

Lesson 28: Class Imbalance

Data Science with Python – BSc Course

Data Science Program

BSc Course

45 Minutes

Previously on L27...

Last Time: Measuring Quality

- Confusion matrix: TP, FP, FN, TN
- Precision, Recall, F1-score
- ROC curve and AUC
- Threshold selection

This Time: When 99% Fails

- What if 99.9% of data is ONE class?
- Why accuracy becomes meaningless
- Techniques to find the rare events
- Cost-aware decision making



L27 gave us the tools to measure – L28 shows why standard metrics fail on imbalanced data

Learning Objectives

The Problem: Fraud occurs in 0.1% of transactions. Default in 2% of loans. How do we train models when positive cases are extremely rare?

After this lesson, you will be able to:

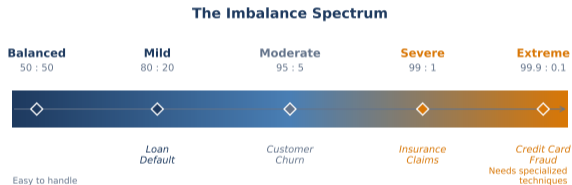
1. **Diagnose** imbalanced datasets and explain why accuracy misleads
2. **Apply** SMOTE to create synthetic minority samples
3. **Use** class weights to rebalance training without resampling
4. **Evaluate** models fairly with PR curves and cost-based thresholds

Finance Application: Fraud detection, default prediction, rare event modeling

The Imbalance Spectrum

Imbalance Is Everywhere in Finance

- Most real-world classification problems are imbalanced
- The harder the imbalance, the more specialized our approach must be

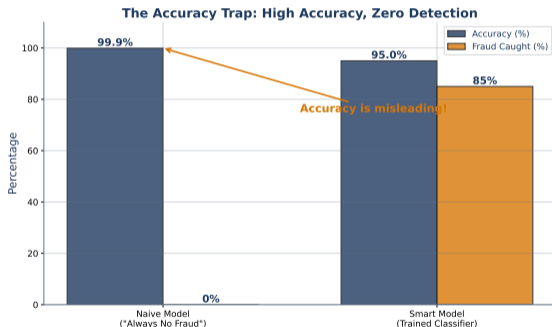


Key insight: Different imbalance levels require different techniques

Why Standard Models Fail

The Needle-in-a-Haystack Problem

- Classify everything as “hay” → right 99.99% of the time
- But you **never find the needle** – which was the whole point!

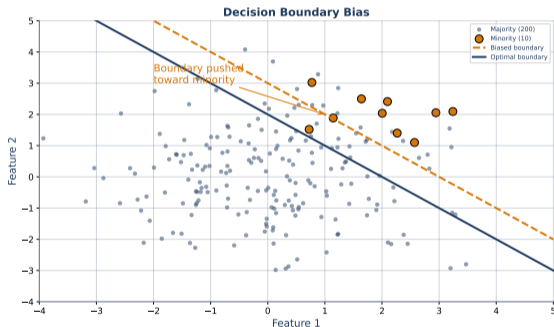


A model predicting “no fraud” always gets 99.9% accuracy – but catches zero fraud

Decision Boundary Bias

Why Standard Models Fail

- The boundary shifts toward the minority class to minimize total errors
- Result: minority samples are consistently misclassified



Standard classifiers push boundary toward minority class to minimize total errors

Three Solution Categories

How Do We Fix This?

1. **Resampling:** Change the data
 - Oversample minority (SMOTE) or undersample majority
2. **Reweighting:** Change the loss function
 - Make minority errors cost more (`class_weight`)
3. **Evaluation:** Change the metric
 - PR curves, F1-score, cost-based thresholds instead of accuracy

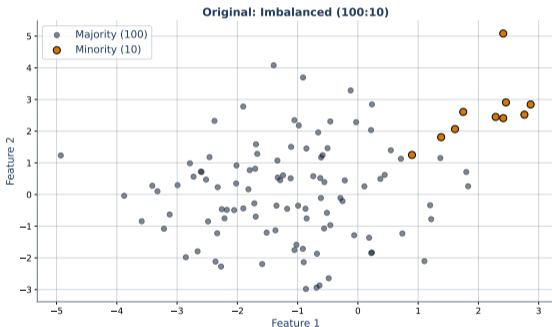
Best practice: Combine all three for robust rare-event classification.

We will cover all three approaches – each alone is often not enough

Sampling: Original Imbalanced Data

Starting Point: 100:10 Class Ratio

- Majority class dominates training
- Model learns to predict majority class by default

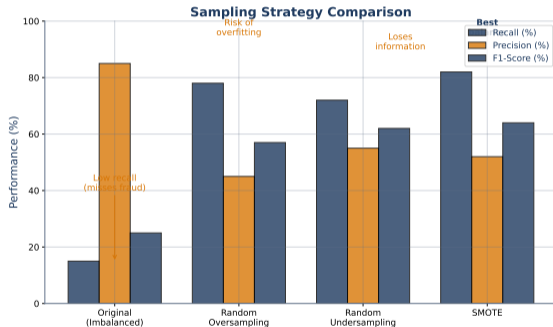


Goal: Rebalance training data while preserving test data distribution

Oversampling vs Undersampling

Two Simple Approaches

- **Oversampling:** Duplicate minority samples (risk: overfitting)
- **Undersampling:** Remove majority samples (risk: losing information)

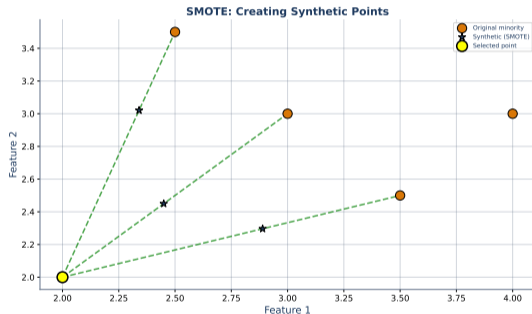


Trade-off: Oversampling risks overfitting, undersampling loses information

SMOTE: The Smart Solution

Instead of Copying, CREATE Synthetic Samples

- Interpolate between nearest neighbors: $x_{new} = x + \alpha(nn - x)$
- Key rule: ONLY on training data, NEVER test data

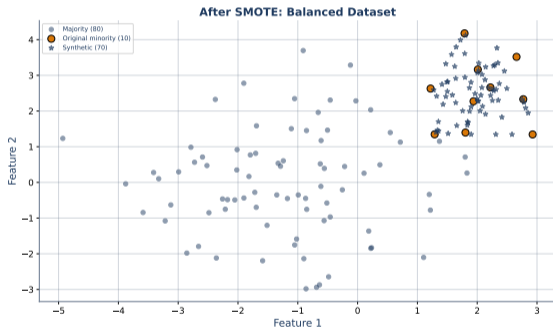


SMOTE generates points along lines connecting minority class neighbors

After SMOTE: Balanced Dataset

SMOTE Result

- Synthetic points fill the minority region naturally
- More variety than duplication – reduces overfitting

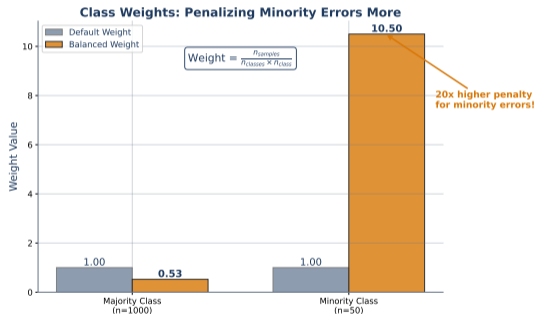


Synthetic samples fill the minority class region without exact duplication

Class Weights: No Resampling Needed

Multiply Minority Errors by Weight

- Use `class_weight='balanced'` in sklearn
- Model “feels” 10x more pain for minority errors

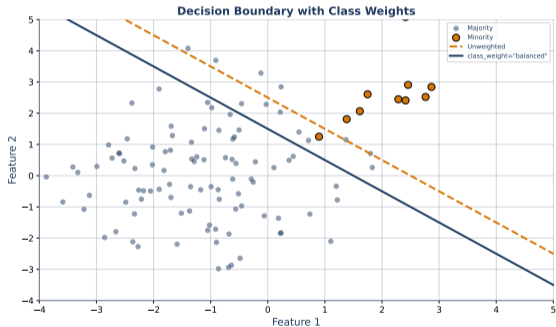


Weight = $n_{samples} / (n_{classes} \times n_{samples,c})$ – each CLASS contributes equally

Weights Shift the Boundary

How Weights Change Classification

- Higher minority weight pushes boundary away from minority class
- More minority samples get correctly classified (higher recall)

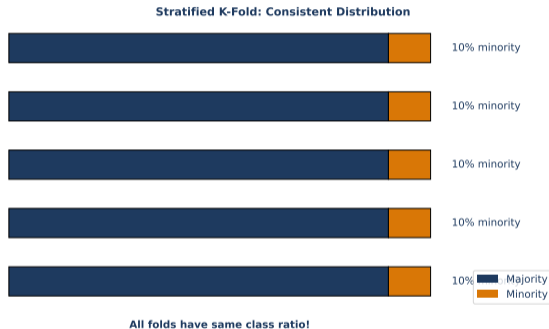


Higher minority weight shifts decision boundary away from minority class

Stratified Cross-Validation

Problem: Random splits may put all rare events in one fold

Solution: StratifiedKFold preserves class ratios in each fold

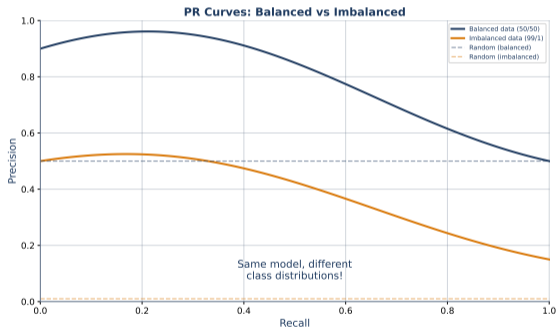


Each fold maintains the same class ratio as the full dataset

PR Curves Beat ROC for Imbalanced Data

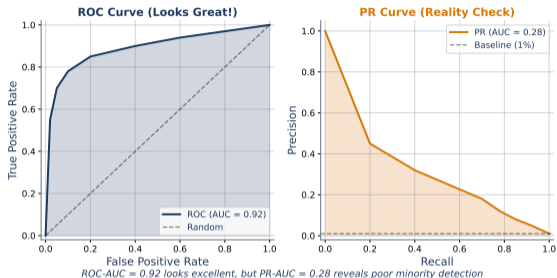
Why PR Curves Matter More

- ROC can look good even with poor minority detection
- PR curve focuses on positive class performance



High AUC-ROC but low AUC-PR = model fails on what matters

PR vs ROC Comparison

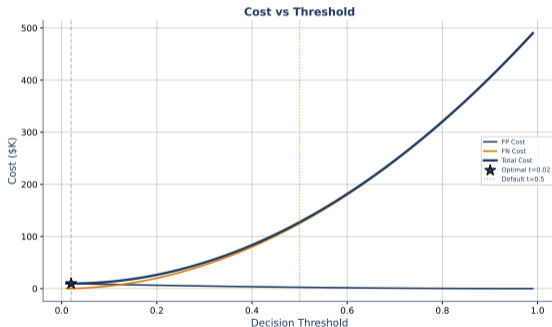


ROC is optimistic for imbalanced data – always check PR curve

Cost-Sensitive Threshold Selection

Threshold Based on Business Costs

- FN costs \$1000 (missed fraud) vs FP costs \$10 (investigation)
- Optimal: $t^* = \frac{C_{FP}}{C_{FP} + C_{FN}} \approx 0.01$ – flag almost everything

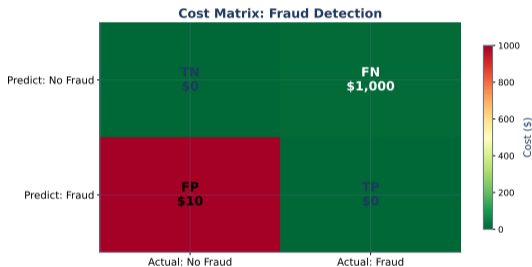


Optimal threshold minimizes expected total cost, not error rate

Expected Profit Calculation

Optimize for Profit, Not Just Metrics

- Expected cost = $FN \times C_{FN} + FP \times C_{FP}$
- Small recall improvement \rightarrow millions in recovered fraud



$$\text{Expected Cost} = FN \times 1,000 + FP \times 10$$

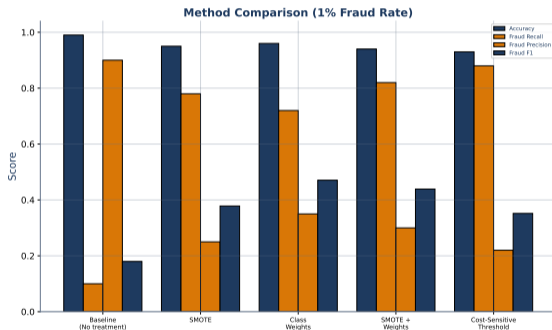
Missing fraud (FN) is 100x more costly than false alarm (FP)

$$\text{Expected Cost} = FN \times \text{Cost}_{FN} + FP \times \text{Cost}_{FP}$$

Method Comparison on Fraud Data

Putting It All Together

- Real-world fraud rates: 0.1–1% positive
- Compare: baseline, SMOTE, class weights, combined

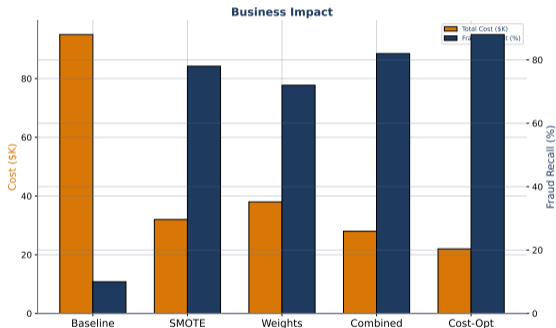


No single method is always best – test on your specific data

Business Impact Analysis

Why This Matters in Practice

- Small recall improvement can mean millions in recovered fraud
- Business stakeholders care about dollars, not F1-scores



Small recall improvement can mean millions in recovered fraud

Complete Fraud Detection Pipeline

Recommended Pipeline Components

1. **Resampling:** SMOTE in imblearn Pipeline (training only)
2. **Classifier:** RandomForest with `class_weight='balanced'`
3. **Validation:** StratifiedKFold cross-validation
4. **Metrics:** F1-score and Average Precision (not accuracy!)
5. **Threshold:** Tune based on business costs

Key Insight: Combine SMOTE + class weights for “double protection” against imbalance.

Full pipeline: SMOTE + class weights + stratified CV + PR evaluation

Five Rules for Imbalanced Classification

1. **Don't trust accuracy** – use F1, Precision, Recall, PR-AUC
2. **Combine techniques** – SMOTE + class weights + stratified CV
3. **Tune the threshold** – default 0.5 is rarely optimal
4. **Consider business costs** – optimize for profit, not just metrics
5. **Validate properly** – never resample test data

When to Use Each:

- Slight imbalance (80/20): Class weights alone
- Moderate (95/5): SMOTE + weights
- Extreme (99.9/0.1): Consider anomaly detection

Industry insight: Ensemble methods (XGBoost, LightGBM) work well for fraud

Hands-On Exercise (25 min)

Task: Build a Fraud Detection Model

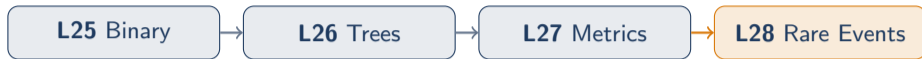
1. Create synthetic imbalanced data (1% fraud rate)
2. Train baseline model – observe the accuracy trap
3. Apply SMOTE and retrain – compare recall
4. Use `class_weight='balanced'` – compare to SMOTE
5. Plot Precision-Recall curve for best model

Deliverable: Comparison table of recall/precision across methods.

Extension: Implement cost-sensitive threshold selection

Module Wrap-Up: The Classification Journey

Module 5 Arc: From Simple Decisions to Rare Events



- L25: Binary decisions (logistic regression)
- L26: Non-linear boundaries (decision trees)
- L27: Measuring quality (confusion matrix, ROC, PR)
- L28: Handling rare events (SMOTE, weights, costs)

Next Module: What if there are NO labels? Welcome to Unsupervised Learning (L29–L32).

From Fisher's iris flowers to fraud detection – the complete classification toolkit