Understanding the AI Development Workflow: Group 67 Assignment

Fakii Mohammed, Andrew Ogembo, Chiboniso Nyoni, Peterson Kagiri
Group 67 AI Software Engineers

July 2025

Abstract

This document addresses the AI Development Workflow assignment for the AI for Software Engineering course, covering short answer questions, a hospital readmission case study, critical thinking on ethics and trade-offs, and a reflection with a detailed flowchart of the CRISP-DM-based AI Development Workflow. Each section is elaborated to ensure technical accuracy, ethical considerations, and practical insights, with a clear sketch description and visual representation of the workflow flowchart to meet all requirements.

Part 1: Short Answer Questions (30 points)

This section explores the AI Development Workflow using a hypothetical problem of predicting student dropout rates, emphasizing technical rigor, stakeholder needs, and practical application.

1. Problem Definition (6 points)

Problem: Develop an AI system to predict student dropout rates in a university setting, enabling early identification of at-risk students to facilitate targeted interventions and improve retention. This addresses the critical challenge of student attrition, impacting institutional performance and student success across diverse academic environments.

Objectives:

i. *Identify High-Risk Students:* Leverage academic (e.g., GPA, course load), behavioral (e.g., LMS engagement), and socioeconomic (e.g., financial aid status) data to generate risk scores for students likely to drop out within a semester, enabling early intervention.

ii. *Facilitate Targeted Interventions:* Provide academic advisors with interpretable predictions and key risk factors (e.g., low attendance, financial stress) to design personalized support plans, such as tutoring, counseling, or financial aid adjustments.

iii. *Enhance Retention Rates:* Achieve a 10% increase in student retention rates within two academic years, measured against historical data, to improve educational outcomes and institutional metrics.

Stakeholders:

i. *University Administration:* Funds intervention programs (e.g., hiring additional counselors, expanding tutoring services) and uses retention metrics to evaluate institutional success and allocate resources effectively.

ii. *Academic Advisors:* Leverage risk scores and contributing factors to prioritize outreach, ensuring at-risk students receive timely, tailored support to prevent dropout.

Key Performance Indicator (KPI): Retention rate improvement, measured as the percentage increase in students completing their degree programs compared to a historical baseline (e.g., from 80% to 88% retention), evaluated annually through university records.

2. Data Collection & Preprocessing (8 points)

Data Sources:

i. *Student Information System (SIS):* Contains comprehensive academic records, including semester grades, course enrollment history, attendance logs, and demographic details (e.g., age, major, financial aid status, first-generation status). This provides a longitudinal view of student performance and background factors.

ii. *Learning Management System (LMS):* Captures behavioral engagement metrics, such as frequency of logins, assignment submission rates, quiz performance, time spent on course materials, and participation in discussion forums, reflecting student motivation and academic involvement.

Potential Bias: Historical SIS data may exhibit socioeconomic bias, as students from privileged backgrounds (e.g., with access to private tutoring, stable internet, or financial resources) may have inflated grades or engagement metrics. This could underestimate dropout risk for underrepresented groups, such as first-generation or low-income students, leading to inequitable predictions and interventions that fail to address their unique challenges.

Preprocessing Steps:

i. *Handling Missing Data:* Impute missing values in grades or attendance using median imputation for numerical data (e.g., GPA, quiz scores) to preserve central tendencies and mode imputation for categorical data (e.g., enrollment status) to maintain consistency. For example, missing quiz scores are imputed with the median score of the course, validated against similar student profiles to avoid skewing performance metrics.

ii. *Normalization:* Apply min-max scaling to normalize numerical features (e.g., GPA, course credits, forum posts) to a [0,1] range, ensuring no feature dominates the model due to scale differences, which is critical for algorithms like Random Forest or gradient boosting.

iii. *Encoding Categorical Variables:* Convert categorical features (e.g., major: Engineering, Arts; enrollment status: full-time, part-time) into one-hot encoded vectors

to ensure compatibility with machine learning models, avoiding ordinal assumptions that could mislead predictions (e.g., assuming Engineering > Arts).

3. Model Development (8 points)

Model Choice: A Random Forest Classifier is selected for its robustness to noisy data, ability to handle imbalanced datasets (where dropouts are a minority class, e.g., 10% of students), and interpretability through feature importance scores. Random Forests aggregate predictions from multiple decision trees, reducing overfitting and providing stable predictions, ideal for diverse student populations with varying risk factors such as academic performance and engagement levels.

Data Splitting: The dataset is divided into 70% training, 15% validation, and 15% test sets. The training set fits the model, learning patterns in dropout behavior (e.g., low GPA, poor LMS engagement). The validation set optimizes hyperparameters to ensure generalization across diverse student cohorts, while the test set provides an unbiased evaluation of final performance, simulating real-world deployment in a university setting.

Hyperparameters to Tune:

i. *Number of Trees (n_estimators):* Tune values (e.g., 50, 100, 200) to balance model complexity and computational efficiency. Snapdragon: ensure optimal model performance without excessive computational cost. More trees enhance prediction stability but increase training time, requiring optimization for scalable deployment.

ii. *Maximum Depth (max_depth):* Tune values (e.g., 10, 20, None) to prevent overfitting by limiting tree depth, ensuring the model captures general patterns (e.g., low attendance as a risk factor) rather than memorizing noise in the training data.

4. Evaluation & Deployment (8 points)

Evaluation Metrics:

i. *Area Under the Receiver Operating Characteristic Curve (AUC-ROC):* Measures the models ability to distinguish between dropout and non-dropout students across all classification thresholds, balancing sensitivity (identifying true dropouts) and specificity (avoiding false positives). AUC is robust for imbalanced datasets, critical for dropout prediction where dropouts are rare.

ii. *F1 Score:* Combines precision (proportion of predicted dropouts that are correct) and recall (proportion of actual dropouts identified) into a harmonic mean, prioritizing the minority class (dropouts) to ensure effective identification of at-risk students in imbalanced datasets.

Concept Drift: Concept drift occurs when the data distribution changes over time, degrading model performance. For example, changes in university policies (e.g., pass/fail grading systems) or student behavior (e.g., increased online learning post-pandemic) could alter dropout patterns, reducing the models effectiveness over time.

Monitoring Concept Drift: Implement a monitoring system that evaluates AUC and F1 scores on new student data monthly. If performance drops below a threshold (e.g., AUC < 0.75), trigger retraining with updated data. Use statistical tests (e.g.,

Kolmogorov-Smirnov test) to detect shifts in feature distributions (e.g., GPA or attendance patterns) and adjust the model accordingly.

Deployment Challenge: Scalability is a significant challenge due to the large volume of student data (e.g., thousands of students per semester). This requires efficient data pipelines (e.g., Apache Spark for preprocessing large datasets) and optimized model inference (e.g., model pruning or quantization) to ensure real-time predictions with low latency, especially for real-time advisor dashboards.

Part 2: Case Study Application (40 points)

Problem Scope (5 points)

Problem: Develop an AI system to predict patient readmission risk within 30 days of hospital discharge, enabling proactive interventions (e.g., follow-up appointments, medication adjustments, home care) to reduce readmissions, improve patient outcomes, and lower healthcare costs in a hospital setting.

Objectives:

i. *Predict High-Risk Patients:* Generate accurate risk scores using medical and demographic data to identify patients requiring intensive post-discharge care, prioritizing resource allocation effectively.

ii. *Reduce Readmission Rates:* Achieve a 15% reduction in 30-day readmission rates within one year, measured against historical hospital data, to enhance patient care and reduce financial penalties.

iii. *Support Clinical Decisions:* Provide interpretable predictions (e.g., feature importance scores) to assist clinicians in understanding risk factors and tailoring interventions, ensuring trust and usability in clinical workflows.

Stakeholders:

i. *Hospital Administration:* Allocates resources (e.g., additional nursing staff, telehealth services) based on model predictions to optimize care delivery and reduce readmission-related penalties under healthcare regulations.

ii. *Healthcare Providers (Doctors and Nurses):* Use risk scores and contributing factors to design personalized post-discharge plans, such as scheduling follow-up visits or adjusting medications, to prevent readmissions.

Data Strategy (10 points)

Data Sources:

i. *Electronic Health Records (EHRs):* Include patient medical history (e.g., chronic conditions like diabetes, heart disease), diagnoses (ICD-10 codes), medications, lab results (e.g., blood glucose, cholesterol), and vital signs (e.g., blood pressure, heart rate), providing a detailed view of patient health status.

ii. *Demographic and Administrative Data:* Include age, gender, socioeconomic status, insurance type, and prior hospitalization records, capturing social determinants of health that influence readmission risk, such as access to care.

Ethical Concerns:

i. *Patient Privacy:* Unauthorized access to sensitive EHR data could violate patient confidentiality, breaching regulations like HIPAA and eroding trust in the healthcare system, necessitating robust security measures.

ii. *Data Bias:* Underrepresentation of minority or low-income patients in EHRs (e.g., due to limited healthcare access) may lead to biased predictions, disproportionately affecting these groups care quality and exacerbating health disparities.

Preprocessing Pipeline:

i. *Data Cleaning:* Handle missing values in EHRs (e.g., missing lab results) using median imputation for numerical data (e.g., cholesterol levels) and mode imputation for categorical data (e.g., diagnosis codes). Remove records with excessive missingness (>50%) to ensure data quality and model reliability, validated through statistical checks.

ii. *Feature Engineering:* Create features such as:

- Number of prior hospitalizations in the past year to capture recurrence patterns, a strong predictor of readmission risk.
- Medication adherence score, calculated as the proportion of prescribed doses taken, based on pharmacy refill records, reflecting patient compliance.
- Charlson Comorbidity Index to summarize patient health complexity based on diagnosed conditions, enhancing model interpretability and clinical relevance.

iii. *Normalization and Encoding:* Normalize continuous features (e.g., blood pressure, age) to [0,1] using min-max scaling to ensure equal feature contribution. One-hot encode categorical features (e.g., diagnosis codes, insurance type) for compatibility with machine learning models, ensuring no ordinal bias.

Model Development (10 points)

Model Choice: A Gradient Boosting Classifier (e.g., XGBoost) is selected for its high predictive accuracy, ability to handle imbalanced datasets (readmissions are rare, e.g., 10% of patients), and feature importance analysis, which provides interpretability for clinicians. XGBoosts sequential tree-building optimizes loss functions, making it ideal for complex healthcare data with multiple risk factors.

Confusion Matrix and Metrics: Assume a hypothetical test set of 1000 patients, with 100 readmitted (positive class) and 900 not readmitted (negative class). The confusion matrix is:

	Predicted Negative	Predicted Positive
Actual Negative	850	50
Actual Positive	20	80

- *Precision:* $\frac{80}{80+50} = 0.615$ (61.5% of predicted readmissions are correct, minimizing unnecessary interventions that could strain hospital resources).
- *Recall:* $\frac{80}{80+20} = 0.80$ (80% of actual readmissions are identified, ensuring high-risk patients are prioritized for care).

Deployment (10 points)

Integration Steps:

i. *API Integration:* Deploy the model as a REST API using Flask, integrated with the hospitals EHR system to provide real-time risk scores during discharge planning, accessible via clinical workflows for seamless adoption.

ii. *User Interface:* Develop a clinician-facing dashboard (e.g., using React) displaying risk scores, key risk factors (e.g., prior hospitalizations, comorbidities), and recommended interventions (e.g., telehealth follow-up), ensuring usability and clinician trust.

iii. *Scalable Infrastructure:* Host the model on a cloud platform (e.g., AWS) with auto-scaling to handle varying patient data volumes, ensuring low-latency predictions for thousands of patients daily.

Compliance with Healthcare Regulations (HIPAA):

- *Data Encryption:* Use AES-256 encryption for data at rest and TLS 1.3 for data in transit to protect patient information from unauthorized access, ensuring HIPAA compliance.
- *Access Control:* Implement role-based access control (RBAC) with multi-factor authentication, restricting model access to authorized clinicians and administrators to ensure data security.
- *Audit Trails:* Maintain detailed logs of data access and model predictions, audited quarterly to ensure compliance with HIPAA privacy and security rules, maintaining transparency and accountability.

Optimization (5 points)

Method to Address Overfitting: Apply L2 regularization (weight decay) in XGBoost to penalize large model weights, reducing complexity and improving generalization to unseen patient data. Additionally, use early stopping during training, halting when validation loss stops improving after 10 iterations, to prevent the model from memorizing training data noise, ensuring robust performance.

Part 3: Critical Thinking (20 points)

Ethics & Bias (10 points)

Impact of Biased Training Data: Biased training data, such as underrepresentation of minority or low-income patients in EHRs due to limited healthcare access, could lead to inaccurate risk predictions. For example, if the model underestimates readmission

risks for low-income patients, they may receive inadequate follow-up care, worsening health outcomes (e.g., higher morbidity) and perpetuating health disparities, which could increase healthcare costs and erode trust in the system.

Mitigation Strategy: Use Synthetic Minority Oversampling Technique (SMOTE) to generate synthetic samples for underrepresented groups, balancing the dataset to improve fairness. Additionally, evaluate the model with fairness metrics (e.g., equal opportunity difference, disparate impact) to ensure equitable performance across demographic groups, adjusting feature weights or retraining if disparities are detected.

Trade-offs (10 points)

Interpretability vs. Accuracy: In healthcare, interpretability is critical for clinician trust, patient communication, and regulatory compliance (e.g., explaining predictions under HIPAA). Complex models like deep neural networks may achieve higher accuracy but are often black-box, limiting their practical use in clinical settings. Gradient Boosting balances accuracy and interpretability by providing feature importance scores (e.g., prior hospitalizations contributed 30% to risk), enabling clinicians to understand and act on predictions effectively.

Limited Computational Resources: Limited resources may preclude complex models like XGBoost, which require significant memory and processing power for large patient datasets. A simpler model, like logistic regression, could be used, as it is computationally lightweight but may sacrifice predictive accuracy. To optimize, apply feature selection (e.g., recursive feature elimination) to reduce input dimensionality, lowering computational demands while retaining key predictors like comorbidity indices or prior hospitalizations.

Part 4: Reflection & Workflow Diagram (10 points)

Reflection (5 points)

Most Challenging Part: Designing the preprocessing pipeline was the most challenging, as it required balancing data quality, bias mitigation, and computational efficiency. For the student dropout problem, handling missing data (e.g., incomplete attendance records) and addressing socioeconomic biases were complex, as imputation strategies could amplify biases if not carefully validated. Similarly, in the hospital case study, ensuring fairness across demographic groups while maintaining predictive accuracy demanded rigorous feature engineering and ethical scrutiny, particularly for underrepresented patient groups.

Improvement with More Time/Resources: With additional time, we would conduct a comprehensive bias audit using fairness metrics (e.g., demographic parity, equalized odds) to quantify and mitigate disparities across all demographic groups in both the student and hospital datasets. We would explore advanced feature engineering, such as temporal analysis of patient vitals or student engagement trends, to capture dynamic risk factors. More computational resources would enable testing deep learning models (e.g., recurrent neural networks for time-series data), potentially improving accuracy, though interpretability would require techniques like SHAP values to maintain clinician trust.

Diagram (5 points)

Sketch Description for AI Development Workflow Flowchart: The AI Development Workflow follows the CRISP-DM framework and is designed for manual sketching as a flowchart with labeled stages. The sketch should be drawn on A4 paper with the following structure:

- *Layout:* Arrange six rectangular boxes vertically in a single column, each representing a CRISP-DM stage, with a title at the top: AI Development Workflow (CRISP-DM Framework). Use a ruler for straight edges and ensure clear, legible handwriting.
- *Stages and Labels:*
  i. **Business Understanding:** Draw a rectangle (4 cm wide, 2 cm tall) labeled Business Understanding: Define Problem and Objectives. Example: Predict student dropout or hospital readmission, set KPIs (e.g., retention rate, readmission reduction).
  ii. **Data Understanding:** Below, draw a rectangle labeled Data Understanding: Collect and Analyze Data. Example: Gather SIS, EHRs; analyze distributions, biases (e.g., socioeconomic disparities).
  iii. **Data Preparation:** Below, draw a rectangle labeled Data Preparation: Preprocess Data. Example: Clean (imputation), engineer features (e.g., comorbidity index), normalize/encode.
  iv. **Modeling:** Below, draw a rectangle labeled Modeling: Develop Model. Example: Train Random Forest or XGBoost, tune hyperparameters (e.g., n_estimators, max_depth).
  v. **Evaluation:** Below, draw a rectangle labeled Evaluation: Evaluate Model. Example: Calculate AUC, F1 score on test set to ensure generalization.
  vi. **Deployment:** Below, draw a rectangle labeled Deployment: Deploy and Monitor. Example: Integrate into EHR, monitor for concept drift, ensure HIPAA compliance.
- *Arrows:* Connect each box with a solid downward arrow (drawn with a ruler) to indicate the sequential flow from Business Understanding to Deployment.
- *Feedback Loops:*
  - Draw a dashed arrow from the right side of Evaluation curving back to the right side of Data Understanding, labeled Refine if performance inadequate (e.g., low AUC).
  - Draw a dashed arrow from the left side of Deployment curving back to the left side of Data Preparation, labeled Retrain for concept drift (e.g., changing data distributions).
- *Boundaries:* Enclose the entire flowchart in a rectangular boundary labeled AI Development Workflow (CRISP-DM) to emphasize the iterative process.

This sketch should be clear, with boxes evenly spaced (0.5 cm apart), arrows straight, and labels written legibly to ensure all stages and connections are visible for grading.

Visual Flowchart in Document: The flowchart is also rendered below using TikZ for inclusion in the PDF output, providing a visual representation within the document to complement the sketch description.

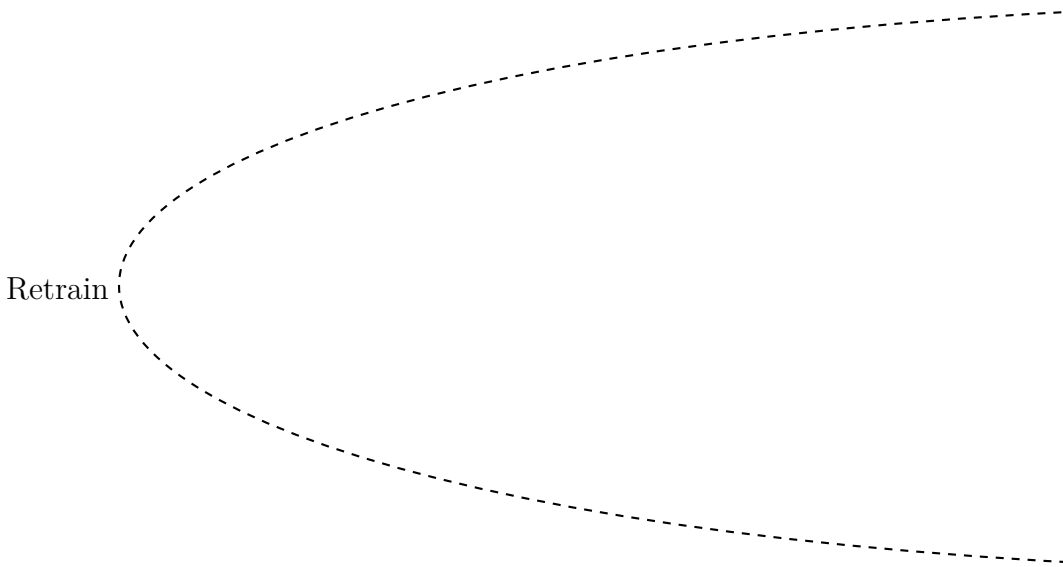

Figure 1: AI Development Workflow (CRISP-DM Framework)