Lesson 28: Class Imbalance

Data Science with Python – BSc Course

45 Minutes

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?

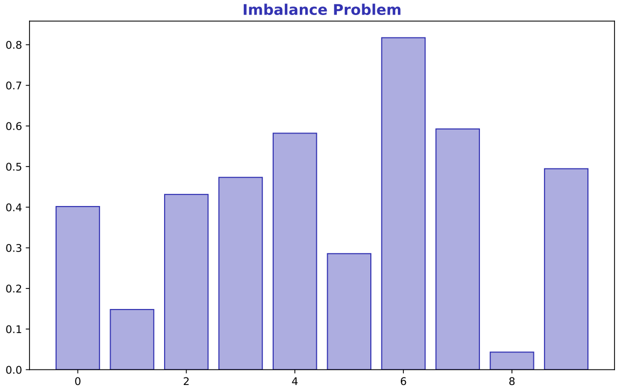
After this lesson, you will be able to:

- Identify and diagnose imbalanced datasets
- Apply SMOTE and other oversampling techniques
- Use class weights to rebalance training
- Evaluate models fairly on imbalanced data

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

Why Standard Models Fail

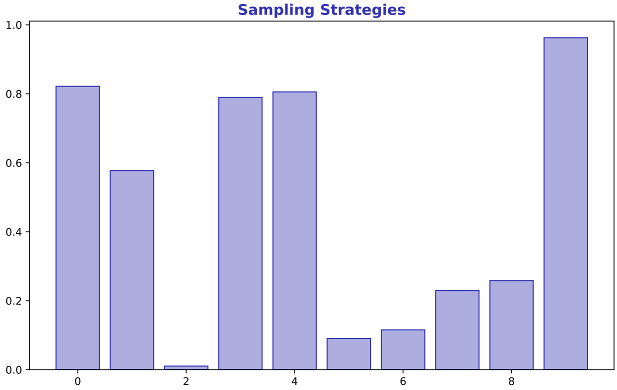
- Model learns to predict majority class (easy 99% accuracy)
- Minority class is ignored because errors are "cheap"



Check: If your model predicts same class 95%+ of the time, you have a problem

Rebalancing the Training Data

- Undersampling: remove majority class samples (loses information)
- Oversampling: duplicate or synthesize minority samples

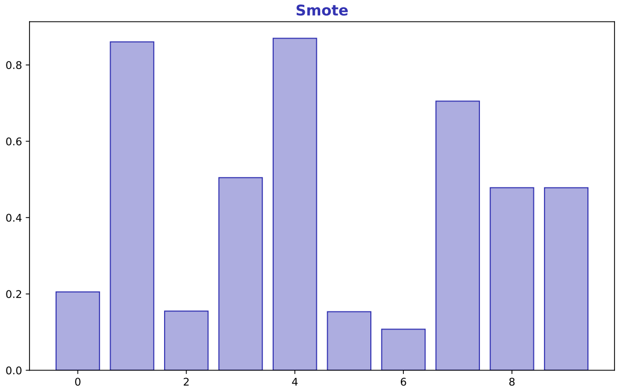


Rule: Only resample training data, never test data – test must reflect reality

SMOTE

Synthetic Minority Over-sampling Technique

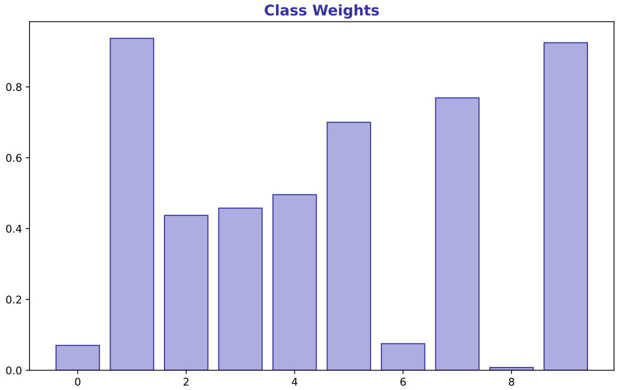
- Creates synthetic samples by interpolating between minority neighbors
- `from imblearn.over_sampling import SMOTE`



SMOTE creates new points along lines between existing minority samples

Rebalancing Without Resampling

- Give higher weight to minority class errors in loss function
- sklearn: `class_weight='balanced'` or custom weights

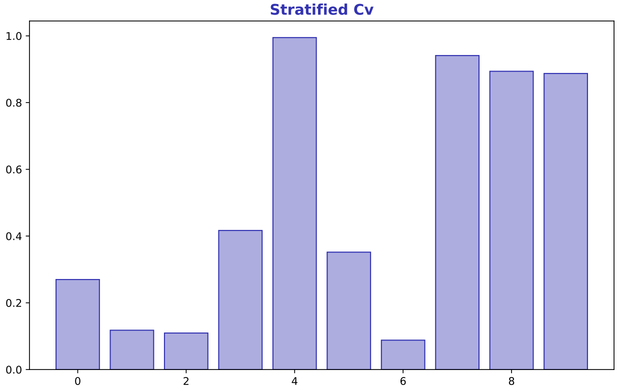


Balanced weights: inversely proportional to class frequency

Stratified Cross-Validation

Preserving Class Proportions

- Regular CV may put all rare events in one fold
- Stratified CV ensures each fold has same class ratio

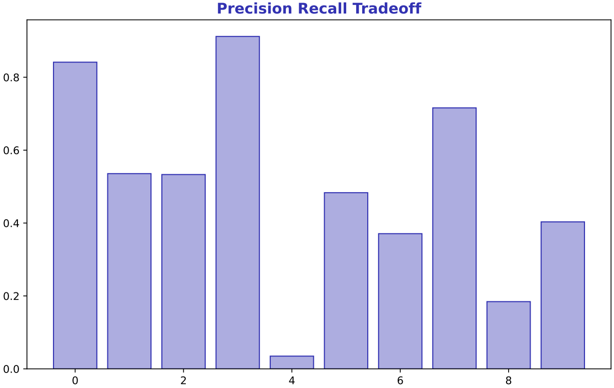


Always use `StratifiedKFold` **for imbalanced classification**

Precision-Recall Trade-off

Finding the Right Balance

- Precision-Recall curve better than ROC for imbalanced data
- Area under PR curve (AP) is more informative than AUC

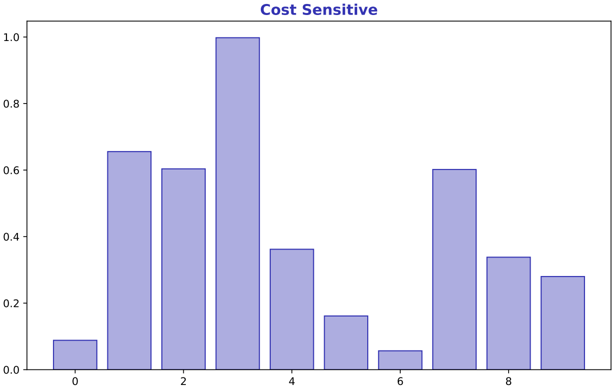


PR curve focuses on positive class – what we care about in imbalanced problems

Cost-Sensitive Learning

Encoding Business Costs

- FN (missed fraud) costs \$1000, FP (false alarm) costs \$10
- Optimal threshold minimizes expected total cost

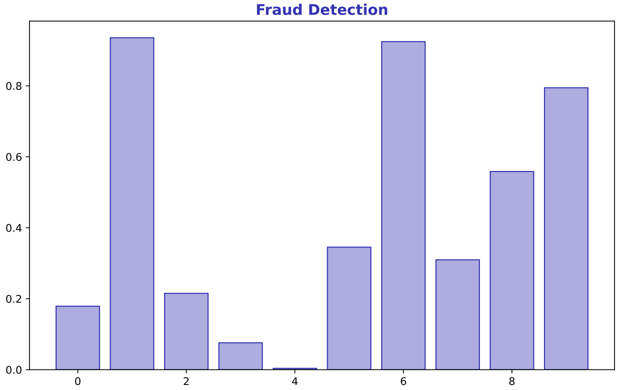


Formula: Expected Cost = FN $\times$ Cost$_{FN}$ + FP $\times$ Cost$_{FP}$

Fraud Detection Example

Putting It All Together

- Real-world fraud rates: 0.1-1% positive
- Pipeline: SMOTE + class weights + stratified CV + PR evaluation



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 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

Lesson Summary

Problem Solved: We can now handle imbalanced datasets common in finance (fraud, default, rare events).

Key Takeaways:

- Accuracy misleads on imbalanced data – use precision/recall
- SMOTE creates synthetic minority samples
- Class weights: alternative to resampling
- Always use stratified CV and PR curves

Next Lesson: KMeans Clustering (L29) – unsupervised learning begins

Memory: SMOTE = Synthetic samples. Class weights = penalize minority errors more.