Project Documentation

Table of Contents

1. Introduction
   - Project Overview
   - Purpose
   - Scope
2. Architecture and Technologies
   - Backend
   - Frontend
   - Database
   - Websockets
3. Folder Structure
4. Setup and Installation
   - Prerequisites
   - Installation Steps
5. Usage
   - Starting the Application
   - Accessing the Application
6. Features
   - Node Spawning and Inference (Alpha Testing)
   - Water Discharge Case File Generation
7. Testing
   - Unit Testing
   - Alpha Testing
8. Contributing
9. Known Issues
10. Future Enhancements
11. Acknowledgements
12. License

1. Introduction

Project Overview

The project is a web application centered around Bayesian networks and water discharge modeling. Developed using Python and the Flask web framework, the application allows users to spawn nodes in a Bayesian network, with an alpha testing phase for inferring relationships between nodes. Additionally, users can generate water discharge case files through a specific endpoint, which can be imported into Netica to influence the network. The project leverages technologies such as SQLite3 for the database, Flask-SocketIO for WebSocket communication, and a combination of JavaScript, HTML, and CSS for the frontend. Deployment is handled through Railway, and the project is version-controlled on GitHub. The scope includes both backend and frontend components, providing a comprehensive platform for users to explore and experiment with Bayesian networks in the context of water discharge modeling.

Purpose

The purpose of this project is to provide a user-friendly web application for exploring Bayesian networks and conducting water discharge modeling. By leveraging Python and the Flask web framework, the project aims to empower users to dynamically spawn nodes within a Bayesian network, facilitating alpha testing for the inference capabilities between these nodes. Additionally, the project offers functionality to generate water discharge case files through a dedicated endpoint. These generated case files can be seamlessly imported into Netica, influencing the underlying network. The overarching goal is to create a comprehensive platform that combines data analysis with Bayesian network modeling, providing users with a toolset for understanding and experimenting with the intricacies of water discharge in a dynamic and interactive environment.

Scope

1. Node Spawning and Inference (Alpha Testing)

In Scope:

- The project includes a feature for users to spawn nodes.
- There is an alpha testing phase for the inference functionality between nodes.

Out of Scope:

- Full production-ready release of the inference feature.
- Details about the specific types of nodes or the complexity of the Bayesian network.

2. Water Discharge Case File Generation

In Scope:

- Users can generate a water discharge case file using a specific endpoint.
- The generated case file can be imported into Netica to affect the network.

Out of Scope:

- Detailed information about how Netica processes the case file.
- Integration with other tools or platforms beyond the specified functionality.

3. General Project Scope

In Scope:

- Use of Python for the backend, Flask for the web framework, SQLite3 for the database, and web technologies (JavaScript, HTML, CSS) for the frontend.
- Deployment on Railway and version control using GitHub.

Out of Scope:

- Detailed information about specific algorithms used in the Bayesian network.
- Deep-dive into the Flask-SocketIO implementation.

2. Architecture and Technologies

Backend

- **Programming Language:** Python, Java, NodeJs
- **Web Framework:** Flask
- **Websockets:** Flask-SocketIO
- **Additional Libraries:**
- **Deployment Platform:** Railway
- **Version Control:** GitHub

Frontend

- **Languages:** JavaScript, HTML, CSS
- **Frontend Frameworks/Libraries:**
- **Websockets Implementation:**

Database

- **Database System:** SQLite3
- **ORM (Object-Relational Mapping):** (Specify if an ORM like SQLAlchemy is used)
- **Database Interactions:**

Websockets

- **Library/Framework:** Flask-SocketIO
- **Purpose:** (Explain the use of websockets in the project, such as real-time communication or other functionalities)
- **Integration with Frontend:** (Describe how websockets interact with the frontend for specific features)

3. Folder Structure

The project follows a structured folder organization to enhance code readability and maintainability:

1. **`.vscode`:**

   - Configuration files for Visual Studio Code.

2. **`docs`:**

   - Documentation files related to the project.

3. **`static`:**

   - **`javascripts`:**
     - Client-side JavaScript files for enhancing web page interactivity.
   - **`stylesheets`:**
     - CSS files for styling the web pages.

4. **`templates`:**

4. Setup and Installation

Prerequisites

Before setting up the project, ensure that the following prerequisites are installed:

1. **Python:**
    - Install Python using conda.
2. **Conda:**
    - Verify that conda is installed by running:

      ```
      conda --version
      ```

      If not installed, you can install Miniconda by following the instructions here.
3. **Virtual Environment:**
    - Create a virtual environment using conda:

      ```
      conda create --name your_environment_name python=3.8
      ```
4. **Dependencies:**
    - Activate the virtual environment:

      ```
      conda activate your_environment_name
      ```
    - Install project dependencies:

      ```
      conda install -c conda-forge flask flask-socketio pandas sqlalchemy  # Add other dependencies as needed
      ```
5. **Database Setup:**
    - SQLite3 is used as the database. No additional setup is required.
6. **Version Control:**
    - Install Git (if not already installed) to manage version control:
        - Git Installation Guide
7. **Railway CLI:**
    - If deploying on Railway, install the Railway CLI:

```
conda install -c conda-forge railway
```

8. **Text Editor or IDE:**
    - Choose a text editor or integrated development environment (IDE) for code editing. Visual Studio Code is recommended and can be configured for Python development.

These prerequisites ensure a smooth setup and execution of the project in a conda environment. Adjust versions and dependencies based on your specific project requirements.

Installation Steps

Follow these step-by-step instructions to set up the project:

1. **Clone Repository:**

```
git clone https://github.com/Daniel-Kenan/BayesianNetworks.git
cd your-repo
```

5. Usage

Starting the Application

[Explain how to start the application.]

Accessing the Application

[Provide information on how users can access the application.]

6. Features

Node Spawning and Inference (Alpha Testing)

During the alpha testing phase, users can explore the following features related to node spawning and inference:

1. **Node Spawning:**
    - Users have the ability to dynamically spawn nodes within the Bayesian network.
    - This feature is currently in alpha testing to gather feedback on user interactions and potential improvements.
2. **Inference (Alpha Testing):**
    - The alpha testing phase includes functionality for making inferences between the spawned nodes.

- Users can experiment with the inference capabilities, and feedback is encouraged to refine and enhance this feature for a future release.

Water Discharge Case File Generation

The water discharge case file generation feature allows users to perform the following actions:

1. **Case File Generation Endpoint:**
    - Users can utilize a specific endpoint to generate a water discharge case file based on natural flows and other relevant parameters.
    - The generated case file is intended to be imported into Netica, influencing the Bayesian network related to water discharge modeling.
2. **Integration with Netica:**
    - The generated case file is designed to be seamlessly integrated with Netica, providing users with a practical tool for affecting and observing changes in the Bayesian network.

These features collectively contribute to the exploration and experimentation with Bayesian networks, specifically in the context of water discharge modeling.

7. Testing

Unit Testing

Unit testing is an integral part of ensuring the reliability and functionality of the project. The following outlines the unit testing process:

1. **Testing Framework:**
    - Unit tests are implemented using a testing framework compatible with Python, such as `pytest` or the built-in `unittest` module.
2. **Test Coverage:**
    - Unit tests cover critical functionalities, including node spawning, inference, and other relevant components.
    - Aim for comprehensive coverage to identify and address potential issues in the codebase.
3. **Automation:**
    - The unit testing process is automated, allowing for regular and consistent testing during development.

Alpha Testing

The alpha testing phase is focused on gathering user feedback and refining the features related to node spawning and inference. Key points for the alpha testing phase include:

1. **User Participation:**

- Alpha testing invites users to actively participate in exploring the node spawning and inference features.
- Users are encouraged to provide feedback on usability, functionality, and any issues encountered.

2. **Feedback Collection:**

    - Establish a systematic approach for collecting user feedback, which may include surveys, feedback forms, or direct communication channels.

3. **Iterative Improvement:**

    - Based on user feedback, iterate on the features to address identified issues and enhance the overall user experience.

4. **Communication:**

    - Maintain clear communication with alpha testers, keeping them informed about updates, changes, and the project's progress.

The alpha testing phase plays a crucial role in refining the project, ensuring that user perspectives are considered in the development process.

8. Contributing

[Guide on how contributors can participate in the project.]

9. Known Issues

We are still finding it hard to link java api to python . java version for netica is not compatible with most laptops .

10. Future Enhancements

The project has a solid foundation, and there are several potential enhancements that can be explored in future iterations:

1. **Enhanced Node Spawning:**

    - Expand node spawning capabilities to include more complex node types and relationships.
    - Introduce a user-friendly interface for configuring node properties.

2. **Production-Ready Inference:**

    - Further develop and refine the inference functionality for a production-ready release.
    - Incorporate advanced algorithms and optimizations for more accurate and efficient inference.

3. **Extended Water Discharge Modeling:**

    - Integrate additional parameters and data sources for a more comprehensive water discharge modeling experience.
    - Explore machine learning techniques to enhance the accuracy of water discharge predictions.

4. **User Authentication and Authorization:**

    - Implement user authentication to secure the application and provide personalized experiences.
    - Introduce role-based access control for different user levels.

5. **Interactive Visualization:**

    - Incorporate interactive data visualization tools to help users better understand the Bayesian network and water discharge patterns.
    - Utilize libraries like Plotly or D3.js for dynamic visualizations.

6. **Integration with External Tools:**

    - Explore integrations with other data analysis and modeling tools beyond Netica.
    - Allow users to export and import data from popular formats.

7. **Scalability and Performance Optimization:**

    - Optimize the application for scalability, allowing it to handle larger datasets and more complex Bayesian networks.
    - Implement caching mechanisms and performance enhancements.

8. **Continuous Integration and Deployment:**

    - Set up continuous integration pipelines to automate testing and deployment processes.
    - Explore additional deployment platforms and containerization for increased flexibility.

These future enhancements aim to take the project to the next level, providing users with an even more powerful and feature-rich platform for Bayesian network exploration and water discharge modeling.

11. Acknowledgements

Thank you Dr Bembe, Theophilus , Prof Gordon , Ms Mellisa Wade

12. License

[Specify the project's license.]