# System State Forecast

## Predicting User State via Resource Metrics

January 31, 2026

System Forecast Team

# Contents

# 1 Project Overview

# Executive Summary

**Goal:** Classify computer state into **Idle**, **Interactive (Light)**, or **Media Watching** in real-time.

**Constraint:** Non-invasive monitoring. No key/mouse logging, only system resources (CPU, RAM, Disk, Network, GPU).

**Outcome:**

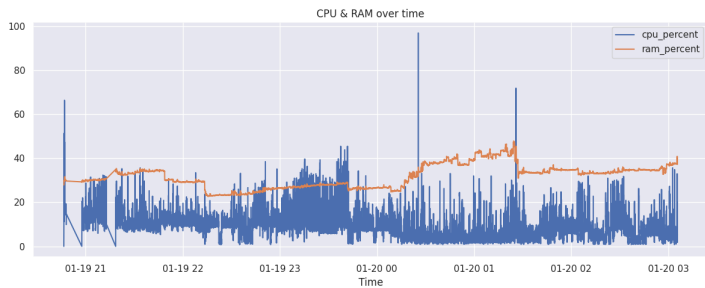
- Random Forest Classifier with 83% accuracy.
- Solved the "Idle vs. Reading" distinction using hardware-aware heuristics (GPU Sleep States).

# Phase 1: The Pivot

**Initial Goal:** Predict CPU/RAM usage to forecast heavy loads.

**The Problem:**

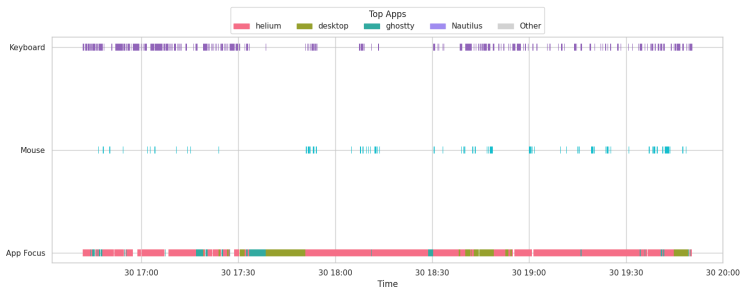
- **Skewed Distribution:** System is idle/low-usage 99% of the time.
- **Unpredictable Peaks:** Spikes are user-driven, not history-driven.



# 2 System Architecture

# Three Main Stages

1. **Data Collection:** Background daemon (`10_comprehensive_activity_log.py`) sampling at 1.0s.
2. **Feature Engineering:** Rolling windows (5s & 30s) to capture trends and variance.
3. **Live Inference:** Random Forest Model + Heuristic Overrides.

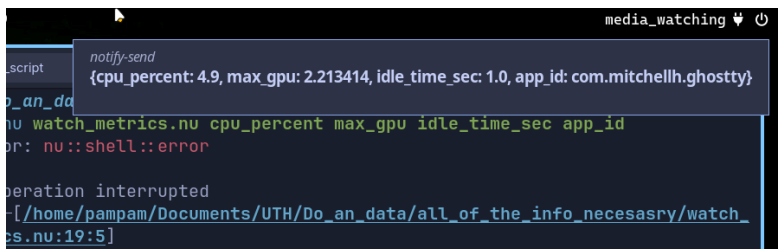


# Data Collection

**Metrics:**

- `psutil`: CPU, RAM, Disk I/O, Network.
- `intel_gpu_top`: RC6 (Sleep), RCS (3D), VCS (Video).
- **Ground Truth:** `libinput` (used only for training labels).

**Verification:** We used a `nu` script to visualize data flow in real-time notifications.



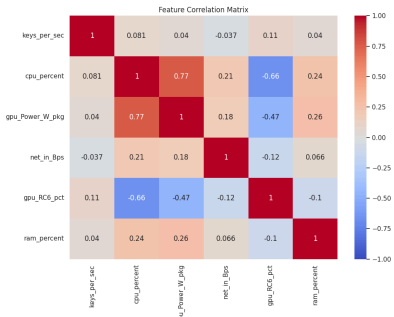
# 3 The Engineering

# Feature Engineering

Raw metrics are too volatile. We calculate **Rolling Features**:

- **Short Window (5s):** Immediate changes (e.g., “Start typing”).
- **Long Window (30s):** Sustained trends (e.g., “Watching Movie”).

**Key Insight:** Standard Deviation (`std`) is crucial. “Idle” has low variance; “Reading” has micro-bursts.



# The “Idle” Detection Challenge

**Problem:**

- **Idle:** CPU 1-2%, GPU Sleep 99%.
- **Reading:** CPU 2-3%, GPU Sleep 97%.
- Statistical distributions overlap > 80%.

**Solution: `max_gpu` & Heuristics**

- **Feature:** `max_gpu` (highest load on **any** engine).
- **Heuristic Override:**

```
if prediction == 'interactive_light':
    if (rc6 > 99.0 and cpu < 5.0) or (max_gpu < 1.0):
        prediction = 'Idle'
```

# 4 Model Performance

# Results

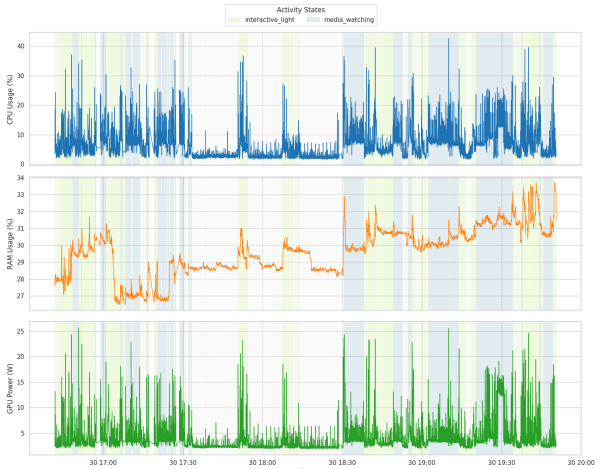
**Algorithm:** Random Forest Classifier

**Performance:**

- Accuracy: 83-87%
- High Precision for “Media Watching” (97%).
- “Idle” requires the heuristic override for reliability.

| Class | Precision | Recall | F1-Score |
|---|---|---|---|
| Idle | 0.70 | 1.00 | 0.82 |
| Interactive | 0.83 | 0.97 | 0.89 |
| Media | 0.97 | 0.77 | 0.86 |

# Visualizing State



# 5 Conclusion

# File Structure

- `10_comprehensive_activity_log.py`: Data Collector.
- `system_only_model/`:
  - `train_model.py`: Trainer.
  - `live_inference.py`: Production Inference.
  - `activity_model.joblib`: Trained Model.

# Future Improvements

- **Hysteresis:** Add a state latch (e.g., “Must be Idle for 3s to switch”) to prevent flickering.
- **Per-Process Analysis:** Hard-coded rules for known media players (mpv, vlc).
- **Profiles:** Different baselines for “Gamer” vs “Coder”.

**Thank You**