Diabetic Readmission Risk Prediction

Design Document

Jonathan Agustin Zack Robertson Lisa Vo

University of San Diego

Contents

1 Introduction

Hospital readmissions, particularly within 30 days of discharge, represent a significant challenge to healthcare systems worldwide, leading to increased costs and potentially adverse patient outcomes. Diabetic patients are especially vulnerable to readmission due to the chronic and complex nature of diabetes management. This document outlines the design of a machine learning (ML) system aimed at predicting 30-day hospital readmissions among diabetic patients. By leveraging historical patient data and advanced analytical techniques, the system seeks to identify high-risk individuals, enabling proactive interventions and contributing to improved healthcare efficiency and patient well-being.

2 Problem Statement

The rising rate of hospital readmissions among diabetic patients presents a multifaceted problem for healthcare providers. It strains resources, increases healthcare expenditures, and can negatively impact patient health and quality of life. The complexity of diabetes management, potential complications, and the need for strict adherence to treatment regimens contribute to the elevated readmission risk for this population.

This project addresses the challenge of accurately predicting 30-day hospital readmissions among diabetic patients. By developing a robust and scalable machine learning system, we aim to analyze a wide range of patient attributes—including demographics, medical history, treatment patterns, and hospital procedures—to identify individuals at high risk of readmission. This predictive capability will empower clinicians to implement targeted interventions, potentially reducing readmission rates and improving overall patient care.

3 Impact Measurement

To assess the effectiveness of the ML system, we will employ a comprehensive set of metrics and analyses. These measures are designed to evaluate not only the technical performance of the model but also its real-world impact on healthcare delivery.

3.1 Model Performance Metrics

The predictive accuracy and reliability of the model are critical for its adoption in clinical settings. We will utilize the following standard classification metrics:

- **Accuracy**: Measures the proportion of correct predictions out of all predictions.
- **Precision**: Assesses the model's ability to correctly identify true positive cases among all positive predictions.
- **Recall (Sensitivity)**: Evaluates the model's capacity to identify all actual positive cases.
- **F1-Score**: Represents the harmonic mean of precision and recall, providing a balance between the two.
- **ROC-AUC Score**: Quantifies the model's ability to distinguish between classes by plotting the true positive rate against the false positive rate.

These metrics will be calculated on a held-out test dataset to provide an unbiased evaluation of the model's performance.

3.2 Reduction in Readmission Rates

A key goal of deploying the ML system is to achieve a measurable reduction in readmission rates among diabetic patients. We plan to:

- **Establish a Baseline**: Determine the current readmission rate prior to implementing the ML system.
- **Post-Implementation Monitoring**: Track readmission rates following deployment to assess changes.
- **Target Reduction**: Aim for at least a 10% decrease in 30-day readmission rates, indicating a significant positive impact.

Comparing these rates before and after implementation will help us assess the practical benefits of the system in a real-world healthcare environment.

3.3 Cost Savings Analysis

Reducing readmissions can result in substantial cost savings for healthcare institutions. We will analyze:

- **Direct Cost Savings**: Calculate the reduction in expenses associated with fewer hospital stays, treatments, and procedures for readmitted patients.
- **Indirect Cost Savings**: Consider savings from improved resource allocation, such as reduced burden on hospital staff and facilities.
- **Return on Investment (ROI)**: Evaluate the financial benefits in relation to the costs incurred in developing and deploying the ML system.

This analysis will provide insights into the economic value of the system for healthcare providers.

3.4 Resource Optimization

Efficient utilization of healthcare resources is essential for enhancing patient care while controlling costs. We will examine:

- **Bed Occupancy Rates**: Assess changes in bed availability and whether reduced readmissions lead to better management of hospital capacity.
- **Staff Workload**: Evaluate the impact on healthcare provider workloads, aiming for a more balanced distribution of patient care duties.
- **Preventive Care Allocation**: Observe whether resources can be reallocated towards preventive measures and patient education due to reduced readmissions.

By monitoring these indicators, we can validate the system's contribution to overall healthcare efficiency and patient care quality.

4 Security, Bias, and Ethical Considerations

In developing an ML system for healthcare, it is imperative to address security, bias, and ethical concerns comprehensively. Patient data is sensitive and subject to strict regulations, and the ML models must be fair and transparent in their predictions.

4.1 Security Checklist

- **Data Privacy Compliance**: Compliance with the Health Insurance Portability and Accountability Act (HIPAA) is mandatory. We will implement strict data governance policies to ensure patient confidentiality. Data de-identification techniques will remove personally identifiable information (PII), and any data sharing will adhere to consent agreements and legal requirements.
- **Secure Data Storage**: All data will be stored in encrypted Amazon S3 buckets using server-side encryption (SSE-S3). Access will be controlled through AWS Identity and Access Management (IAM) roles and policies, restricting permissions to authorized personnel only. Regular backups and recovery tests will prevent data loss.
- **Data Transmission Security**: We will enforce HTTPS protocols for all data transmission between services, utilizing SSL/TLS encryption to protect data in transit against interception and unauthorized access.
- **Access Control**: Role-Based Access Control (RBAC) will define user permissions based on specific roles and responsibilities. Regular audits of access logs will be conducted to detect and respond to any unauthorized activities.
- **Incident Response Plan**: A comprehensive incident response strategy will be developed, outlining procedures for incident detection, containment, eradication, recovery, and follow-up. Roles and responsibilities will be clearly defined to ensure an effective response to any security incidents.

4.2 Bias and Ethical Concerns

- **Bias Identification and Mitigation**: We recognize the potential for biases related to age, gender, race, or socioeconomic status. Regular audits will be performed to detect such biases using techniques like Disparate Impact Analysis. If biases are identified, we will adjust training data and model parameters to ensure equitable treatment across all patient groups.
- **Model Explainability and Transparency**: To foster trust and facilitate clinical adoption, we will employ interpretable models or integrate explainability tools such as SHAP (SHapley Additive exPlanations). This approach provides clinicians with clear insights into the factors influencing each prediction.
- **Informed Consent and Data Usage**: Patient consent forms will include authorization for data usage in predictive modeling. Transparency about data collection methods, storage practices, and purposes will be maintained to build trust with patients and stakeholders.
- **Compliance with Ethical Guidelines**: The project will align with ethical standards set by professional bodies like the American Medical Association (AMA). Training will be provided to staff on data ethics, privacy, and responsible AI practices to cultivate an ethical culture.

By proactively addressing these considerations, we aim to build a system that is not only effective but also trustworthy and respectful of patient rights.

5 Solution Overview

Our proposed solution integrates robust data handling, advanced modeling techniques, seamless deployment strategies, and continuous monitoring processes. Each component is designed to contribute to an effective and reliable ML system for predicting hospital readmissions.

5.1 System Architecture

The overall architecture of the proposed ML system is designed to facilitate seamless data flow, robust model training, and efficient deployment. The system comprises multiple components, each responsible for specific tasks, integrated to function cohesively.

5.1.1 System Architecture Diagram

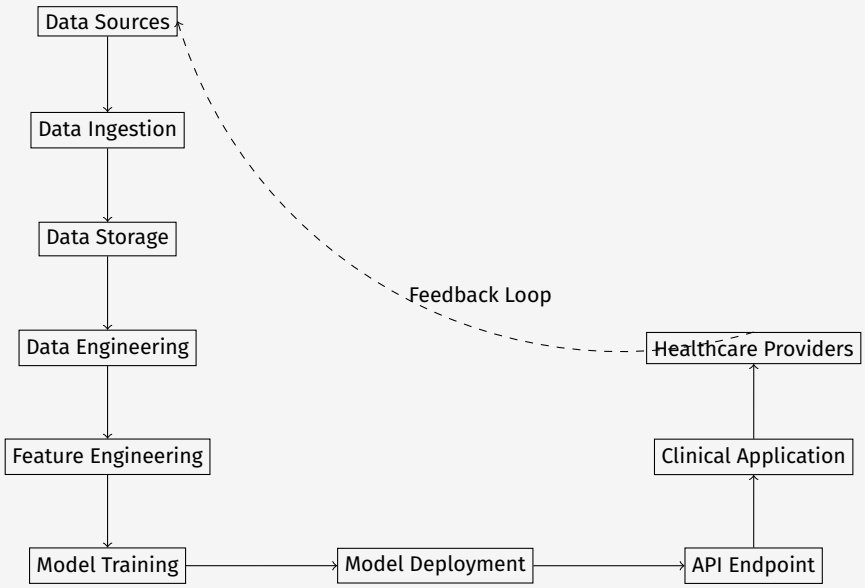

Figure 1: System Architecture Diagram

Figure 1: System Overview Diagram illustrating the flow from data sources through model deployment to clinical application and feedback integration.

5.2 Data Sources

We utilize the **Diabetes 130-US hospitals for years 1999-2008 Data Set** from the UCI Machine Learning Repository, accessed via the Hugging Face Datasets library under the identifier `aai540-group3/diabetes-readmission`.

5.2.1 Dataset Characteristics

- **Size**: The dataset contains 101,766 patient records, providing a substantial sample for model training and evaluation.
- **Features**: It includes 55 attributes encompassing demographics (age, gender, ethnicity), medical history (diagnoses, comorbidities), laboratory results, medications, and hospitalization details (length of stay, number of procedures).
- **Target Variable**: The `readmitted` indicator (0 or 1) signifies whether a patient was readmitted within 30 days of discharge.

Table 1: Summary of Dataset Features

Feature Type	Description
Demographics	Age, Gender, Ethnicity
Medical History	Diagnoses (primary and secondary), Comorbidities
Lab Results	Number of lab procedures, lab test results
Medications	Number of medications, types of medications administered
Hospitalization Details	Length of stay, number of inpatient visits, number of procedures
Other	Discharge disposition, admission type, payer code

Table 1: Summary of the types of features included in the dataset.

5.2.2 Versioning

- **Consistency**: The dataset is versioned (`v1.0`) to ensure consistency and reproducibility across different experiments and development stages.
- **Storage**: Raw data is stored in a structured format within the repository (`data/raw/data.csv`), enabling version control through tools like Git and DVC (Data Version Control).

5.3 Data Engineering

Effective data engineering is crucial for preparing the dataset for modeling. Our process entails data ingestion, cleaning, and splitting.

5.3.1 Ingestion

Using the custom script `ingestion.py`, we perform the following steps:

- **Data Loading**: Employ the Hugging Face Datasets API to reliably fetch the dataset.
- **Data Export**: Save the dataset as a CSV file in the `data/raw` directory, facilitating version control and transparency.
- **Logging**: Implement logging mechanisms to document the ingestion process, ensuring traceability and facilitating debugging if anomalies arise.

5.3.2 Cleaning

The `cleaning.py` script addresses data quality issues to enhance model performance:

- **Duplicate Removal**: Identified and removed 1,437 duplicate entries to prevent data redundancy and bias.
- **Missing Values Handling**:
  - **Numerical Variables**: Imputed missing values using the median, reducing the impact of outliers and skewed distributions.
  - **Categorical Variables**: Filled missing values with the mode (most frequent value), maintaining category integrity.

- **Data Type Consistency**:
  - Converted numerical features to `float32` for computational efficiency.
  - Transformed binary categorical features into boolean types for clarity and proper handling in modeling.
- **Feature Reduction**:
  - Removed features with more than 50% missing data, such as `weight` and `payer_code`, to avoid introducing bias or noise.
  - Eliminated irrelevant identifiers (e.g., patient IDs) to prevent data leakage and protect privacy.
- **Normalization and Standardization**: Applied where appropriate to ensure that features are on comparable scales, which is beneficial for certain algorithms.

5.3.3 Splitting

The `splitting.py` script divides the dataset into training and testing sets:

- **Training Set**: Consists of 80% of the data, used for model training.
- **Testing Set**: Comprises 20% of the data, reserved for evaluating model performance.
- **Stratification**: Ensured that the distribution of the target variable (`readmitted`) is consistent across both sets, maintaining the representativeness of the sample.
- **Reproducibility**: Used a fixed `random_state=42` to enable consistent results in subsequent runs, facilitating collaboration and comparison.

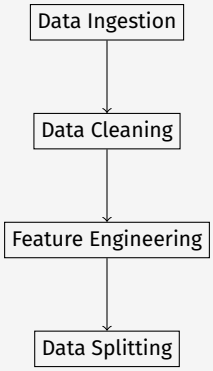

Figure 2: Data Engineering Pipeline

Figure 2: Data processing pipeline from ingestion to data splitting.

5.4 Feature Engineering

Effective feature engineering can significantly enhance model performance by creating new variables that capture underlying relationships in the data.

5.4.1 New Feature Creation

Using `build_features.py`, we engineered several new features:

Table 2: Engineered Features

Feature Name	Description
`total_procedures`	Sum of `num_lab_procedures` and `num_procedures`; represents the overall procedural burden.
`total_visits`	Sum of `number_outpatient`, `number_emergency`, and `number_inpatient`; reflects the frequency of healthcare interactions.
`avg_procedures_per_visit`	`total_procedures` divided by `total_visits`; captures procedural intensity per visit.
`lab_procedure_ratio`	`num_lab_procedures` divided by `total_procedures`; indicates the focus on diagnostic procedures.
`medication_intensity`	`num_medications` divided by `time_in_hospital`; captures the rate of medication administration.

Table 2: New features created during feature engineering.

5.4.2 Polynomial Features

To capture non-linear relationships, we introduced polynomial features:

- **Implementation**: Used Scikit-learn's `PolynomialFeatures` with a degree of 2.
- **Application**: Applied to selected numerical features based on their correlation with the target variable.
- **Advantages**: Allows the model to learn complex patterns that linear features might not capture.
- **Considerations**: Monitored for potential issues such as multicollinearity and overfitting; performed feature selection post-generation to retain only features contributing positively to model performance.

5.5 Model Training and Evaluation

The modeling phase involved training two distinct models: Logistic Regression and AutoGluon's `TabularPredictor`. Both models were evaluated to determine their effectiveness.

5.5.1 Models Implemented

Logistic Regression

- **Algorithm**: Utilized Scikit-learn's `LogisticRegression`, known for its simplicity and interpretability.
- **Hyperparameters**:
  - **Penalty**: L2 regularization to prevent overfitting.
  - **Solver**: 'lbfgs', suitable for smaller datasets and supports L2 regularization.
  - **Max Iterations**: Set to 1000 to ensure convergence, especially important given the complex feature set.
  - **Random State**: Set to 42 for reproducibility.

- **Preprocessing Steps**:
  - **Scaling**: Applied `StandardScaler` to standardize numerical features.
  - **Imputation**: Used `SimpleImputer` to handle missing values.
  - **Encoding**: Employed one-hot encoding for categorical variables.
- **Training Process**:
  - Loaded preprocessed training data from `data/processed/logistic_regression/train_preprocessed.csv`.
  - Trained the model and saved it as `models/logistic_regression/model.pkl`.

AutoGluon TabularPredictor

- **Algorithm**: Leveraged AutoGluon's automated machine learning capabilities to systematically explore various models and hyperparameters.
- **Configuration**:
  - **Presets**: Set to 'best_quality' to prioritize predictive performance.
  - **Time Limit**: Restricted to 3600 seconds to balance computational resources and exploration depth.
  - **Hyperparameters**: Customized settings for Gradient Boosting Machines (GBM) with `num_boost_round` set to 100.
  - **Verbosity**: Level 2 to provide detailed logs during training.
- **Automated Processing**:
  - AutoGluon handles missing values, categorical encoding, and feature scaling internally.
- **Training Process**:
  - Used the raw training data (`data/processed/train.csv`).
  - Trained models are stored under `models/autogluon/`.

5.5.2 Evaluation Metrics and Results

Both models were evaluated on the test set, yielding the following results:

Table 3: Model Performance Metrics

Metric	Logistic Regression	AutoGluon
Accuracy	46.01%	63.30%
Precision	69.49%	63.18%
Recall	46.01%	63.30%
F1-Score	55.18%	63.24%
ROC-AUC	64.65%	68.09%

Table 3: Comparison of model performance metrics.

Analysis

- **AutoGluon Performance**: Demonstrated superior performance across most metrics compared to Logistic Regression. The higher accuracy and ROC-AUC indicate better overall predictive ability.
- **Logistic Regression**: Achieved higher precision, suggesting it was better at predicting positive cases when it predicted them but had lower recall and overall accuracy.
- **ROC-AUC Improvement**: AutoGluon's ROC-AUC of 68.09% shows better discrimination between classes compared to Logistic Regression's 64.65%.

The results indicate that while both models have room for improvement, AutoGluon provides a better foundation for further development due to its higher accuracy and balanced performance across metrics.

5.5.3 Visualizations

To gain deeper insights into model behavior, we generated several visualizations:

- **Confusion Matrix**: Showed the model's performance in terms of true positives, true negatives, false positives, and false negatives.
- **ROC Curve**: Illustrated the trade-off between true positive rate and false positive rate at various thresholds.
- **Feature Importances**:
  - **Logistic Regression**: Analyzed coefficients to understand the impact of each feature.
  - **AutoGluon**: Used built-in methods to extract and plot feature importance rankings.

5.6 Model Deployment

5.6.1 Serialization and Versioning

- **Model Saving**:
  - **Logistic Regression**: Serialized using `joblib` for efficient storage and loading.
  - **AutoGluon**: Utilized AutoGluon's `.save()` and `.load()` methods.
- **Version Control with DVC**:
  - Managed datasets and models using DVC to track changes and maintain consistency.
  - Configured remote storage on AWS S3 to handle large files and facilitate collaboration.

5.6.2 Infrastructure as Code

- **Terraform Configuration**:
  - Defined infrastructure resources (e.g., AWS S3 buckets, IAM policies) using Terraform scripts in the `terraform/` directory.
  - Enabled reproducible, version-controlled infrastructure provisioning.

5.6.3 Deployment Strategy

- **Endpoint Exposure**:
  - Plan to deploy the model as a RESTful API, making it accessible for integration with hospital systems.
  - Utilize AWS services like SageMaker or containerization technologies (Docker, Kubernetes) for scalability and ease of management.
- **Integration with Clinical Workflows**:
  - Collaborate with IT departments to incorporate the model's predictions into Electronic Health Record (EHR) systems.
  - Ensure outputs are presented in a user-friendly manner, facilitating clinicians' ability to act on the predictions.

5.6.4 Deployment Architecture Diagram

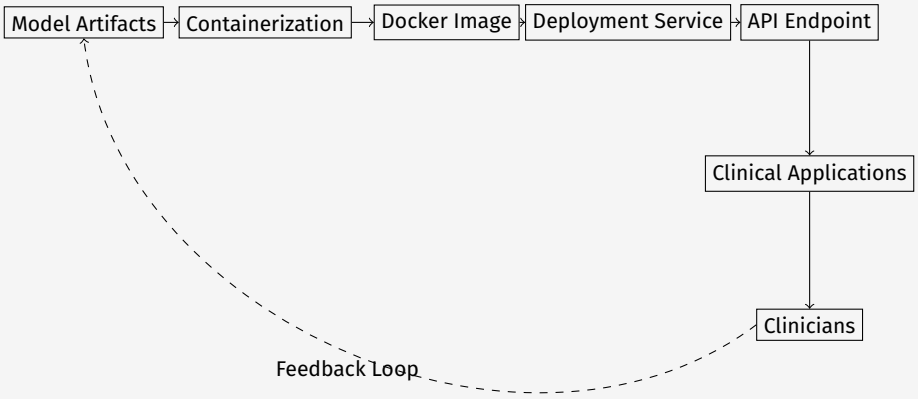

Figure 3: Model Deployment Architecture

Figure 3: Deployment architecture diagram showing the transition from model artifacts to deployment and integration with clinical applications.

5.7 Model Monitoring

5.7.1 DVCLive Integration

- **Real-Time Tracking**:
  - Incorporated DVCLive to log metrics, parameters, and artifacts during training and evaluation.
  - Stored logs in the `dvclive/` directory for easy access and visualization.

5.7.2 DVC Studio Usage

- **Experiment Management**:
  - Connected the repository to DVC Studio for interactive dashboards.
  - Enabled comparison of experiments, tracking of model performance over time, and identification of trends.

5.7.3 Automated Alerts and Retraining

- **Performance Monitoring**:
  - Established thresholds for key metrics to detect when model performance degrades.
  - Configured alerts using AWS CloudWatch and SNS to notify the team promptly.
- **Data Drift Detection**:
  - Implemented mechanisms to identify changes in data distributions that could impact model validity.
  - Scheduled periodic retraining and evaluation to ensure the model remains accurate and relevant.

5.8 Continuous Integration and Continuous Deployment (CI/CD)

5.8.1 GitHub Actions Workflow

The CI/CD pipeline, defined in `.github/workflows/mlops-pipeline.yml`, automates the development process.

Pipeline Stages

1. **Trigger Conditions**:
   - Initiated on code pushes to the `main` branch or upon pull request merges involving relevant files.
2. **Environment Setup**:
   - Checks out the repository using `actions/checkout`.
   - Sets up Python 3.11 environment with `actions/setup-python`.
   - Installs project dependencies specified in `requirements.txt`.
3. **DVC Configuration**:
   - Configures DVC remotes to connect to AWS S3 storage.
   - Pulls data and models using `dvc pull`.
4. **Pipeline Execution**:
   - Runs DVC pipeline stages defined in `dvc.yaml` using `dvc exp run -run-all`.
   - Captures outputs, metrics, and artifacts.
5. **Artifacts Management**:
   - Pushes updated data and models to remote storage using `dvc push`.

- Commits changes to relevant files back to the repository.

6. **Automated Commits**:
   - Utilizes `git-auto-commit-action` to automate the commit process.

5.8.2 CI/CD Pipeline Diagram

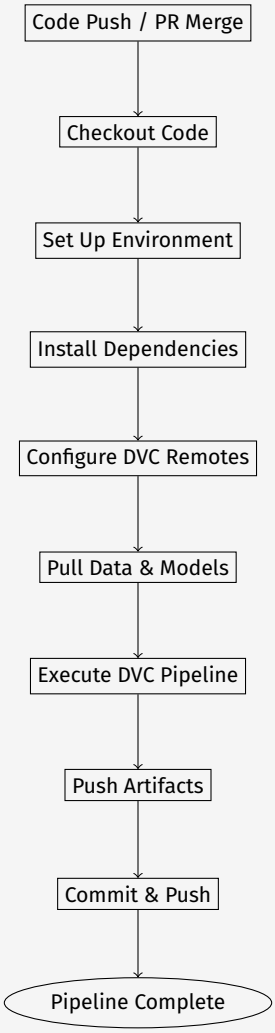

Figure 4: CI/CD Pipeline Diagram

Figure 4: CI/CD Pipeline Diagram illustrating the automated steps from code changes to updated models and data in the repository.

5.8.3 Benefits

- **Automation**: Streamlines repetitive tasks, allowing the team to focus on development and improvement.
- **Reproducibility**: Ensures every run is documented and can be replicated, enhancing reliability and trust.
- **Collaboration**: Facilitates teamwork by maintaining a consistent codebase and artifact versions.

6 Conclusion

This design document presents a comprehensive machine learning system aimed at predicting 30-day hospital readmissions for diabetic patients. By integrating advanced data engineering practices, sophisticated modeling techniques, and robust deployment strategies, the proposed solution is poised to make a significant impact in healthcare settings.

Emphasis on security, ethical considerations, and bias mitigation ensures that the system not only performs effectively but also responsibly. The use of tools like DVC and Terraform enhances reproducibility and scalability. Additionally, automated CI/CD pipelines facilitate continuous improvement and collaboration.

Moving forward, collaboration with healthcare professionals will be crucial to refine the system, ensuring that it integrates seamlessly into clinical workflows and truly enhances patient care.