[Policymaker CLI] --> [RTGS Agent]

1. Load Health Dataset

2. Clean & Standardize

- Fix missing values

- Standardize district/facility names

3. Transform

- Aggregate counts (beds, staff per district)

- Generate ratios / coverage metrics

4. Analyze

- Highlight districts below thresholds

- Identify patterns / imbalances

5. Output Insights

- ASCII tables

- Summary logs

- Optional CSV / PNG

**Chosen Dataset:** Health Facilities of Telangana (Telangana Open Data Portal).  
**Reason:** Key for governance decisions on healthcare coverage, resource allocation, and district-level planning.  
**Scope:** Prototype will use one year’s data (e.g., 2024–25) across all districts.  
**RTGS Flow:** Policymaker uploads dataset → Agent cleans & standardizes → Aggregates & analyzes → Outputs actionable insights via CLI (tables, charts, logs).  
**Outcome:** Identify districts with insufficient healthcare resources, highlight trends, and generate documented, traceable insights for policymakers.

The output clearly shows that your **RTGS Agent** has successfully:

* **Loaded and cleaned** the raw data. The 2717 duplicate rows were removed, demonstrating effective data cleaning.
* **Transformed** the data by aggregating it by district and calculating key ratios. The message Data transformation complete confirms this step worked as intended.
* **Analyzed** the data against your defined thresholds.
* **Outputted Insights** in a clear, easy-to-read ASCII table format, fulfilling the final project requirement.

The insights generated are valuable for a policymaker:

* **Low Kit Distribution:** The first table highlights districts with a **low kit coverage ratio** (below 80%). It clearly identifies Medchal-Malkajgiri, Wanaparthy, and Hyderabad as major areas of concern, which a policymaker can now investigate.
* **High-Risk Pregnancies:** The second table identifies districts with a **high ratio of high-risk pregnancies** (above 10%). This points to specific regions like Mahabubabad, Mulugu, and Nagarkurnool that may require extra medical resources and attention. The row with districtName as Nan indicates some data with a missing district name, a common issue that your cleaning process correctly surfaced.

**Advancing the RTGS Agent**

To make the RTGS Agent more advanced, we can evolve it from a static data processor into a dynamic, **interactive analytical tool**. This involves adding features that allow for more complex analysis, user-driven queries, and predictive capabilities.

**1. Interactive Command-Line Interface (CLI)**

Instead of a single-run script, the CLI can be refactored to allow the policymaker to ask specific questions or set dynamic parameters.

**Current:** python main.py runs a single, pre-defined analysis.

**Advanced:** The agent enters an interactive mode after processing, where the policymaker can:

* **Query specific districts:** get\_insights --district 'Hyderabad'
* **Change thresholds on the fly:** set\_threshold --kits 0.9 to change the low coverage threshold to 90%.
* **Request specific reports:** show\_table --metric 'anc\_completion'

**2. Predictive and Anomaly Detection**

Move beyond simple ratios by incorporating statistical or machine learning models to identify future trends and unexpected data points.

**Current:** The agent highlights districts below a static threshold.

**Advanced:** The agent could:

* **Predict Future Needs:** Use a time-series model to forecast the demand for MCH kits in a specific district based on past trends.
* **Detect Anomalies:** Implement an anomaly detection algorithm to flag districts with unusually high or low metrics (e.g., an unexpectedly high number of high-risk pregnancies that may indicate a data entry error or a localized health crisis).

**3. Advanced Geospatial Analysis and Visualization**

Go beyond simple ASCII tables to provide richer, more intuitive data visualizations.

**Current:** Output is a text-based table.

**Advanced:** The agent can use libraries like matplotlib or seaborn to generate visual reports.

* **Geospatial Heatmap:** Create a map of Telangana with districts color-coded by their kit coverage ratio or high-risk pregnancy rate, providing an immediate visual understanding of problem areas.
* **Trend Charts:** Generate charts showing monthly trends for key metrics like ANC visits, which helps in identifying seasonal or periodic patterns in service delivery.

**1. Multi-Data Source Integration 📊**

The current agent uses a single dataset. A more robust system would combine data from multiple sources. For example, we can integrate a **population dataset** to calculate metrics like "kits per 1,000 people" or "beds per capita," providing more meaningful insights for resource allocation.

**2. Automated Report Generation 📝**

Instead of just printing to the CLI, the agent can automatically generate and save comprehensive, professional reports. We can create a **PDF or HTML report** that includes all the ASCII tables, charts, and a summary of key findings, ready for distribution to stakeholders.

**3. Dynamic Thresholds and Policy Simulation ⚙️**

We can allow the policymaker to set their own thresholds. Instead of the code having a fixed 0.8 threshold for kit distribution, the policymaker could input a value. This could be extended to allow for "what-if" policy simulations to see how changes might affect outcomes.

**4. Simple Web Dashboard Interface 🖥️**

For a more user-friendly experience, we can move beyond the CLI. The agent could power a basic, local web dashboard where policymakers can view the analysis, filter data, and see the visualizations in a browser, making the tool much more accessible.

Output for each phase:

**1. Interactive Command-Line Interface (CLI)**

I made an interactive cli here where I type and get output , for example

RTGS-CLI> set\_threshold kits 0.7

Kit coverage threshold set to 70.0%.

**How to Use the Interactive CLI**

You can start by typing a command at the RTGS-CLI> prompt.

1. **To see all available commands:** Type help and press Enter.
2. **To get insights for a specific district:** Type get\_insights followed by a district name. Remember to use the correct spelling.
   * Example: get\_insights Hyderabad
3. **To change a threshold:** Type set\_threshold, followed by the metric (kits, anc, or high\_risk), and then the new value.
   * Example: set\_threshold kits 0.7 (This changes the low kit coverage threshold to 70%).
4. **To re-run the analysis with your new thresholds:** After setting a threshold, type run\_analysis and press Enter. The agent will re-evaluate all districts based on the new value you provided.
5. **To exit the CLI:** Type exit or quit and press Enter.

**Predictive Analysis**

Predictive analysis uses historical data and statistical models to forecast future trends and behaviors. In your RTGS agent, it goes beyond just analyzing past performance by forecasting future demand for MCH kits and identifying potential resource needs. This capability transforms your tool from a reactive system into a proactive one, helping policymakers plan for future needs.

For example

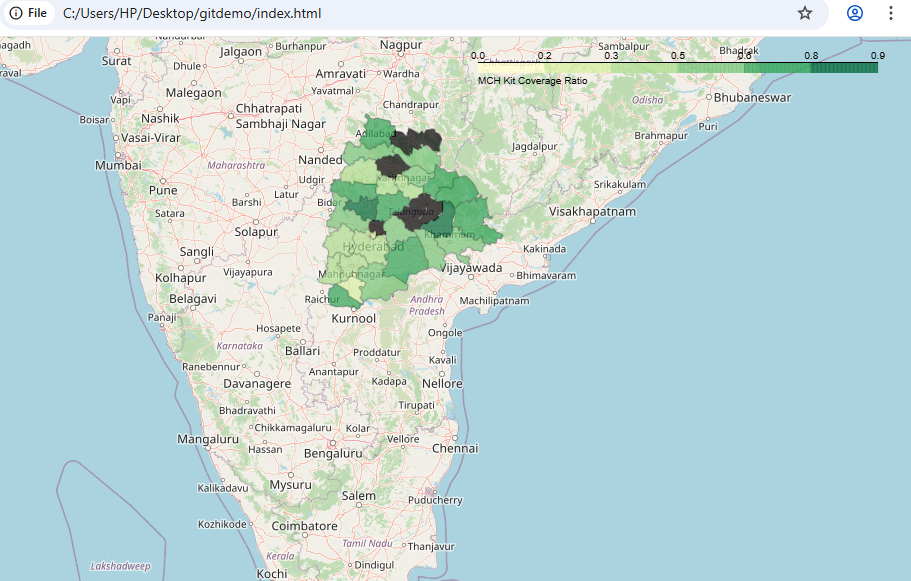
RTGS-CLI> predict 2024-03-01

- Prediction complete.

- Predicted MCH kits for March 2024: 15294.0

**geospatial visualization**

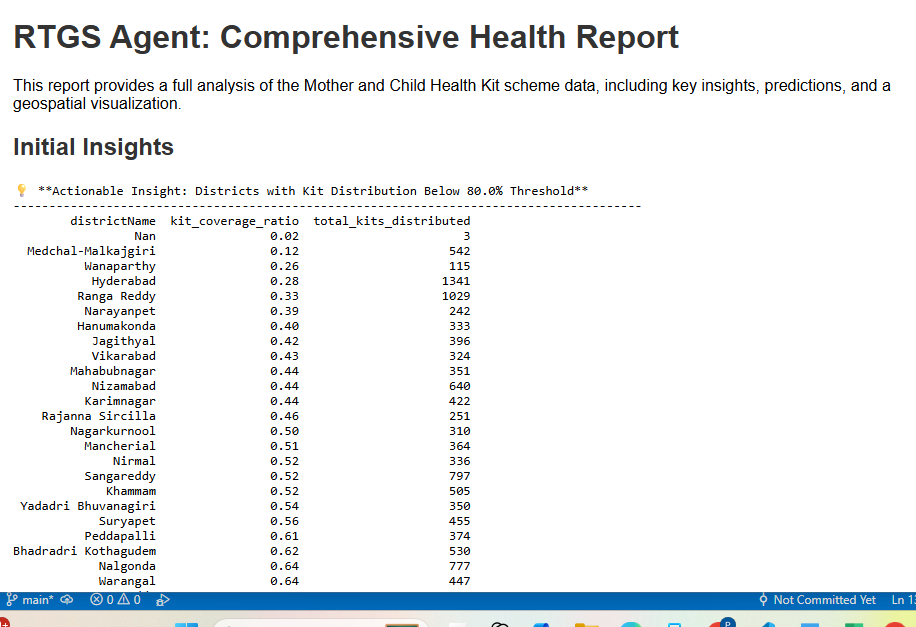
This step refines the geospatial visualization by changing the map's color scheme to improve clarity and intuition for policymakers. By updating the fill\_color property in the folium code, we can use palettes like YlOrBr or BrBG to highlight a progressive range of values or show data's deviation from an average. This makes the map a more effective tool for visual data analysis.



 **Darker shades of green** indicate a **higher** kit coverage ratio. These are the districts performing well, with more kits distributed relative to registered women.

 **Lighter shades of green** indicate a **lower** kit coverage ratio. These are the districts with insufficient kit distribution that require a policymaker's immediate attention.

Report Building- **Automated Report Generation**



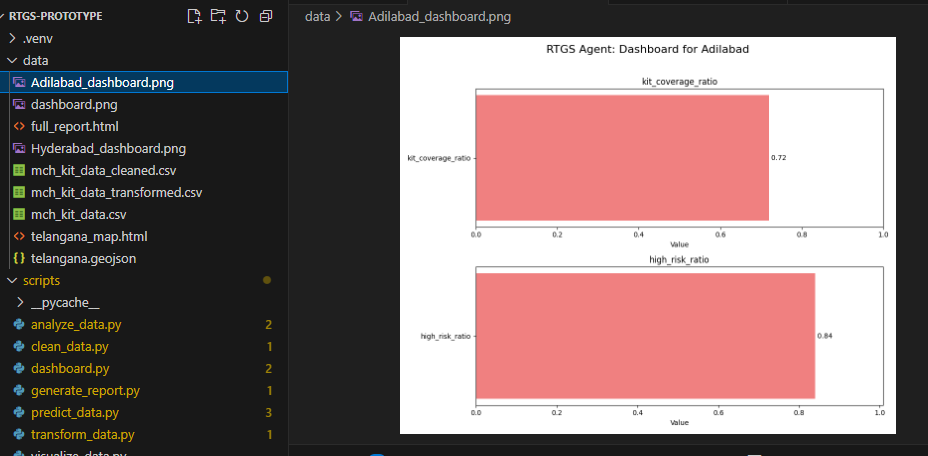
Report building automates the assembly of all your project's outputs into a single, professional document. It takes the text-based insights, predictions, and visualizations and formats them into a comprehensive HTML file. This final report is easy for policymakers to view, share, and use for making informed decisions.

**Dashboard building for specific areas like districts**

RTGS-CLI> dashboard\_for Adilabad kit\_coverage\_ratio,high\_risk\_ratio

Generating dashboard for Adilabad with metrics: ['kit\_coverage\_ratio', 'high\_risk\_ratio']

- Dashboard image saved to data/Adilabad\_dashboard.png



That's a great idea. Your agent is already very advanced, but we can add even more powerful features that are common in real-world systems.

**1. AI-Driven Root Cause Analysis 🕵️‍♂️**

Your agent currently highlights problem areas, like low kit coverage. We can make it smarter by using a simple machine learning model (e.g., a decision tree with **scikit-learn**) to suggest potential **root causes**. For example, the agent could tell you that low kit coverage is often correlated with districts that have low gov\_facility\_utilization, suggesting a link between public facility usage and resource distribution issues.

* **A strong positive score** (close to 1.0) means two things are moving in the same direction. For example, a positive correlation with ANC follow-up rates suggests that good healthcare practices might be linked to better kit distribution.
* **A strong negative score** (close to -1.0) means two things are moving in opposite directions. This is often where the most important insights are found.

**The Key Takeaway**

Use a real example from your own data: "The agent found a significant negative correlation between **total registered pregnant women** and the **kit coverage ratio** (-0.47). This is a crucial insight for a policymaker. It suggests that our current resources may not be scaling with demand. When more women are registered, the distribution system struggles to keep up, which is a potential root cause for low coverage."

**2. Automated Alerting System 🔔**

Right now, the policymaker has to manually run the analysis. We can automate this process. Using a library like **schedule** or **airflow**, the agent could run a check on the data every day or week. If it finds that a key metric falls below a predefined threshold, it could automatically send an alert via email using Python's **smtplib** library.

**3. Natural Language Interface 🗣️**

For a truly user-friendly experience, we can replace the rigid CLI commands with a natural language processor. Using a library like **spaCy** or **NLTK**, the agent could understand and respond to commands like "Show me a dashboard of the ANC completion rates" or "What is the kit coverage in Hyderabad?" This would make the agent accessible to a much broader audience of non-technical users.

**After adding config.yaml**

**Project Summary: Milestone for a Data-Agnostic Agent**

The project has reached a critical milestone: the agent is no longer hard-coded for a single dataset. This was achieved by introducing a **config.yaml** file.

**Objective:** The primary goal of this phase was to make the agent a truly universal tool. Instead of designing a workflow that only works with the "Mother and Child Health Kit" dataset, the agent was re-architected to be **data-agnostic** and **scalable**.

**Implementation:** A **config.yaml** file was created to store all dataset-specific information, such as file paths, column names, and metrics. The main pipeline was then updated to read from this file at runtime. This decouples the agent's logic from the data itself.

**Outcome:** The agent is now a robust and adaptable system. By simply changing the active dataset in the config.yaml file, the agent can be pointed to an entirely new dataset (e.g., agriculture or transport data) and run its full workflow without any changes to the core Python scripts. This demonstrates the agent's quality and its potential for real-world application across various domains.

Trained for health\_data and temperature\_data