

**GROUP 21**

**POWER LEARN PROJECT AI FOR SOFTWARE ENGINEERING**

**WEEK 5 ASSIGNMENT**

**GROUP MEMBERS;**

1. **BRILLIANT MWENDWA**
2. **LOWELL OWUOR**
3. **EMMANUEL BARAKA**
4. **TEDDY BRIAN**

### **PART 1; SHORT ANSWER QUESTIONS**

### **1. Problem Definition**

**Hypothetical AI Problem:** ***Determining the number of jobs to be negatively impacted by AI in the hotel industry in Mombasa****.*

#### **Objectives:**

1. Estimate the number and type of hotel jobs likely to be automated or reduced due to AI implementation over the next 5–10 years.
2. Identify departments and roles within hotels most susceptible to automation (e.g., front desk, housekeeping, food service).
3. Provide data-driven recommendations for workforce upskilling or reskilling in response to potential AI disruptions.

#### **Stakeholders:**

1. **Hotel Management and Owners** – Interested in understanding workforce changes and planning future investments in automation.
2. **Local Government and Employment Agencies** – Concerned with job security, labor market impact, and policy planning for economic stability.

#### **Key Performance Indicator (KPI):**

* Accuracy of job impact predictions compared to actual employment changes over time, measured through retrospective validation (e.g., within ±10% of actual job losses over a set period).

### **2. Data Collection & Preprocessing (8 points)**

#### **Data Sources:**

1. **Hotel Employment Records in Mombasa:** Historical and current data on job roles, employee counts, turnover rates, and AI adoption levels from hotel HR systems.
2. **Industry Reports and Surveys:** National and international hospitality industry studies (e.g., from the Kenya Tourism Board or World Travel & Tourism Council) on automation trends, AI deployment, and labor market forecasts.

#### **Potential Bias in the Data:**

* **Urban Bias:** Hotels in central Mombasa may be overrepresented in the data due to better digital records and accessibility, potentially underestimating AI's impact in smaller or rural hotel establishments.

#### **Preprocessing Steps:**

1. **Handling Missing Data:** Use imputation techniques (e.g., mean/mode imputation for numerical/categorical fields or model-based imputation) to fill in missing job counts or automation indicators.
2. **Standardizing Job Titles:** Normalize varying job titles (e.g., "Room Attendant" vs. "Housekeeper") into consistent categories to ensure comparability across datasets.
3. **Feature Encoding:** Convert categorical variables (e.g., department, job role) into numerical form using one-hot encoding or label encoding for model compatibility.

### **3. Model Development (8 points)**

#### **Chosen Model:**

* **Random Forest Regressor**

**Justification:**

* Random Forest is well-suited for handling structured/tabular data with mixed feature types (numerical and categorical).
* It can model non-linear relationships and interactions between features without requiring heavy preprocessing like normalization.
* It is robust to overfitting and provides feature importance insights, which is valuable for interpreting the key drivers of job impact.

#### **Data Splitting:**

* **70% Training Set:** Used to train the model on historical job and AI adoption data.
* **15% Validation Set:** Used during model development for hyperparameter tuning and preventing overfitting.
* **15% Test Set:** Held out until final evaluation to assess generalization performance on unseen data.

#### **Hyperparameters to Tune:**

1. **n\_estimators (Number of Trees):** Controls the number of trees in the forest. More trees can improve performance but increase training time.
2. **max\_depth (Maximum Tree Depth):** Limits how deep each tree can grow. Tuning this helps balance model complexity and avoid overfitting.

### **4. Evaluation & Deployment (8 points)**

#### **Evaluation Metrics:**

1. **Mean Absolute Error (MAE):** Measures the average magnitude of prediction errors in units of jobs impacted. It is easy to interpret and appropriate for understanding the real-world impact of prediction inaccuracies.
2. **R² Score (Coefficient of Determination):** Indicates how well the model explains the variance in the target variable. Useful for assessing the model’s overall goodness-of-fit and how much of the variability in job impact is captured by the features.

#### **Concept Drift:**

* **Definition:** Concept drift refers to changes in the underlying data patterns over time that can degrade model performance. In this case, factors like new AI technologies, regulatory changes, or shifting customer preferences may alter how AI impacts hotel jobs.
* **Monitoring Strategy:** Track prediction errors (e.g., MAE) over time and compare them against recent actual job data. If errors increase consistently, it may indicate concept drift. Also, re-evaluate model performance periodically using updated data and consider retraining if necessary.

#### **Technical Challenge – Scalability:**

* **Issue:** As more hotels and job types are included, the model may need to handle larger, more diverse datasets in real time or on a schedule.
* **Solution:** Optimize the model pipeline using batch processing, scalable cloud services (e.g., AWS SageMaker, Google AI Platform), and model versioning to ensure consistent, efficient deployment across different environment

**PART 2; CASE STUDY**

### **Problem Scope**

**Problem:** Develop an AI system to predict whether a patient will be readmitted within 30 days after hospital discharge**.**

**Objectives:**

1. Identify high-risk patients to enable targeted interventions.
2. Reduce avoidable readmissions and associated costs.
3. Improve patient care quality and hospital performance metrics.

**Stakeholders:**

* Hospital administrators
* Physicians and care coordinators

### **Data Strategy (10 points)**

**Data Sources:**

* **Electronic Health Records (EHRs):** Diagnosis codes, lab results, length of stay, medications.
* **Demographic Data:** Age, gender, socioeconomic status.
* **Previous Admissions Data:** Readmission history, discharge summaries.

**Ethical Concerns:**

1. **Patient Privacy**: Handling sensitive health data must comply with HIPAA and ensure secure data storage and processing.
2. **Bias in Predictions:** Historical biases (e.g., unequal treatment based on race or income) can be learned by the model and lead to unfair outcomes.

**Data Preprocessing.**

The data preprocessing was done as follows;

I'll now open and review the Jupyter notebook file you uploaded to explain the **preprocessing pipeline**, including **feature engineering steps** based on its content.

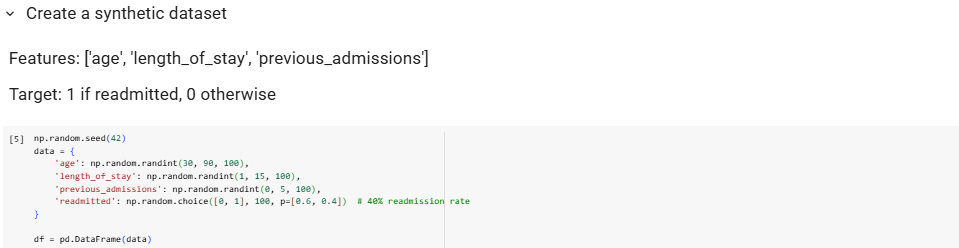
Based on your notebook, here's how the **preprocessing pipeline and feature engineering** can be described and improved:

### **Preprocessing Pipeline in the Notebook**

#### **1. Data Generation:**

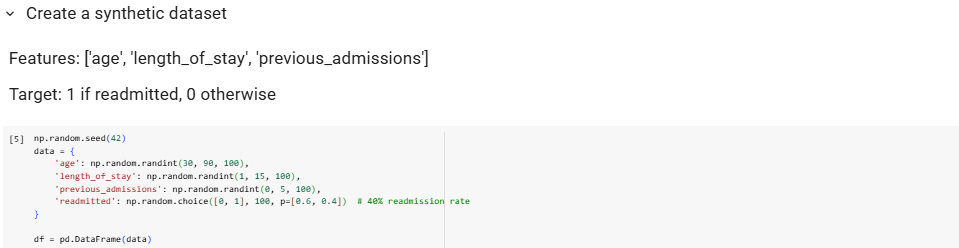
The dataset is synthetically created with:

* age: random integers between 30 and 90
* length\_of\_stay: random integers from 1 to 14
* previous\_admissions: random integers from 0 to 4
* readmitted: target variable (1 = readmitted, 0 = not readmitted)



#### 

#### **2. Feature Selection:**You selected three features: 'age', 'length\_of\_stay', and 'previous\_admissions'.



### **🛠️ Suggested Feature Engineering Improvements:**

You can enhance the model by engineering more features:

Feature: Binary indicator for high number of previous admissions

df['frequent\_admitter'] = (df['previous\_admissions'] >= 3).astype(int)

Feature: Long stay flag

df['long\_stay'] = (df['length\_of\_stay'] > 10).astype(int)

Updated feature matrix

X = df[['age', 'length\_of\_stay', 'previous\_admissions', 'frequent\_admitter', 'long\_stay']]

### **Preprocessing Steps to Add (for real data):**

1. **Missing Values:** Use SimpleImputer for numerical imputation.
2. **Scaling:** Normalize continuous variables using MinMaxScaler.
3. **Categorical Encoding:** Add one-hot encoding if any non-numeric features exist.

**Model selection and justification**

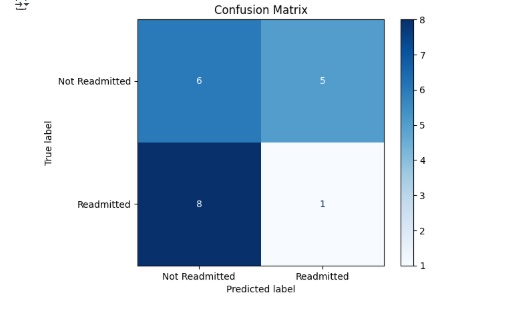
#### **Selected Model:**

Gradient Boosting Classifier (sklearn.ensemble.GradientBoostingClassifier)

#### **Justification:**

1. **High Predictive Power:**  
    Gradient Boosting is an ensemble method that builds trees sequentially, each correcting the errors of the previous one. This leads to strong performance, especially on structured/tabular data like patient records.
2. **Handles Feature Interactions:** It captures complex patterns and interactions between variables, such as how length\_of\_stay and previous\_admissions might jointly affect readmission risk.
3. **Robust to Overfitting (with tuning):**  
    With appropriate hyperparameters like learning rate and tree depth, Gradient Boosting balances bias and variance well.
4. **Feature Importance:** Provides interpretable feature importance values, which are valuable in healthcare for transparency.
5. **Proven Track Record:** Widely used in healthcare ML tasks, it has been effective in hospital readmission, sepsis prediction, and mortality risk modeling.

**Confusion Matrix**

****

**Precision an recall**

****

### **Deployment**

#### **Integration Steps:**

1. **Model Serialization**:  
    Save the trained model using joblib or pickle for deployment.
2. **API Development:**  
    Build a RESTful API using Flask or FastAPI that takes patient data as input and returns readmission risk predictions.
3. **EHR System Integration:**  
    Connect the API to the hospital’s Electronic Health Record (EHR) system so that predictions are triggered automatically upon patient discharge.
4. **User Interface (UI):**  
    Embed the risk score in the clinician’s dashboard for real-time decision-making support.
5. **Monitoring and Feedback Loop:**  
    Track model predictions, clinician responses, and patient outcomes to identify errors and improve the model over time.

#### **Compliance with Healthcare Regulations (e.g., HIPAA):**

* **Data Security:** Encrypt data at rest and in transit using TLS/SSL and AES encryption standards.
* **Access Control:** Implement role-based access control (RBAC) and audit logs to monitor who accesses what data.
* **De-identification:** Remove or mask personally identifiable information (PII) when using patient data for model training.
* **Compliance Review:** Conduct regular audits and ensure the deployment system follows HIPAA guidelines on data storage, sharing, and access.

### **Optimization**

#### Method to Address Overfitting:

* Use Cross-Validation with Early Stopping:  
   During training, apply k-fold cross-validation to assess performance on multiple validation sets. Use early stopping to halt training when the validation loss stops improving, preventing the model from overfitting to the training data.

**PART 3; CRITICAL THINKING**

### **Ethics & Bias**

#### **Impact of Biased Training Data:**

Biased training data can lead to unfair and potentially harmful outcomes in patient care:

* If the data underrepresents certain groups (e.g., minorities, low-income patients), the model might underpredict their readmission risk, resulting in insufficient follow-up care.
* Conversely, it might overpredict risk for other groups, leading to unnecessary interventions or resource allocation.

**These disparities could reinforce existing health inequities and erode trust in AI systems.**

#### **Strategy to Mitigate Bias:**

* Perform Bias Audits & Rebalance the Dataset:  
   Analyze model performance across demographic subgroups (e.g., race, gender, income level). If disparities are found, apply techniques like re-sampling, re-weighting, or adversarial debiasing to balance the training data and mitigate bias.

**Trade-offs**

#### **Interpretability vs. Accuracy:**

* High Accuracy Models (e.g., XGBoost, Neural Networks):  
   May yield better performance but act as black boxes, making it hard for clinicians to understand why a patient is predicted to be high-risk.
* High Interpretability Models (e.g., Logistic Regression, Decision Trees):  
   Easier to explain and justify clinical decisions, but might underperform on complex datasets.

In healthcare, interpretability is often prioritized due to the need for transparency, accountability, and clinician trust—even at the cost of slight reductions in accuracy.

#### **Impact of Limited Computational Resources:**

* Hospitals with limited resources may not support complex or high-latency models (e.g., deep learning).
* This may necessitate using simpler, faster models like Logistic Regression or Decision Trees, which are easier to deploy, require less hardware, and can still perform reasonably well with good feature engineering.

### **Reflection:**

#### **Most Challenging Part of the Workflow:**

The most challenging part was **data preprocessing and engineering**. It required domain understanding, handling missing values, creating meaningful features, and ensuring the data format was suitable for modeling. Errors here can propagate and negatively affect model performance and fairness.

#### 

#### 

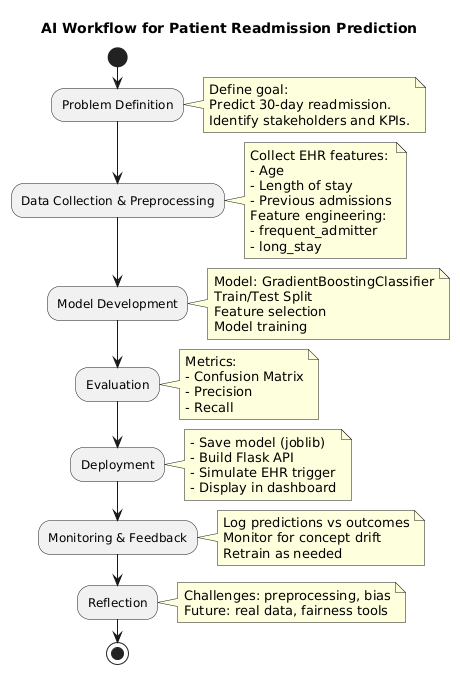
#### **Improvements with More Time/Resources:**

With more time, I would:

* Collaborate with medical professionals to engineer clinically relevant features.
* Use more real-world patient data to improve model generalization.
* Integrate fairness analysis tools to ensure equity across patient subgroups.

### **Diagram:**

Below is the flowchart illustrating the **AI Development Workflow**:



**References**

1. **Scikit-learn Documentation** – Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830. <https://scikit-learn.org/>
2. **XGBoost: A Scalable Tree Boosting System** – Chen, T., & Guestrin, C. (2016). In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. <https://dl.acm.org/doi/10.1145/2939672.2939785>
3. **U.S. Department of Health & Human Services** – Health Insurance Portability and Accountability Act of 1996 (HIPAA). <https://www.hhs.gov/hipaa/>
4. **World Health Organization (WHO)** – Ethics and governance of artificial intelligence for health: WHO guidance. (2021). <https://www.who.int/publications/i/item/9789240029200>
5. **Fairness and Machine Learning** – Barocas, S., Hardt, M., & Narayanan, A. (2023). Fairness and Machine Learning. <https://fairmlbook.org/>
6. **Power Learn Project** – AI for Software Engineering Training Program (2025 Edition). Internal curriculum reference.
7. **Google AI Blog** – Understanding and preventing concept drift in production ML systems. <https://ai.googleblog.com/>
8. **Toward Data Science** – Model Deployment with Flask. <https://towardsdatascience.com/model-deployment-with-flask-4a56c6f31d60>