Part 1 – Question 1: Problem Definition

1. problem definition

Problem statement*:*
Develop an AI system that predicts the probability that a currently enrolled university student will drop out of their degree program within the next semester, using historical academic, behavioral, and demographic dat.

2. Objectives and stakeholders

- **Objective 1**: Accurately identify students at high risk of dropping out within the next semester so that the university can intervene early.
- **Objective 2**: Reduce the overall semester-to-semester dropout rate by enabling targeted academic and counseling support based on the model's risk scores.
- **Objective 3**: Provide actionable and interpretable risk factors (e.g., low attendance, failing grades) that support academic advisors in decision-making rather than acting as a black-box tool.

Stakeholders

- **Stakeholder 1 – University administration and management**: They care about retention rates, funding, and institutional reputation, all of which are influenced by dropout rates.
- **Stakeholder 2 – Students and academic advisors**: Students are directly affected by interventions based on model predictions, and advisors rely on the system to prioritize support and avoid unfairly labeling students.
- **Stakeholder 3 – Lecturers**
- **Stakeholder 4 – Scholarship sponsors**

3. KPI (1 key performance indicator)

A strong KPI:

- **Primary KPI**:
    - Measure the success of the AI system by tracking the percentage reduction in the dropout rate among students flagged as high risk, relative to the previous year's baseline without the AI-driven interventions.

Part 1, Question 2: Data Collection & Preprocessing

Step 1 – Identify Two Data Sources

The success of an AI-driven student dropout prediction system depends on selecting rich, relevant data sources. Two key sources include:

- **Academic Records and Performance Logs:** This encompasses students' course grades, GPA histories, exam scores, attendance, and assignment submission rates. Academic records

offer direct indicators of academic engagement and performance, which are critical to predicting dropout risk.

- **Demographic and Enrollment Data:** Data such as age, gender, socioeconomic status, scholarship eligibility, enrollment mode (full-time/part-time), and degree program. These factors often correlate strongly with dropout rates and provide context beyond pure academic performance.

Step 2 – Explain One Potential Bias in the Data

A common challenge in educational data is **socioeconomic bias**. If the dataset contains a disproportionately high number of students from privileged backgrounds … whether through scholarships, financial aid, or full-time study … the learned model may underestimate dropout risk for underrepresented groups, such as working students or those from low-income families. This bias can result in unfair or ineffective interventions, potentially exacerbating inequality within academic institutions.

Step 3 – Outline Three Preprocessing Steps

1. **Handling missing values:**
    - Employ imputation techniques such as filling missing grades or attendance data with the median/mean value of the respective feature, or flag records for exclusion if missingness exceeds a reasonable threshold.
2. **Normalization of features:**
    - Scale continuous features (GPA, age, exam score) via min-max or z-score normalization, ensuring one feature does not dominate model learning due to its raw value range.
3. **Encoding categorical variables:**
    - Convert categorical columns such as “major”, “enrollment status”, or “gender” using one-hot encoding. This step ensures all model inputs are numerical and algorithms interpret these factors effectively.

Effective preprocessing transforms noisy, inconsistent raw data into a structured, model-ready format, maximizing the reliability and impact of machine learning results.

Part 1, Question 3: Model Development

3.1 Model Choice & Justification

Recommended Model:

- Random Forest Classifier

Justification:

- Random Forests are widely used for educational dropout prediction because they handle both numerical (grades, GPA) and categorical (major, status) variables without special preprocessing.
- They naturally manage missing data, reduce the risk of overfitting through ensemble averaging, and provide feature importance rankings, which boosts interpretability (important in educational contexts).
- Recent studies demonstrate their high performance, with accuracy and F1-scores often above 0.80 for student dropout tasks.

In a real workflow, you could also compare Logistic Regression (simple, explainable) with more complex models like XGBoost or Neural Networks, but Random Forest is strong for a small-to-mid feature set and when interpretability is a plus.

3.2 Data Splitting

Proper data splitting prevents overfitting and allows unbiased evaluation. You should:

- **Training set (60%)**: Used fit model parameters.
- **Validation set (20%)**: Used for hyperparameter tuning.
- **Test set (20%)**: Only used at the very end to estimate final performance.

Use stratified splitting to ensure both dropout and non-dropout cases are well represented in each set, especially if the dataset is imbalanced (as is common in dropout prediction).

3.3 Hyperparameters to Tune (and Why)

1. **Number of Trees (`n_estimators`):**
    - Controls the size of the forest (number of decision trees). More trees can increase accuracy but also computational cost—so you want to find a point of diminishing returns.
2. **Maximum Tree Depth (`max_depth`):**
    - Limits how deep each individual tree can grow. A very deep tree may overfit, while a shallow one may underfit—tuning this helps strike a balance between bias and variance.

Extra: You could also tune the minimum samples per split (`min_samples_split`) or the criterion for splits (`gini` or `entropy`), but `n_estimators` and `max_depth` usually have the biggest impact.

Part 1, Question 4: Evaluation & Deployment

4.1 Evaluation Metrics (2 metrics & their relevance)

In the context of student dropout prediction, it's critical to minimize the number of students at risk who are missed by the model (false negatives), while still keeping the number of false alarms (false positives) manageable. Two appropriate metrics are:

1. Recall (Sensitivity):

- **Definition:** Recall measures the proportion of actual dropouts that were correctly predicted by the model.
- **Relevance:** High recall ensures that most at-risk students are identified, even if this means flagging a few who wouldn't actually drop out. In universities, it's often preferable to "over-warn" rather than let truly at-risk students slip through unnoticed.

2. Precision:

- **Definition:** Precision measures the proportion of students predicted at-risk who actually do drop out.
- **Relevance:** Precision is important to avoid overwhelming advisors and wasting resources on unnecessary interventions. High precision means targeted interventions are more likely to reach those who genuinely need help.

Other commonly used metrics include F1-score, Accuracy, and AUC-ROC, but recall and precision directly address key outcomes in dropout prediction contexts.

4.2 Concept Drift

Concept drift refers to changes over time in the statistical properties of the input features or the underlying relationship between inputs and dropout risk, leading to a model's declining performance after deployment. For example, changes in university policies, pandemic-driven remote learning, or economic shifts could affect dropout patterns, making the model's learned relationships outdated.

Monitoring after deployment:

- Continuously track performance metrics like recall, precision, and dropout rates on new data.
- Set up alerting if metrics degrade significantly.
- Periodically retrain the model using recent data to ensure relevance.

4.3 Technical Deployment Challenge: Scalability

One major technical challenge during deployment is **scalability …** ensuring the system can handle predictions for thousands (or tens of thousands) of students quickly, especially during peak periods like semester enrollment. If the model is slow or cannot scale horizontally (adding more machines), delays could harm intervention timing or user trust. Addressing this may require optimizing code, choosing efficient deployment platforms, or enabling batch processing.

Part 2: Case Study Application – Hospital Readmission Risk

2.1 Problem Scope (5 points)

Problem Definition: Develop an AI system for predicting the risk that a discharged patient will be readmitted to the hospital within 30 days.

Objectives:

1. Accurately identify patients at high risk of readmission to enable timely preventive care.
2. Reduce 30-day readmission rates by targeting interventions more effectively.
3. Increase resource efficiency for hospital staff by prioritizing cases using predicted risk scores.

Stakeholders:

- Hospital management and care teams (they want to lower costs and improve patient health outcomes).
- Patients (whose care decisions and outcomes may change based on risk predictions).

2.2 Data Strategy (10 points)

Data Sources

- **Electronic Health Records (EHRs):** Medical history, diagnoses, treatments, discharge summaries, medications.
- **Demographic Data:** Age, gender, socioeconomic status, and insurance type.

Ethical Concerns

1. **Patient Privacy:** Health data is highly sensitive—misuse or accidental leakage could harm individuals or breach regulations (e.g., HIPAA).
2. **Algorithmic Bias:** If training data underrepresents certain groups (e.g., minorities, uninsured patients), predictions may be unfair or less accurate for those populations.

Preprocessing Pipeline

1. **Data Cleaning:** Remove duplicate records, handle inconsistent entries, and detect data quality issues.
2. **Handling Missing Values:** Impute missing lab results/fields using median value or clinically justified approaches.
3. **Feature Engineering:** Extract temporal features (days since last admission, count of chronic conditions) and encode categorical variables (diagnosis codes, discharge location) with one-hot encoding.

4. **Normalization:** Scale continuous variables such as age or lab measurements.

2.3 Model Development (10 points)

Model Selection & Justification

- **Gradient Boosted Trees (e.g., XGBoost):** Excellent for tabular, heterogeneous hospital data, interpretable (via feature importance), robust to missing values, and achieves strong predictive performance in medical tasks.

Hypothetical Confusion Matrix & Precision/Recall Calculation

Suppose the model on a test set produced:

	Actual Readmitted	**Actual Not Readmitted**
Predicted Yes	40 (TP)	20 (FP)
Predicted No	10 (FN)	130 (TN)

- ***Precision:*** $Precision=TP/(TP+FP)= 40/(40+20)=0.67$
- **Recall:** $Recall=TP/(TP+FN)=40/(40+10)=0.80$

2.4 Deployment (10 points)

Integration Steps

1. Develop REST API or integrate model into EHR software.
2. Automate risk prediction on discharge, surfacing risk scores to care teams.
3. Log predictions and monitor model performance for ongoing improvement.

Ensuring Compliance

- Audit data handling for HIPAA compliance: encrypt patient data, restrict access, log all usage.
- Validate model fairness and accuracy for all subgroups, documenting risks and mitigation plans.
- Update governance protocols for AI-assisted decision making, with oversight by medical ethics boards.

2.5 Optimization – Preventing Overfitting (5 points)

Method: Implement cross-validation and early stopping during model training; use regularization techniques (e.g., limiting tree depth, increasing min_child_weight) to avoid the model fitting noise in training data. Monitor validation metrics and retrain only if generalization improves.

Part 3: Critical Thinking

3.1 Ethics & Bias (10 points)

How might biased training data affect patient outcomes in the readmission scenario?

If the training data underrepresents certain groups … such as minorities, lower-income patients, uninsured individuals, or the elderly … the model may systematically underrate the readmission risk for these groups. That could lead to:

- *Missed interventions*: At-risk patients don't get preventive care, suffering worse outcomes or being readmitted unnecessarily.
- *Resource misallocation*: Hospital resources might be wasted on lower-risk patients falsely flagged as "at risk" while higher-risk but underrepresented patients are ignored.
- *Amplified inequity*: Marginalized groups suffer disproportionately, potentially undermining trust and violating ethical or legal standards (like fairness regulations in healthcare).

Strategy to mitigate bias

- Audit model performance *separately* for each demographic group (age, race, insurance status).
- If disparities are found, rebalance the training dataset using oversampling, data augmentation, or instance weighting for underrepresented groups.
- Optionally, apply fairness constraints or thresholds directly in training.

> “Bias doesn’t just cause technical inaccuracy—it can directly harm patient care and legal compliance. Rigorous subgroup testing and data balancing are essential, not optional.”

3.2 Trade-offs (10 points)

Interpretability vs Accuracy in Healthcare

- More *interpretability* (e.g., using Logistic Regression or Decision Trees) enables doctors and care teams to see *why* a prediction was made, supporting trust and actionable decisions. This

is vital for medical settings where clinical oversight and justification are required by law/regulation, and black-box predictions often get rejected by practitioners and auditors.

- More *accuracy* (e.g., using deep neural networks, complex ensemble models) can increase predictive performance, but may obscure reasoning—leading to "black-box" outputs that are harder to scrutinize, justify, or contest in a clinical environment.
- The ideal solution is to find the *minimum acceptable trade-off*: Use the most accurate model possible that clinicians can still interpret, possibly with model-agnostic explanation tools (like SHAP values).

Impact of Resource Constraints on Model Choice

If the hospital has limited computational resources:

- Large, complex models (deep learning, big ensembles) become impractical—they may require more memory, training time, and speed than the hospital's systems can handle.
- Resource-constrained hospitals should prefer simpler, efficient models (like Logistic Regression or small Random Forests) that deliver fast predictions, are easy to deploy on standard hardware, and simplify maintenance.
- Sometimes *accuracy must be sacrificed for affordability and stability*—as no prediction at all is better than an unreliable or unsustainable one.

Part 4: Reflection & Workflow Diagram

4.1 Reflection (5 points)

Most challenging part:

The most challenging part of the AI development workflow for the hospital readmission case was ensuring the **fairness and robustness of the predictive model**. Healthcare datasets are often imbalanced and rife with systemic biases; identifying, quantifying, and mitigating these biases requires more than standard preprocessing and modeling … it demands in-depth subgroup analysis, ethics review, and transparent stakeholder communication. This stage is also where technical decisions have the highest risk of impacting real-world patient care, so both technical and ethical scrutiny are unavoidable.

Improving the approach with more resources:

With more time and resources, I would integrate more advanced **fairness auditing tools** (such as model explainers and group-performance metrics), include broader, more diverse data sources, and establish a continuous feedback loop with clinical end-users. This would both improve the model's fairness and support more meaningful, acceptable deployment.

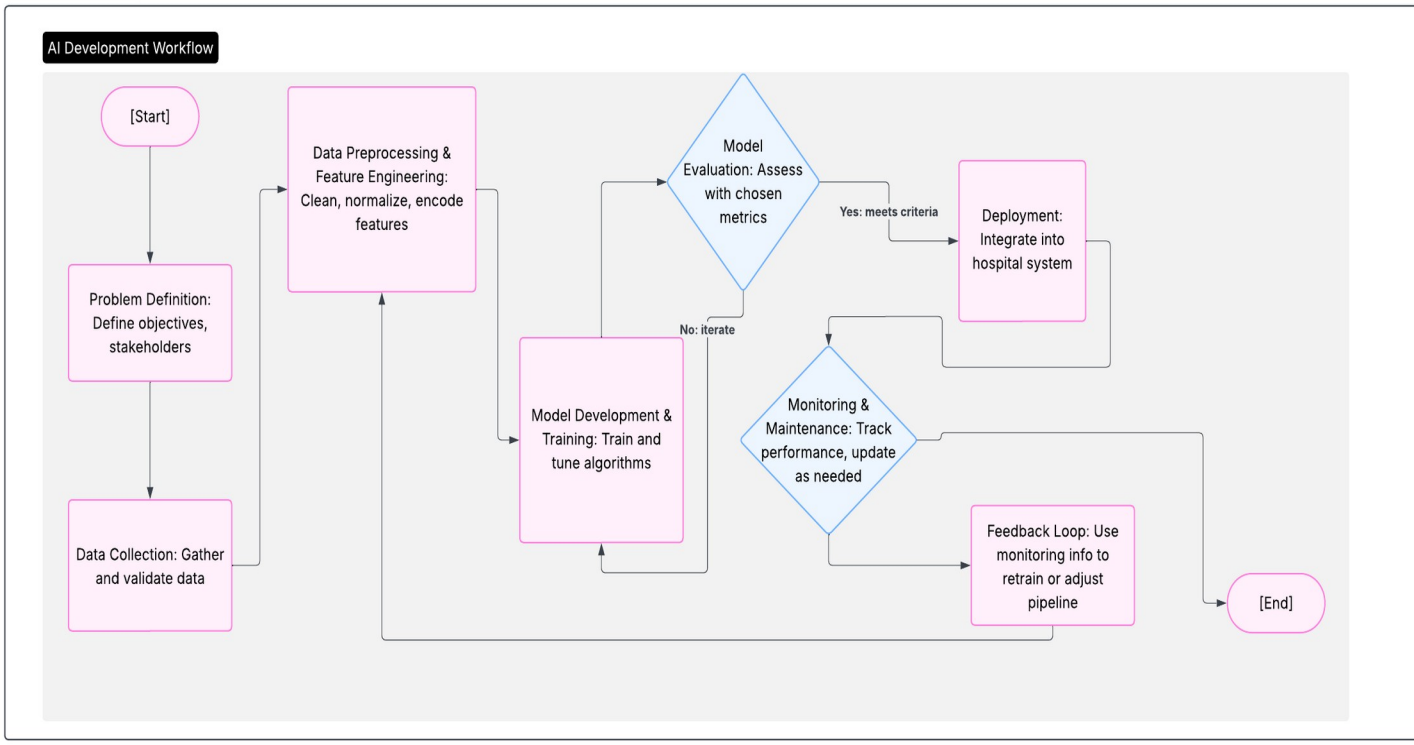
4.2 AI Development Workflow Diagram (5 points)

Below is a typical flowchart outlining the AI development workflow for this case study. (In your PDF, you should actually *draw* this as a diagram using flowchart tools like Lucidchart, draw.io, or

hand-sketched and scanned—since markdown cannot depict flow diagrams. Here, you'll find a structured, stepwise breakdown so you can accurately represent each stage.)

AI Development Workflow Steps:

1. **Problem Definition**
    - Clearly define task, objectives, and stakeholders.
2. **Data Collection**
    - Gather data from EHRs, demographics, etc.
    - Identify and ensure data quality and compliance.
3. **Data Preprocessing & Feature Engineering**
    - Clean, impute, normalize, encode features.
    - Engineer new features based on domain needs.
4. **Model Development**
    - Select and train model(s) (e.g., XGBoost, Random Forest).
    - Tune hyperparameters.
5. **Model Evaluation**
    - Perform train/validation/test split.
    - Evaluate using precision, recall, and other relevant metrics.
    - Audit for bias and fairness.
6. **Deployment**
    - Integrate model into clinical systems (e.g., via API, EHR integration).
    - Develop user interfaces and decision support tools.
7. **Monitoring & Maintenance**
    - Monitor real-world model performance for concept drift and errors.
    - Maintain, retrain, and update model as needed.
8. **Feedback & Continuous Improvement**
    - Collect feedback from users (clinicians, patients, admins).
    - Update data sources, features, and models to improve over time.


AI Development Workflow
[Start]
Problem Definition: Define objectives, stakeholders
Data Collection: Gather and validate data
Data Preprocessing & Feature Engineering: Clean, normalize, encode features
Model Development & Training: Train and tune algorithms
Model Evaluation: Assess with chosen metrics
Yes: meets criteria
No: iterate
Deployment: Integrate into hospital system
Monitoring & Maintenance: Track performance, update as needed
Feedback Loop: Use monitoring info to retrain or adjust pipeline
[End]