Design Document

Patient Readmission Prediction

Rohith R - **EP21B030**
Afreen - **NA21B050**

April 2025

1 High-Level Design

- Data is ideally ingested on a periodic basis (e.g., daily or weekly) from an API endpoint. However, due to access restrictions—such as paid APIs requiring industry credentials—we used static data for demonstration purposes. While suboptimal, this allowed us to showcase the pipeline functionality within course constraints.
- Perform thorough Exploratory Data Analysis (EDA) and identify potential features to engineer.
- Automate the data ingestion and preprocessing (including feature engineering) using Apache Airflow DAGs.
- Track all versions of data using Data Version Control (DVC).
- Conduct model experimentation using MLflow. Log all relevant metrics, hyperparameters, artifacts, and data versions for reproducibility.
- Deploy the best-performing model using FastAPI as the backend service. A Streamlit-based frontend provides a user-friendly interface for end users.
- Monitor system-level metrics using Node Exporter and Prometheus. A Grafana dashboard is used for visualization and real-time monitoring.
- Implement unit tests for all modular components to ensure reliability and correctness.

The machine learning pipeline is visually summarized in the flowchart file: `flowchart.drawio`.

2 Low-Level Design

Refer to `project_structure.txt` for an overview of the directory layout.

- `airflow/dags`: Contains two DAG scripts—`load_data.py` and `process_data.py`—as well as a utility script `utils/add_feature.py` for feature engineering. Airflow is containerized using the `docker-compose.yaml` file located in the `airflow` directory. Although our use case involves static data, the DAGs are designed with periodic scheduling capabilities for scalability.
- `data`: Stores both raw and processed datasets. Includes the `.git.dvc` configuration used for data versioning.
- `models`:
  - `models/config`: Contains a YAML configuration file listing various models and hyperparameters to be tested.
  - `models/training`: Includes training scripts that log experiments via MLflow. Outputs include the best-performing model, label encoders, and scalers, which are later used for inference and stored in the `serving` directory.
- `monitoring`: Includes Prometheus, Node Exporter, and the JSON configuration for the Grafana dashboard used to visualize metrics.
- `notebooks`:
  - `notebooks/analysis_src`: Contains auxiliary scripts used during the analysis phase.
  - `EDA.ipynb`: The main EDA notebook.
  - `Test.ipynb`: Used for validating feature engineering logic.
- `serving`:
  - `serving/backend`: Contains the FastAPI app and its Dockerfile. Provides endpoints for both single and batch predictions.
  - `serving/frontend`: Contains the Streamlit app and its Dockerfile for a graphical user interface.
  - `serving/docker-compose.yaml`: Used to launch both the frontend and backend services together.
- `tests`: Contains unit test scripts covering various modular functions to ensure correctness and maintainability.