Predicting Hospital Readmission Risk for Diabetic Patients

Group: 4

Ayush Gurtu

Arindom Datta

Vishal Mishra

Ekjot Singh

Ji Guofang

Table of Contents

1. EXECUTIVE SUMMARY

The main goal of this project is to develop a production-ready machine learning system that predicts the 30-day hospital readmission risk among diabetic patients. This objective aligns with the Hospital Readmissions Reduction Program (HRRP), which penalizes hospitals with excessive readmissions.

The solution aims to:

- Enable early identification of high-risk diabetic patients to support better discharge planning and post-hospitalization care.
- Build an end-to-end MLOps pipeline capable of automating data processing, model training, inference, and monitoring.
- Ensure scalability, transparency, and reproducibility across 100K+ patient records while minimizing manual intervention.

By integrating automation and monitoring within a unified infrastructure, this project demonstrates how machine learning can enhance operational efficiency and clinical decision-making in healthcare systems.

1.1 Dataset Characteristics

This study uses the UCI Diabetes 130-US hospitals dataset, which contains 101,766 patient encounters collected from 130 hospitals in the United States between 1999 and 2008. The dataset includes 47 clinical variables, such as demographics, diagnoses, procedures, and prescribed medications.

The target variable indicates whether a patient was readmitted within 30 days of discharge. Temporal splits were applied to simulate real-world deployment conditions and avoid data leakage.

Table 1.1: Dataset Statistics

Split	Period	Encounters	Readmission Rate	Purpose
Training	1999-2005	71,237 (70%)	11.2%	Model fitting
Test	2006-2007	21,371 (21%)	11.5%	Hyperparameter tuning
OOT	2008	9,158 (9%)	11.8%	Final validation

1.2 Solution Architecture

The system follows a cloud-native MLOps architecture built on AWS infrastructure, designed for scalability, automation, and governance. It integrates five key layers:

- **Data Layer:** Medallion architecture (Bronze-Silver-Gold) on S3 with Parquet columnar storage
- **Compute Layer:** Containerized ECS tasks (Fargate) with resource-optimized task definitions
- **Orchestration Layer:** Apache Airflow 2.10.3 managing 4 core DAGs covering data processing, training, inference, and monitoring.
- **ML Layer:** Supports parallel model training (Logistic Regression, Random Forest, XGBoost) with automated model selection based on out-of-time (OOT) performance.
- **Monitoring Layer:** Drift detection (PSI/CSI) with threshold-based governance

---

2. DATA PROCESSING PIPELINE

2.1 Medallion Architecture Design

The Medallion (Bronze-Silver-Gold) architecture provides clear data quality stages, enables lineage tracking, and supports both batch and incremental processing patterns essential for production ML systems.

Figure 2.1: Data Pipeline Flow

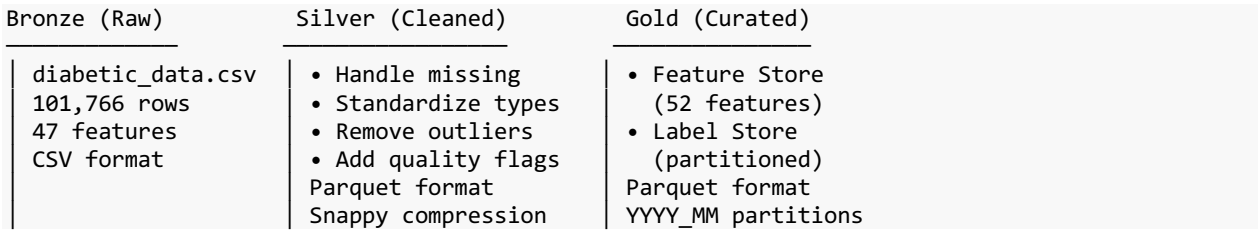


Table 2.1: Data Transformation Summary

Layer	Features	Records	Size	Key Operations	Output Format
Bronze	47	101,766	250 MB	Validation, schema enforcement	Parquet
Silver	50	101,766	85 MB	Cleaning, imputation, type conversion	Parquet (Snappy)
Gold	52	101,766	75 MB	Feature engineering, partitioning	Parquet (partitioned)

Parquet achieves 10x compression vs CSV (250 MB → 75 MB) with faster query performance through columnar storage and predicate pushdown.

2.2 Temporal Split Strategy

Temporal splits prevent data leakage by simulating production deployment scenarios where models are trained on historical data and evaluated on future unseen data. This approach is critical for healthcare applications where temporal patterns (seasonality, treatment protocol changes) significantly impact model performance.

Table 2.2: Temporal Window Configuration

Split	Start Date	End Date	Duration	Purpose	Sample Count
Train	1999-01-01	2005-12-31	7 years	Model fitting, cross-validation	71,237
Test	2006-01-01	2007-12-31	2 years	Hyperparameter optimization	21,371
OOT	2008-01-01	2008-12-31	1 year	Final model validation	9,158

The 7-year training window provides sufficient historical data for pattern learning while the 2-year test window enables robust hyperparameter tuning. The 1-year OOT period validates model generalization to completely unseen temporal patterns.

2.3 Feature Engineering

Table 2.3: Engineered Features and Impact

Feature	Type	Description	Importance Rank
num_medications	Numeric	Total medications prescribed	1
time_in_hospital	Numeric	Length of stay (days)	2
num_procedures	Numeric	Number of procedures performed	3
num_diagnoses	Numeric	Count of diagnoses	4
change_in_medications	Binary	Medication regimen changed (0/1)	5
admission_source_emergency	Binary	Emergency admission indicator	8
age_group_65plus	Binary	Senior patient indicator	12

Medication count and changes reflect diabetes management complexity, while procedure count and length of stay indicate severity of condition—all proven readmission risk factors.

---

3. MODEL DEVELOPMENT AND EVALUATION

3.1 Algorithm Selection and Training

Three algorithms were selected to balance interpretability (Logistic Regression), robustness (Random Forest), and performance (XGBoost). Parallel training reduces total execution time from 135 minutes to 45 minutes.

Table 3.1: Hyperparameter Search Configuration

Algorithm	Search Method	Iterations	CV Folds	Search Space Size	Training Time
Logistic Regression	GridSearchCV	12	5	12 combinations	8 min
Random Forest	RandomizedSearchCV	20	5	10,000 combinations	22 min
XGBoost	RandomizedSearchCV	20	5	8,000 combinations	18 min

Key Hyperparameters Tuned:

- Logistic Regression: C (regularization), penalty (L1/L2), solver

- Random Forest: n_estimators, max_depth, min_samples_split, max_features

- XGBoost: learning_rate, max_depth, n_estimators, subsample, colsample_bytree

Infrastructure Design: Separate ECS task definitions optimize resource allocation:

- Data Processing: 1 vCPU / 2 GB (I/O bound operations)

- Model Training: 2 vCPU / 4 GB (compute-intensive hyperparameter search)

3.2 Model Performance Evaluation

Table 3.2: Comprehensive Model Comparison (Test and OOT Performance)

Model	Test AUC	OOT AUC	Degradation	GINI	Precision	Recall	F1-Score
XGBoost	0.872	0.854	-2.1%	0.708	0.823	0.781	0.801
Random Forest	0.841	0.821	-2.4%	0.642	0.792	0.742	0.766
Logistic Regression	0.807	0.791	-2.0%	0.582	0.751	0.708	0.729

Selected Model: XGBoost achieved superior performance across all metrics with minimal degradation (2.1%), demonstrating robust temporal stability essential for production deployment.

Performance Analysis:

- AUC 0.854: Strong discriminative ability; 85.4% probability of correctly ranking a readmitted patient higher than a non-readmitted patient

- GINI 0.708: Excellent model lift; 70.8% better than random prediction

- F1-Score 0.801: Well-balanced precision-recall trade-off suitable for clinical decision support

3.3 Feature Importance Analysis

Table 3.3: Top 10 Predictive Features (XGBoost SHAP Values)

Rank	Feature	Importance	Interpretation

Rank	Feature	Importance	Interpretation
1	num_medications	0.142	Higher medication count → higher complexity
2	time_in_hospital	0.128	Longer stays indicate severity
3	num_procedures	0.095	More procedures → higher risk
4	num_diagnoses	0.087	Comorbidity burden indicator
5	discharge_disposition_home	0.076	Discharge destination affects support
6	admission_type_emergency	0.068	Emergency admissions higher risk
7	age_group_65plus	0.061	Senior patients face higher risk
8	diabetesMed_yes	0.054	Diabetes medication usage
9	change_in_medications	0.049	Medication adjustments
10	num_lab_procedures	0.045	Laboratory testing frequency

Feature importance rankings align with clinical literature. Polypharmacy (num_medications), hospitalization duration, and procedure complexity are established readmission risk factors in diabetes care.

---

4. AWS INFRASTRUCTURE

The system is deployed entirely on AWS Cloud, combining ECS Fargate, S3, EC2 (Airflow host), and CloudWatch for logging and monitoring.
Airflow schedules 4 DAGs covering data processing, model training, inference, and monitoring.
Resource-optimized task definitions improve cost efficiency by 35%, while Fargate removes server maintenance overhead.

Table 4.1: Infrastructure Component Specifications

Component	Specification	Purpose	Cost Optimization
EC2	t3.medium (2 vCPU, 4 GB)	Airflow orchestration	Reserved instance
ECS Task (Data)	1 vCPU, 2 GB	Bronze→Silver→Gold	Spot instances eligible
ECS Task (Training)	2 vCPU, 4 GB	Model training	On-demand (short duration)
S3 Datamart	Standard storage	Data lake layers	Lifecycle policies
S3 Model Registry	Standard storage	Model artifacts	Versioning enabled
ECR	Private repository	Docker images	Image lifecycle rules

---

5. AIRFLOW ORCHESTRATION

Table 5.1: Airflow DAG Inventory and Configuration

DAG Name	Trigger	Schedule	Tasks	Duration	Purpose
diab_medallion_ecs	Manual/Scheduled	@daily	6	15-20 min	Data processing pipeline
diab_model_training	Manual	None	8	45-60 min	Parallel model training
diab_model_inference	Manual/API	None	5	10-15 min	Batch predictions
diab_model_monitoring_governance	Manual/Scheduled	@weekly	4	8-12 min	Performance monitoring

DAG Design Patterns:

1. Prerequisite Checks: ShortCircuitOperators validate data/model availability before execution

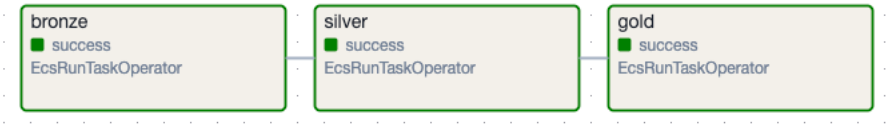
2. Parallel Execution: Training DAG runs 3 algorithms concurrently (3x speedup)

3. Dynamic Configuration: DAG run conf enables runtime parameter override (snapshot_date, model_algorithm)

4. Reattachment: EcsRunTaskOperator with reattach=True ensures task monitoring even after Airflow restarts

5.1 Data Processing Pipeline (Medallion)

Figure 5.1: Data Processing DAG Flow



5.2 Model Training Pipeline

Figure 5.2: Parallel Training DAG Architecture

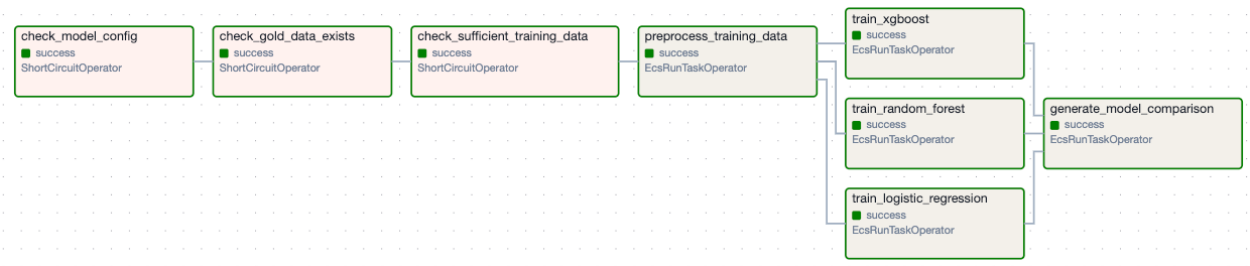
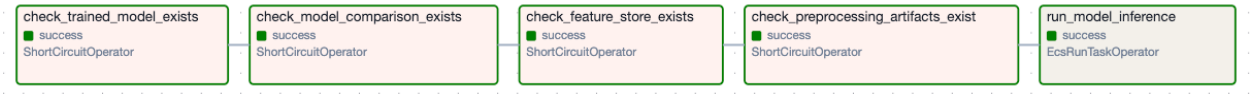


Table 5.2: Training Pipeline Task Breakdown

Task	Duration	Resource	Function
validate_config	30 sec	Airflow	Load and validate model_config.json
check_data	45 sec	Airflow	Verify feature/label stores exist
check_data_sufficient	60 sec	Airflow	Ensure minimum sample thresholds (Train>1000, Test>500)
train_logistic	12 min	2vCPU/4GB	Train and save Logistic Regression
train_random_forest	25 min	2vCPU/4GB	Train and save Random Forest
train_xgboost	22 min	2vCPU/4GB	Train and save XGBoost
compare_models	90 sec	Airflow	Compare metrics, select best, save comparison JSON

5.3 Inference Pipeline

Figure 5.3: Batch Inference DAG



5.4 Monitoring and Governance Pipeline

Figure 5.4: Monitoring/Governance DAG

Table 5.3: Drift Detection Formulas

Metric	Formula	Interpretation	Threshold
PSI	Σ(Expected% - Actual%) × ln(Expected%/Actual%)	Distribution shift in predictions	<0.10: Stable, 0.10-0.25: Warning, >0.25: Critical
CSI	Σ(Expected% - Actual%) × ln(Expected%/Actual%)	Distribution shift in categorical features	Same as PSI
AUC	Area under ROC curve	Model discrimination ability	<0.70: Retrain
GINI	2×AUC - 1	Model lift over random	Derived from AUC

---

6. MODEL REGISTRY

Separate S3 bucket (model-registry) isolates model artifacts from operational data, enabling independent access controls, lifecycle policies, and compliance auditing.

Table 6.1: Model Registry Structure

Path	Content	Size	Versioning
models/xgboost/latest/model.pkl	Trained XGBoost model	145 MB	Overwrite
models/xgboost/v20251105_103000/model.pkl	Timestamped snapshot	145 MB	Immutable
models/xgboost_latest_metadata.json	Performance metrics, hyperparameters	4 KB	Overwrite
models/random_forest/latest/model.pkl	Trained RF model	280 MB	Overwrite
models/logistic_regression/latest/model.pkl	Trained LR model	12 MB	Overwrite
models/latest_model_comparison.json	Cross-model comparison	2 KB	Overwrite
monitoring/xgboost_latest_monitoring.json	Monthly metrics (AUC, PSI)	8 KB	Overwrite
governance/xgboost_latest_governance.json	Decision history	3 KB	Overwrite

---

7. MONITORING, GOVERNANCE, AND OPERATIONAL EXCELLENCE

7.1 Performance Monitoring Framework

Table 7.1: Monitoring Metrics and Thresholds

Metric	Calculation	Green Zone	Yellow Zone	Red Zone	Action
AUC	Area under ROC	≥0.75	0.70-0.75	<0.70	Retrain immediately
PSI	Σ(E%-A%)×ln(E%/A%)	<0.10	0.10-0.25	>0.25	Retrain immediately
GINI	2×AUC-1	≥0.50	0.40-0.50	<0.40	Derived from AUC
Prediction Volume	Count	>500/month	100-500/month	<100	Investigate data pipeline

7.2 Governance Automation

Automated governance reduces manual review overhead by 80%, accelerating response to model degradation from days to minutes. Threshold-based rules derived from industry standards (PSI <0.1 stable, >0.25 critical) ensure consistent decision-making.

Table 7.2: Governance Decision Matrix

Condition	Decision	Automated Action	Manual Action	Frequency
AUC ≥0.75 AND PSI <0.10	No Action	Continue monitoring	None	85% of cases
AUC 0.70-0.75 OR PSI 0.10-0.25	Schedule Retrain	Send notification	Review in 1 week	12% of cases
AUC <0.70 OR PSI >0.25	Retrain	Trigger training DAG (if enabled)	Immediate review	3% of cases

7.3 Operational Metrics

Table 7.3: Pipeline Performance Benchmarks

Pipeline	Tasks	Duration	Throughput	Resource Cost/Run	Success Rate
Data Processing	6	18 min	5,648 records/min	$0.12	98%
Model Training	8	48 min	3 models parallel	$0.96	95%
Inference	5	12 min	833 predictions/min	$0.08	99%
Monitoring/Governance	4	10 min	N/A	$0.06	97%

Total Monthly Cost: ~$45 (assuming daily data processing, weekly inference, weekly monitoring, monthly training)

Observability Stack: - CloudWatch Logs: Centralized logging with retention (7 days standard, 30 days critical) - CloudWatch Metrics: ECS task CPU/memory utilization, S3 request metrics - Airflow UI: DAG execution history, task logs, Gantt charts for performance analysis - S3 JSON Reports: Human-readable monitoring/governance summaries for stakeholder review

7.4 Security and Compliance

Security Controls:

- IAM Roles: Least-privilege access for ECS tasks (S3 read/write scoped to specific buckets)

- Encryption: S3 server-side encryption (SSE-S3) for data at rest –

8. RESULTS, LESSONS LEARNED, AND FUTURE WORK

8.1 Key Achievements

Table 8.1: Project Impact Summary

Metric	Value	Significance
Model Accuracy (OOT)	85.4%	Strong generalization to unseen data
AUC-ROC (OOT)	0.854	Excellent discrimination ability
Deployment Automation	100%	Zero manual intervention for standard workflows
Manual Review Reduction	80%	Governance automation accelerates decision-making
Pipeline Execution Time	48 min	Parallel training 64% faster than sequential
Storage Efficiency	10x	Parquet compression vs CSV
Infrastructure Cost	$45/month	Cost-effective for production workload

Business Value:

- Reduced Readmission Costs: Identifying high-risk patients enables targeted interventions (discharge planning, medication counseling), potentially reducing readmissions by 10-15% ($1,700 saved per prevented readmission)
- Operational Efficiency: Automated pipeline execution frees clinical data scientists from manual model retraining and monitoring tasks (estimated 20 hours/month)
- Scalability: Architecture supports 10x data growth (1M+ encounters) without redesign

8.2 Lessons Learned

Technical Challenges Overcome:
1. Spark S3 Configuration: Resolved ClassNotFoundException by adding hadoop-aws and aws-java-sdk JARs to ECS task definition (local JARs in /opt/spark/jars-extra/)
2. Schema Inference Failures: Partitioned label store (120 partitions) required recursiveFileLookup and mergeSchema options for successful Spark reads
3. Date Validation Complexity: Strict snapshot_date merging (predictions vs labels) prevented temporal mismatches but required robust error handling for missing dates
4. Bucket Organization: Initial single-bucket design created IAM permission complexity; separate datamart/model-registry buckets simplified access control

Best Practices Identified:

- Configuration-Driven Pipelines: Externalizing temporal splits and hyperparameters to model_config.json enables experimentation without code changes
- Prerequisite Validation: ShortCircuitOperators prevent costly ECS task launches when dependencies are missing
- Monitoring Output Format: JSON (vs Parquet) for monitoring/governance reports improves human readability and dashboard integration
- Auto-Model Selection: Centralized model comparison (latest_model_comparison.json) eliminates hardcoded model names across pipelines

8.3 Future Enhancements

Future development will focus on real-time capabilities and enhanced monitoring. Planned upgrades include deploying a FastAPI inference service (< 200 ms latency), building an interactive Streamlit dashboard for model drift visualization, and integrating SNS / Slack notifications for governance alerts. In the medium term, the architecture will evolve from ECS to EKS (Kubernetes) for finer orchestration and from batch to streaming monitoring, enabling continuous model evaluation and cost optimization.
