Healthcare Billing Analytics

Prepared by: Nowsherwan Khan & Muhammad Idrees
Submitted to: Sir Shayan Ali

1. Use Case Selection

For this project, I have selected the **Healthcare Billing Analytics** use case from the Healthcare Services domain. This use case focuses on analyzing large volumes of healthcare claims data to identify billing inefficiencies, predict claim denials, and optimize revenue cycle management using big data and machine learning techniques.

2. Define the Scope (Business Problem Statement)

Clear Business Problem:
I will build a machine learning model that predicts healthcare claim denials before submission. The goal is to identify high-risk claims early, reduce financial losses for healthcare providers, and improve cash flow through proactive intervention and optimized billing processes.

Healthcare organizations process millions of claims annually. Each denied claim represents significant administrative costs and delayed revenue. Using big data analytics, we can process historical billing patterns, payer behavior, and clinical documentation to classify each claim's denial probability and provide actionable recommendations.

3. Dataset Information

Dataset Link: https://github.com/CMSgov/beneficiary-fhir-data/wiki/Synthetic-Data

Source: CMS (Centers for Medicare & Medicaid Services) - Publicly available synthetic dataset

Brief Metadata:

Attribute	**Value / Description**
Number of rows	~1.2 million synthetic medical claims
Number of features	48 attributes
File size	~5.2 GB (compressed)
Class label	Binary: denied vs. approved claims
Data type	Structured claims data with mixed types
Imbalance	Denial cases typically 5–15% of total claims

Data Composition:

- Patient demographics (age, gender, insurance type)
- Provider information (NPI, specialty, location)
- Clinical data (ICD-10 codes, CPT codes, HCPCS codes)
- Billing information (charged amounts, allowed amounts, payment details)
- Temporal data (service dates, submission dates, adjudication dates)
- Payer information (payer type, plan details, contract terms)

4. Big Data Pipeline Design

Data Ingestion:

- Real-time streaming via Apache Kafka for claim submissions
- Batch ingestion using Azure Data Factory for historical data

- API connectors for payer data integration

Storage:

- Raw data → Azure Data Lake Gen2 (Bronze layer)
- Processed data → Delta Lake (Silver/Gold layers)
- Feature store → Azure SQL Database
- Models → MLflow Model Registry

Processing:

- Azure Databricks for distributed transformations
- Spark Streaming for real-time claim scoring
- dbt (data build tool) for transformation orchestration
- Great Expectations for data quality validation

Visualization:

- Power BI dashboards for executive stakeholders
- Tableau reports for operational teams
- Custom alerting system for high-risk claims
- Grafana for real-time system monitoring

5. Machine Learning Methodology

A. Pre-Processing Strategy

1. Overview:
Preprocessing goals include cleaning healthcare claims data, handling missing values, encoding categorical medical codes, creating temporal and behavioral features, scaling numerical features, and addressing class imbalance while maintaining temporal validity.

2. Missing Values:

- **Detection:** Comprehensive missing value analysis across all features
- **Simple Imputation:**
    - Numeric features: Median imputation for billing amounts and clinical values
    - Categorical features: "Unknown" category for missing provider or patient data
- **Advanced Imputation:**
    - MICE (Multiple Imputation by Chained Equations) for correlated clinical features
    - Domain-guided imputation using medical coding standards
- **Drop Criteria:**
    - Features with >60% missing rate
    - Invalid clinical code combinations
    - Claims with missing critical dates

Strategy Summary:

- Numeric columns → Median imputation
- Categorical columns → "Unknown" imputation
- Clinical codes → Expert-guided imputation
- Temporal columns → Drop claims with missing critical dates

3. Handling Categorical Variables:

Column	Type	Recommended Encoding
gender	Low-cardinality	One-Hot Encoding

Column	Type	Recommended Encoding
insurance_type	Low-cardinality	One-Hot Encoding
provider_specialty	Medium-cardinality	Target Encoding
diagnosis_codes	High-cardinality	Frequency Encoding
procedure_codes	High-cardinality	Target Encoding
payer_id	Medium-cardinality	One-Hot Encoding
patient_id	High-cardinality	Drop (privacy)
provider_id	High-cardinality	Target Encoding

Encoding Choices:

- One-Hot Encoding for low-cardinality features
- Target/Frequency Encoding for high-cardinality medical codes
- Drop personally identifiable information (PII)

4. Feature Engineering:

Temporal Features:

- submission_lag_days, adjudication_time
- day_of_week, month_of_year
- time_since_last_claim

Clinical Features:

- diagnosis_code_count, procedure_code_count
- code_complexity_score, comorbidity_indicator

Financial Features:

- amount_to_contract_ratio, historical_average_ratio

- payer_specific_adjustment

Operational Features:

- provider_denial_rate_30d, payer_specific_denial_rate
- claim_edit_failure_score, credentialing_status

Behavioral Features:

- customer_num_claims_30d, submission_pattern_consistency
- payer_mix_complexity

Scaling:

- StandardScaler → Amounts, time durations
- RobustScaler → Features with outliers
- No scaling → Encoded categorical features and binary indicators

B. Recommended Machine Learning Algorithm: XGBoost

Why XGBoost?

1. Handles imbalanced data efficiently
2. High performance on structured healthcare claims
3. Robust to missing values and high-cardinality features
4. Scalable in big data environments
5. Explainable for regulatory compliance
6. Supports real-time predictions

C. Dataset Analysis

1. Size & Volume:

- 1.2 million claims, 3+ years of data

- Requires distributed processing and memory optimization

2. Number of Features:

- 48 original features expanding to 100+ after engineering

3. Class Imbalance:

- Denial rate: 5–15%
- Stratified sampling and class weighting required

Data Types Breakdown:

Type	Columns	Examples
Numeric	25+	billed_amount, allowed_amount
Categorical	15+	diagnosis_codes, provider_specialty
Temporal	5+	service_date, submission_date
Clinical	10+	ICD-10 codes, CPT codes, HCPCS codes
Operational	8+	payer_id, provider_credentials

6. Implementation Plan

1. Library Selection (Python Stack):

- Core: pandas, numpy, scikit-learn, xgboost, imbalanced-learn, mlflow, optuna, shap
- Big Data & Streaming: pyspark, kafka-python, azure-storage-blob
- Serving: fastapi, uvicorn, docker, prometheus-client
- Healthcare Specific: fhir-resources, icd10-cm, pyhive

2. Pseudo-code / High-level Logic:

- Batch Training Pipeline (weekly) → MLflow tracking, Delta Lake ingestion, feature engineering, preprocessing, XGBoost training, evaluation, MLflow registration
- Real-time Scoring Pipeline → Kafka streaming, preprocessing, scoring, alerting
- Model Serving → FastAPI endpoints /predict and /batch_predict

3. Evaluation Metrics:

- Primary: PR-AUC, Recall at 95% Precision, False Positives per 1,000 Claims, Business Cost Savings, Avg Revenue per Claim
- Secondary: ROC-AUC, F1-score, Detection Latency

4. Hyperparameter Tuning Strategy:

- Tool: Optuna with SparkTrials
- Objective: Maximize PR-AUC with cost constraints
- Parameters: max_depth, learning_rate, subsample, colsample_bytree, scale_pos_weight, reg_alpha, reg_lambda

5. Deployment & Monitoring Plan:

- Automated weekly retraining
- Shadow and canary deployments
- Monitoring: data quality, model performance, business metrics, system metrics
- Alerts & retraining triggers: performance degradation, data drift, changes in denial patterns
- Rollback strategy: maintain last 3 model versions, automated/manual rollback

Prepared by: Nowsherwan Khan & Muhammad Idrees
Date: 2025-11-21