Hospital Readmission Prediction Using the AI Development Workflow

Course: AI for Software Engineering

Student: Kibiwott Kamoo

Table of Contents

PART 1 — SHORT ANSWER QUESTIONS (30 POINTS)

1. Problem Definition

Hypothetical AI Problem:

Predicting early student dropout in an online learning platform.

Objectives:

1. Identify students at high dropout risk within the first four weeks.
2. Provide actionable insights for instructors and advisors.
3. Reduce overall dropout rate by enabling targeted interventions.

Stakeholders:

Students are directly affected by predictions and interventions. Instructors and academic advisors use predictions to support struggling learners.

Key Performance Indicator (KPI):

Area Under the ROC Curve (AUC) — evaluates the ability of the model to distinguish likely dropouts from non-dropouts.

2. Data Collection and Preprocessing

Data Sources:

1. Learning Management System logs, including activity time, number of quiz attempts, and forum participation.
2. Student demographic and enrollment data, such as age, program of study, prior academic performance, and socioeconomic information.

Potential Bias:

Students with limited internet connectivity may appear disengaged, which could unfairly skew risk scores.

Preprocessing Steps:

1. Handle missing log data using imputation techniques such as mean, median, or K-nearest neighbor imputation.
2. Normalize engagement metrics through min-max scaling to ensure consistent input for machine learning algorithms.
3. One-hot encode categorical fields such as program or major to allow models to process them numerically.

3. Model Development

Model Choice:

Random Forest is selected for its ability to handle mixed data types, robustness against noisy or missing data, and interpretability via feature importance.

Data Split:

70% of data is used for training, 15% for validation, and 15% for testing.

Hyperparameters to Tune:

1. The number of trees (n_estimators), which affects model accuracy.
2. Maximum depth of the trees (max_depth) to prevent overfitting.

4. Evaluation and Deployment

Evaluation Metrics:

1. Precision — ensures that the model does not incorrectly flag many low-risk students.
2. Recall — ensures the model identifies most students who are actually at risk of dropping out.

Concept Drift:

Occurs when patterns in data change over time, such as new platform features or changes in student behavior. Monitoring involves real-time dashboards that compare expected vs. actual model performance.

Technical Deployment Challenge:

Scalability is a key challenge when handling tens of thousands of predictions per hour during peak usage periods.

PART 2 — CASE STUDY APPLICATION

Scenario:

A hospital wants an AI system to predict patient readmission within 30 days of discharge.

1. Problem Scope (5 points)

Definition:

Build a predictive model that estimates whether a patient will be readmitted within 30 days of discharge.

Objectives:

1. Reduce preventable hospital readmissions.
2. Optimize resource allocation within the hospital.
3. Improve overall quality-of-care metrics.

Stakeholders:

Hospital administration, physicians, medical staff, insurance providers, and patients.

2. Data Strategy

Data Sources:

1. Electronic Health Records (EHRs) including past admissions, comorbidities, and vital signs.
2. Patient demographics and medical history.
3. Laboratory tests, diagnosis codes, medications, and procedure records.

Ethical Concerns:

1. Privacy and HIPAA compliance to prevent misuse of sensitive health data.

2. Bias against minority groups if historical data reflects unequal healthcare access, leading to unfair readmission predictions.

Preprocessing Pipeline:

1. Remove inconsistent or unknown markers and standardize missing values.
2. Convert diagnosis codes to categorical variables.
3. Standardize numerical fields like lab test values.
4. One-hot encode categorical variables for model compatibility.
5. Engineer additional features such as total hospital visits and chronic condition scores.

3. Model Development

Selected Model:

XGBoost Classifier is chosen for high predictive performance and its ability to handle class imbalance effectively.

Performance Metrics:

Precision is calculated as the proportion of true positive predictions among all positive predictions, while recall measures the proportion of true positives identified among all actual positive cases. Hypothetical calculations indicate that the model is reasonably balanced in detecting readmissions while minimizing false alarms.

4. Deployment

Integration Steps:

1. Save the trained model as a .pkl file.
2. Wrap it within a FastAPI or Flask microservice.

3. Expose a /predict endpoint for easy access.
4. Integrate predictions into the hospital EHR system interface for clinicians.
5. Log all predictions for auditing purposes and to detect performance drift over time.

Regulatory Compliance:

Encrypt data at rest and in transit, enforce role-based access controls, maintain audit logs, and follow HIPAA's minimum necessary data requirements.

5. Optimization

Method to Reduce Overfitting:

Apply early stopping during XGBoost training to terminate training when performance on the validation set ceases to improve.

PART 3 — CRITICAL THINKING

1. Ethics and Bias (10 points)

Impact of Biased Data:

If historical data reflects unequal healthcare treatment for minority groups, the model may incorrectly label these patients as high-risk or low-risk, leading to inequitable care decisions.

Mitigation Strategy:

Use fairness-aware reweighting or equal opportunity metrics during model evaluation to ensure predictions are not systematically biased against specific populations.

2. Trade-offs

Interpretability vs Accuracy:

Highly accurate models such as XGBoost or neural networks are often less transparent, whereas interpretable models like logistic regression are easier to understand but may sacrifice accuracy. In healthcare, interpretability is critical, but hybrid approaches such as SHAP explanations allow clinicians to see feature contributions without losing model performance.

Limited Computational Resources:

Hospitals may lack GPUs for heavy computation. Simpler models such as Random Forest or logistic regression can still provide actionable insights with lower computational cost.

PART 4 — REFLECTION AND WORKFLOW DIAGRAM

Reflection (5 points)

The most challenging part of the workflow was **data preprocessing**, particularly handling missing data and mixed-format diagnosis codes. With more resources, implementing automated data validation, a dedicated feature store, and a CI/CD pipeline for machine learning models would improve reproducibility and efficiency.

Workflow Diagram (5 points)

The AI development workflow for hospital readmission prediction consists of the following stages:

1. **Problem Definition:** Define objectives, stakeholders, and KPIs.
2. **Data Collection:** Gather patient data from EHRs, demographics, lab results, and medical history.

3. **Data Preprocessing:** Clean data, handle missing values, standardize numeric fields, encode categorical variables, and engineer relevant features.
4. **Model Selection:** Choose an appropriate predictive model (e.g., XGBoost).
5. **Training and Validation:** Split data into training, validation, and test sets, tune hyperparameters, and monitor performance.
6. **Evaluation:** Assess metrics such as precision, recall, and AUC, and check for bias and fairness.
7. **Deployment:** Package the model in a microservice, integrate with hospital systems, and ensure regulatory compliance.
8. **Monitoring and Optimization:** Track performance over time, detect concept drift, and retrain models as needed.

Appendix — Python Code and Diagrams

```
from graphviz import Digraph

# Create workflow diagram
dot = Digraph(comment='Hospital Readmission Prediction Workflow')

# Define nodes
dot.node('A', 'Problem Definition')
dot.node('B', 'Data Collection')
dot.node('C', 'Data Preprocessing')
dot.node('D', 'Model Selection')
dot.node('E', 'Training and Validation')
dot.node('F', 'Evaluation')
dot.node('G', 'Deployment')
dot.node('H', 'Monitoring & Optimization')

# Define edges
dot.edges(['AB', 'BC', 'CD', 'DE', 'EF', 'FG', 'GH'])

# Render diagram
dot.render('workflow_diagram', format='png', cleanup=True)
```