# Part 2: Case Study Application – Predicting 30-Day Hospital Readmission Risk

## 1. Problem Scope (5 Points)

### 1.1 Problem Statement

Hospital readmissions within 30 days are a persistent global healthcare challenge. They often indicate incomplete recovery, inadequate discharge planning, or lack of post-discharge support. High readmission rates increase treatment costs, burden healthcare systems, and negatively affect patient satisfaction. To address this, the hospital aims to develop an AI-based predictive model that estimates the probability of a patient being readmitted within 30 days of discharge. Using historical Electronic Health Record (EHR) data, such as demographics, diagnoses, previous admissions, medications, and discharge details-the model will flag high-risk patients for early intervention and follow-up.

### 1.2 Objectives

**Main Objective:**
Develop and deploy an interpretable machine learning model that predicts 30-day hospital readmission risk using structured EHR data.

**Specific Objectives:**

1. Collect and preprocess anonymized EHR data for model training.
2. Design a preprocessing pipeline with feature engineering and data validation.
3. Compare baseline and advanced ML models (Logistic Regression, XGBoost).
4. Evaluate model accuracy using confusion matrix, precision, and recall.
5. Deploy the final model securely and ensure compliance with health data regulations.

### 1.3 Stakeholders:

Patients, Doctors & Nurses, Hospital Administrators, IT & Data Science Team, Health Information Officers, and Policy Makers & Insurers all have vested interests in ensuring data privacy, clinical accuracy, and effective AI integration**.**

### 1.4 Success Criteria:

- Model precision ≥ 80% and recall ≥ 70%.
- Seamless EHR integration (FHIR/HL7 compliant).
- SHAP-based interpretability for clinical acceptance.
- HIPAA-compliant storage, encryption, and access control.

# 2. Data Strategy (10 Points)

### 2.1 Data Sources

- **Electronic Health Records (EHRs):** Diagnoses (ICD-10 codes), lab results, vitals, discharge summaries.
- **Demographic Data:** Age, sex, insurance type, socioeconomic index.
- **Utilization History:** Number of prior admissions, emergency visits, length of stay.
- **Medication & Treatment Data:** Medication counts, comorbidities, follow-up plans.

### 2.2 Ethical Concerns

1. **Patient Privacy:** EHRs contain highly sensitive health data. Strict data anonymization, encryption, and secure access control are mandatory.
2. **Bias & Fairness:** The dataset may underrepresent certain populations (e.g., rural or low-income groups), leading to skewed predictions. Continuous bias auditing is necessary.

### 2.3 Preprocessing Pipeline

| Step | Description |
|---|---|
| **1. Data Collection** | Securely extract EHR data through APIs or SQL connections. |
| **2. Data Cleaning** | Handle duplicates, inconsistent timestamps, and missing discharge info. |
| **3. Handling Missing Data** | Apply median imputation for numeric values and “Unknown” category for categorical. |
| **4. Feature Engineering** | Create num_prev_admissions, length_of_stay, comorbidity_index, days_since_last_visit, follow_up_scheduled. |
| **5. Encoding & Scaling** | One-hot encode categorical features and scale continuous ones. |
| **6. Label Definition** | 1 = readmitted within 30 days, 0 = not readmitted. |
| **7. Data Split** | 70% training, 15% validation, 15% testing — ensuring stratified sampling. |

```
# data_preprocessing.py
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
# Sample readmission dataset
df = pd.read_csv("patient_records.csv")
# Fill missing data
df['follow_up_scheduled'].fillna(0, inplace=True)
df.fillna(df.median(numeric_only=True), inplace=True)
# Encode categorical variables
categorical_features = ['gender', 'insurance_type']
encoder = OneHotEncoder(drop='first', sparse_output=False)
encoded = encoder.fit_transform(df[categorical_features])
encoded_df = pd.DataFrame(encoded, columns=encoder.get_feature_names_out())
# Merge encoded columns
df = pd.concat([df.drop(categorical_features, axis=1), encoded_df], axis=1)
# Split data
X = df.drop('readmitted_30days', axis=1)
y = df['readmitted_30days']
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.3)
```

# 3. Model Development (10 Points)

### 3.1 Model Selection

**Model Chosen:** Gradient Boosting (XGBoost)

**Justification:**

- Handles mixed data types and missing values efficiently.
- Provides feature importance for interpretability.
- Delivers strong performance for structured EHR data.

**Baseline Comparison:** Logistic Regression and Random Forest for benchmarking.

### 3.2 Model Training & Evaluation (Python Code)

```
# model_development.py
import xgboost as xgb
from sklearn.metrics import confusion_matrix, precision_score, recall_score
# Train XGBoost
model = xgb.XGBClassifier(
    learning_rate=0.1, n_estimators=100, max_depth=4,
    subsample=0.8, colsample_bytree=0.8, random_state=42)
model.fit(X_train, y_train)
# Predictions
y_pred = model.predict(X_test)
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
print("Confusion Matrix:\n", cm)
print(f'Precision: {precision*100:.2f}%")
print(f'Recall: {recall*100:.2f}%")
```

**Hypothetical Output:**

| Metric | Value |
|---|---|
| Confusion Matrix | [[370, 20], [30, 80]] |
| Precision | 80.0% |
| Recall | 72.7% |

### 3.3 Interpretation

- The model correctly identifies **73%** of patients who will be readmitted.
- **Precision = 80%** means that when the model predicts “readmission,” it is correct 4 out of 5 times.
- Clinicians can confidently use these predictions to trigger follow-up protocols.

## 4. Deployment (10 Points)

### 4.1 Integration Steps

1. Containerize the model (using Docker).
2. Expose via REST API so EHR systems can send patient discharge data and receive risk scores.
3. Embed predictions in EHR dashboards with visual indicators (Low / Medium / High risk).
4. Automate alerts for high-risk patients to the care management team.
5. Set up monitoring for model drift, performance decay, and false alarm rates.
6. Train users (nurses, doctors, coordinators) on interpretation and ethical use.

4.2 Regulatory Compliance (HIPAA)

- Encrypt all patient data in transit (TLS) and at rest (AES-256).
- Restrict access with role-based controls.
- Maintain audit logs for all data access and predictions.
- Use de-identified data during model training.
- Sign Business Associate Agreements (BAA) with all third-party vendors.
- Regularly conduct bias and safety audits before clinical rollout.

# 5. Optimization (5 Points)

### Technique: Stratified K-Fold Cross-Validation

To avoid overfitting and ensure consistent results, use **Stratified 5-Fold Cross-Validation**, preserving the readmission rate across folds.

```
from sklearn.model_selection import StratifiedKFold, cross_val_score

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

scores = cross_val_score(model, X, y, cv=cv, scoring='precision')

print("Mean Precision:", scores.mean())
```

This ensures reliable generalization and helps tune hyperparameters effectively.

### References


1. IBM AI Fairness 360 Toolkit — https://aif360.mybluemix.net/
2. U.S. Department of Health & Human Services — HIPAA Privacy Rule Summary
3. T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.