Technical Report: Healthcare Provider Fraud Detection

1. Executive Summary

This report documents the development, training, and evaluation of a machine learning system designed to detect potential fraud among healthcare providers. The objective was to identify fraudulent providers based on aggregated claims data.

The project followed a standard data science lifecycle:

1. **Data Aggregation:** Transforming transactional claim-level data into provider-level feature vectors.
2. **Modeling:** Experimenting with tree-based, linear, and support vector models while addressing severe class imbalance (~90:10).
3. **Optimization:** Tuning hyperparameters using Cross-Validation.
4. **Evaluation:** Analyzing errors to understand the business impact of False Positives vs. False Negatives.

---

2. Data Exploration and Feature Engineering

2.1 Data Understanding

The raw data consisted of three primary datasets:

- **Beneficiary Data:** Demographic and health conditions of patients.
- **Inpatient/Outpatient Data:** Transactional claims including reimbursement amounts, deductibles, and diagnosis codes.
- **Labels:** The target variable PotentialFraud (Yes/No) mapped to the Provider ID.

Initial Findings:

- The dataset is highly imbalanced: **9.77% Fraudulent** vs **90.23% Non-Fraudulent** providers.
- There is a many-to-one relationship between claims and providers, necessitating an aggregation strategy to create a provider-level dataset.

2.2 Feature Engineering Strategy

To train the model, we transformed the data from "per-claim" granularity to "per-provider" granularity. The following aggregation logic was applied:

1. **Volume Features:**
   - TotalClaims: Count of claims filed by the provider.
   - UniqueBeneficiaries: Count of distinct patients.
   - UniquePhysicians: Count of distinct attending physicians associated with the provider.
2. **Financial Features:**
   - InscClaimAmtReimbursed (Sum, Mean, Std): High total reimbursements or high variance can indicate fraudulent activity.
   - DeductibleAmtPaid (Sum, Mean): Aggregated patient deductible payments.
3. **Patient Health Profile (Chronic Conditions):**
   - The original beneficiary data coded chronic conditions as 1 (Yes) and 2 (No). We preprocessed this to 1 and 0.
   - We computed the **mean** of these flags per provider (e.g., ChronicCond_Diabetes_mean). This effectively represents the **percentage** of a provider's patients suffering from specific chronic conditions.
4. **Renal Disease Indicator:**
   - Converted raw 'Y'/'0' values to binary 1/0 and aggregated by provider.

2.3 Data Cleaning

- Missing values in the aggregated dataset were dropped to ensure data quality.
- The target variable PotentialFraud was encoded: Yes = 1, No = 0.

3. Modeling Strategy and Candidate Selection

3.1 Handling Class Imbalance

Given the ~10% positive class rate, standard training would bias models toward the majority class. We addressed this using **Class Weighting** rather than resampling (SMOTE/Undersampling). This preserves the original data distribution while penalizing the model more for missing fraudulent cases.

- **Sklearn Models:** Used class_weight_dict

- **XGBoost:** Used scale_pos_weight, calculated as ratio of negative to positive samples.

3.2 Candidate Models

We experimented with five algorithm families to capture different data relationships:

1. **Logistic Regression (LR**)
2. **Random Forest (RF)**
3. **Decision Tree (DT)**
4. **Support Vector Machine (SVM)**
5. **XGBoost**

---

4. Experiment Log and Hyperparameter Tuning

4.1 Baseline Comparison

We first trained all models with default parameters (incorporating class weights).

Initial Results (Ranked by F1-Score):

1. **XGBoost:** F1: 0.624 | Precision: 0.62 | Recall: 0.62 | AUC: 0.936
2. **Random Forest:** F1: 0.583 | Precision: 0.73 | Recall: 0.49 | AUC: 0.938
3. **Logistic Regression:** F1: 0.575 | Precision: 0.43 | Recall: 0.87 | AUC: 0.913
4. **SVM:** F1: 0.547 | Precision: 0.39 | Recall: 0.92 | AUC: 0.930
5. **Decision Tree:** F1: 0.428 | Precision: 0.40 | Recall: 0.46 | AUC: 0.691

Insight:

- **XGBoost** provided the best immediate balance.
- **Logistic Regression** and **SVM** achieved very high recall (>85%) but suffered from low precision (<45%), meaning they flagged too many legitimate providers as fraud (high False Positive rate).
- **Random Forest** had high precision but missed half the fraud cases (Recall ~49%).

4.2 Rationale for Final Candidate Selection

Based on the baseline metrics, we selected **XGBoost** and **Logistic Regression** for hyperparameter tuning. The rationale for this decision was:

1. Why XGBoost? (The Balanced Performer)

- **Performance:** It achieved the highest baseline **F1-Score (0.62)**. In fraud detection, a balance is critical; we need to catch fraud (Recall) without overwhelming investigators with false leads (Precision).
- **Efficiency:** It offers faster training speeds and better scalability than SVM for tabular data.

2. Why Logistic Regression? (The High-Sensitivity Baseline)

- **Recall Performance:** It achieved an impressive **Recall of 0.87**, significantly higher than the tree-based models. It captures the vast majority of fraud cases.
- **Interpretability:** As a linear model, it provides direct coefficients, making it easy to explain exactly *which* feature increased the fraud risk (e.g., "Each 1% increase in Renal Disease patients increases fraud odds by X").

4.3 Hyperparameter Tuning (Two-Phase Strategy)

We implemented a two-phase tuning strategy to first explore the parameter space and then validate the findings using robust cross-validation.

Phase 1: Using fixed hold-out test set (End of Modeling Notebook)

- **Objective:** Conduct a broad, rapid search across a wide range of hyperparameters on a fixed hold-out test set to identify the most promising model configurations.
- **XGBoost Tuning Grid:**
  - n_estimators: [100, 300, 500, 700]
  - max_depth: [3, 5, 7, 9]
  - learning_rate: [0.01, 0.05, 0.1, 0.2]
  - **Best XGBoost Configuration:** n_estimators=300, max_depth=7, learning_rate=0.2.
  - **Metrics:** F1-Score: 0.635, Precision: 0.67, Recall: 0.60.
- **Logistic Regression Tuning Grid:**

    - C: [0.01, 0.1, 1, 10]
    - penalty: ['l2', ‘l1’, ‘elasticnet’, none]
    - solver: ['liblinear', 'saga', 'lbfgs', 'newton-cg'
    -]
    - **Best LR Configuration:** C=0.01, penalty='l2', solver='liblinear'.
    - **Metrics:** F1-Score: 0.575, Precision: 0.43, Recall: 0.87.

Phase 2: Robust Validation with Cross-Validation (Beginning of Evaluation Notebook)

- **Objective:** Use **5-Fold Stratified Cross-Validation** (GridSearchCV) to validate the optimal configurations from Phase 1, ensuring the model's performance generalizes reliably across multiple data subsets. The optimization metric remained the **F1-Score**.
- **XGBoost Tuning Grid:**
    - **Final Best XGBoost Parameters:** learning_rate=0.1, max_depth=5, n_estimators=100.
    - **Validated Outcome:** The model's recall improved to **0.71**, indicating better sensitivity to fraud, despite a small drop in the raw F1-score (**~0.595**) compared to Phase 1. The cross-validated result is deemed more stable.
- **Logistic Regression Tuning Grid:**
    - **Final Best LR Parameters:** C=0.01, penalty='l2', solver='liblinear'.
    - **Validated Outcome:** The tuning maintained the core strength of high recall (**0.83**), confirming its role as the secondary, high-sensitivity model.

---

5. Final Evaluation and Error Analysis

5.1 Selected Model Performance (XGBoost Tuned)

- **Confusion Matrix:**
    - True Negatives: 859

    - False Positives: 74
    - False Negatives: 28
    - True Positives: 73
- **Total Cost Calculation** (Assumed: FN Cost = $1000, FP Cost = $10):
    - **XGBoost Cost:** $28,740
    - *Comparison:* Logistic Regression cost was lower ($16,210) due to massive recall, but this comes at the operational cost of investigating 121 false leads vs XGBoost's 74. XGBoost is preferred for a balanced investigative workload.

5.2 Error Analysis

We analyzed specific misclassified samples to understand model weaknesses.

False Positives (Legit flagged as Fraud):

- **Characteristics:** These providers often had very high InscClaimAmtReimbursed_sum (e.g., $460,400) and DeductibleAmtPaid_sum.
- **Insight:** The model conflates "High Volume/High Cost" with "Fraud". Large, legitimate hospitals are at risk of being flagged.
- **Mitigation Strategy:** Future feature engineering should normalize financial amounts by beneficiary count (e.g., "Cost per Patient") to distinguish volume from fraud intensity.

False Negatives (Fraud flagged as Legit):

- **Characteristics:** many FN samples showed high total reimbursements ($255090), large numbers of unique beneficiaries, and high deductible sums ($32150), yet their per-claim statistics and chronic-condition ratios were often mid-range (not extreme).
- **Insight:** These fraudulent providers look like high-volume, legitimate providers — large totals but without clear, extreme outlier signatures — so the model treats them as normal.
- **Mitigation Strategy:** We need temporal features (e.g., "Spike in claims over 1 month") or network analysis features (e.g., "Shared beneficiaries with other fraudulent providers").

6. Conclusion and Final Recommendation

Based on the analysis, we propose two models based on the need for balance or catching most fraudulent cases:

Primary Model: XGBoost

- **Reasoning:** For a fraud detection task where both precision and recall matter, XGBoost provides the most reliable balance.
- **Business Case:** Investigating fraud is resource-intensive. If we deploy Logistic Regression, 57% of the alerts generated will be false alarms, leading to investigator fatigue and wasted resources. XGBoost provides a more targeted list of high-probability suspects, making it the most sustainable choice for the primary detection engine.

Secondary Model: Logistic Regression

- **Reasoning:** Even though its F1-score is lower than XGBoost, Logistic Regression is ideal when minimizing false negatives is the primary business goal (83% Recall).
- **Business Case:** This model should be run periodically (e.g., quarterly) to audit providers that XGBoost cleared.