Group4

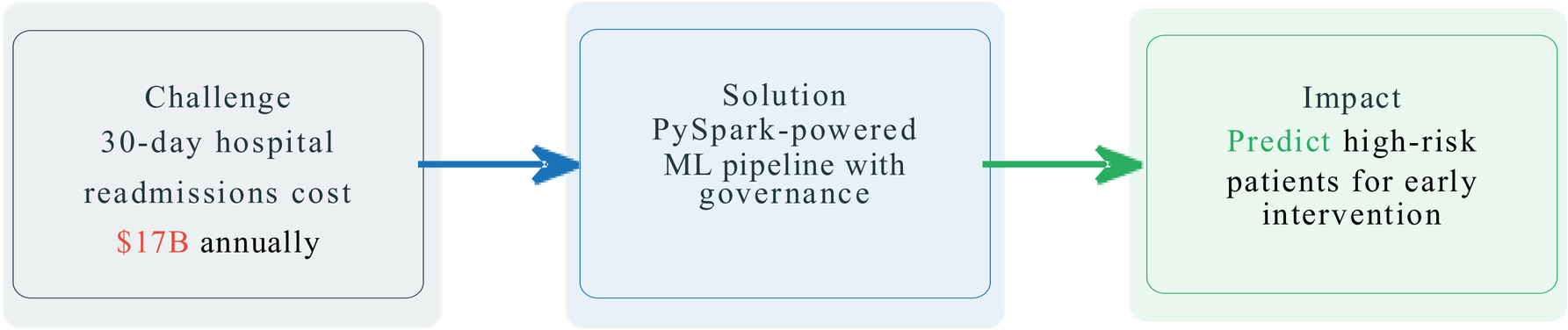
PREDICTING HOSPITAL READMISSION RISK FOR DIABETIC PATIENTS USING MACHINE LEARNING

Ayush Gurtu
Arindom Datta
Vishal Mishra
Ekjot Singh
Ji Guofang

INTRODUCTION

- 30-day hospital readmissions cost U.S. healthcare systems over $17 billion annually.
- Diabetic patients are particularly vulnerable to early readmission due to chronic care needs.
- Hospitals need data-driven solutions to identify high-risk patients before discharge.

Objective: Build a PySpark-powered scalable ML pipeline to predict readmission risk and enable proactive clinical interventions.



120

Monthly Snapshots

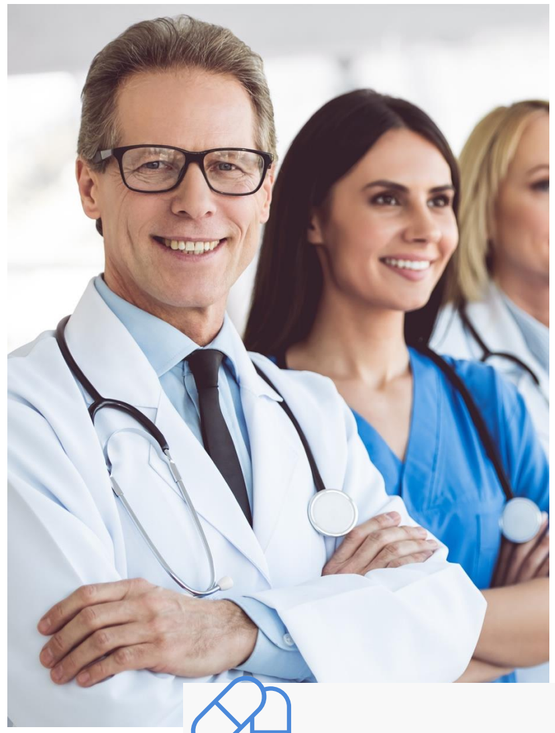
6

ML Models

Auto

Governance

ML FORMULATION



Target Variable (Y):
A binary variable indicating whether the patient was readmitted within 30 days:
Readmitted = 1
Not readmitted = 0

Problem Type:
This use case is a supervised binary classification problem.

Objective:

Predict the probability of hospital readmission within 30 days for diabetic patients.

Features (X) are derived from the following categories:

1. Patient Demographics

- Race (African American, Caucasian, Asian, Hispanic)
- Gender (binary: is_female)
- Age midpoint (e.g., [60–70) → 65)

2.Admission-Related Features

- Admission severity score (based on admission type: Emergency = 3, Urgent = 2, Elective = 1)
- Admission source risk score

3.Glycemic Control & Medication Features

- Poor glucose control indicator (based on max serum glucose and A1C results)
- Metformin usage (ordinal encoding)
- Insulin usage (ordinal encoding)
- Diabetes medication treatment status
- Medication density = medication intensity / diagnosis density

4.Diagnosis & Visit Features

- Primary diagnosis code (ordinal encoding by ICD categories: Circulatory, Respiratory, Diabetes, Digestive, Injury, Neoplasm, Musculoskeletal, Genitourinary, etc.)
- Secondary and tertiary diagnosis codes
- Severity × total visits

TECHNOLOGY STACK

Our comprehensive MLOps platform leverages a carefully curated technology stack designed for enterprise-grade machine learning operations. This architecture combines industry-leading frameworks for data processing, model training, and deployment orchestration, ensuring scalability, reliability, and performance at every layer.

Core Framework & Runtime

- Flask 3.0.0 web framework (UI)
- Python 3.12 runtime
- OpenJDK 21 for Spark execution
- Werkzeug 3.0.1 web server

Data Processing & Analytics

- Apache Spark (PySpark 3.5.5)
- Pandas 2.2.3 & NumPy 2.2.2
- Apache Parquet via PyArrow 14.0.1
- Medallion Architecture (Bronze-Silver-Gold)

Machine Learning

- scikit-learn 1.6.1 framework
- Logistics Regression, XGBoost, Random Forest
- GridSearchCV: Hyperparameter Tuning
- Pickle model serialization

Cloud & Storage Infrastructure

- Amazon S3 for scalable cloud storage
- boto3 1.34.0 AWS SDK integration
- Parquet format for efficient data storage
- Optimized for distributed processing

Orchestration & Workflow

- Apache Airflow for workflow management
- Amazon ECS (Fargate): Containerized task execution
- Automated pipeline scheduling
- Real-time job monitoring using CloudWatch

Development & Containerization

- Docker and Docker Compose enable consistent, reproducible environments across development and production.
- Amazon ECR: For Docker Image Registry.
- Terraform for Infrastructure as Code.

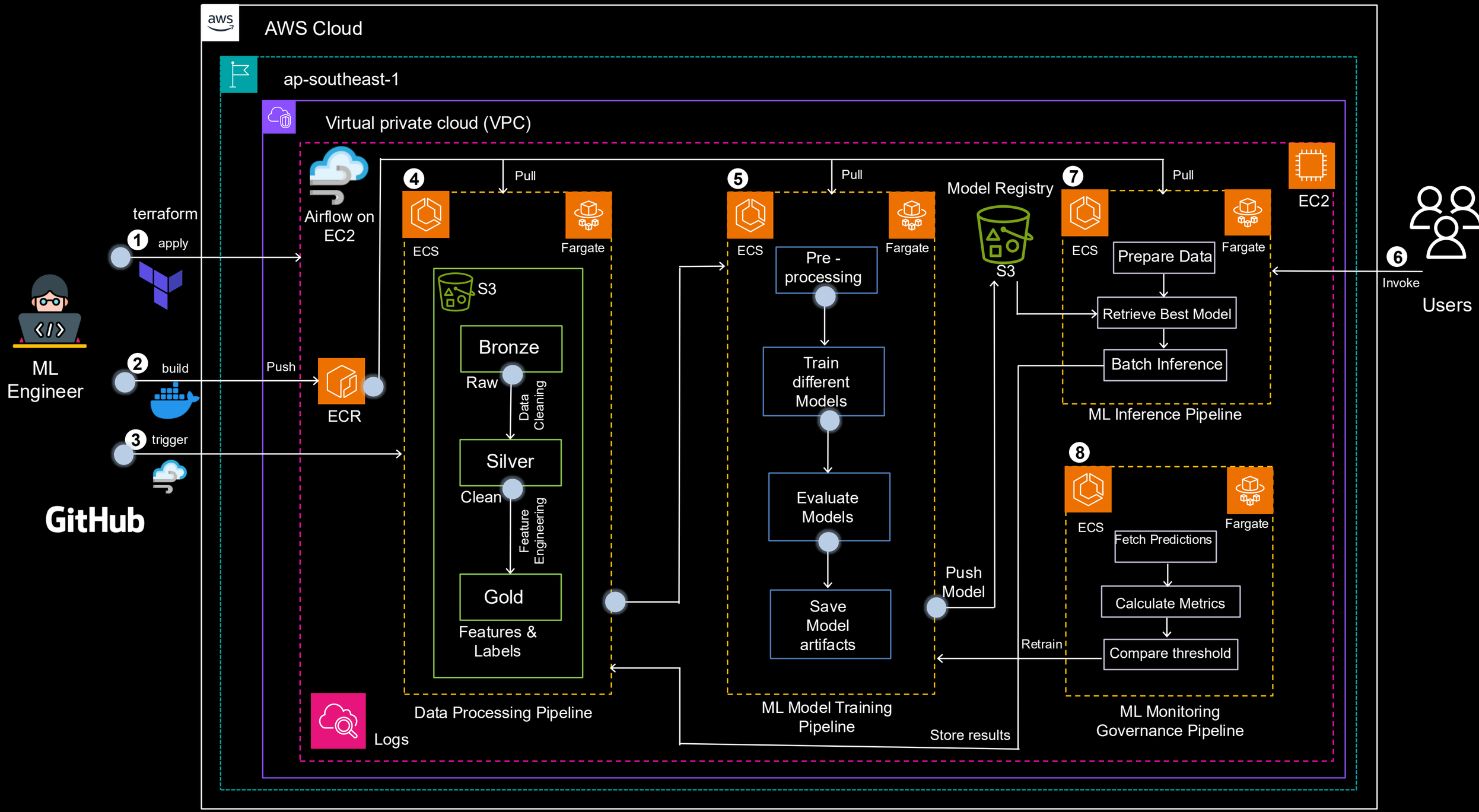
Model Inference & Serving

- Batch inference pipeline for scalable predictions.
- Automated Model Selection (Best Model from comparison).
- Manual Upload via UI: Adhoc predictions using Dag trigger.

Web Technologies

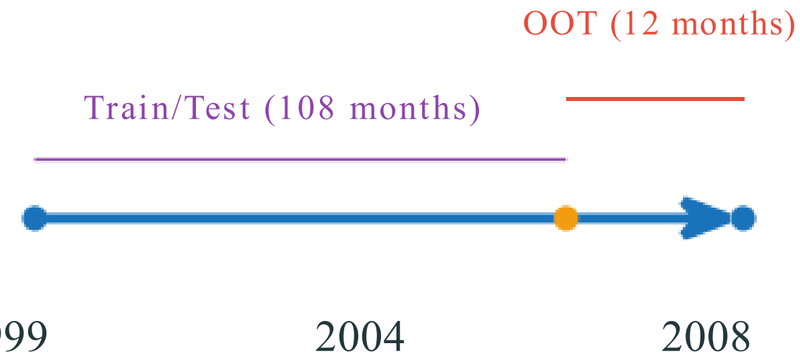
Flask REST API powers the backend services, while HTML5, CSS3, and JavaScript create responsive user interfaces. Flask-CORS 4.0.0 ensures secure cross-origin resource sharing for modern web applications.

ARCHITECTURE DIAGRAM


AWS Cloud
ap-southeast-1
Virtual private cloud (VPC)
terraform
1 apply
ML Engineer
2 build
Push
ECR
3 trigger
GitHub
Airflow on EC2
4
Pull
ECS
Fargate
S3
Bronze
Raw
Data Cleaning
Silver
Clean
Feature Engineering
Gold
Features & Labels
Data Processing Pipeline
Logs
5
Pull
ECS
Fargate
Pre - processing
Train different Models
Evaluate Models
Save Model artifacts
ML Model Training Pipeline
Model Registry
S3
Push Model
Store results
7
Pull
ECS
Fargate
Prepare Data
Retrieve Best Model
Batch Inference
ML Inference Pipeline
EC2
6
Invoke
Users
8
ECS
Fargate
Fetch Predictions
Calculate Metrics
Compare threshold
Retrain
ML Monitoring Governance Pipeline

DATASET AND PREPROCESSING

Dataset overview:



Total Snapshots 120 months
Features Severity, visits, meds
Target 30-day readmission

Validation Out-of-Time (OOT)

Metric	Value
Time Span	1999–2008 (120 months)
Total Records	101,766
Feature Dimensions	17 features
Monthly Records	784–868 (relatively stable)
Data Integrity	100% (labels and features fully matched)

Category	Number of Features	Example Features
Demographics	6	Race (4 binary variables), Gender, Age midpoint
Admission-Related	2	Admission severity score, Admission source risk score
Medical Treatment	5	Glucose control, Medication usage, Medication density
Diagnosis Features	3	Primary / Secondary / Tertiary diagnosis codes (ICD classification)
Derived Features	1	Severity × Total visits

DATA PROCESSING ARCHITECTURE AND FEATURE ENGINEERING

Raw Data (CSV)

↓

[Bronze Layer] Raw Storage

• 120 CSV files
• 101,766 raw records

↓

[Silver Layer] Data Cleaning

120 Parquet files (~70% compression)
• Data type conversion, missing value handling, validation

↓

[Gold Layer] Feature Engineering

• Label Store: 101,766 records (encounter_id, label, snapshot_date)
• Feature Store: 101,766 records (17 features + ID + date)

KEY FEATURE ENGINEERING

1. Categorical Feature Encoding:

Race: One-hot encoding (4 binary features)

Gender: Binary encoding (is_female)

Age Range → Midpoint Value (e.g., [60–70) → 65)

2.Business Rule Mapping:

Admission Type → Severity Score

(Emergency = 3, Urgent = 2, Elective = 1)

Diagnosis ICD Code → 11 Disease Categories

(e.g., Circulatory, Respiratory, Diabetes, etc.)

3.Derived Features:

medication_density = medication_intensity / (diagnosis_density + 1)
severity_x_visits = severity_score × total_visits
poor_glucose_control = abnormal_glucose_or_A1C_flag

MODEL TRAINING

Evaluation Objectives

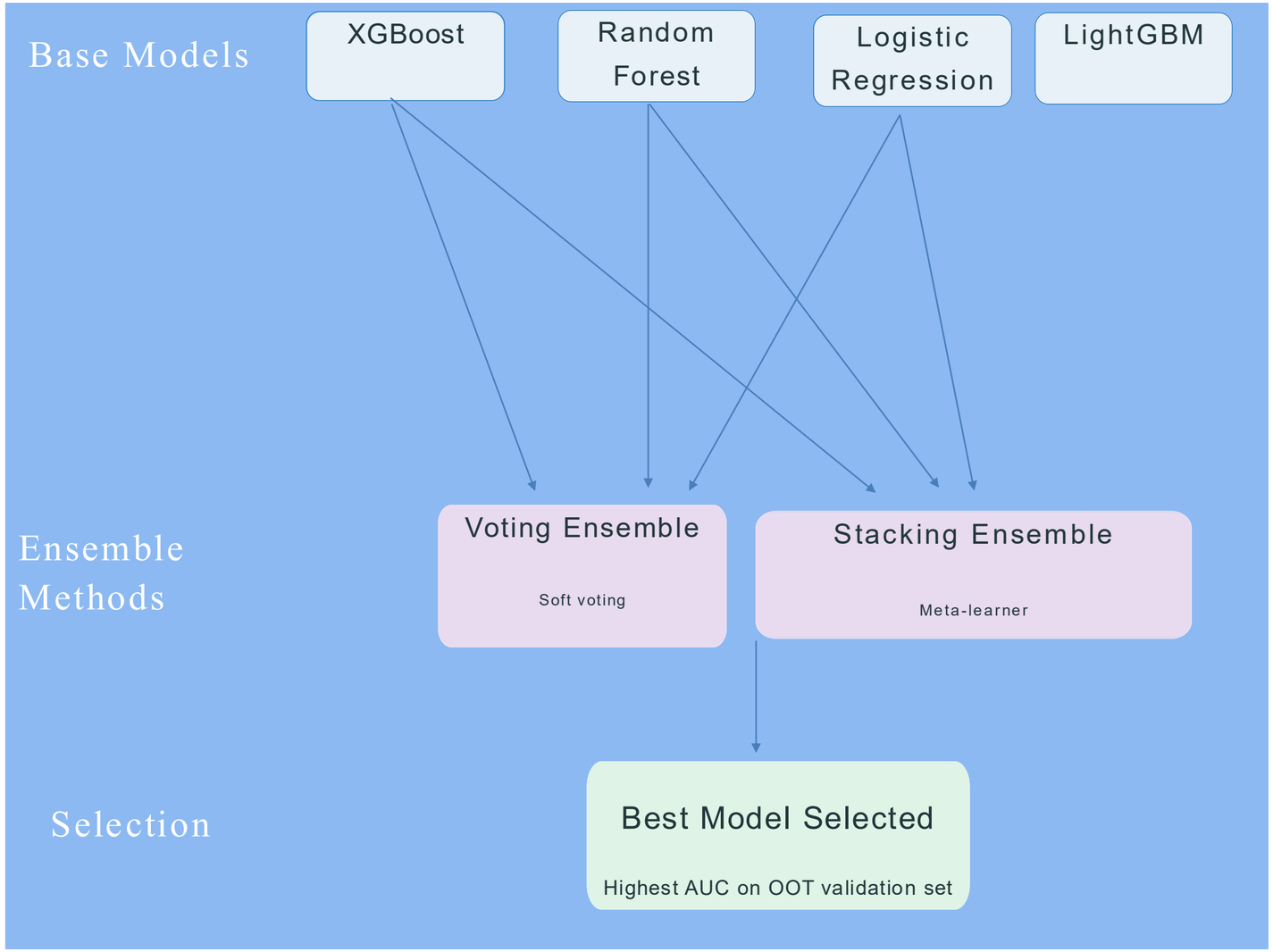
- Evaluate the performance of multiple machine learning models for diabetes prediction.
- Identify the best-performing model for deployment in the production environment.

Dataset Splitting (Time-Series Partitioning)

- Training / Testing Period: 1999-01-01 to 2007-12-31 (108 months)
  - Training Set: 73,507 records (80%)
  - Test Set: 18,377 records (20%)
- Out-of-Time (OOT) Validation Period: 2008-01-01 to 2008-12-31 (12 months)
  - OOT Set: 9,882 records
- Positive Sample Ratio: ~11% (balanced class distribution)

Evaluation Metrics

- AUC-ROC: Measures the model's discriminative ability.
- GINI Coefficient: Calculated as 2 × AUC - 1; widely used in business applications.
- Cross-Validation: 3-fold CV applied for hyperparameter optimization.

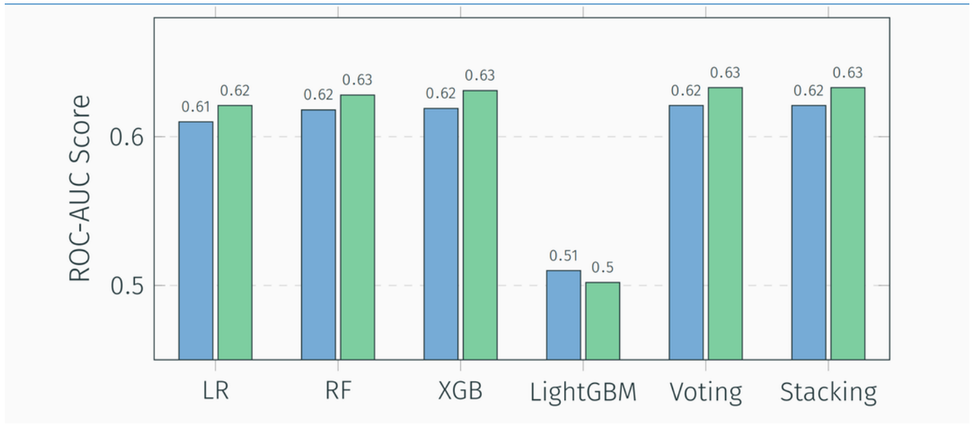


Train (80%) • Test (20%) • OOT (12 months) for temporal validation

MODEL PERFORMANCE EVALUATION RESULTS

```
================================================================================
MODEL COMPARISON
================================================================================
                Model  Train_AUC  Test_AUC  OOT_AUC  Train_GINI  Test_GINI  OOT_GINI
Logistic Regression   0.613081  0.609971 0.621208    0.226161   0.219941  0.242416
      Random Forest   0.634191  0.618444 0.627624    0.268381   0.236889  0.255248
            XGBoost   0.694146  0.618542 0.630988    0.388291   0.237084  0.261976
    Voting Ensemble   0.657657  0.621018 0.633399    0.315314   0.242037  0.266797
  Stacking Ensemble   0.675990  0.620924 0.633445    0.351979   0.241848  0.266891
           LightGBM   0.511981  0.509807 0.502073    0.023962   0.019613  0.004147

Best model by OOT GINI: Stacking Ensemble
OOT GINI: 0.2669
```



- Stacking Ensemble achieves best performance
- Consistent results across test and OOT validation

Key Findings

1. Best Model: Stacking Ensemble (OOT GINI = 0.2669)

2.XGBoost showed stable performance; the ensemble achieved slight improvement over individual models.

3.LightGBM performed poorly, indicating it may not be well-suited for this task.

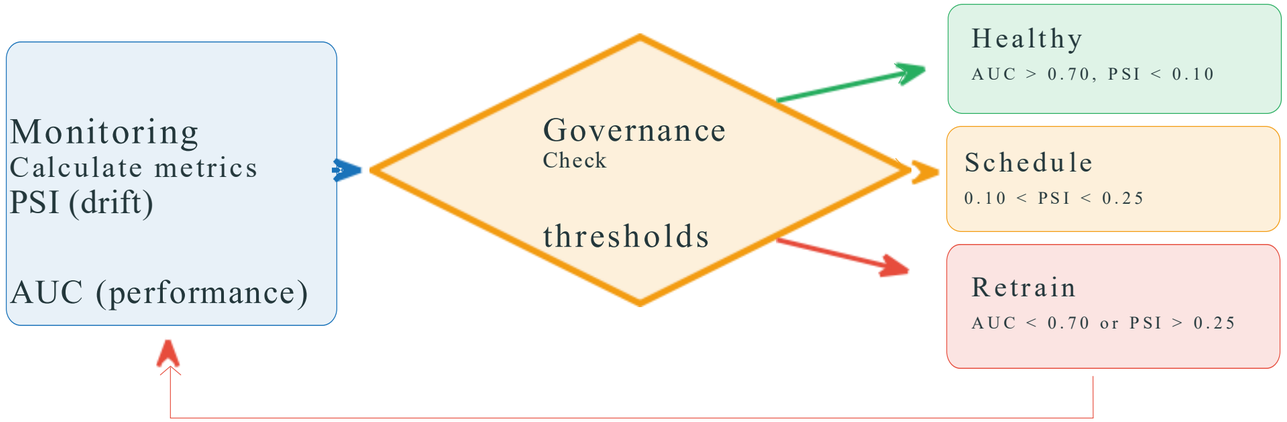
4.All models demonstrated stable performance on the OOT dataset, with no signs of overfitting.

Model Stability Analysis

- The differences in metrics between Train, Test, and OOT datasets are minimal, indicating good generalization capability.
- The Stacking Ensemble model demonstrates the most balanced performance across all three datasets.
- Recommendation: Deploy the Stacking Ensemble model as the production model.

MODEL MONITORING

Hospital Information System → **Feature Engineering Service** → **Stacking Ensemble Model API** → **Clinical Decision Support System (CDSS)**



Monitoring Type	Metric	Method	Alert Threshold
Model Performance	AUC / GINI	Monthly calculation	Drop > 5%
Prediction Drift	PSI (Population Stability Index)	Compare predicted distribution	PSI > 0.2 (moderate), > 0.5 (severe)
Feature Drift	CSI (Characteristic Stability Index)	Compare feature distributions	CSI > 0.2
System Performance	API response time, success rate	Real-time monitoring	Response > 100 ms or success rate < 99.5%

Real-Time Prediction:

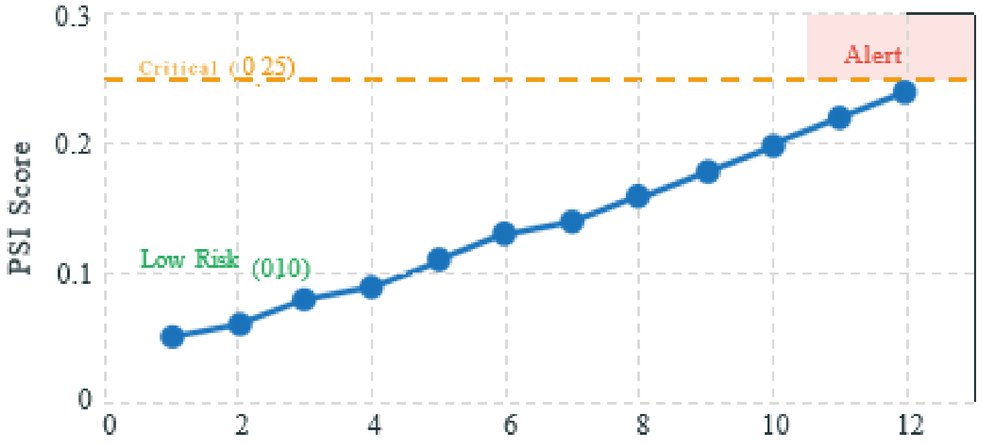
- Triggered at patient discharge.
- Response time < 100 ms per request.
- Supports RESTful API for seamless integration with hospital systems.

Batch Prediction:

- Executed daily or weekly for large-scale patient datasets.
- Used for trend analysis, reporting, and care planning.

Model Version Management:

- Supports A/B testing between model versions.
- Enables automatic rollback to the previous stable model if performance degradation is detected.



Prediction Month

Drift detected at month 11 → Auto governance triggered

Prevents model degradation before performance drops

CLINICAL IMPACT & BUSINESS VALUE

30%
Reduction
Readmissions

$5M
Savings
Annual cost

Early
Intervention
Care plans

Predict → Identify → Intervene → Follow-up → Measure

Proactive care • Better outcomes • Lower costs

MODEL INTEGRATION IN CLINICAL WORKFLOW

Patient Admission

During Hospitalization: Data Collection & Feature Computation

- Collect demographic information
- Record diagnosis codes
- Monitor glucose control
- Track medication usage

24 Hours Before Discharge: Model Prediction Triggered

- Automatically compute 17 features
- Call prediction API
- Generate risk score (0–1)
- Classify risk level

Risk Levels and Recommended Actions

High Risk (Probability ≥ 0.7)

- Extend hospitalization by 1–2 days
- Develop detailed discharge plan
- Schedule follow-up within 48 hours
- Provide 24/7 emergency contact hotline
- Assign case manager

Medium Risk (0.3 < Probability < 0.7)

- Standard discharge procedure
- Telephone follow-up within 1 week
- Provide educational materials

Low Risk (Probability ≤ 0.3)

- Routine discharge process
- Standard discharge guidance

CONCLUSION

Technical Achievements

Category	Key Achievements
Data Processing	• Implemented a three-layer data lake architecture (Bronze–Silver–Gold) • Processed 120 months of historical data • Successfully handled 101,766 complete records
Feature Engineering	• Developed 17 engineered features • Applied multiple encoding techniques (One-hot, Ordinal, Rule-based Mapping) • Created 5 derived features for clinical interpretation
Model Development	• Built Random Forest model (100 trees, max depth = 10) • Implemented via PySpark ML Pipeline • Conducted feature importance analysis for interpretability
System Architecture Design	• Designed a modular and scalable architecture • Supported incremental data processing • Enabled distributed computation for large-scale hospital data

Model Performance:

Stacking Ensemble OOT GINI: 0.2669 (Best)

Improved 0.0020 GINI compared to the single model (XGBoost)

Stable generalization with no overfitting

Key Technical Innovations:

- **Ensemble Learning Methods:**

Voting Ensemble: Simple and efficient

Stacking Ensemble: Achieved optimal performance (OOT GINI: 0.2669)

- **Strict Time-Series Partitioning:**

OOT set uses future 12-month data

Prevents data leakage and provides a realistic model evaluation

- **Model Monitoring System:**

PSI/CSI drift detection

Automated monitoring and alert mechanism

FUTURE IMPROVEMENT DIRECTIONS

Near Term (Q1-Q2)

- Real-time API — FastAPI endpoint
- SHAP Analysis — Model explainability
- Dashboard — Clinical decision support

Long Term (Q3-Q4)

- Deep Learning — Temporal patterns (LSTM)
- Spark Streaming — Real-time processing
- MLflow — Experiment tracking

THANK YOU