Technical Documentation

dataNow! Data Analysis Chatbot

Naveen Prabakar

August 3, 2025

1 Overview

dataNow! is an intelligent chatbot platform designed to facilitate data exploration, visualization, statistical analysis, and machine learning operations via a user-friendly chat interface. Built with Flask, it leverages popular Python data science libraries and seamlessly integrates with MongoDB for robust, persistent storage of large datasets.

2 Architecture

2.1 Technology Stack

- **Backend Framework:** Flask (Python)
- **Data Science Libraries:** pandas, seaborn, matplotlib, numpy, scipy, scikit-learn
- **ML Integrations:** scikit-learn (ML models), OpenAI API (code/insight generation), Gemini API (graph/table analysis)
- **Database:** MongoDB (pymongo)
- **Session Management:** Flask session
- **Frontend:** HTML rendered with Flask's Jinja2 templating
- **Miscellaneous:** pdfkit (PDF generation), Pillow (Image processing)

3 Core Features

3.1 Data Upload & Storage

- **Endpoint:** `/upload` (POST)
- **File Support:** CSV
- **Storage Logic:**
  - Small files are serialized and stored in the Flask user session.
  - Larger files (over `MAX_SESSION_SIZE`) are stored in MongoDB as JSON.
- Upload and temporary folders are created if they do not exist.

3.2 Chat Interface

- **Endpoint:** `/` (GET)
- Initializes with a welcome message, example commands, and resets session data.

3.3 Data Query, Insight, and Visualization

1. **Data Processing**
    - User commands are sent to the OpenAI API with dataframe info and description.
    - OpenAI returns only the Python code required.
    - Code is executed securely on the server in a restricted namespace.
2. **Output Handling**
    - **Tabular:** Converted to HTML and tracked for report inclusion.
    - **Visualizations:** Saved as images (Base64) for embedding.
    - **Other:** Returned as plain messages or text.
3. **Examples of Supported Prompts**
    - Data exploration (showing rows, summaries)
    - Data manipulation (sort, filter)
    - Visualization (bar, line, box, scatter, heatmap)
    - ML tasks (regression, classification, clustering)
    - Statistical analysis (mean, t-tests, correlations)

3.4 AI-Powered Table & Graph Analysis

- **Functions:** `analyze_table`, `analyze_graph`
- **API:** Gemini (Google Generative AI)
- **Input:** Saved table (HTML) or plot (PNG)
- **Output:** Expert textual insights added to exported PDF reports

3.5 Report Generation

- **Endpoint:** `/download_pdf`
- All tables and graphs, with AI-generated commentary, are combined into an HTML report and converted to PDF.

3.6 Chat History Retrieval

- **Endpoint:** `/get_chat_history`
- Returns session chat history in JSON.

3.7 Helper Commands

- `!help` — Returns sample prompts.
- `!info` — Returns information about the system.

4 Session and State Management

- **Session Variables:** Store DataFrame, chat history, results, and prompts.
- **Global Variables:** `lista` (all outputs for PDF), `prompts` (paths of tables/graphs).

5 Security Considerations

- **API Keys:** Loaded securely from environment variables.
- **Code Execution:** Restricted environment, no Python imports allowed in executed code.
- **File Operations:** Checks and gracefully handles file and data errors.

6 Customization and Extension

- Easily add endpoints for new analyses.
- MongoDB supports large-scale data.
- UI and backend extensible for advanced features and templates.

7 Deployment

- **Default**: Flask server at `0.0.0.0:10000`
- **Deployment Recommendation**: Run behind a reverse proxy (e.g., nginx), enable HTTPS, use production environment variables.

8 Example Workflow

1. User uploads CSV to `/upload`.
2. DataFrame gets stored appropriately.
3. User sends a data analysis command.
4. Server queries LLM API, executes code, and returns output (table, plot, or analysis).
5. User downloads report with embedded charts and AI insights.

9 Notes

- Designed for analysts, students, and business users; no code required on the client side.
- Modular and ready for extension by engineering teams or individuals.