# Task 2 Case Study:

**Build an end-to-end AI solution for hospitals to reduce patient readmission within 30 days of discharge.**

## 1. Problem Scope/ Definition

Hospital readmissions within 30 days of discharge represent a significant challenge in the whole world. They often indicate premature discharge, inadequate post-discharge care, or complications not initially detected. These readmissions lead to increased healthcare costs, patient dissatisfaction, and strain on medical resources. African healthcare systems, for example, in South Africa, where public hospitals face high bed occupancy rates (e.g., ~85% in major facilities like Chris Hani Baragwanath Hospital) and limited resources. These readmissions strain hospital capacity, exacerbate staff shortages, and increase patient mortality risks, especially in rural areas with limited access to follow-up care.

It is in light with this background that this project focuses on developing an Artificial Intelligence AI-driven predictive model that can identify patients at high risk of readmission within 30 days, allowing healthcare providers to intervene proactively. This system ultimately helps save billions of dollars, reduce mortality rates, increase patient satisfaction. Above all, the system can help governments and hospital administrators and Clinicians to strengthening tele-medication and providing continuous support to those at high risk of readmission.

**Objectives**:

a. **Predict Readmission Risk**: Accurately identify high-risk patients with at least 80% recall within 30 days using clinical and demographic data at the time of discharge to ensure effective follow-up

b. **Reduce operational costs:** Minimize unnecessary readmissions to improve patient outcomes and reduce associated costs for hospitals and insurance systems.

c. **Support Clinical Decision-Making**: Provide hospital staff with risk scores and interpretable predictions to inform discharge planning and follow-up strategies.

**Stakeholders**

This project has a huge span of stakeholders. Starting from Government, Provincial down to the hospital level. However, listing all stakeholders is beyond the scope of this project, but the most critical stakeholders were identified as below:

**Healthcare Providers (Doctors, Nurses, Case Managers)**

They will use the model's outputs to inform discharge decisions, follow-up care plans, and patient education.

**Hospital Administrators:**

Responsible for improving care quality, operational efficiency, and meeting readmission reduction targets set by policy or insurance providers.

## 2. Data Sources

In terms of data sources, in order to build a relevant model, two data sources were used as outlined below:

Primary dataset

**Kaggle Hospital Readmission dataset**: The data has rich features set for model development. It consists of demographics, comorbidities, discharge data and outcomes.

However, this data is from US and as such, might not be fully applicable to African context. Therefore, I decided to use the Tygerberg Hospital study to help simulate certain features into the original dataset. These features will make the model more applicable and generalizable in African context. Therefore, below are the features simulated into the dataset:

**Local Context Simulation**

- Drawn from Tygerberg Hospital study (2014–2015): 11,826 admissions with 10.5% readmissions (Dreyer & Viljoen (2019).
- Adapted features: Charlson comorbidity index(CCI)[i], common African conditions (HIV, TB).
- Simulate hospital type and resource-access flags for regional relevance.

**Ethical Concerns**

**Patient Privacy & Data Protection**: Training requires strict anonymization and POPIA compliance; encryption and informed consent are necessary.

**Fairness & Representation**: Ensure model doesn't underpredict high-risk patients from rural or low-resource clinics by bias audits and potential oversampling of minority subgroups.

**Feature engineering:** Feature engineering conducted during the project also can have its own bad side. For instance, the simulated data might not be up to date or fail to mimic reality 100%. Leading to a development of a model that produces low levels of accuracy and recall.

## 3. **Model Development -** model selection, evaluation set up and metrics

Based on the problem to be solved, it is clear it is a complex classification problem. Many or several factors usually lead to the classification into one class or the other. As such, the Random Forest Classifier was chosen as our predictive model.

**Why Random Forest Classifier:**

Structured Clinical Data Compatibility: Random Forests handle tabular data with mixed types (numerical + categorical), which suits Electronic Health Records (EHRs).

Robustness and Accuracy: They are resistant to overfitting, especially with cross-validation, and perform well on healthcare problems with many features.

Interpretability: Feature importance scores help clinicians understand which factors (e.g., comorbidity score, age, HIV status) most influence predictions. This is important to build trust and ethical accountability.

Scalability: Random Forests are well known to train relatively fast and can be scaled across medium to large datasets that are well common in hospital systems.

**Train-Test Split**

Rather than well-known 80/20% split of the train test data, in this project, I wanted to also introduce cross validation. As such 15% of the data will be set aside as validation set. In order to build a good model, stratified splitting will also be used to maintain similar proportion of readmissions across all sets. As a result, this is how the data was split. The dataset will be split as follows:

70% Training Set: this is the set used to fit the model.

15% Validation Set: this set is used for hyperparameter tuning.

15% Test Set:   this set is used to evaluate how the model performs on unseen data.

Below is a code snippet of how the splitting will be performed:

```
from sklearn. model_selection import train_test_split

X = df.drop("readmitted", axis=1)

y = df["readmitted"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y, random_state=42)

X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, stratify=y_temp, random_state=42)
```

**Confusion Matrix**

Confusion matrix is a model evaluation strategy to deep dive into models ability to truly identify True Positives(TP), True Negatives(TN) and also check how confused the model was in falsely identifying False Positive(FP) and False Negatives(FN). This directly speaks to model precision and recall.

| | **Predicted: Yes** | **Predicted: No** |
|---|---|---|
| **Actual: Yes** | **TP = 20** | **FN = 10** |
| **Actual: No** | **FP = 5** | **TN = 65** |

**Precision and Recall**

Precision = TP / (TP+FP)

=20 / (20+5)

=20/25 =0.80

*i.e. when we predict readmission, we're right 80% of the time*.

**High precision** means we avoid falsely alarming patients who wouldn't be readmitted.

**Recall**

Recall = TP /(TP+FN)

20 / (20+30) = 0.67

*i.e. in the above hypothetical data, it means the model managed too catch 67% of actual admissions. Why in my own, would not be good performance!*

**Moderate recall** means there's room to improve on catching more at-risk patients, perhaps by adjusting thresholds or using ensemble models.

**4. Model Deployment**

Deployment in this case means sending the well trained and evaluated as well as optimized model into production. This means integrating the model into the real Hospital environment or system to start assisting the administrators and Doctors by predicting and flagging patients at high risk of readmission. In order  for this to take place, the following steps should be taken:

Step 1: Export Trained Model

The fully trained model is exported from the notebook or python file as a *.pkl* file or a *.joblib* file. This process rebuilds the model in its entirety from different parts and make it ready to be deployed in production environment.

Step 2: Build a REST API Interface

This stage involves building a lightweight Python based REST API (Representational State Transfer) using Flask or FastAPI Python frameworks. This allows the Hospital systems (Electronic Health Records software) to send and communicate patient data to the model and receive prediction results).

Step 3: Embed into Discharge Workflow

The final proposal is to  deploying the model via a RESTful API using Flask. When a patient is discharged, their relevant features (e.g., age, comorbidities, length of stay) are sent to the API. The model returns a prediction indicating whether they are at high risk of being readmitted within 30 days. This prediction can be used to trigger follow-up protocols such as early outpatient visits or home care.

The model could also be integrated into the hospital's Electronic Health Record system to enable real-time risk scoring.

Step 4: Monitoring & Maintenance

Once deployed, model performance should continuously be monitored  using metrics such as precision, recall and false alarm rates. This assists in checking if the model overtime remains effective or if there is any concept drift. A dashboard should also be used to display the following Weekly Predictions and Number of High risks predictions.

In order to comply with international laws like HIPAA, the following steps were also taken into consideration:

Data Anonymization - during training, all Personally Identifying Information (PII) should be deducted.

Use secure Infrastructure – the model and patient data must be stored in an encrypted database with limited control through role-based access control.

Strengthen explainability and Clinical trust

Model predictions should be interpretable and make sense through SHAP or feature importance. This helps to boost Clinicians' trust in the model and give them clear justification on why that prediction was made than the other.

**Optimization**

To address potential overfitting, one method that has already been mentioned is using cross validation during model development. During training, a 5-fold cross-validation was applied to the model. This ensures the model generalizes well to unseen patient cases and avoids overly optimistic performance on the training set.

In a nutshell, using cross-validation ensures model generalizes well to diverse patients and prevent bias toward overrepresented conditions.

Real Optimization

During running the validation set *(y_val_pred = rf_model.predict(X_val), I realized the model was not doing well.* Many Fasle Negatives! In fact, I was using a seriously unbalanced dataset for training. From 30k patients, only 10% were readmitted. This means that the model would even try to guess out Negatives and get away with it. This is the reason why using confusion matrix was the best. First, the model had the below metrics

```
Confusion Matrix:

[[3320  629

[415 236]]

Precision Score: 0.18

Recall Score: 0.25
```

This performance was not good for such lifesaving model. I performed further parameter tuning to optimize which included introducing GriSearchCv which automates **hyperparameter tuning**, increasing n_estiamtors.

Even after finding the best GriSearchCv parameters, Recall score stayed at 28%. After looking at the dataset again and conformed how imbalanced data is the issue than the model. I then chose to implement SMOTE (Synthetic Minority Oversampling Technique to try and balance the data.

After training a Random Forest model on a SMOTE-balanced dataset, we achieved the following results on the test set:

- **Precision**: 0.12
- **Recall**: 0.30
- **F1 Score**: 0.17

While precision is low, recall improved significantly, which is crucial for a healthcare use case where identifying patients at risk of readmission is more important than avoiding false positives.

This model can serve as a **screening tool** to flag potentially high-risk patients for further review.

References:

Dreyer, R., & Viljoen, A. J. (2019). *Evaluation of factors and patterns influencing the 30-day readmission rate at a tertiary-level hospital in a resource-constrained setting in Cape Town, South Africa*. South African Medical Journal, 109(3). Retrieved from [AJOL] doaj.org+8

[i] The **Charlson Comorbidity Index** is a clinical scoring system that predicts a patient's risk of death or hospital readmission by assigning weights to **17 comorbid (chronic) conditions**.