End-to-End Machine Learning: Diabetes Readmission Prediction

A workflow analysis utilizing Databricks, Scikit-Learn, and MLflow.

GOAL

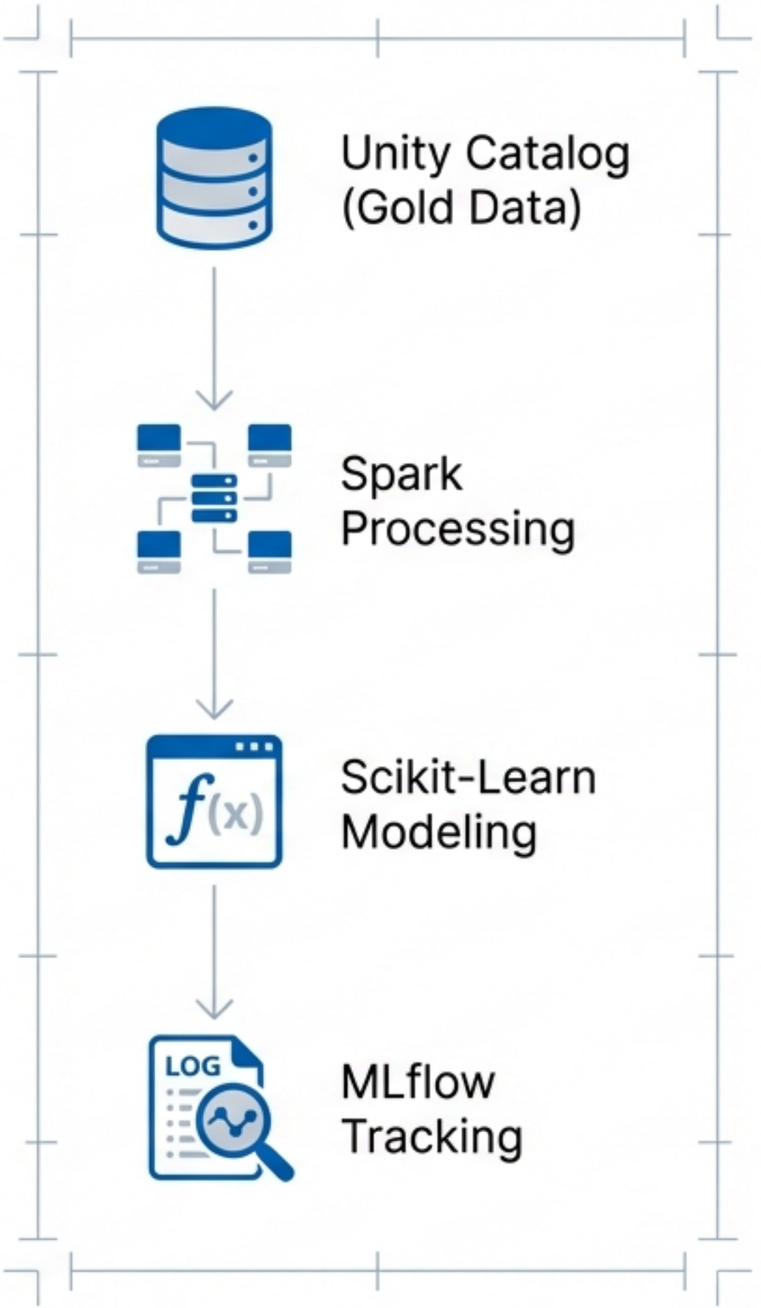
Predict 30-day hospital readmissions

STACK

PySpark, Pandas, Scikit-Learn, MLflow

ENV

Databricks Runtime 15.x / Python 3.12



From Raw Records to Risk Prediction

The Objective

The Challenge

The healthcare system needs to identify diabetes patients at risk of returning to the hospital within 30 days. We utilize a binary classification approach where **y = readmitted_30**. The goal is to intervene early and improve patient outcomes.

```
# Binary Classification Target
target_variable = 'readmitted_30'
# Goal: Early Intervention
objective = 'Improve Patient Outcomes'
```

The Solution & Result

Executive Summary

Key Stats

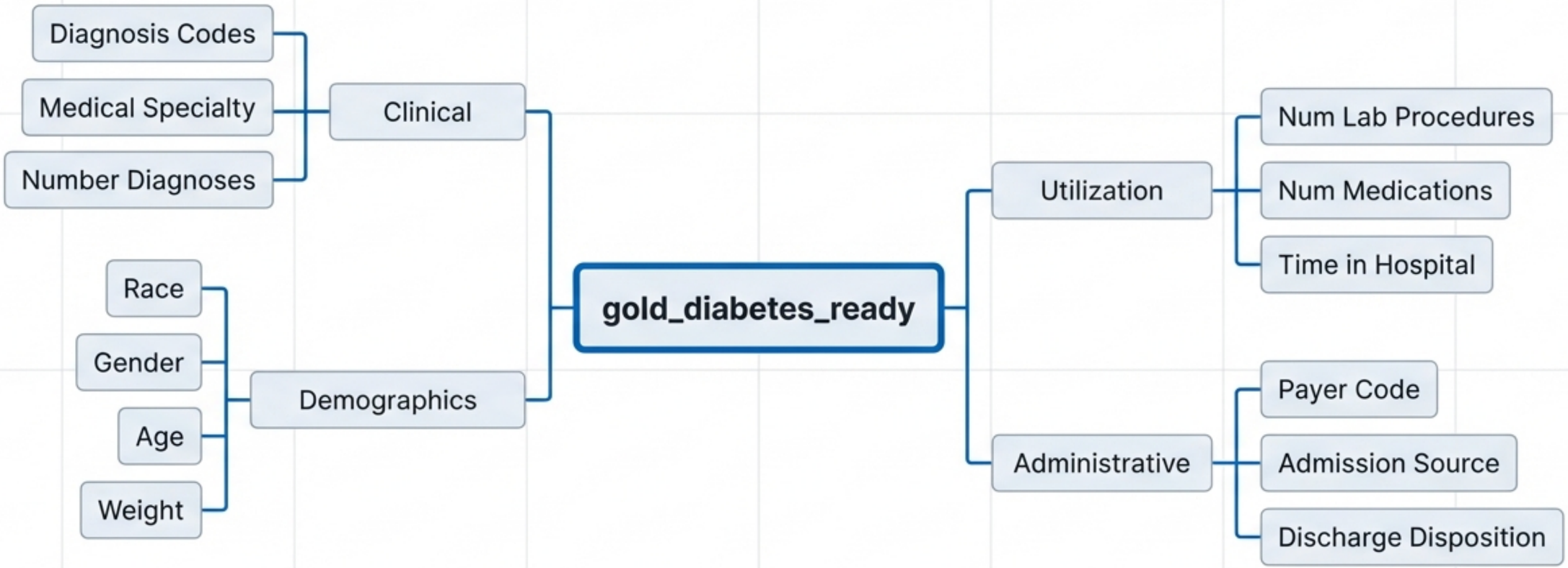
Data Source **Gold-Certified Unity Catalog**	Model Type **Logistic Regression Baseline**
Primary Outcome (AUC) **0.641**	Top Predictor **Prior Inpatient Visits** Coefficient: 0.317

Sourcing Validated Data from the Gold Layer

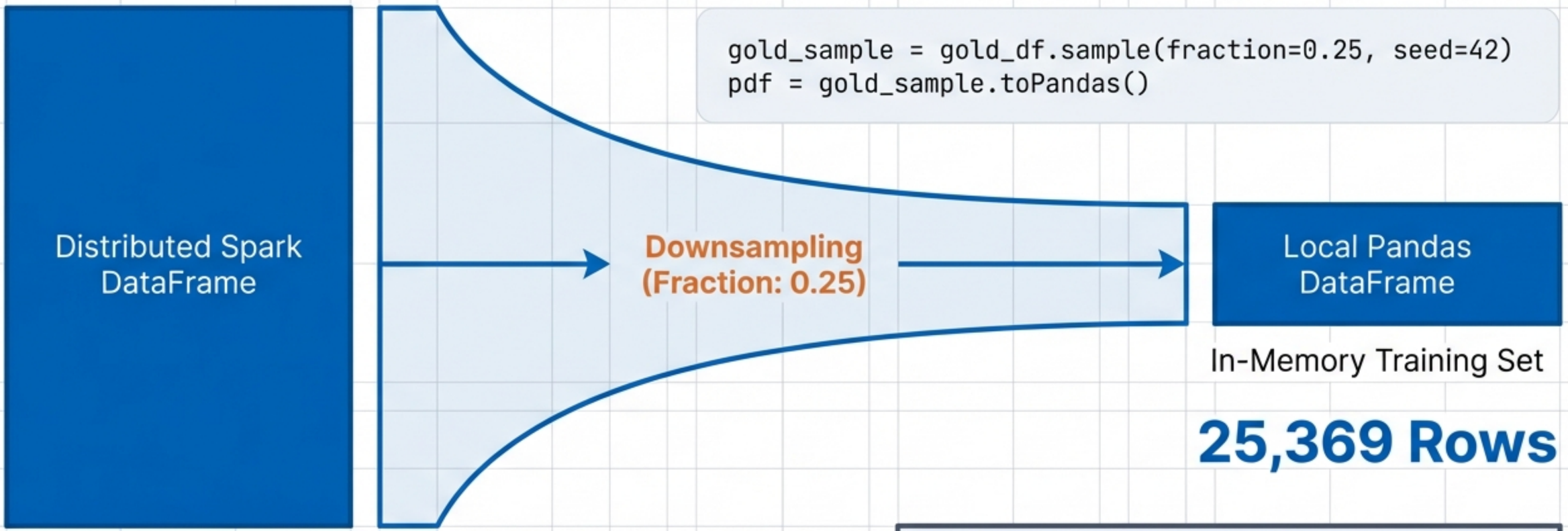
The workflow begins by ingesting a curated dataset, ensuring downstream models are built on reliable ground truth.

Ingestion Source

```
spark.table('diabetes_readmissions.gold_diabetes_ready')
```



Strategic Sampling for Efficient Prototyping




```
gold_sample = gold_df.sample(fraction=0.25, seed=42)
pdf = gold_sample.toPandas()
```

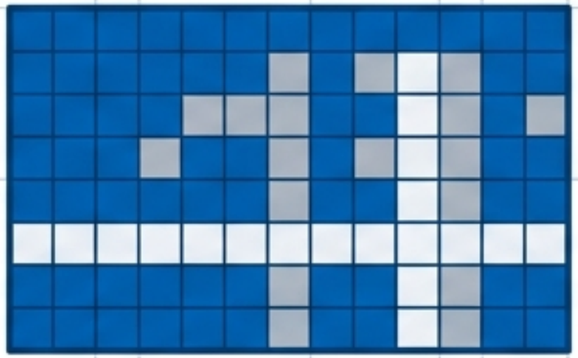
Moving full-scale big data into a local Scikit-Learn context requires memory management. We apply a fixed seed for reproducibility.

Isolating the Target Variable

The dataset is separated into features and a target variable for machine learning.

Local Pandas DataFrame

Feature Selection



Matrix X (Features)

Patient, hospital, and treatment info.

(25369, 47)

```
X = pdf.drop(columns=['readmitted_30'])
```

Vector y (Target)

Binary Flag (Readmitted 30 Days).

(25369,)

```
y = pdf['readmitted_30']
```

Encoding Categorical Attributes

Automated Label Encoding

Machine learning models require numerical input. The pipeline iterates through all “object” type columns to transform text into integers.

```
for col in X.columns:
    if X[col].dtype == 'object':
        X[col] = LabelEncoder().fit_transform(X[col])
```

Before and After

Race	Gender	Specialty
Caucasian	Female	Pediatrics
AfricanAmerican	Male	Cardiology

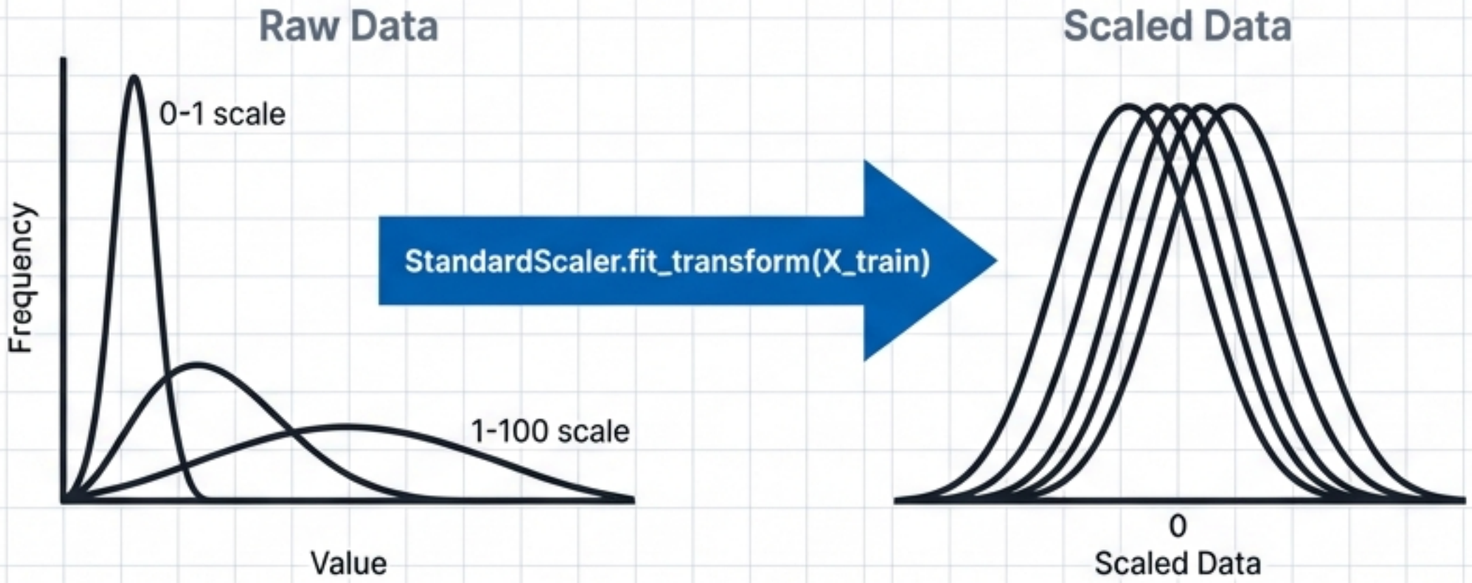
Race	Gender	Specialty
2	0	14
0	1	4

Standardization and Train-Test Splitting

Inter = train_test_split(random_state=42)

Feature Scaling Strategy

Logistic Regression uses distance-based optimization. We apply StandardScaler to normalize feature ranges.



Crucial Note

To prevent data leakage, the scaler is fitted ONLY on the training set, then applied to the test set.

```
# Scale features
# Scale features using StandardScaler
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Training the Logistic Regression Baseline

```
lr_scaled = LogisticRegression(
    max_iter=1000,
    solver='lbfgs'
)
lr_scaled.fit(X_train_scaled, y_train)
```

Increased Iterations: Guarantees convergence for high-dimensional data (47 features).

Solver: Limited-memory BFGS, optimized for smaller datasets in memory.

Input: Scaled Training Matrix.

Experiment Tracking with MLflow

Ensuring reproducibility and version control for model artifacts.

Run: Logistic_Regression_Baseline

Parameters & Metrics

Experiment Path:
/Shared/diabetes_readmission_lr

Metric Logged: AUC
0.641453647480148

Artifacts

model
(sklearn)

MLmodel
File

conda.yaml

Signature

Input Example Logged: Yes

Schema Enforcement: Active

First 5 rows of X_train logged for validation.

```
# STEP 9: Track with MLflow
# Log the model and metrics to MLflow for experiment tracking.
import mlflow
import mlflow.sklearn

mlflow.set_experiment("/Shared/diabetes_readmission_lr")

with mlflow.start_run(run_name="Logistic_Regression_Baseline"):
    mlflow.log_metric("AUC", auc)
    mlflow.sklearn.log_model(
        lr_scaled,
        "logistic_regression_model",
        input_example=X_train.iloc[:5]
    )
    print("MLflow run logged successfully")
```

Swiss Clinical Engineering

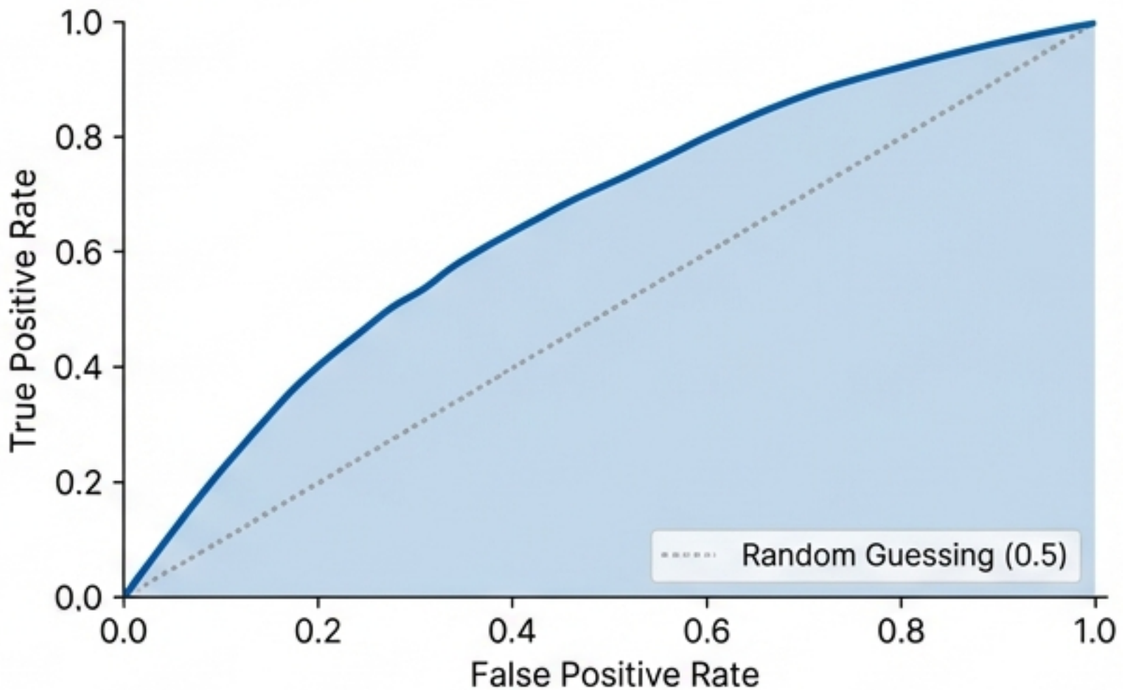
Evaluation: Establishing a Baseline

0.641

ROC AUC Score

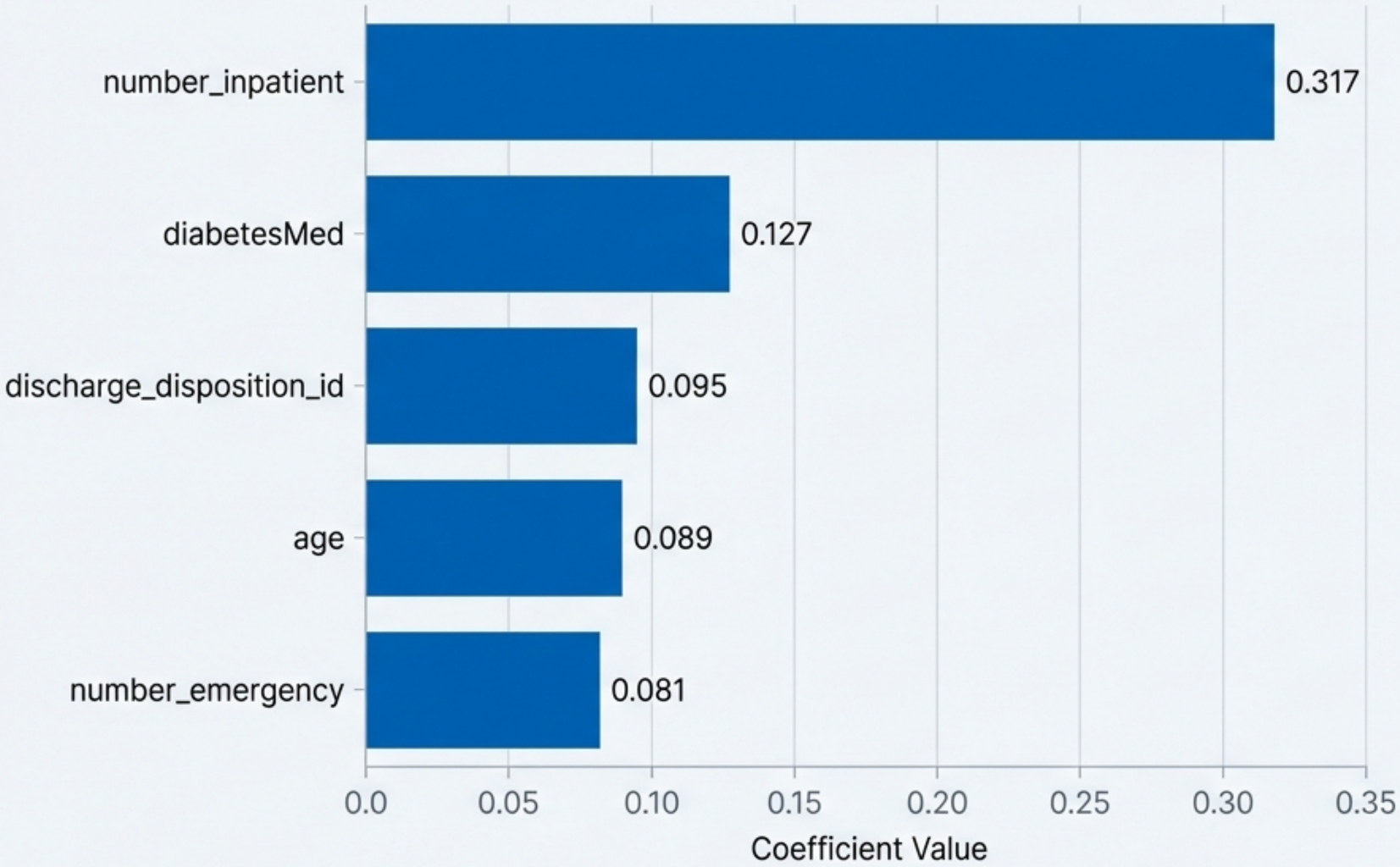
Evaluated on Test Set (N= ~5,074)

```python
# STEP 8: Evaluate model
# Evaluate the trained model using ROC AUC score on
the test set.
from sklearn.metrics import roc_auc_score

y_prob = lr_scaled.predict_proba(X_test_scaled)[:, 1]
auc = roc_auc_score(y_test, y_prob)

print("AUC:", auc)
```



The model performs better than random guessing, validating that signal exists in the dataset. However, the score suggests non-linear relationships that a linear baseline misses.

Drivers of Readmission Risk



Clinical Interpretation

- **Utilization History:** Frequent past inpatient visits are the **single strongest predictor** of readmission.
- **Medication Complexity:** Patients dependent on diabetes medication show **higher risk profiles**.
- **Discharge Status:** Where a patient goes post-discharge (home vs. nursing facility) **significantly impacts** return rates.

Conclusions & Optimization Strategy

Project Status

Successfully implemented a tracked, reproducible pipeline from Gold data to an interpretable baseline interpretable baseline model. Established a performance floor of AUC 0.64.

Feature Refinement

Data strategy:
Investigate 'number_inpatient' correlation. Transition from Label Encoding to One-Hot Encoding for nominal variables like Race and Payer Code to remove artificial ordinality.

Next Steps

The current linear boundary is insufficient. Propose testing non-linear, tree-based ensembles:

- Random Forest Classifier
- XGBoost / LightGBM
- Hyperparameter tuning via Hyperopt