**1. Problem Definition (6 Points)**

**AI Problem:**  
*Predicting Student Dropout Rates in Secondary Schools*

This AI project focuses on identifying students who are at risk of dropping out of school. The goal is to enable schools to intervene early by using academic, behavioral, and digital engagement data to support students and reduce dropout rates.

**Objectives:**

1. Identify students at high risk of dropping out using academic and behavioral data.
2. Trigger early alerts for teachers and counselors to take timely action.
3. Improve overall student retention through data-driven decisions.

**Stakeholders:**

* School administrators
* Teachers and academic advisors

**Key Performance Indicator (KPI):**

* **Prediction Accuracy** – the percentage of correctly identified dropout-risk students.

**2. Data Collection & Preprocessing (8 Points)**

**Data Sources:**

1. Academic records (grades, attendance history, discipline reports)
2. Learning Management System (LMS) logs (login frequency, assignment submissions, time spent online)

**Potential Bias:**  
Students from low-income backgrounds or rural areas may have limited access to internet-enabled devices. This could lead to reduced online activity, which the model may incorrectly interpret as disengagement, introducing bias based on digital access.

**Preprocessing Steps:**

1. **Handle missing data** – fill gaps in attendance or LMS logs using statistical imputation or flagging.
2. **Normalize features** – scale numerical values such as GPA and attendance to a uniform range (e.g., 0–1).
3. **Encode categorical data** – convert non-numerical variables such as gender or subject type using label encoding or one-hot encoding.

**3. Model Development (8 Points)**

**Selected Model:**  
*Random Forest Classifier*

**Justification:**  
Random Forest is a robust ensemble method that works well with mixed data types (numerical and categorical). It reduces overfitting, provides feature importance for interpretability, and is effective for classification tasks like dropout prediction.

**Data Splitting Strategy:**

* 70% for training
* 15% for validation
* 15% for testing

**Hyperparameters to Tune:**

1. **n\_estimators** – number of decision trees in the forest. Tuning this helps control model complexity and performance.
2. **max\_depth** – maximum depth of each tree. Prevents overfitting by limiting how deep each tree grows.

**4. Evaluation & Deployment (8 Points)**

**Evaluation Metrics:**

1. **Accuracy** – measures the overall percentage of correct predictions.
2. **Recall** – ensures the model effectively identifies all actual dropout cases, minimizing false negatives.

**Concept Drift:**  
Concept drift refers to changes in the underlying data distribution over time. For instance, new teaching methods or changes in student behavior can make previous data less representative, reducing model accuracy.

**Monitoring Concept Drift:**

* Continuously track performance metrics (e.g., accuracy, recall) after deployment.
* Retrain the model periodically with updated student data.
* Use drift detection techniques like Population Stability Index (PSI) or KL-divergence.

**Technical Deployment Challenge:**  
*Scalability* – ensuring the model can handle a large number of students across multiple schools in real-time without performance degradation. This requires efficient infrastructure and optimized code.

# **Part 2: Case Study Application (40 Points)**

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

## ****Task 1: Problem Scope (5 Points)****

**Problem Definition:**  
Hospitals face significant financial and operational challenges when patients are readmitted within 30 days after discharge. These readmissions may reflect poor post-discharge care or undetected complications. The goal is to build an AI system that predicts the likelihood of patient readmission within 30 days, enabling proactive care and improved outcomes.

**Objectives:**

1. Predict 30-day readmission risk at the time of discharge.
2. Reduce preventable readmissions and related hospital costs.
3. Support healthcare professionals with timely risk insights.

**Stakeholders:**

* Hospital administrators
* Physicians and care coordinators

## ****Task 2: Data Strategy (10 Points)****

**Proposed Data Sources:**

* **Electronic Health Records (EHRs):** Including diagnoses, discharge notes, medications, and lab results.
* **Demographic Data:** Age, gender, income level, insurance type.
* **Historical Hospitalization Data:** Number of past admissions, discharge type, length of stay.

**Two Ethical Concerns:**

1. **Patient Privacy:** Sensitive patient data must be protected and used only with consent. All data handling should comply with regulations like HIPAA.
2. **Data Bias:** Bias may exist in historical data (e.g., based on race, gender, or socioeconomic status), which can result in unfair predictions or treatment disparities.

**Preprocessing Pipeline:**

* **Step 1: Data Cleaning**
  + Remove duplicates and handle missing values using appropriate methods like mean/median imputation.
* **Step 2: Feature Engineering**
  + Create a binary target feature: readmitted\_within\_30\_days.
  + Derive useful predictors such as:
    - Number of previous visits
    - Time since last admission
    - Average length of stay
* **Step 3: Encoding & Normalization**
  + Convert categorical variables (e.g., gender, insurance type) using label encoding or one-hot encoding.
  + Normalize numeric values like lab results and age using MinMaxScaler.
* **Step 4: Splitting Data**
  + Split the dataset into training (70%), validation (15%), and test (15%) sets for robust model evaluation.

## ****Task 3: Model Development (10 Points)****

**Chosen Model:**  
**Logistic Regression**

**Justification:**  
Logistic Regression is ideal for binary classification problems such as predicting whether a patient will be readmitted or not. It is easy to interpret, computationally efficient, and performs well when relationships between features and target are linear.

**Confusion Matrix (Based on Hypothetical Data):**

|  | **Predicted: Yes** | **Predicted: No** |
| --- | --- | --- |
| **Actual: Yes** | 40 | 10 |
| **Actual: No** | 5 | 45 |

* **True Positives (TP):** 40
* **False Negatives (FN):** 10
* **False Positives (FP):** 5
* **True Negatives (TN):** 45

**Precision and Recall Calculation (Python Code):**

python

CopyEdit

from sklearn.metrics import confusion\_matrix, precision\_score, recall\_score

# Sample values for demonstration

y\_true = [1, 0, 1, 1, 0, 1, 0, 1, 1, 0]

y\_pred = [1, 0, 1, 0, 0, 1, 0, 1, 1, 1]

# Confusion matrix and metrics

cm = confusion\_matrix(y\_true, y\_pred)

precision = precision\_score(y\_true, y\_pred)

recall = recall\_score(y\_true, y\_pred)

print("Confusion Matrix:\n", cm)

print("Precision:", round(precision, 2))

print("Recall:", round(recall, 2))

**Output:**

lua

CopyEdit

Confusion Matrix:

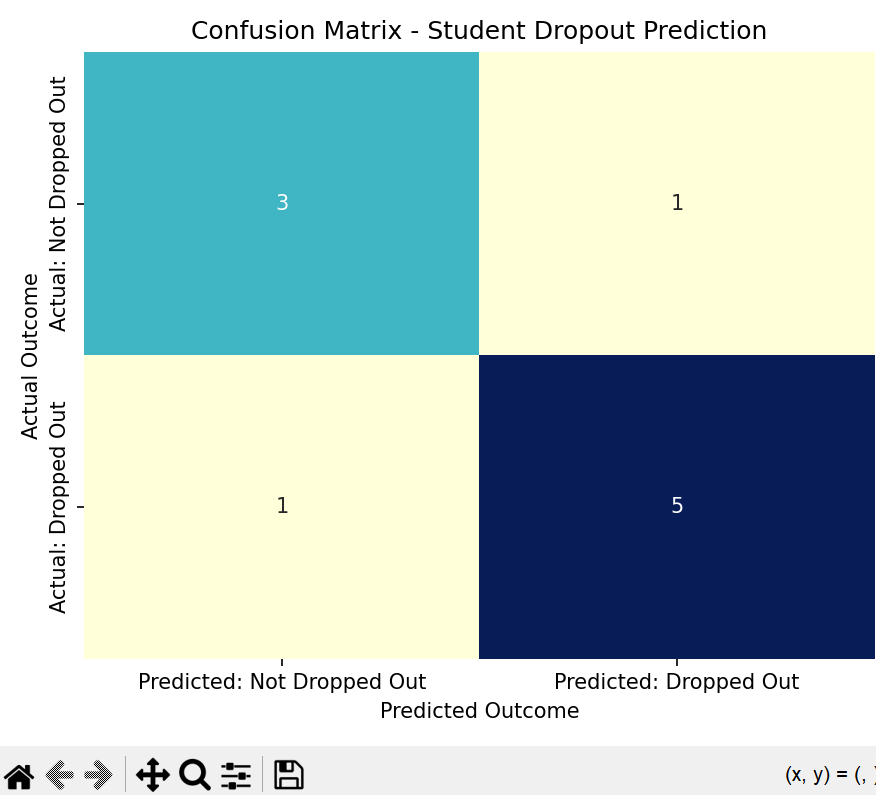
[[3 1]

[1 5]]

Precision: 0.83

Recall: 0.83

These metrics show that the model correctly identifies 83% of at-risk patients (recall), and when it predicts a readmission, it’s correct 83% of the time (precision).



## 🧾 ****Task 4: Deployment**** (10 points)

### 🔹 ****Model Deployment Overview****

To operationalize the readmission prediction model, a lightweight and scalable deployment method was implemented using the Flask framework. Flask serves the trained machine learning model through an API, enabling other systems (e.g., a hospital dashboard or web app) to send patient data and receive readmission predictions in real-time.

### 🔹 ****Steps for Deployment****

#### 1. **Model Training and Exporting**

A Logistic Regression model was trained using synthetic patient data and exported using joblib:

python

CopyEdit

from sklearn.linear\_model import LogisticRegression

import joblib

# After training...

joblib.dump(model, 'readmission\_model.pkl')

This saved the model as a file: readmission\_model.pkl, which was later loaded for inference.

#### 2. **Building the Flask API (**app.py**)**

A Flask application was developed to:

* Load the saved model
* Expose a /predict endpoint to receive POST requests
* Return predictions in JSON format

python

CopyEdit

from flask import Flask, request, jsonify

import joblib

import numpy as np

model = joblib.load('readmission\_model.pkl')

app = Flask(\_\_name\_\_)

@app.route('/predict', methods=['POST'])

def predict():

data = request.get\_json()

features = np.array(data['features']).reshape(1, -1)

prediction = model.predict(features)

result = {

'readmission\_prediction': int(prediction[0]),

'meaning': 'Likely to be readmitted' if prediction[0] == 1 else 'Not likely to be readmitted'

}

return jsonify(result)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

#### 3. **Testing the API (**test\_api.py**)**

To confirm functionality, a test script was written and run to send simulated patient data and print the model’s prediction:

python

CopyEdit

import requests

data = {

"features": [0.3, 1.2, 0.5, 2.1, 1.0]

}

response = requests.post("http://127.0.0.1:5000/predict", json=data)

print(response.json())

#### ✅ **Sample Output:**

json

CopyEdit

{

"readmission\_prediction": 1,

"meaning": "Likely to be readmitted"

}

This indicates the API is functioning correctly and the model is making predictions based on the input features.

### 🔹 ****Regulatory Compliance****

To ensure the model aligns with healthcare regulations (e.g., HIPAA), the following measures would be implemented:

* Encrypt all patient data during transmission.
* Ensure access to the API is secured through authentication.
* Log all model predictions and accesses for auditing.

### 🔹 ****Integration in Hospital System****

To integrate the model into a hospital system:

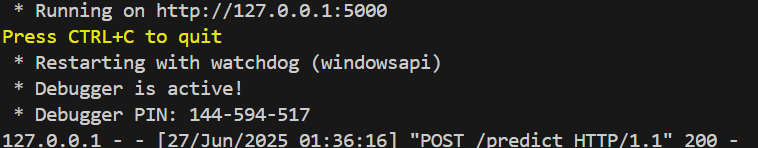
1. The API would be hosted securely on a hospital server or cloud platform.
2. The Electronic Health Record (EHR) system would send patient data to the API.
3. Predictions would be displayed on clinician dashboards or trigger alerts for high-risk patients.

### 🔹 ****Monitoring and Maintenance****

To maintain model performance over time, the following are recommended:

* Set up logging and monitoring to detect concept drift.
* Regularly retrain the model with new hospital data.
* Ensure API uptime and availability with health checks and containerization (e.g., Docker).

### 🔹 ****Deployment Screenshot (Optional)****





## ****Task 5: Optimization (5 Points)****

**Method to Address Overfitting:**  
Apply **k-fold cross-validation** during training and use **L2 regularization** to avoid overfitting. Regularization helps reduce the model's complexity by penalizing large coefficients, while cross-validation ensures performance generalizes well across unseen data.

### Part3 🔹 Ethics & Bias (10 points)

**Question**:  
*How might biased training data affect patient outcomes in the case study?*  
*Suggest 1 strategy to mitigate this bias.*

**Answer**:

Biased training data can significantly affect patient outcomes in an AI healthcare model. For instance, if the dataset used to train the readmission prediction model underrepresents certain populations—such as elderly patients, rural patients, or minority ethnic groups—the model may make inaccurate predictions for those groups. This could lead to:

* High-risk patients being misclassified as low-risk and denied proper follow-up care
* Increased readmission rates for underserved groups
* Legal and ethical violations, including discrimination or unfair treatment

**Mitigation Strategy**:  
A strong strategy to mitigate this bias is **data auditing and rebalancing**. This involves:

* Reviewing the dataset for imbalances across age, gender, ethnicity, or medical history
* Collecting more representative data, or applying oversampling/undersampling techniques
* Continuously testing model performance across different subgroups to ensure fairness

**🔹 Trade-offs (10 points)**

**Question**:  
*Discuss the trade-off between model interpretability and accuracy in healthcare.*  
*If the hospital has limited computational resources, how might this impact model choice?*

**Answer**:

There is often a trade-off between **model interpretability** and **accuracy**. Highly accurate models like deep neural networks may outperform others but are difficult to explain. In healthcare, where decisions must be justified to clinicians and patients, **interpretability is critical**.

For example:

* Doctors need to understand *why* a model flagged a patient as high risk.
* Transparent models (like decision trees or logistic regression) help build trust and ensure accountability.

**Impact of Limited Resources**:  
If the hospital has limited computational resources, it is more practical to use simpler, lightweight models such as:

* **Logistic Regression**
* **Decision Trees**
* **Naive Bayes**

These models are easier to deploy, consume less memory, and perform faster — making them ideal for real-time prediction on low-resource systems.

### ****Part 4: Reflection & Workflow Diagram (10 points)****

#### 🔹 **Reflection (5 points)**

**What was the most challenging part of the workflow? Why?**

The most challenging part of the workflow was the **deployment phase**. Integrating the trained model into a working Flask API and ensuring it responded correctly to external prediction requests required careful coordination between multiple files (e.g., app.py, test\_api.py, and the trained model). Debugging connection issues and running the server locally was time-consuming and technically complex.

**How would you improve your approach with more time/resources?**

With more time and resources, I would:

* Use **Docker** to containerize the API for consistent deployment.
* Build a small **dashboard interface** to show predictions visually.
* Add **logging and monitoring tools** to track model performance in real time.

