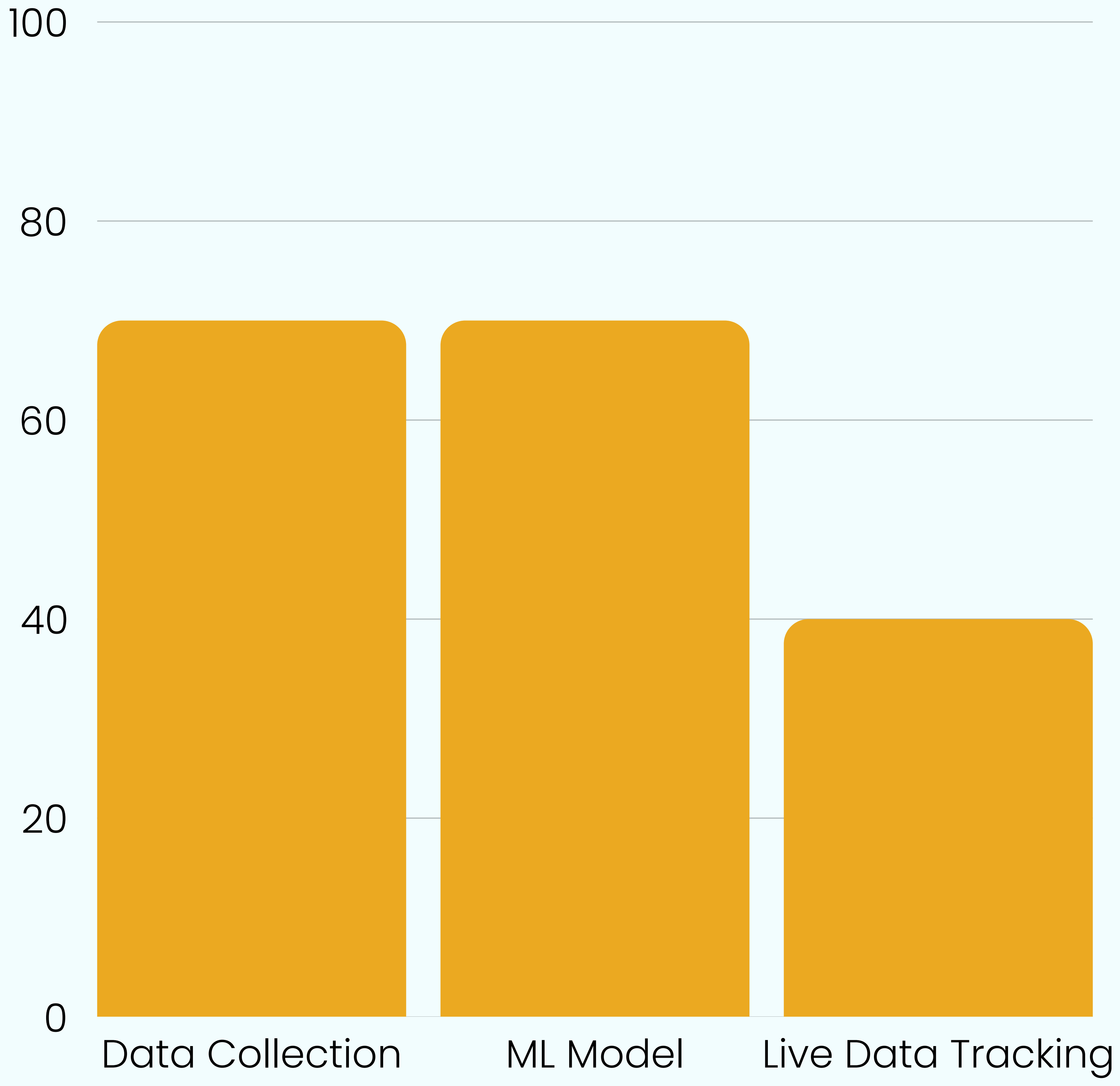
Robust for classification (Good/Alert/Bad).

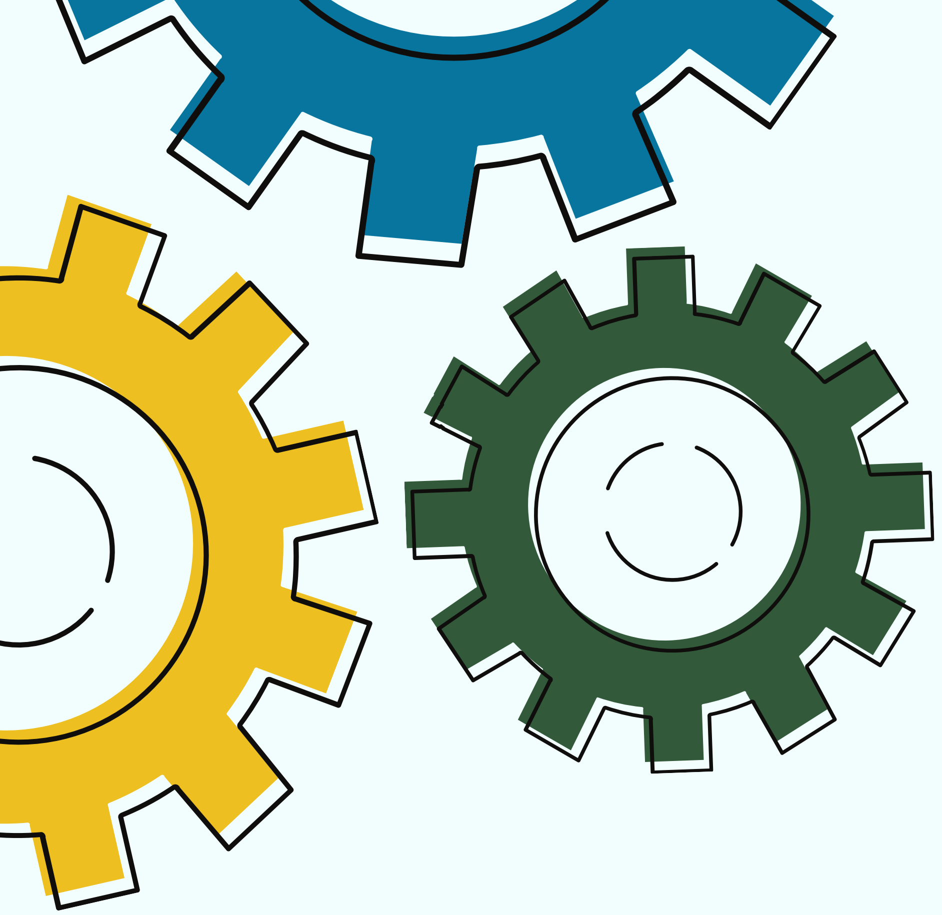
Predict pod failures using current metrics and error flags.

How Random Forest Is Used in the Failure Prediction System

1. Uses pod metrics and error flags at a single point in time.
2. Predicts pod health as Good, Alert, or Bad.
3. Learns from labeled data, balanced via undersampling for consistent performance.
4. Aggregates multiple decision trees to handle noisy metrics and reduce overfitting.
5. Provides feature importance and SHAP values to identify key failure drivers.

Current Status





Thank You!