Case Study Application: Predicting Patient Readmission

1. Problem Scope (5 points)

Problem Definition:

Predict whether a patient will be readmitted within 30 days of discharge.

Objectives:

- Reduce hospital readmission rates.
- Improve patient outcomes.
- Reduce operational costs.

Stakeholders:

- Hospital Administration
- Medical Staff (Doctors, Nurses)
- Patients
- Data Science and IT Team
- Regulatory Bodies

2. Data Strategy (10 points)

A. Proposed Data Sources:

- Electronic Health Records (EHR): patient history, diagnosis, lab results, vital signs.
- Demographics: age, gender, ethnicity, income.
- Hospital Utilization: prior admissions, discharge dates, length of stay.
- Social Determinants: insurance, zip code, family support.

B. Ethical Concerns:

1. Patient Privacy: Ensure data anonymization and encryption.
2. Bias and Fairness: Prevent biased training by monitoring for demographic skews.

C. Preprocessing Pipeline:

- Handle missing values with imputation.
- Scale numeric values.
- One-hot encode categorical features.

- Split the dataset into training and testing sets.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer

df = pd.read_csv("patient_data.csv")
numeric_features = ['age', 'length_of_stay', 'lab_result1', 'lab_result2']
categorical_features = ['gender', 'ethnicity', 'diagnosis_code']

numeric_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

categorical_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer([
    ('num', numeric_pipeline, numeric_features),
    ('cat', categorical_pipeline, categorical_features)
])
```

3. Model Development (10 points)

A. Model Chosen: Random Forest Classifier

- Justification: Handles mixed data types, robust to overfitting, interpretable.

B. Evaluation Metrics:

We use confusion matrix, precision, and recall.

Hypothetical output shows good balance between false positives and false negatives.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, precision_score, recall_score

X = df.drop("readmitted", axis=1)
y = df["readmitted"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

```
from sklearn.pipeline import make_pipeline
model = make_pipeline(preprocessor, RandomForestClassifier(n_estimators=100, random_state=42))
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("Confusion Matrix:", cm)
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
```

4. Deployment (10 points)

A. Integration Steps:

1. Serialize the model with joblib or pickle.

2. Deploy the model using Flask/FastAPI as a REST API.

3. Connect API to hospital systems like EHR via HL7/FHIR.

4. Add monitoring and retraining logic.

B. HIPAA Compliance:

- Anonymize and encrypt data.

- Use secure APIs with authentication.

- Conduct privacy assessments and keep audit logs.

5. Optimization (5 points)

Method to prevent overfitting:

Use cross-validation and hyperparameter tuning (GridSearchCV) to select optimal parameters and avoid overfitting.

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'randomforestclassifier__n_estimators': [100, 200],
    'randomforestclassifier__max_depth': [5, 10, None]
}

grid_search = GridSearchCV(model, param_grid, cv=5, scoring='recall')
```

```
grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)
```