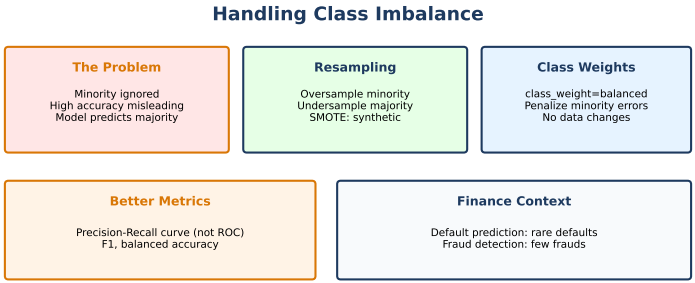
Lesson 28 Summary: Class Imbalance

Data Science with Python – Key Concepts

Data Science Program

Class Imbalance Overview



class_weight="balanced" | *SMOTE()* | *RandomUnderSampler()*

Imbalanced data is common in fraud, default, and churn prediction

The Problem

Why imbalance hurts:

- **Majority bias:** Model predicts the common class
- **Misleading accuracy:** 99% majority = 99% accuracy
- **Lost minority:** Rare but important events missed

In fraud detection, the fraud class is typically less than 1%

Oversampling

Add more minority samples:

- **Random:** Duplicate minority samples
- **SMOTE:** Generate synthetic samples
- **Caution:** May cause overfitting

Always oversample training data only, not test data

Undersampling

Remove majority samples:

- **Random:** Randomly remove majority
- **Tomek links:** Remove ambiguous samples
- **Caution:** Loses potentially useful information

Works well when majority class has redundancy

Class Weights

Algorithm-level adjustment:

- **Method:** Penalize errors on minority more heavily
- **sklearn:** class_weight='balanced'
- **Advantage:** No data modification needed

Class weights are often the simplest effective solution

SMOTE

Synthetic Minority Oversampling:

- **How:** Create synthetic samples between neighbors
- **Benefit:** Adds diversity, not just duplicates
- **Library:** imbalanced-learn (imblearn)

SMOTE is the most popular oversampling technique

Appropriate Metrics

Beyond accuracy:

- **Precision-Recall curve:** Better than ROC for imbalance
- **F1 score:** Balances precision and recall
- **Balanced accuracy:** Average of per-class recalls

Use PR-AUC instead of ROC-AUC for highly imbalanced data

Threshold Adjustment

Lower the decision threshold:

- **Default 0.5:** Too high for rare class
- **Lower threshold:** Catches more minority (more recall)
- **Tradeoff:** Also increases false positives

Optimize threshold based on business costs

Best Practices

Recommended workflow:

- **Start simple:** Try class_weight='balanced' first
- **Then SMOTE:** If class weights don't work
- **Always:** Evaluate on original (imbalanced) test set

Never resample your test set

Quick Reference

Essential Techniques:

Method	Code
Class weights	`LogisticRegression(class_weight='balanced')`
Random oversample	`RandomOverSampler().fit_resample(X, y)`
SMOTE	`SMOTE().fit_resample(X, y)`
Undersample	`RandomUnderSampler().fit_resample(X, y)`
Balanced acc.	`balanced_accuracy_score(y, pred)`

Install imbalanced-learn: pip install imbalanced-learn