Emergent.sh Analytics Intern

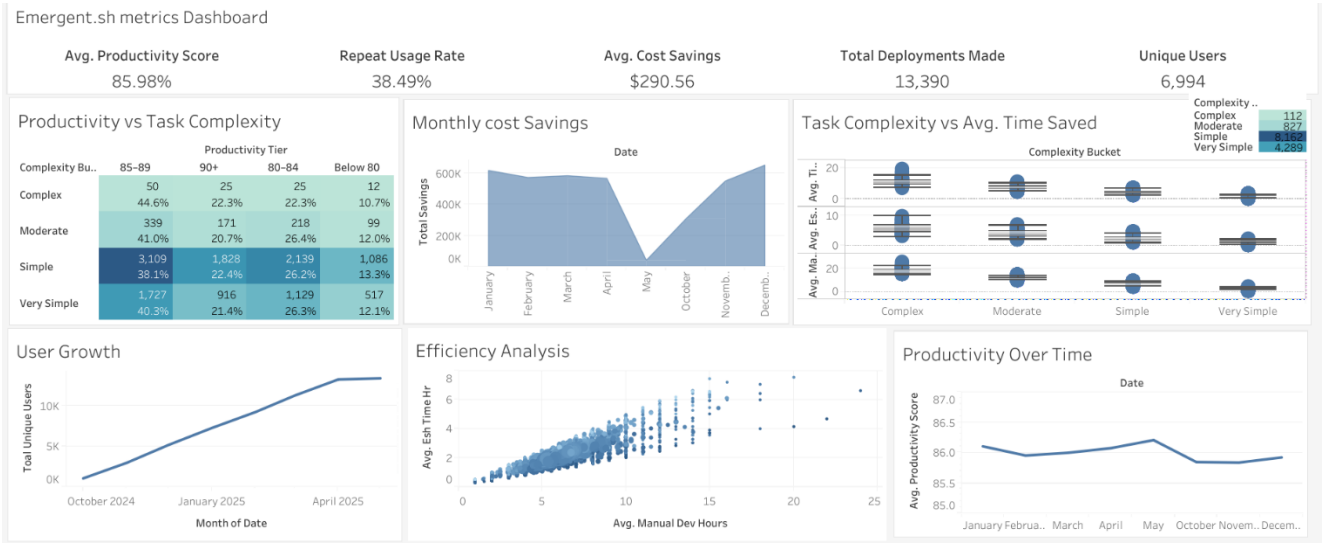
Gautam Manu

B.Tech CSE (2026)

Jaypee Institute of Information Technology, Noida-62

GitHub: https://github.com/gautam642/Emergent.sh-metrics

Tableau link:
https://public.tableau.com/views/Book1_17476387685290/Emergent_shmetricsDashboard?:language=en-US&:sid=&:redirect=auth&:display_count=n&:origin=viz_share_link



Agentic Insights Assignment

1. Assignment Overview

Objective: Pitch the Agentic (Emergent.sh) Platform to a data-savvy investor by demonstrating impact through three core lenses: effectiveness, stickiness, and ROI.

Deliverables:

- **Three key metrics** to prove product efficacy
- **Mock dashboard** (Tableau/Google Sheets) with synthetic data
- **Narrative insights** (three lines) interpreting the data

My approach involved: ideation via Gemini, synthetic event/data simulation, metric computation, and Tableau dashboard creation.

2. Key Metrics: Definitions, Formulae & Rationale

2.1 Agentic Productivity Score

- **Definition:** Composite index measuring quality (success rate) and speed (time saving).
- **Formula:** Productivity Score = 0.7 * Success Rate % + 0.3 * (1 - Avg Platform Time / Manual Dev Time) * 100
- **Rationale:** 70% weight on delivery quality (interruption-free runs), 30% on time savings—prioritizing reliability.

2.2 30-Day Repeat Usage Rate

- **Definition:** Proportion of users who run ≥2 projects within 30 days of their first usage.
- **Formula:** Repeat Usage Rate (%) = (Number of users with at least one revisit within 30 days / Total unique users) * 100
- **Rationale:** Gauge platform stickiness; top-quartile benchmarks are ~35–45% for dev tools.

2.3 Avg. Cost Savings per Deployment

- **Definition:** Dollar savings comparing manual vs. AI-assisted build cost (at $50/hr vs. $1/hr).
- **Formula:** Cost Savings = (Manual Hours * 50) - (Avg Platform Hours * 1)
- **Rationale:** Direct ROI metric for investors; highlights financial value delivered per project.

3. Data Generation & Preprocessing Pipeline

We synthesized a dataset of 13,390 AI builds spanning October 2024–April 2025.

3.1 Idea Generation via Gemini API

```
# High-volume ideation using Google Gemini (gemini-2.0-flash)
prompt = (
  f"Generate {batch_size} unique 1–2 line project ideas, "
  "with estimated manual dev hours (e.g., '4 hr'). Return JSON array."
)
# Rate-limited to 15 calls/min, 1499/day
# Dedupe via rapidfuzz;
```

- **Outcome:** project_ideas.csv with {id, idea, manual_dev_hours}.
- **Reasoning:** Demonstrate an AI-driven use case and seed realistic per-project complexity.

3.2 Cleaning & Enrichment

- **Load & parse JSON** into DataFrame.
- **Extract numeric hours** from strings; convert to float.
- **Drop duplicates** via fuzzy matching (threshold 90%).

3.3 Date & User Event Simulation

Assign random date within 2024-10-01 to 2025-04-30.

```
df['date'] = np.random.choice(pd.date_range('2024-10-01','2025-04-30'), len(df))
```

Simulate user signup & revisit events:

- Monthly cohorts (200 → 5,000 new users per month).
- 40% chance of forced "return" 5–30 days later.
- Additional visits via exponential inter-arrival (λ=1/15 days).

3.4 Data Smoothing & Cohort Assignment

1. Match each project to a user active on that date (exact or random fallback).

2. Recompute `repeat_flag`: revisit ≤30 days marked `1`.

3. Cap runs: max 5 projects per user in any rolling 7-day window; otherwise impose 7–30 day cooldown.

3.5 Metric Calculations

For each record:

baseline_cost_usd = manual_dev_hours * 50

esh_cost_usd = avg_time_hr * 1

cost_savings_usd = baseline_cost_usd - esh_cost_usd

success_rate_pct = max(0, 100 - success_flag*5)

productivity_score = clip(

  0.7*success_rate_pct + 0.3*(1 - avg_time_hr/manual_dev_hours)*100,

  0,100

)

Save final DataFrame to project_metrics_smoothed.csv.

4. KPIs & Dashboard Components

Below are the chosen visualizations and why each was selected to tell our story.

4.1 Top KPI Strip

- **Metrics:** Avg. Productivity, Repeat Usage %, Avg. Cost Savings, Total Deployments, Total Unique Users.
- **Purpose:** Instant snapshot of platform health and scale.

4.2 Productivity vs. Task Complexity Heatmap

- **Axes:** Complexity buckets (Very Simple → Complex) × Productivity tiers.
- **Why:** Shows quality distribution across task difficulty—evidence that even complex tasks largely succeed.

4.3 Monthly Cost Savings Area Chart

- **Y:** Total $ savings per month
- **Why:** Tracks financial impact over time; highlights seasonal lift and growth.

4.4 Task Complexity vs. Avg. Time Saved Scatter + Box Whisker

- **X:** Manual Dev Hours
- **Y:** Avg. Platform Time Hours
- **Why:** Visualizes absolute time saved by task size; boxplots reveal consistency and variance.

4.5 User Growth Line Chart

- **Series:** Monthly unique users
- **Why:** Demonstrates adoption trajectory and market fit velocity.

4.6 Efficiency Analysis Bubble Scatter

- **X:** Manual Hours, **Y:** Platform Hours, **Bubble size/color:** Productivity Score
- **Why:** Multi-dimensional view of per-project ROI—higher scores at lower Y/X ratios.

4.7 Productivity Over Time Line Chart

- **Y:** Avg. Productivity Score by month
- **Why:** Monitors platform quality stability and seasonality; assures long-term consistency.

5. Insights & Next Steps

- **Strong ROI:** ~$290 savings per deployment.
- **High reliability:** Avg. productivity ≈86%.
- **Healthy stickiness:** 38.5% 30-day repeat rate.
- **Growth opportunity:** Increase volume and top-tier performance on complex tasks.