

## **AI DEVELOPMENT WORKFLOW ASSIGNMENT**

### **Assignment: Understanding the AI Development Workflow**

**Author: Peter Kamau Mwaura**

## **1 PART 1: SHORT ANSWER QUESTIONS (30 POINTS)**

### **1.1 Problem Definition (6 points)**

- **Hypothetical AI Problem:** "Predicting Student Dropout Rates in Online Higher Education."
- **Three Objectives:**
  1. To identify at-risk students early in the semester for timely intervention.
  2. To understand the key factors (e.g., engagement, grades) contributing to dropout risk.
  3. To improve overall student retention and academic success rates.
- **Two Stakeholders:**
  1. **Students:** The primary subjects who will be impacted by the interventions.
  2. **University Administration/Academic Advisors:** The primary users who will act on the predictions to provide support.
- **1 Key Performance Indicator (KPI): Precision.** In this context, we want to minimize false positives (incorrectly labelling a student as at-risk) to ensure advisors' time is spent efficiently on students who genuinely need help. A high precision means when the model says a student is at-risk, it is very likely to be correct.

## 1.2 Data Collection & Preprocessing (8 points)

- **Two Data Sources:**
  1. **Learning Management System (LMS) Data:** Login frequency, assignment submission times, quiz scores, video lecture engagement, forum participation.
  2. **Student Information System (SIS) Data:** Demographic information, past academic performance, course enrolment history, financial aid status.
- **One Potential Bias: Socioeconomic Bias.** Data from LMS (like access to high-speed internet for video lectures) or SIS (like financial aid status) could inadvertently reflect a student's socioeconomic background. A model trained on this might perform poorly for students from underrepresented or low-income backgrounds, unfairly flagging them as at-risk due to factors beyond their academic ability.
- **Three Preprocessing Steps:**
  1. **Handling Missing Data:** For numerical features (e.g., quiz scores), impute missing values with the median. For categorical features (e.g., major), impute with the mode. Document the percentage of missing data for each feature.
  2. **Encoding Categorical Variables:** Convert categorical data (e.g., 'department', 'course\_type') into numerical format using One-Hot Encoding to avoid imposing an arbitrary ordinal relationship.
  3. **Feature Scaling/Normalization:** Apply StandardScaler to normalize numerical features (e.g., login count, GPA) to have a mean of 0 and a standard deviation of 1. This is crucial for models that rely on distance calculations or gradient descent.

## 1.3 Model Development (8 points)

- **Model Choice & Justification: Gradient Boosting Classifier (e.g., XGBoost).** Justification: It provides high predictive accuracy, can handle mixed data types (numerical and categorical), and captures complex, non-linear relationships between student activities

and dropout risk. It is also less prone to overfitting compared to a single decision tree and often performs well on structured tabular data like this.

- **Data Splitting:** The data will be split chronologically to avoid data leakage.
  - **Training Set (70%):** Used to train the model.
  - **Validation Set (15%):** Used for hyperparameter tuning and model selection during development.
  - **Test Set (15%):** Used only once, at the very end, to provide an unbiased evaluation of the final model's performance.
- **Two Hyperparameters to Tune:**
  1. **learning\_rate:** Controls the contribution of each tree. A lower rate makes the model more robust but requires more trees (slower to train).
  2. **max\_depth:** Controls the complexity of each individual tree. Tuning this helps prevent overfitting (if too deep) or underfitting (if too shallow).

## 1.4 Evaluation & Deployment (8 points)

### Two Evaluation Metrics:

1. **Precision-Recall Curve (PRC) & Area Under PRC (AUPRC):** Since predicting dropout is likely a class-imbalanced problem (few students actually drop out), the PRC is more informative than the ROC curve. It focuses on the performance of the positive (at-risk) class.
2. **F2-Score:** The harmonic mean of precision and recall. An F2-Score weights recall higher than precision, which is desirable here. We are willing to have some false positives to ensure we catch as many true at-risk students as possible (high recall), but not so many that it overwhelms advisors (hence, F2, not F1).

**Concept Drift & Monitoring:** Concept drift is the change in the statistical properties of the target variable or input features over time, which degrades the model's performance. In this case, student behaviour and dropout factors might change between semesters.

- **Monitoring:** I would monitor the model's performance by regularly (e.g., monthly) calculating the F2-Score on a newly labelled batch of recent student data. A significant drop would signal potential concept drift. I would also monitor the data distribution (e.g., average login frequency) for shifts.

**One Technical Challenge During Deployment: Scalability and Integration.** The model must be served as an API that can be integrated into the university's existing student portal or advisor dashboard. This requires building a scalable backend (e.g., using Flask/FastAPI and containerizing with Docker) to handle prediction requests from thousands of students and advisors simultaneously, especially during peak times like the start of a semester.

---

## **2 PART 2: CASE STUDY APPLICATION: AI SYSTEM FOR PREDICTING PATIENT READMISSION RISK**

### **2.1 Problem Scope (5 points)**

#### **2.1.1 Problem Definition**

The problem involves developing an AI system to predict the risk of patient readmission within 30 days of discharge from a hospital. Readmissions are costly and indicate potential gaps in care quality, patient management, or post-discharge support. The system aims to identify high-risk patients early, enabling targeted interventions to reduce readmission rates.

#### **2.1.2 Objectives**

- Accurately predict readmission risk using historical patient data.
- Enable proactive care planning, such as follow-up appointments or home care services.
- Reduce healthcare costs by minimizing unnecessary readmissions.

- Improve patient outcomes and satisfaction.

### 2.1.3 Stakeholders

- **Hospital Administrators:** Interested in cost savings and operational efficiency.
- **Healthcare Providers (Doctors and Nurses):** Use predictions to prioritize patient care and interventions.
- **Patients:** Benefit from personalized care plans that prevent readmissions.
- **Insurers and Payers:** Concerned with reducing claim costs and improving reimbursement models.
- **Data Scientists and IT Teams:** Responsible for model development, maintenance, and integration.

## 2.2 Data Strategy (10 points)

### 2.2.1 Proposed Data Sources

- **Electronic Health Records (EHRs):** Include diagnoses (ICD codes), medications, lab results, vital signs, and treatment histories.
- **Demographic Data:** Age, gender, race/ethnicity, socioeconomic status, and insurance type.
- **Admission and Discharge Data:** Length of stay, admission type (emergency vs. elective), discharge disposition, and previous admission history.
- **External Sources:** Public health data (e.g., comorbidity indices) or wearable device data if available.

### 2.2.2 Two Ethical Concerns

1. **Patient Privacy:** Handling sensitive health data raises risks of breaches or unauthorized access, potentially violating patient rights and leading to identity theft.
2. **Bias and Fairness:** Data may reflect historical biases (e.g., underrepresentation of certain demographics), resulting in discriminatory predictions that disproportionately affect vulnerable groups like minorities or low-income patients.

### 2.2.3 Preprocessing Pipeline

1. **Data Collection and Integration:** Aggregate data from EHRs and demographics into a unified dataset.
2. **Handling Missing Values:** Impute missing numerical values with medians and categorical with modes; flag excessive missingness for exclusion.
3. **Feature Engineering:**
  - Create derived features like "number of previous admissions," "total length of stay," and "comorbidity score" (e.g., Charlson Comorbidity Index).
  - Encode categorical variables (e.g., one-hot encoding for diagnoses).
  - Normalize numerical features (e.g., age, lab values) using standardization.
4. **Outlier Detection:** Use statistical methods (e.g., IQR) to identify and cap/remove outliers.
5. **Train-Test Split:** Split data chronologically (e.g., 70% train, 15% validation, 15% test) to simulate real-world deployment.
6. **Feature Selection:** Use techniques like recursive feature elimination or correlation analysis to retain relevant features.

## 2.3 Model Development (10 points)

### 2.3.1 Model Selection and Justification

Selected Model: **Random Forest Classifier.**

**Justification:** Random Forest is suitable for this task as it handles mixed data types (numerical and categorical), is robust to overfitting through ensemble averaging, provides feature importance for interpretability, and performs well on imbalanced datasets common in healthcare (e.g., low readmission rates). It is also computationally efficient and can be trained on large datasets.

### 2.3.2 Confusion Matrix and Metrics (Hypothetical Data)

Using hypothetical data with 300 predictions:

- True Positives (TP): 100 (correctly predicted readmissions)
- False Positives (FP): 20 (incorrectly predicted readmissions)
- True Negatives (TN): 150 (correctly predicted no readmissions)
- False Negatives (FN): 30 (incorrectly predicted no readmissions)

**Confusion Matrix:**

$$\begin{bmatrix} 150 & 20 \\ 30 & 100 \end{bmatrix}$$

**Calculations:**

- **Precision** =  $\frac{TP}{(TP + FP)} = \frac{100}{(100 + 20)} = 0.833$  (83.3%)
- **Recall** =  $\frac{TP}{(TP + FN)} = \frac{100}{(100 + 30)} = 0.769$  (76.9%)

These metrics indicate the model is reasonably accurate but could improve recall to catch more at-risk patients.

## **2.4 Deployment (10 points)**

### **2.4.1 Steps to Integrate the Model into the Hospital's System**

1. **Model Training and Validation:** Train the model on historical data, validate using cross-validation, and serialize (e.g., using joblib) for deployment.
2. **API Development:** Wrap the model in a REST API (e.g., using Flask or FastAPI) to accept patient data and return risk scores.
3. **Integration with EHR System:** Embed the API into the hospital's EHR software (e.g., via HL7 interfaces or direct database queries) to trigger predictions at discharge.
4. **User Interface:** Develop a dashboard for clinicians to view risk scores and recommendations.
5. **Testing and Rollout:** Conduct pilot testing in a subset of wards, monitor performance, and gradually scale.

6. **Monitoring and Maintenance:** Implement logging for predictions, retrain periodically with new data, and set up alerts for model drift.

#### 2.4.2 Ensuring Compliance with Healthcare Regulations (e.g., HIPAA)

- **Data Encryption and Anonymization:** Encrypt data in transit and at rest; use de-identification techniques (e.g., remove PHI) for training.
- **Access Controls:** Implement role-based access (e.g., only authorized personnel can view predictions).
- **Audit Trails:** Log all model interactions for compliance audits.
- **Regular Audits:** Conduct HIPAA compliance reviews and penetration testing.
- **Vendor Agreements:** If using third-party tools, ensure they meet HIPAA standards.

### 2.5 Optimization (5 points)

#### 2.5.1 Method to Address Overfitting

Propose **Cross-Validation with Regularization:** Use k-fold cross-validation during training to evaluate model performance on unseen data. Apply L2 regularization (Ridge) in the Random Forest or underlying base models to penalize complex features, reducing overfitting by discouraging reliance on noise in the training data. This ensures the model generalizes better to new patients.

---

## 3 PART 3: CRITICAL THINKING (20 POINTS)

### 3.1 Ethics & Bias (10 points)

- **Impact of Biased Data:** If the training data is biased (e.g., against a racial minority that historically received less comprehensive care), the model will learn these patterns. It might systematically **under-predict** the readmission risk for these patients. Consequently, they would not receive the proactive interventions, leading to worse health outcomes and

perpetuating existing health disparities. This is a serious case of an AI model causing "allocative harm."

- **Mitigation Strategy: Pre-processing with Reweighting.** Analyse the training data for disparities across sensitive attributes (e.g., race, gender). Apply techniques like reweighting the training instances so that the model pays more attention to examples from underrepresented groups, effectively balancing its learning signal. This should be coupled with rigorous fairness audits using metrics like "Equalized Odds" post-training.

### 3.2 Trade-offs (10 points)

- **Interpretability vs. Accuracy:** In healthcare, a slightly less accurate but interpretable model (like Logistic Regression) is often preferred over a "black box" model (like a complex Neural Network). The reason is **trust and accountability**. A doctor must understand the model's reasoning to make an informed clinical decision. For example, knowing that "a high sodium level was the primary driver of this risk score" is actionable. Without interpretability, the model's output is just a number that is difficult to act upon and dangerous to blindly follow.
  - **Impact of Limited Computational Resources:** Limited resources would heavily favor simpler models. Training and deploying a large Neural Network or even an ensemble method like Random Forest at scale can be expensive. A hospital might opt for a highly optimized Logistic Regression or a shallow Decision Tree that can run efficiently on their existing servers, even if it means sacrificing a few percentage points of predictive accuracy. The focus would be on robust feature engineering to get the most out of a simpler, faster, and cheaper model.
-

## **4 PART 4: REFLECTION & WORKFLOW DIAGRAM (10 POINTS)**

### **4.1 Reflection (5 points)**

#### **4.1.1 Most Challenging Part:**

The most challenging part was navigating the **trade-offs in the "Model Development" stage**, particularly between accuracy and interpretability for the hospital case study. Choosing a complex model like XGBoost might yield a higher F2-Score, but justifying the choice of a simpler Logistic Regression model required a deeper understanding of the real-world context where trust and explainability are non-negotiable constraints. This highlights that software engineering for AI is not just about maximizing metrics but about building a system that fits seamlessly and responsibly into a human-centric process.

#### **4.1.2 Improvement with More Time/Resources:**

With more time and resources, I would focus on:

1. **Advanced Feature Engineering:** Incorporating more temporal data (e.g., vital sign trends during the hospital stay) using time-series analysis.
2. **Robust MLOps Pipeline:** Implementing a full MLOps pipeline with continuous integration/continuous deployment (CI/CD), automated retraining triggers based on performance monitoring, and a more sophisticated concept drift detection system.
3. **User-Centric Design:** Conducting extensive user interviews with doctors to design the most effective way to present the risk score and explanations within their workflow.

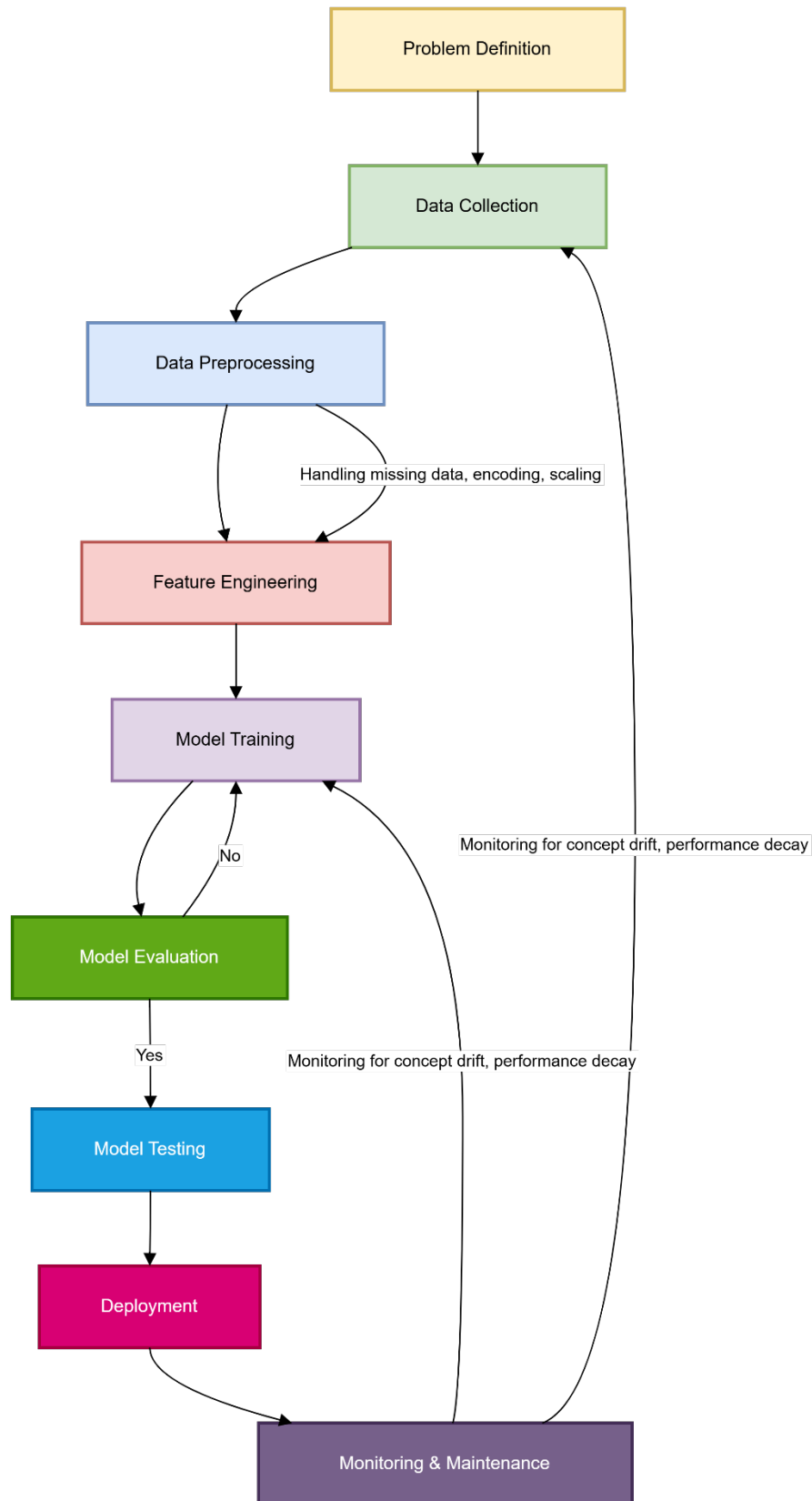


Figure 1: AI Development Workflow

### **Stages:**

1. **Business & Problem Definition:** Understand the "why," define objectives, stakeholders, and KPIs.
2. **Data Collection & Annotation:** Gather and label data from relevant sources.
3. **Data Preprocessing & Feature Engineering:** Clean data, handle missing values, and create informative features.
4. **Model Selection & Training:** Choose an algorithm, train it on the prepared data.
5. **Model Evaluation & Validation:** Rigorously test the model on unseen data using relevant metrics.
6. **Model Deployment & Integration:** Serve the model via an API and integrate it into the existing software system.
7. **Live Monitoring & Maintenance:** Continuously monitor for performance decay and concept drift.
8. **Model Retraining / Updating:** Periodically retrain the model with new data to maintain its accuracy and relevance.

## **5 REFERENCES**

1. IBM Cloud Education. (2020, September 15). *CRISP-DM*. IBM. <https://www.ibm.com/cloud/learn/crisp-dm>
2. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow* (2nd ed.). O'Reilly Media.
3. U.S. Department of Health & Human Services. (2013, July 26). *Summary of the HIPAA Privacy Rule*. <https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html>

4. Molnar, C. (2022). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. <https://christophm.github.io/interpretable-ml-book/>
5. Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.