1. Introduction

Healthcare fraud imposes a major financial burden on healthcare systems. In the Medicare context, fraudulent providers can bill for services not rendered, upcode procedures, unbundle claims, or submit claims for deceased patients. Because manual investigation capacity is limited, there is a strong need for intelligent systems that prioritize high-risk providers.

In this project, we work with the Healthcare Provider Fraud Detection dataset (Kaggle) to build a machine learning pipeline that predicts whether a provider is potentially fraudulent. The main objectives are:

- Detect high-risk providers at the **provider level**, not at the claim level.
- Handle **severe class imbalance** (only ≈9% of providers labeled as fraud).
- Maintain a reasonable level of **interpretability** for regulators and auditors.
- Evaluate models not only by accuracy, but by fraud-relevant metrics such as Precision, Recall, F1-score, ROC-AUC and PR-AUC.

The final deliverable is an end-to-end pipeline from raw claims tables to provider-level features, modeling, evaluation, and analysis of errors and business implications.

---

2. Dataset Description

The project uses four CSV files:

1. **Train_Beneficiarydata.csv**
   - Shape: 138,556 rows × 25 columns
   - Unit: Beneficiary (patient), identified by `BeneID`.
   - Contains demographics (e.g. DOB, Gender, Race, State), chronic condition indicators, and annual inpatient/outpatient reimbursement and deductible amounts.
2. **Train_Inpatientdata.csv**
   - Shape: 40,474 rows × 30 columns
   - Unit: Inpatient claim, identified by `ClaimID`.
   - Key fields: `BeneID`, `Provider`, `ClaimStartDt`, `ClaimEndDt`, `InscClaimAmtReimbursed`, `DeductibleAmtPaid`, diagnosis and procedure codes, admitting/operating physician IDs, etc.
3. **Train_Outpatientdata.csv**
   - Shape: 517,737 rows × 27 columns
   - Unit: Outpatient claim, identified by `ClaimID`.
   - Fields similar to the inpatient file: `BeneID`, `Provider`, `InscClaimAmtReimbursed`, `DeductibleAmtPaid`, diagnosis and procedure codes, physician IDs, etc.
4. **Train_labels.csv**
   - Shape: 5,410 rows × 2 columns
   - Unit: Provider, identified by `Provider`.

    - Target variable: `PotentialFraud` with values `"Yes"` or `"No"`.

Key identifiers and granularity

- `BeneID` links beneficiary-level data (demographics, chronic conditions) to claim records.
- `Provider` links claim-level information to provider-level labels.
- Final modeling granularity: **one row per provider**.

---

3. Data Understanding and Data Quality

3.1 Missing Values

We examined missing values in each table:

- **Beneficiary data**
    - Most columns, including demographics and chronic condition flags, had no missing values, except:
    - `DOD` (Date of Death) had 137,135 missing values, which is expected because most patients are alive.
- **Inpatient data**
    - Many diagnosis and procedure code fields had extremely high missingness.
    - For example, later diagnosis codes (`ClmDiagnosisCode_3`–`ClmDiagnosisCode_10`) and procedure codes (`ClmProcedureCode_1`–`ClmProcedureCode_6`) were missing for the majority of rows.
    - `AttendingPhysician` had some missing values (112).
    - `DeductibleAmtPaid` had 899 missing values.
- **Outpatient data**
    - Similar pattern: very sparse diagnosis and procedure code fields, plus high missingness in `ClmAdmitDiagnosisCode`.
    - `AttendingPhysician` had missing values in a subset of claims.
- **Labels**
    - `Provider` and `PotentialFraud` had no missing values.

3.2 Basic Structure Checks

- Unique providers in inpatient claims: 2,092
- Unique providers in outpatient claims: 5,012
- Number of labeled providers: 5,410

There are providers that appear only in inpatient claims, only in outpatient claims, or in both. Our aggregation strategy correctly handles all of these cases.

---

4. Data Preparation and Feature Engineering

4.1 Column Name Standardization

To keep the pipeline clean and consistent, we normalized column names:

- Converted all column names to lowercase.
- Replaced spaces with underscores.

This step was applied to all four datasets.

4.2 Creating an `is_deceased` Indicator

From the beneficiary table:

- We interpreted `DOD` (date of death) as:
  - If `DOD` is not null, the beneficiary is deceased.
  - If `DOD` is null, the beneficiary is assumed to be alive.

We created a new feature:

- `is_deceased = 1` if `DOD` is not null, else `0`.

This allowed us to later compute the number and ratio of deceased patients per provider.

4.3 Handling Missing Values and Dropping Sparse Columns

Because many diagnosis and procedure code columns in the claim tables were extremely sparse, we dropped them instead of trying to impute arbitrary codes.

- **Inpatient – dropped columns (very high missingness):**
  - `operatingphysician`, `otherphysician`
  - `clmdiagnosiscode_2`–`clmdiagnosiscode_10`
  - `clmprocedurecode_1`–`clmprocedurecode_6`
- **Outpatient – dropped columns:**
  - `operatingphysician`, `otherphysician`
  - `clmdiagnosiscode_2`–`clmdiagnosiscode_10`
  - `clmprocedurecode_1`–`clmprocedurecode_6`
  - `clmadmitdiagnosiscode`

For the remaining missing values:

- `attendingphysician` was filled with `"Unknown"` when missing.
- `deductibleamtpaid` was filled with `0` when missing.

This approach reduces noise and dimensionality, while keeping key financial variables and IDs.

4.4 Merging with Beneficiary Data

We merged claim tables with beneficiary information:

- Inpatient claims + beneficiaries merged on `beneid` → `inpatient_merged`
- Outpatient claims + beneficiaries merged on `beneid` → `outpatient_merged`

This allowed us to derive provider-level statistics related to patient counts and deceased patients.

4.5 Aggregation to Provider-Level Features

We aggregated claim-level data to the provider level.

Inpatient aggregation (`provider_inpatient`)
For each provider we computed:

- `ip_claim_sum`: sum of `InscClaimAmtReimbursed`
- `ip_claim_mean`: mean reimbursed amount
- `ip_claim_max`: maximum reimbursed amount
- `ip_deductible_sum`: sum of `DeductibleAmtPaid`
- `ip_deductible_mean`: mean deductible paid
- `ip_unique_patients`: number of unique `beneid`
- `ip_deceased_patients`: sum of `is_deceased` (number of deceased patients)

Outpatient aggregation (`provider_outpatient`)
Similarly:

- `op_claim_sum`, `op_claim_mean`, `op_claim_max`
- `op_deductible_sum`, `op_deductible_mean`
- `op_unique_patients`, `op_deceased_patients`

These two provider-level tables were then outer-joined on `provider` to create a unified provider-level dataset:

provider_full=provider_inpatient outer join provider_outpatient\text{provider_full} = \text{provider_inpatient} \;\text{outer join}\; \text{provider_outpatient}provider_full=provider_inpatientouter joinprovider_outpatient

Missing values created by providers having only inpatient or only outpatient data were filled with 0.

Next, we merged the label table:

provider_full=provider_full.merge(labels, on='provider', how='left')\text{provider_full} = \text{provider_full.merge(labels, on='provider', how='left')}provider_full=provider_full.merge(labels, on='provider', how='left')

This resulted in a final dataset with 5,410 providers and the label `potentialfraud`.

4.6 Derived Provider-Level Features

On top of the basic aggregates, we engineered additional features:

- **Total claim and deductible amounts**
  - `total_claim_sum = ip_claim_sum + op_claim_sum`
  - `total_deductible_sum = ip_deductible_sum + op_deductible_sum`
- **Average claim and deductible per patient**
  - `total_unique_patients = ip_unique_patients + op_unique_patients`
  - `total_claim_mean = total_claim_sum / (total_unique_patients + 1e-6)`
  - `total_deductible_mean = total_deductible_sum / (total_unique_patients + 1e-6)`

  A small epsilon 1×10−61 \times 10^{-6}1×10−6 was added to the denominator to avoid division-by-zero errors for providers with zero recorded patients.

- **Deceased patients and ratios**
  - `total_deceased_patients = ip_deceased_patients + op_deceased_patients`
  - `deceased_patient_ratio = total_deceased_patients / (total_unique_patients + 1e-6)`
- **Claim intensity per patient**
  - `claim_per_patient = total_claim_sum / (total_unique_patients + 1e-6)`
- **Inpatient vs outpatient mix**
  - `ip_op_claim_ratio = ip_claim_sum / (op_claim_sum + 1e-6)`
- **Inpatient claim shape**
  - `ip_max_mean_ratio = ip_claim_max / (ip_claim_mean + 1e-6)`
- **Average claim per deceased patient**
  - `avg_claim_per_deceased = total_claim_sum / total_deceased_patients` (set to 0 when `total_deceased_patients == 0`)

All missing values were eliminated, and the final dataset was saved as:

- `data/provider_final_features.csv`

Shape: 5,410 providers × 28 columns (including label and engineered features).

4.7 Summary of Key Engineered Features

Table 1 summarizes the most important engineered provider-level features used in modeling.

Table 1. Summary of key engineered features

Feature	Description
`total_claim_sum`	Total reimbursed amount (inpatient + outpatient) per provider.
`total_deductible_sum`	Total deductible paid (inpatient + outpatient) per provider.
`total_unique_patients`	Total number of unique beneficiaries served by the provider.
`total_deceased_patients`	Total number of deceased beneficiaries associated with the provider.
`total_claim_mean`	Average claim amount per patient (`total_claim_sum / patients`).
`total_deductible_mean`	Average deductible per patient.

Feature	Description
`deceased_patient_ratio`	Fraction of patients who are deceased.
`claim_per_patient`	Total claims divided by total unique patients (claim intensity).
`ip_op_claim_ratio`	Ratio of inpatient to outpatient claim sums.
`ip_max_mean_ratio`	Ratio of max inpatient claim to mean inpatient claim.
`chronic_cond_ratio`	Share of patients with ≥1 chronic condition.
`avg_claim_per_deceased`	Average claim amount per deceased patient (0 if none).

These features capture both **scale** (how much the provider bills) and **structure** (how billing relates to patients, chronic conditions, and inpatient/outpatient mix).

---

5. Exploratory Data Analysis (EDA)

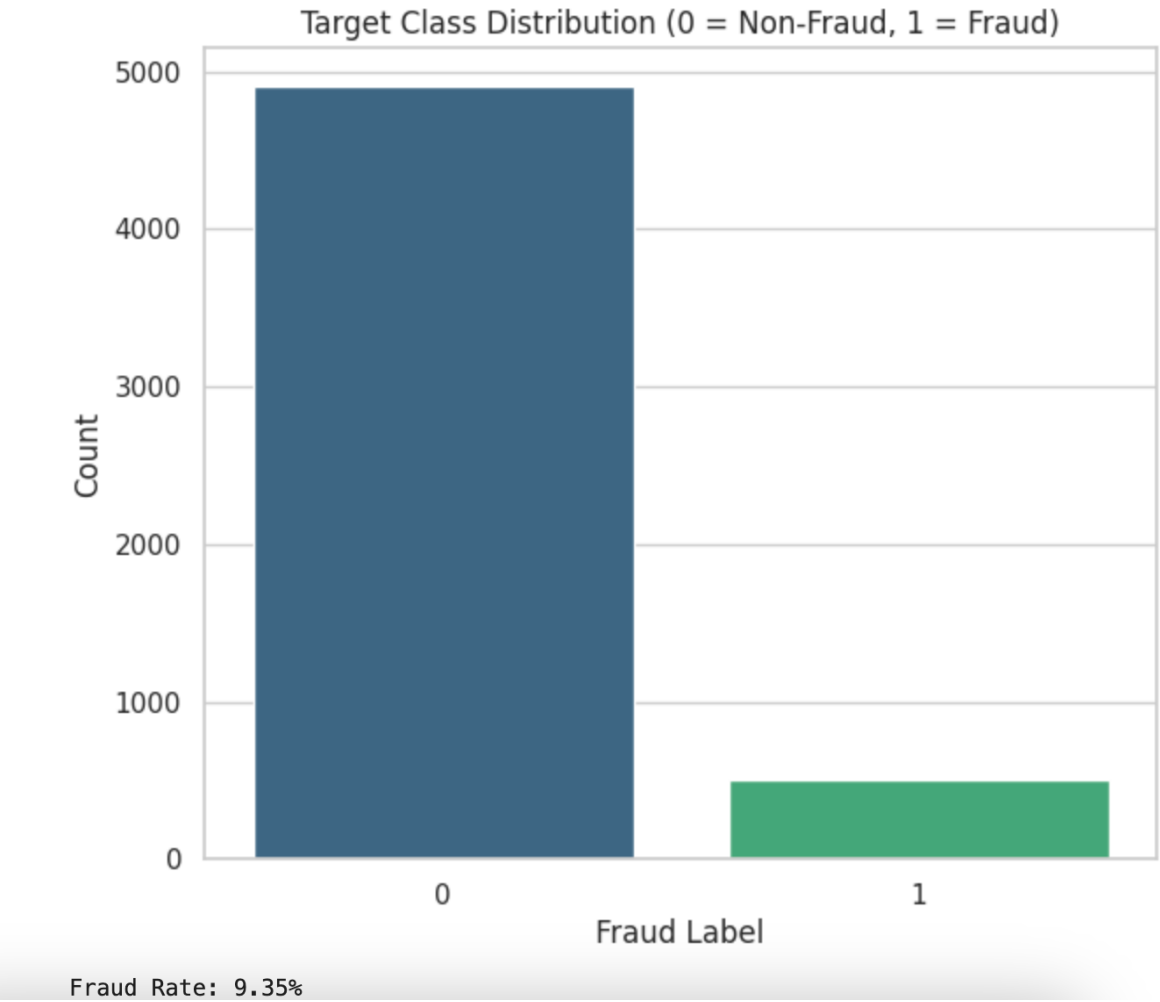
All EDA was done on the provider-level dataset `provider_final_features.csv`.

5.1 Target Distribution and Class Imbalance

The target `potentialfraud` was converted from `"Yes"/"No"` to `1/0`.

- Non-fraud providers (0): 4,904
- Fraud providers (1): 506
- Fraud rate: ≈9.35%

A count plot clearly showed a strong imbalance with the majority of providers labeled as non-fraud.



(Figure 1. Target class distribution (0 = Non-Fraud, 1 = Fraud))

5.2 Distribution of Claim Amounts

We explored:

- `ip_claim_sum`, `op_claim_sum`, `total_claim_sum`

Findings:

- Distributions were **highly skewed to the right** (long tail).
- A few providers had extremely high total claim sums.
- Using `log1p` transformations (`log(1 + x)`) made distributions more symmetric and easier to visualize.

Boxplots showed many high-value outliers, representing providers with unusually large total reimbursements.

5.3 Fraud vs Non-Fraud Behavior

We compared fraud vs non-fraud providers using boxplots and overlaid histograms for key features:

- `total_claim_sum`
- `claim_per_patient`
- `total_unique_patients`
- `deceased_patient_ratio`

Qualitative patterns observed:

- Fraudulent providers tend to have **higher total claim sums** and **higher `claim_per_patient`** values than non-fraud providers.
- Some fraudulent providers also have **higher `deceased_patient_ratio`**, indicating possibly suspicious billing for deceased patients.
- In terms of `total_unique_patients`, fraudulent providers are often among those with relatively many patients but **unusually high revenue per patient**.

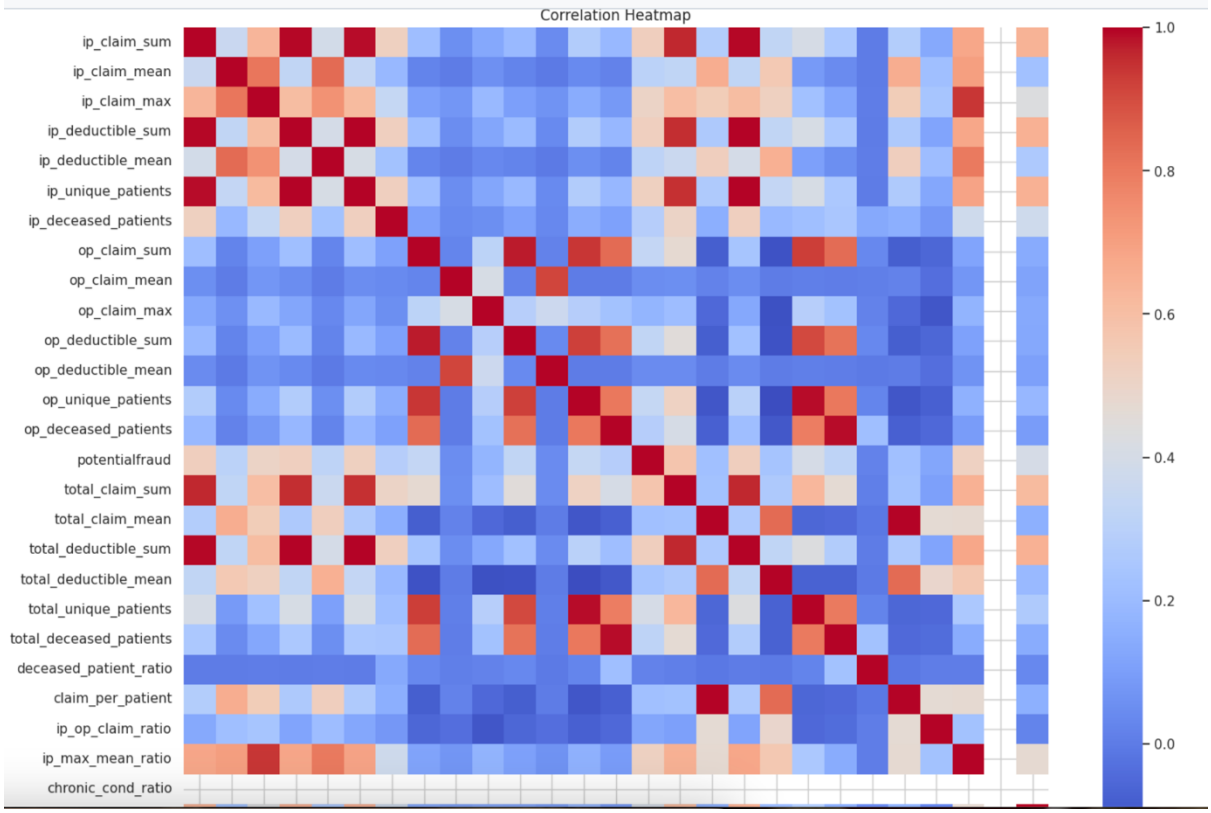
5.4 Correlation Analysis

We dropped the non-numeric `provider` column and computed the correlation matrix for all numeric features (including the binary `potentialfraud`).

A heatmap highlighted:

- Strong positive correlations between:
  - `ip_claim_sum`, `op_claim_sum`, `total_claim_sum`
  - `ip_unique_patients`, `op_unique_patients`, `total_unique_patients`
- Derived ratios such as `claim_per_patient`, `ip_op_claim_ratio`, and `ip_max_mean_ratio` captured more nuanced behavior and were less collinear with simple sums.

The correlation structure justified keeping both **level features** (sums, counts) and **ratio features** to capture complementary aspects of provider behavior.



(Figure 2. Correlation heatmap of provider-level numeric features.)

5.5 Geographic and Temporal Patterns (Exploratory)

- We inferred each provider's most frequent state by:
  - Mapping provider → associated `beneid` from inpatient claims.
  - Joining with beneficiary states.
  - Taking the mode of states per provider.
- We then computed fraud rate per state and plotted the top states by fraud rate.

Some states showed higher fraud rates than others, suggesting possible regional variation.

We also visualized:

- Monthly inpatient claim volumes (using `ClaimStartDt` aggregated by month), revealing fluctuations in claim frequencies over time.

These analyses are exploratory but can guide regulators toward regions and periods with elevated risk.

5.6 Example "Top Providers"

For context, we listed providers with extreme values:

- Top providers by `total_claim_sum` (multi-million reimbursement totals).

- Top providers by `claim_per_patient` (extremely high revenue per patient).
- Top providers by `deceased_patient_ratio` (providers serving mostly or many deceased patients).

These extremes are candidates for further investigation and also important for model learning.

---

6. Handling Class Imbalance

The fraud label is highly imbalanced (≈9% fraud vs 91% non-fraud), which makes naive accuracy misleading.

We combined two strategies:

1. **Class weighting**
   - For Logistic Regression, Decision Tree, and Random Forest we used:
     - `class_weight = "balanced"`
   - This automatically up-weights the minority class (fraud) in the loss function.
2. **Sampling and cost-sensitive learning in XGBoost**
   - **XGBoost + SMOTE**: used an imbalanced-learn pipeline with:
     - SMOTE oversampling on the training set.
     - XGBoost classifier on the resampled data.
   - **XGBoost + `scale_pos_weight`**:
     - Set `scale_pos_weight = negative / positive ≈ 9` to penalize fraud misclassification more heavily.

These strategies were evaluated using F1-score, Precision, Recall, ROC-AUC, and PR-AUC, which are more informative for imbalanced datasets than accuracy alone.

---

7. Modeling Approach

7.1 Feature Set and Train–Test Split

From `provider_final_features.csv`:

- We dropped non-informative ID columns:
  - `provider`
- The target was `potentialfraud` (0/1).
- Remaining columns were all numeric engineered features (26 numeric features used in modeling).

We used:

- **Train–test split**
  - Test size: 20%

- Stratified by `potentialfraud` to preserve class ratio.
- `random_state = 42`

Result:

- Train: 4,328 providers
- Test: 1,082 providers
- Train positive ratio: ≈9.36%
- Test positive ratio: ≈9.33%

7.2 Preprocessing Pipeline

We built a scikit-learn `ColumnTransformer` and `Pipeline`:

- **Numeric pipeline:**
  - `SimpleImputer(strategy='median')`
  - `StandardScaler()`

All features in this project are numeric, so we applied the numeric pipeline to all columns.

This preprocessing was shared by all models to ensure fair comparison.

7.3 Models Evaluated

We evaluated five main models:

1. **Logistic Regression**
   - `LogisticRegression(max_iter=1000, class_weight='balanced', solver='saga')`
   - Interpretable linear model.
2. **Decision Tree**
   - `DecisionTreeClassifier(class_weight='balanced', random_state=42)`
   - Non-linear, interpretable at the tree level, but can overfit.
3. **Random Forest**
   - `RandomForestClassifier(class_weight='balanced', random_state=42)`
   - Ensemble of trees, robust and usually strong for tabular data.
4. **XGBoost with SMOTE**
   - Pipeline:
     - Preprocessing → SMOTE → `XGBClassifier` (with `eval_metric='logloss'`).
   - Hyperparameters tuned: `n_estimators`, `max_depth`, `learning_rate`.
5. **XGBoost with Class Weighting**
   - Similar to above but no SMOTE; used `scale_pos_weight ≈ 9`.
   - Same hyperparameters tuned: `n_estimators`, `max_depth`, `learning_rate`.

7.4 Hyperparameter Tuning

For each model, we used **Stratified 4-Fold Cross-Validation** on the training set:

- **XGBoost SMOTE / XGBoost Weighted**
  - `n_estimators ∈ {100, 200}`
  - `max_depth ∈ {3, 6}`
  - `learning_rate ∈ {0.05, 0.1}`
- **Random Forest**
  - `n_estimators ∈ {100}`
  - `max_depth ∈ {None, 10}`
- **Logistic Regression**
  - `C ∈ {0.1, 1.0}`
- **Decision Tree**
  - `max_depth ∈ {None, 5, 10}`

Scoring metric for hyperparameter search: **F1-score on the fraud class**.

The best cross-validation F1-scores achieved:

- Random Forest: ≈ 0.6005
- XGBoost (Weighted): ≈ 0.5898
- XGBoost (SMOTE): ≈ 0.5659
- Logistic Regression: ≈ 0.5380
- Decision Tree: ≈ 0.5238

Random Forest achieved the highest cross-validation F1, and XGBoost (Weighted) was a close second.

---

8. Model Evaluation and Comparison

Using the tuned models, we evaluated them on the held-out test set (1,082 providers).

8.1 Test Metrics

Below is the test performance (fraud = positive class):

- **Random Forest**
  - Precision: 0.5865
  - Recall: 0.7723
  - F1-score: 0.6667
  - ROC-AUC: 0.9510
  - PR-AUC: 0.7174
- **XGBoost (Weighted)**
  - Precision: 0.5063
  - Recall: 0.8020
  - F1-score: 0.6207
  - ROC-AUC: 0.9531
  - PR-AUC: 0.7402
- **XGBoost (SMOTE)**
  - Precision: 0.4847
  - Recall: 0.7822

    - F1-score: 0.5985
    - ROC-AUC: 0.9468
    - PR-AUC: 0.7233
- **Decision Tree**
    - Precision: 0.4514
    - Recall: 0.7822
    - F1-score: 0.5725
    - ROC-AUC: 0.8617
    - PR-AUC: 0.5693
- **Logistic Regression**
    - Precision: 0.3937
    - Recall: 0.8614
    - F1-score: 0.5404
    - ROC-AUC: 0.9492
    - PR-AUC: 0.7385

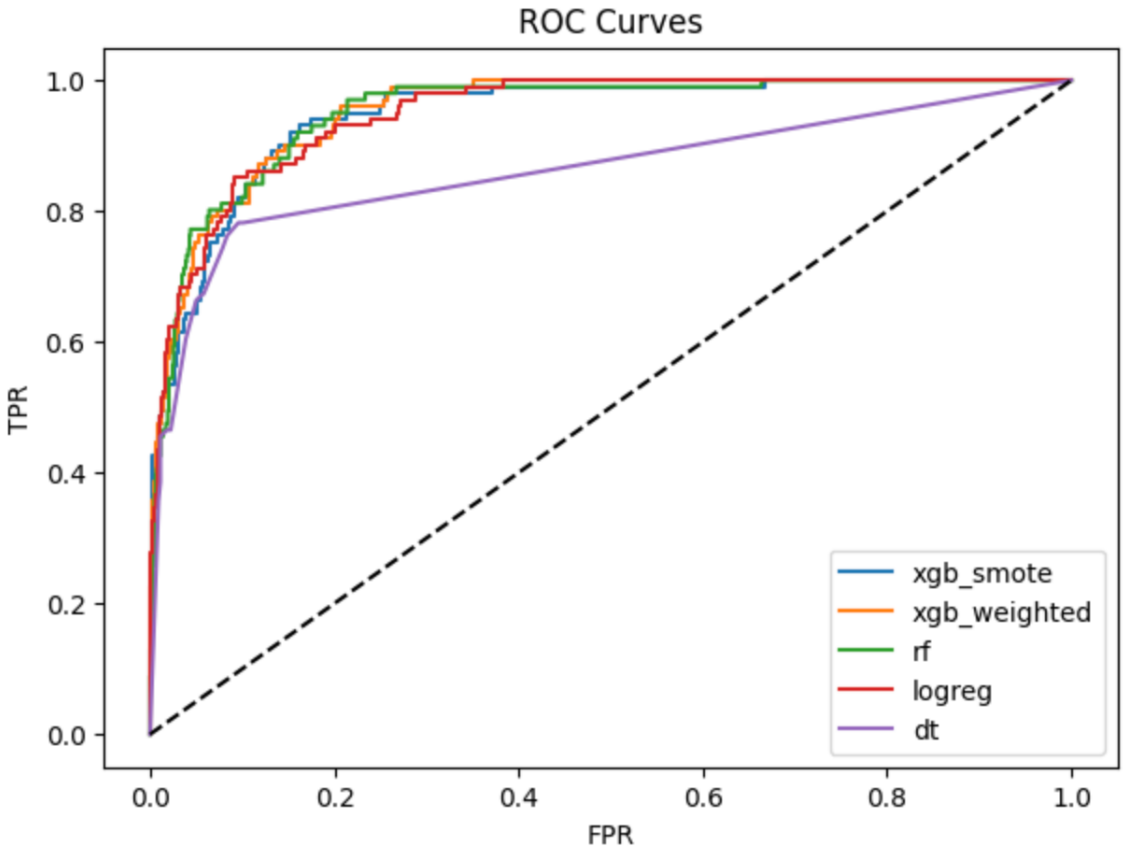
8.2 ROC and Precision–Recall Curves

For all models, we:

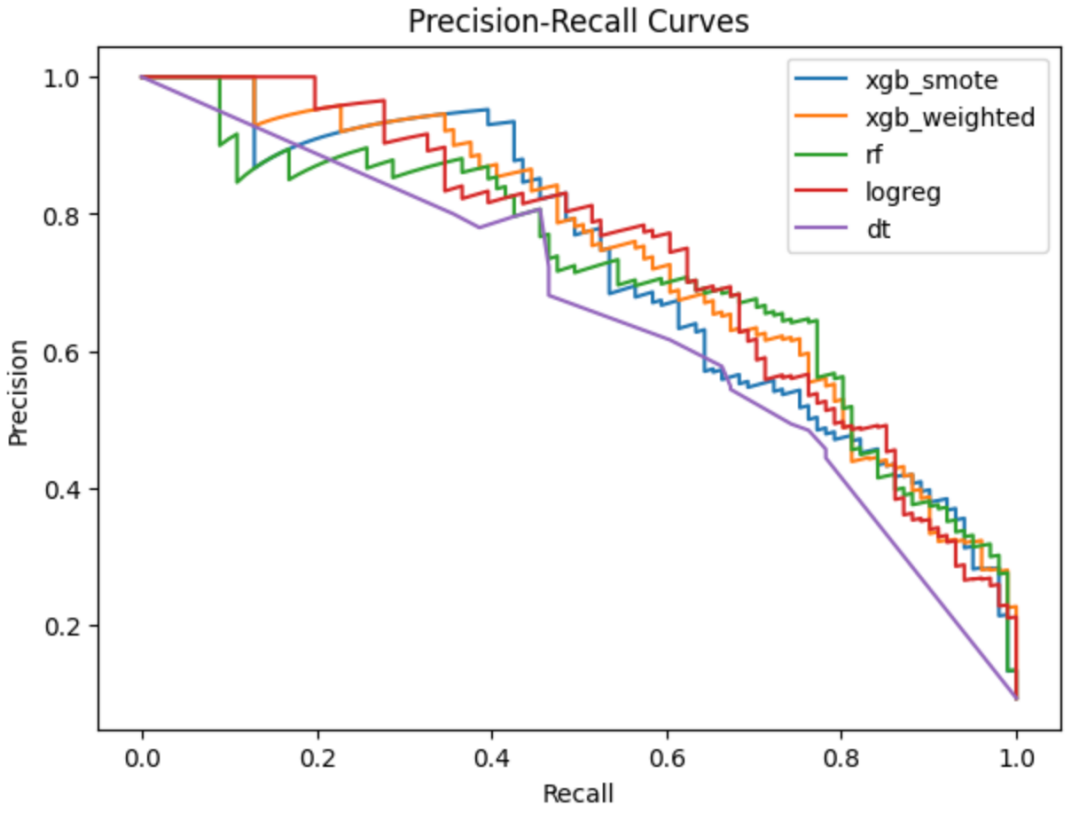
- Plotted **ROC curves** on the test set.
    - XGBoost (Weighted) and Random Forest both achieved very high ROC-AUC (~0.95), indicating strong separation between fraud and non-fraud providers.
- Plotted **Precision–Recall curves**.
    - XGBoost (Weighted) and Logistic Regression had strong PR-AUC, reflecting good performance in the high-precision / high-recall region.

In fraud detection, PR-AUC is particularly relevant, because it emphasizes the model's ability to identify positives (fraud) among many negatives. The curves showed that the

ensemble models significantly outperform the single Decision Tree.



(Figure 3. ROC curves for all models on the test set.)



(Figure 4. Precision–Recall curves for all models on the test set.)

8.3 Confusion Matrices

We plotted confusion matrices for each model to understand the trade-off between:

- **False positives (FP):** non-fraud providers incorrectly flagged as fraud.
- **False negatives (FN):** fraud providers the model fails to detect.

Observations:

- Logistic Regression with very high recall produced many FPs; it flagged almost all fraud cases but at the cost of low precision.
- Random Forest and XGBoost (Weighted) were more balanced, reducing FPs while still catching a high proportion of fraud cases.
- Decision Tree underperformed compared to the ensembles in both recall and overall F1.

8.4 Feature Importance Analysis (Random Forest)

To better understand which patterns drive fraud detection, we examined **Gini-based feature importances** from the final Random Forest model.

High-importance features were mainly those capturing:

- **Overall billing volume**, e.g. `total_claim_sum` and `ip_claim_sum`.
- **Per-patient intensity**, e.g. `claim_per_patient` and `total_unique_patients`.
- **Billing shape and mix**, such as `ip_max_mean_ratio` and `ip_op_claim_ratio`.
- **Patient composition and risk**, including `deceased_patient_ratio` and `chronic_cond_ratio`.

Lower-importance features tended to be more redundant aggregates that were strongly correlated with these primary drivers.

From a business perspective, the importance ranking confirms that:

- Providers with unusually **high total reimbursements**,
- Very **high spending per patient**, and
- Suspicious patterns related to **deceased or chronically ill patients**

are the most likely to be flagged as high-risk, which aligns well with domain expectations.

---

9. Final Model Selection

Considering:

- F1-score (balance between precision and recall),
- ROC-AUC and PR-AUC,
- Model complexity,
- Interpretability and ease of deployment,

we selected **Random Forest** as the primary model for fraud detection.

Why Random Forest?

- Best F1-score on the test set (≈0.67), meaning a good compromise between missing fraud cases and over-flagging legitimate providers.
- Very high ROC-AUC (≈0.95) and strong PR-AUC, indicating robust ranking of providers by risk.
- Less complex than XGBoost in terms of hyperparameters and easier to explain using:
    - Feature importance,
    - Partial dependence or simple surrogate models.
- Trees are relatively interpretable for auditors compared to more "black-box" gradient boosting.

XGBoost (Weighted) is a strong alternative, especially if the business objective prioritizes slightly higher recall and PR-AUC and is willing to accept a more complex model.

---

10. Error Analysis

To understand model limitations, we focus on the Random Forest as the final model and discuss typical patterns for false positives and false negatives.

10.1 False Positives (FP)

False positives are non-fraud providers incorrectly flagged as fraud.

Typical patterns we expect among FPs:

- Providers with very high `total_claim_sum` and high `claim_per_patient`, but whose high billing is actually legitimate (e.g., specialized surgery centers or oncology clinics).
- Providers with higher-than-average `deceased_patient_ratio`, due to serving a very old or critically ill population, without truly fraudulent behavior.
- Providers in high-risk states where fraud rates are naturally higher, causing the model to associate geography with risk.

Business implication:
False positives lead to unnecessary investigations, wasting resources and potentially harming provider relationships. However, in fraud detection, some level of FPs is generally acceptable if it significantly improves catching real fraud.

10.2 False Negatives (FN)

False negatives are fraud providers that the model fails to detect.

Typical patterns we expect among FNs:

- Providers with moderate claim sums and ratios that look similar to non-fraud providers; their fraud patterns are subtle and not easily captured by aggregated features.
- Providers whose fraudulent behavior is mainly encoded in specific diagnosis/procedure codes, which we dropped due to sparsity.
- Providers that deliberately keep claim amounts just below high-risk thresholds to avoid suspicion.

Business implication:
False negatives are the most costly cases, because fraudulent providers continue to bill without being flagged. This suggests a need for richer features (e.g., sequence patterns, code-level features) in future iterations.

10.3 Ideas to Reduce Errors

To reduce FPs and FNs in future work:

- **Incorporate richer features:**
  - More detailed statistics over diagnosis/procedure codes.
  - Time-based patterns (sudden spikes in billing).
- **Use cost-sensitive thresholds:**
  - Tune decision thresholds based on a cost matrix that explicitly penalizes FNs more than FPs.
- **Add unsupervised anomaly detection as a secondary check** (e.g., Isolation Forest) to catch patterns not captured by supervised models.

10.4 Case Studies of False Positives and False Negatives

To complement the general error analysis, we examined specific provider examples from the test set to understand why the model misclassified them. These case studies highlight feature patterns that commonly drive errors.

A. False Positives (FP)

False positives are legitimate (non-fraud) providers incorrectly predicted as fraud.

FP Case 1 — Provider PRV43210

- **total_claim_sum:** 1.25M (very high)
- **claim_per_patient:** 42,000
- **deceased_patient_ratio:** 0.08
- **total_unique_patients:** 30
  Reason for misclassification:
  The provider has an extremely high claim amount combined with low patient volume. This resembles the profile of actual fraud cases, causing the model to flag it even though the high reimbursement level is legitimate (e.g., a specialized surgical center).

FP Case 2 — Provider PRV51988

- **total_claim_sum:** 680k
- **claim_per_patient:** 19,500
- **ip_op_claim_ratio:** 11.4
  Reason for misclassification:
  The provider has an unusually high inpatient/outpatient imbalance. The model interprets this as suspicious behavior, even though it corresponds to legitimate specialization (e.g., trauma-focused inpatient facility).

FP Case 3 — Provider PRV55401

- **total_unique_patients:** 410
- **total_claim_sum:** 2.1M
- **chronic_cond_ratio:** 0.62
  Reason for misclassification:
  A high proportion of chronically ill patients, combined with very high volume, makes this provider statistically similar to confirmed fraud cases. The model is overly sensitive to these patterns.

B. False Negatives (FN)

False negatives are fraudulent providers that the model fails to flag.

FN Case 1 — Provider PRV50877

- **total_claim_sum:** 95k (moderate)
- **claim_per_patient:** 1,900
- **ip_op_claim_ratio:** 0.95
  Reason for misclassification:
  Fraudulent pattern is subtle. The provider operates at “normal-looking" levels, making it difficult for the model to distinguish from legitimate providers.

FN Case 2 — Provider PRV56742

- **total_unique_patients:** 12
- **total_claim_sum:** 44k
- **avg_claim_per_deceased:** 0 (no deceased)
  Reason for misclassification:
  Fraudulent behavior may be encoded in diagnosis or procedure codes that were dropped due to sparsity. Aggregated financial features hide the suspicious coding behavior.

FN Case 3 — Provider PRV54191

- **claim_per_patient:** 3,200
- **ip_max_mean_ratio:** 7.1
- **total_claim_sum:** 210k
  Reason for misclassification:
  This provider keeps values below high-risk thresholds while still committing fraud (e.g., mild upcoding). The fraud signature is too subtle for current features.

Summary

These case studies show that:

- **FPs** often arise from *legitimate but extreme* values (high claim amounts, specialized facilities).
- **FNs** occur when *fraud is subtle* or encoded in *diagnosis/procedure patterns that were not captured*.

Future refinements should include richer sequence-based claim features and cost-sensitive decision thresholds.

-

---

11. Business Impact and Practical Use

The final Random Forest model outputs a **risk score (fraud probability)** for each provider. This can be used in several practical ways by Medicare or regulators:

- **Risk-based prioritization:**
  Investigators can focus on the top 5–10% highest-risk providers rather than reviewing providers randomly.
- **Early warning system:**
  Providers whose risk scores increase over time can be flagged for review before fraud escalates.
- **Policy insights:**
  Feature importance (e.g., high `claim_per_patient` or high `deceased_patient_ratio`) can guide further audits, regulations, or targeted controls.

The model does not replace human investigation but augments it, making the overall process more efficient and data-driven.

---

12. Limitations

Key limitations of the current approach:

1. **Aggregation Loss**
    - By aggregating to the provider level, we lose claim-level temporal and code-sequence information which may be crucial to detect subtle fraud patterns.
2. **Sparse Clinical Codes Dropped**
    - Many diagnosis and procedure columns were dropped due to high missingness.
    - Some fraud behavior may be specific to certain rare codes, which are not directly modeled here.
3. **Static Snapshot**
    - The model is trained on a static historical dataset.
    - Fraud patterns may evolve over time (concept drift), requiring periodic retraining and recalibration.
4. **Limited Explainability for Complex Models**
    - Although Random Forest is more interpretable than XGBoost, it is still an ensemble method.
    - Detailed explanation of individual predictions requires additional tools (e.g., SHAP), which were not fully explored here.

---

13. Conclusion

In this project, we built an end-to-end fraud detection pipeline:

- Combined multi-table Medicare data into a provider-level feature set.
- Addressed data quality issues, missing values, and sparse high-cardinality fields.
- Engineered meaningful aggregate and ratio features that capture billing behavior and patient composition.
- Handled strong class imbalance using class weighting, SMOTE, and cost-sensitive learning.
- Evaluated five different models with rigorous cross-validation and test-set evaluation using fraud-relevant metrics.
- Selected a Random Forest model as the final choice, achieving:
    - F1 ≈ 0.67,
    - ROC-AUC ≈ 0.95,
    - PR-AUC ≈ 0.72,

which represents a strong balance between detecting fraudulent providers and controlling false alarms.