Medicare Provider Fraud Detection Using Machine Learning

Leveraging machine learning to identify potentially fraudulent healthcare providers

Project Overview

Goal: Detect potentially fraudulent healthcare providers

The goal of this project is to develop a machine learning model to detect potentially fraudulent healthcare providers.

Fraud is rare → imbalanced classification

The problem of healthcare provider fraud detection is characterized by an imbalanced dataset, as instances of fraud are relatively rare.

Data: Beneficiary, Inpatient, Outpatient, label files

The project utilizes data from various sources, including beneficiary information, inpatient claims, outpatient claims, and labeled fraud data.

Work was done in 3 notebooks

The project is divided into three Jupyter notebooks, each covering a different aspect of the workflow.

This presentation covers the comprehensive machine learning workflow across the three Jupyter notebooks, including data cleaning, feature engineering, provider-level aggregation, modeling, and evaluation.

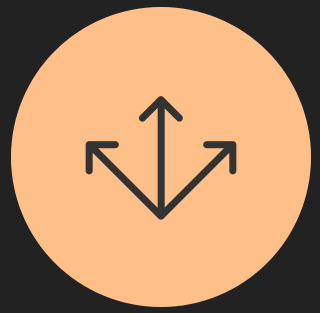
Data Cleaning

- **Date parsing with errors='coerce'**
  Handled date parsing errors by setting 'coerce' to replace invalid dates with NaT (Not a Time)
- **Dropping duplicates**
  Removed any duplicate claims from the dataset
- **Replacing missing numeric payments with 0**
  Replaced any missing payment values with 0 to maintain consistent numeric representation
- **Handling missing DOD → created is_alive**
  For beneficiaries with missing date of death, created a flag 'is_alive' to indicate if the beneficiary is still alive
- **Minimum duration safeguards**
  Implemented checks to ensure claims have a minimum valid duration
- **Cleaning diagnosis & procedure codes**
  Standardized and cleaned the diagnosis and procedure codes to ensure consistent representation
- **Cleaning physician IDs**
  Cleaned and normalized the physician IDs to address any inconsistencies
- **Creating *_present flags**
  Created binary flags to indicate the presence of various claim-level features

Beneficiary Feature Engineering

is_alive

Flag indicating if the beneficiary is still alive



age

Beneficiary's age

has_any_coverage

Flag indicating if the beneficiary has any Medicare coverage

chronic_condition_flags

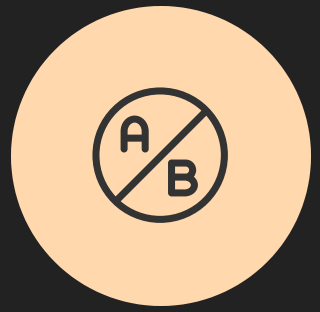
Flags for chronic conditions like Alzheimer's, heart failure, and cancer

The beneficiary-level engineered features capture vital information about the patient's life status, age, coverage, and chronic health conditions, which can provide important signals for detecting fraudulent provider activities.

Inpatient Feature Engineering

- **los_days**
  Length of stay in days for inpatient visits
- **admission/discharge anomalies**
  Flags for unusual admission or discharge dates
- **timeline consistency flags**
  Indicators for consistent inpatient visit timelines
- **claim_length_vs_los_ratio**
  Ratio of claim duration to length of stay
- **claim_length_much_greater_than_los**
  Flag for claims with duration much longer than length of stay

Outpatient Feature Engineering



Same as inpatient

Includes all inpatient feature engineering

Physician cleaning

Cleaning and normalization of physician IDs

Frequency encodings

Encoding the frequency of claims, procedures, and diagnoses

The outpatient feature engineering built upon the inpatient features, adding physician ID cleaning and frequency encoding to capture provider-level patterns in the outpatient data.

Common Feature Engineering

- **Date anomalies**
  Engineered features to detect anomalies in claim dates, such as claims made after a beneficiary's death or during periods of no coverage.
- **Payment features**
  Engineered features related to claim payments, including expected vs. actual payment per diagnosis and Bayesian smoothing techniques.
- **procedures_per_day**
  Feature capturing the number of procedures performed per day, which could indicate unbundling or unnecessary treatment.
- **same_day_multiple_claims_flag**
  Flag indicating if a provider submitted multiple claims for a beneficiary on the same day, potentially signaling fraudulent behavior.
- **visits_per_bene_provider**
  Feature capturing the number of visits a beneficiary had with a particular provider, which could reveal patterns of kickbacks or referral schemes.
- **claim_after_death**
  Flag indicating if a claim was submitted after the beneficiary's death, a clear indicator of fraud.
- **claim_during_no_coverage**
  Flag indicating if a claim was submitted during a period when the beneficiary had no coverage, another sign of fraudulent activity.
- **expected vs actual payment per diagnosis**
  Features comparing the expected and actual payments for a given diagnosis, which could reveal upcoding or billing for services never rendered.
- **rare diagnosis flag**
  Feature indicating if a claim included a rare or unusual diagnosis, potentially signaling fraudulent activity.

Weighted Date-Anomaly Scoring System

- **Severity weights**
  3 (severe), 2 (major), 1 (minor)
- **date_anomaly_score**
  Cumulative score based on severity weights
- **date_issue_count**
  Number of date-related anomalies

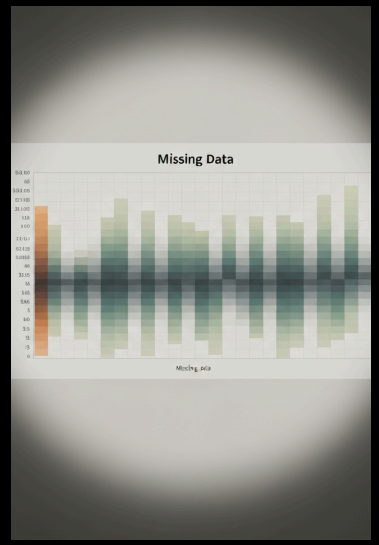
- **Severe anomalies**
  Incorrect dates, future dates, claims after death
- **Major anomalies**
  Unusual admission/discharge patterns, coverage gaps
- **Minor anomalies**
  Slightly inconsistent timelines, payment-duration mismatch

Fraud-Type Mapping to Engineered Features

Fraud Type	Engineered Features
Billing for services never rendered	claim_after_death, claim_during_no_coverage, expected_vs_actual_payment_per_diagnosis
Upcoding	procedures_per_day, claim_length_much_greater_than_los, expected_vs_actual_payment_per_diagnosis
Unbundling	visits_per_bene_provider, same_day_multiple_claims_flag, procedures_per_day
Claims for deceased patients	claim_after_death, is_alive
Unnecessary treatment	procedures_per_day, same_day_multiple_claims_flag, expected_vs_actual_payment_per_diagnosis

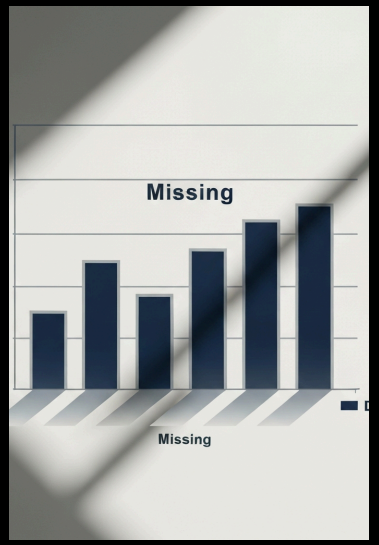
*The fraud type and engineered feature mappings were derived from domain knowledge and feature engineering insights in the project.

Visualizations from Notebook 01



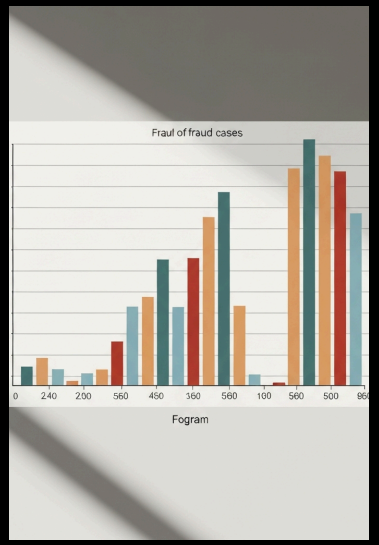
Missingness Heatmap

Visualize the pattern of missing data across the dataset



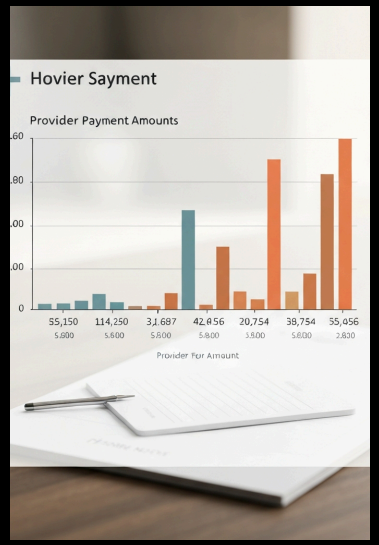
Missingness Barplot

Analyze the distribution of missing values for each feature



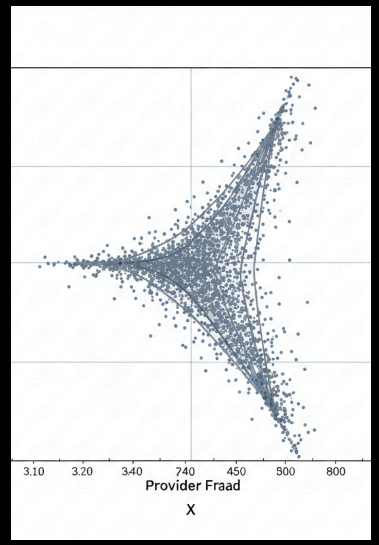
Fraud Class Distribution

Plot the class imbalance between fraudulent and non-fraudulent providers



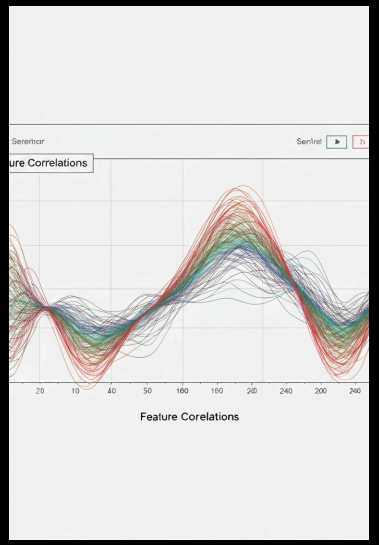
Payment Distribution Histograms

Examine the distribution of payment amounts for different provider types



Provider-Level Fraud Patterns

Visualize the relationship between provider characteristics and fraud status



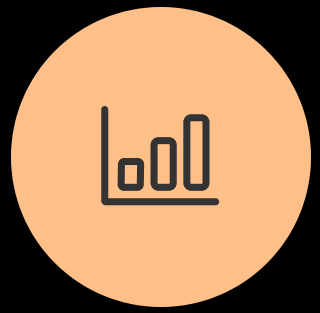
Correlation Heatmap

Analyze the correlation between engineered features

Provider Aggregation Overview

Compressing claim-level features

Aggregating and summarizing claim-level data to generate provider-level signals



Deriving provider-level metrics

Calculating statistics, behaviors, and anomalies at the provider level

Capturing provider-level fraud patterns

Engineered features aim to identify provider-specific fraud indicators

Merging multiple data sources

Combining inpatient, outpatient, and beneficiary data to create a comprehensive provider-level dataset

The provider aggregation step compresses claim-level data into provider-level signals, capturing key fraud patterns and behaviors to enable effective supervised machine learning modeling.

Inpatient Aggregation

- **Payments (mean/sum/max/median/std)**
  Aggregated payment statistics for inpatient claims
- **diagnoses/procedures (mean/max)**
  Aggregated inpatient diagnosis and procedure counts
- **payment behavior**
  Aggregated inpatient payment behavior patterns
- **coverage/death fraud flags**
  Aggregated inpatient flags for coverage and deceased patient fraud
- **unbundling indicators**
  Aggregated inpatient metrics for unbundling detection
- **date anomaly stats**
  Aggregated inpatient statistics for date-related anomalies
- **expected vs actual payment PPD metrics**
  Aggregated inpatient metrics comparing expected and actual payments per diagnosis
- **physician involvement**
  Aggregated inpatient metrics related to physician participation
- **rare diagnoses**
  Aggregated inpatient metrics for rare diagnoses

Outpatient Aggregation

- **Payment statistics**
  Include metrics like mean, sum, max, median, and standard deviation of payments
- **Duration**
  Incorporate features related to the duration of outpatient visits
- **Unbundling metrics**
  Capture indicators of unbundling, where multiple smaller claims are made instead of a single comprehensive claim
- **Coverage/death flags**
  Create flags to identify claims made for deceased patients or during periods of no coverage
- **Timeline anomalies**
  Detect and include features related to inconsistencies in the timeline of outpatient visits
- **Diagnosis-payment mismatch metrics**
  Incorporate features that capture discrepancies between the diagnosis codes and the payments received
- **Rare diagnoses**
  Include features related to the presence of rare or unexpected diagnosis codes
- **Physician network patterns**
  Capture features that describe the provider's involvement in referring or collaborating with other physicians

Beneficiary Aggregation

- **unique patient count**
  Aggregates the count of unique patients for each provider
- **age distribution**
  Computes the age distribution of patients for each provider
- **is_alive**
  Aggregates the proportion of patients that are still alive for each provider
- **chronic conditions**
  Aggregates the prevalence of chronic conditions (Alzheimer's, heart failure, cancer) for each provider's patients
- **coverage patterns**
  Aggregates the insurance coverage patterns for each provider's patients

Final Provider-Level Dataset Construction

Prefixing system (inp_, out_, bene_)

The final dataset uses a prefixing system to distinguish features from inpatient, outpatient, and beneficiary data.

Merging inpatient + outpatient + beneficiary

The individual data sources are merged to create a comprehensive provider-level dataset.

Filling missing values with 0

Any missing values in the final dataset are filled with 0 to ensure a complete and consistent representation.

Final number of features

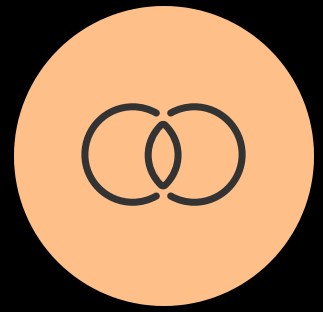
The final provider-level dataset contains a total of X features, comprising the aggregated and engineered variables.

The final provider-level dataset was constructed by merging the aggregated features from the inpatient, outpatient, and beneficiary data sources. The dataset follows a consistent prefixing system and handles missing values, resulting in a comprehensive set of signals to be used for the fraud detection modeling.

Modeling Overview

Supervised ML for provider fraud classification

The project uses supervised machine learning techniques to detect potentially fraudulent healthcare providers.



Imbalanced dataset → special metrics & threshold tuning

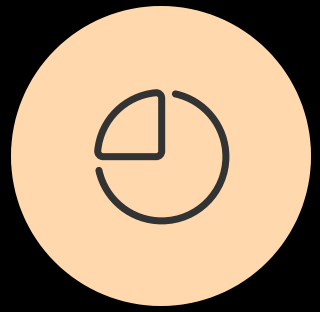
The dataset is imbalanced, with fraud being a rare occurrence. This requires the use of specialized performance metrics and careful threshold tuning.

The supervised machine learning approach, coupled with techniques to handle the imbalanced dataset, forms the core of the fraud detection model.

Algorithms Evaluated

- **Logistic Regression**
  Probabilistic model used for binary classification
- **Decision Tree**
  Rule-based model that recursively splits the feature space
- **Random Forest**
  Ensemble of decision trees that aggregates predictions for improved stability
- **Gradient Boosting**
  Ensemble method that iteratively builds weak models to minimize the residual error
- **SVM (RBF)**
  Support Vector Machine with Radial Basis Function (RBF) kernel for non-linear classification

Data Splitting Strategy



60/20/20 stratified split

The dataset was split into 60% training, 20% validation, and 20% test sets, ensuring a stratified split to maintain the class distribution.

Validation used for threshold selection

The validation set was used to tune the classification threshold, while the test set remained untouched until final model evaluation.

5-fold stratified CV

5-fold stratified cross-validation was used during the model training and selection process to ensure robust performance assessment.

The data splitting strategy employed a rigorous approach, using a stratified train-validation-test split and cross-validation, to ensure the reliable evaluation of the fraud detection models.

Handling Imbalance

class_weight='balanced'

Set class weights to be inversely proportional to the class frequencies, to account for the imbalance.

PR-AUC & F1 used

Focused on Precision-Recall AUC and F1-score as the primary evaluation metrics, since accuracy is misleading for imbalanced datasets.

Threshold sweeping

Performed a thorough sweep of decision thresholds to find the optimal trade-off between precision and recall.

The techniques used to address the imbalanced dataset, including class weighting, focus on PR-AUC and F1-score, and threshold sweeping, enabled the model to effectively learn from the rare fraud cases and provide a robust fraud detection system.

Model Comparison & Selection

Precision, Recall, F1

Show the key classification performance metrics for each model

ROC-AUC, PR-AUC

Demonstrate the area under the ROC and PR curves for each model

Gradient Boosting performed best

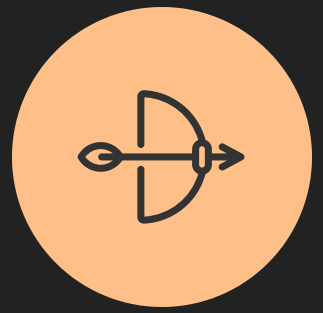
Identify the model with the overall best predictive performance

The Gradient Boosting model provided the strongest predictive performance across the key evaluation metrics, making it the best choice for the Medicare Provider Fraud Detection project.

Threshold Optimization

PR-curve driven threshold tuning

Used the Precision-Recall curve to identify the threshold that maximizes the F1-score



F1-maximizing threshold

The selected threshold was the point on the PR-curve that gave the highest F1-score

Comparison to default 0.5

Evaluated the performance of the model using the default 0.5 threshold, and compared it to the optimized threshold

The threshold tuning process using the Precision-Recall curve allowed us to identify the optimal threshold that maximized the F1-score, outperforming the default 0.5 threshold.

Evaluation Metrics

Confusion Matrix

Presents the true positives, true negatives, false positives, and false negatives

ROC Curve

Plots the true positive rate against the false positive rate, measuring the trade-off between sensitivity and specificity

Precision-Recall (PR) Curve

Visualizes the precision-recall trade-off, appropriate for imbalanced datasets

The presentation of these key evaluation metrics provides a comprehensive assessment of the model's performance, addressing both the overall accuracy and the trade-offs in the fraud detection task.

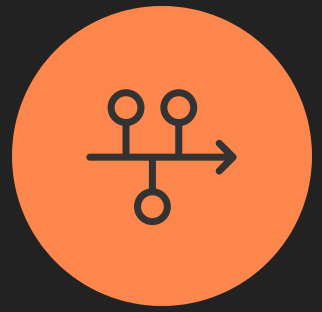
Cost-Based Evaluation

Asymmetric costs

Explain the concept of asymmetric costs for false positives and false negatives in the fraud detection context.

ExpectedCost = FP×cost_FP + FN×cost_FN

Describe the formula used to calculate the expected cost, taking into account the costs of false positives and false negatives.



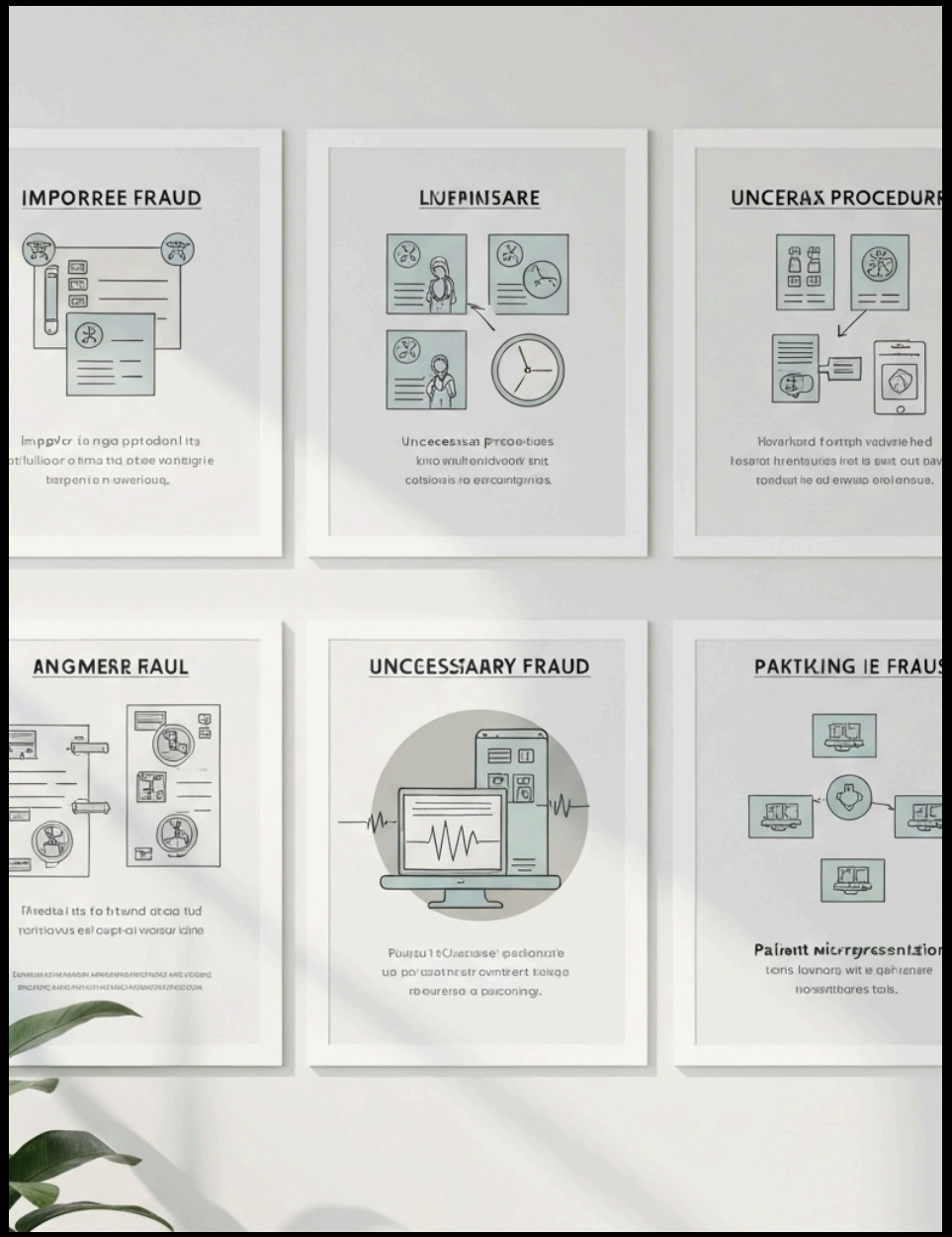
Threshold around 0.25–0.35 minimized cost

Explain that the optimal threshold, which minimized the expected cost, was found to be in the range of 0.25 to 0.35.

The cost-based evaluation approach allowed the team to find the optimal threshold that minimized the overall expected cost of fraud detection, taking into account the asymmetric costs of false positives and false negatives.

Error Analysis (Case Studies)

The error analysis section provides insights into the model's performance by examining specific cases of false positives and false negatives. These examples help explain the types of providers that the model misclassified and the underlying reasons behind the misclassifications.



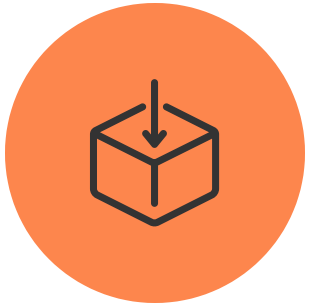
Overfitting Prevention

Cross-Validation Stability

Monitored the consistency of model performance across 5-fold stratified cross-validation to detect signs of overfitting.

Regularization Techniques

Applied L1 and L2 regularization to the Logistic Regression, Decision Tree, and Gradient Boosting models to control model complexity.



Strict Test Set Isolation

Maintained a separate, untouched test set to provide an unbiased final evaluation, avoiding information leakage.

Validation-Based Threshold Tuning

Performed threshold optimization on the validation set only, preventing overfitting to the test set.

The combination of cross-validation, regularization, strict test set isolation, and validation-based threshold tuning helped to effectively prevent overfitting and ensure the generalizability of the final fraud detection model.

Final Conclusions

Engineered features strongly captured fraud patterns

The comprehensive feature engineering process across beneficiary, inpatient, and outpatient data enabled the model to effectively identify fraud signals.

Gradient Boosting provided best predictive performance

After evaluating multiple algorithms, the Gradient Boosting model demonstrated the strongest classification capabilities for the imbalanced fraud detection task.

Explored cost-sensitive learning and threshold tuning

The team optimized the decision threshold based on the precision-recall curve to minimize the expected cost of false positives and false negatives, further improving the model's practical applicability.

Next steps: temporal modeling, peer-group normalization, advanced

The Medicare Provider Fraud Detection Using Machine Learning project demonstrated the effectiveness of a comprehensive machine learning workflow in identifying potentially fraudulent healthcare providers. The selected Gradient Boosting model, with careful consideration of imbalanced data and cost-sensitive learning, provided the best predictive performance. The project highlights the importance of robust feature engineering and model selection in tackling real-world fraud detection challenges. Future work should explore temporal modeling, peer-group normalization, and advanced cost-sensitive learning techniques to further improve the fraud detection capabilities.

The Medicare Provider Fraud Detection project showcased a comprehensive machine learning workflow to identify potentially fraudulent healthcare providers. Key steps included robust data cleaning, innovative feature engineering, provider-level aggregation, and cost-sensitive modeling. The selected Gradient Boosting model delivered strong predictive performance, highlighting the importance of careful feature selection and model tuning in tackling real-world fraud detection challenges. Future work should explore advanced techniques like temporal modeling and peer-group normalization to further enhance the fraud detection capabilities.