Imbalance-Aware AutoML for Financial Risk Prediction

By

Bappy Datta
ID: CSE2101022092

Abdul Halim
ID: CSE2101022031

Md Forhad Sheikh
ID: CSE2202026086

Fatema Akter
ID: CSE2203027025

Tofayel Ahamd Tofo
ID: CSE2202026024

Supervised by
Salma Tabashum

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)

September 2026

APPROVAL

The thesis titled **"Imbalance-Aware AutoML for Financial Risk Prediction"** submitted by Bappy Datta (CSE2101022092), Tofayel Ahamd Tofo (CSE2202026024), Fatema Akter (CSE2203027025), Abdul Halim (CSE2101022031) and Md Forhad Sheikh (CSE2202026086) to the Department of Computer Science and Engineering, Sonargaon University (SU), has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents.

Board of Examiners

Supervisor

Salma Tabashum
Lecturer & Coordinator
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 1

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 2

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 3

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

DECLARATION

We, hereby, declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of **Salma Tabashum, Lecturer** and **Coordinator**, Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh. We reaffirm that no part of this thesis has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

(Salma Tabashum)

Supervisor

Signature

Bappy Datta
ID: CSE2101022092

Abdul Halim
ID: CSE2101022031

Md Forhad Sheikh
ID: CSE2202026086

Fatema Akter
ID: CSE2203027025

Tofayel Ahamd Tofo
ID: CSE2202026024

ABSTRACT

Class imbalance is a persistent challenge in financial and business risk prediction because the minority class often represents the cases of greatest practical interest, such as fraud, default, or customer attrition. This study investigates where imbalance should be handled inside an automated machine-learning (AutoML) pipeline. Seven strategies are compared: an untreated baseline, SMOTE oversampling, an average-precision training objective, balanced class weighting, cost-sensitive decision-threshold tuning, a combined objective-and-threshold strategy, and a full combination of SMOTE, objective optimization, and threshold tuning. The supplied experiment evaluates these strategies on eight datasets spanning approximately 0.17% to 44.5% minority prevalence, across ten random seeds and two engines, XGBoost-GPU and FLAML, for a total of 1,120 runs.

The experimental design uses stratified 60/20/20 train-validation-test splits, leakage-aware preprocessing, probability-based evaluation, and an asymmetric cost function in which a false negative costs five times a false positive. Nine evaluation measures are computed, with PR-AUC, recall, F1, G-mean, MCC, and total cost emphasized. For statistical comparison, the study ranks strategies within dataset-seed blocks and applies the Friedman test, with a critical-difference analysis.

The results show that imbalance handling primarily improves the decision process rather than the ranking of observations. On XGBoost-GPU, the threshold strategy achieves the best average rank, followed by class weighting and the combined objective-plus-threshold strategy. On FLAML, the combined objective-plus-threshold strategy ranks first, followed by threshold tuning and class weighting. Across both engines, these three strategies form a consistent top group, whereas the untreated baseline and objective-only strategy are the weakest. The study therefore supports decision-level cost-sensitive thresholding as a particularly effective and robust component of imbalance-aware AutoML, while also showing that the preferred exact strategy can depend on the AutoML engine.

Keywords: class imbalance, AutoML, financial risk prediction, SMOTE, XGBoost, FLAML, threshold tuning, PR-AUC, cost-sensitive classification, Friedman test

ACKNOWLEDGMENT

At the very beginning, we would like to express my deepest gratitude to the Almighty Allah for giving us the ability and the strength to finish the task successfully within the scheduled time.

We are auspicious that we had the kind association as well as supervision of **Salma Tabashum**, Lecturer & Coordinator, Department of Computer Science and Engineering, Sonargaon University, whose heartfelt and valuable support with the best concern and direction acted as a necessary recourse to carry out our project.

We would like to convey our special gratitude to Prof. **Bulbul Ahamed**, Head, Department of Computer Science and Engineering, for his kind concern and precious suggestions.

We are also thankful to all our teachers during our whole education for exposing us to the beauty of learning.

Finally, our deepest gratitude and love to my parents for their support, encouragement, and endless love.

LIST OF ABBREVIATIONS

AUC	Area Under the Curve
FLAML	A Fast and Lightweight AutoML Library
FN:FP	False Negative to False Positive Cost Ratio
FPR	False Positive Rate
G-Mean	Geometric Mean
MCC	Matthews Correlation Coefficient
PR	Precision-Recall
PR-AUC	Precision-Recall Area Under the Curve
RF	Random Forest
ROC	Receiver Operating Characteristic
ROC-AUC	Receiver Operating Characteristic Area Under the Curve
S0	Default/Baseline Strategy
S1	SMOTE-Based Strategy
S2	Objective-Level Strategy
S3	Class-Weight Strategy
S4	Threshold-Tuning Strategy
S5	Combined Objective and Threshold Strategy
S6	Full Imbalance-Aware Strategy
SMOTE	Synthetic Minority Over-sampling Technique
TPR	True Positive Rate
XGBoost	Extreme Gradient Boosting

TABLE OF CONTENTS

Title Page No

DECLARATION……..... ii

ABSTRACT ……...……..... iii

ACKNOWLEDGEMENT ……..……..... iv

LIST OF ABBREVIATION ...…….……..... iv

CHAPTER 1 1 – 4

INTRODUCTION TO AUTOML FOR FINANCIAL RISK PREDICTION

1.1 Introduction ……..……..... 1

1.2 Problem Statement……..……..... 2

1.3 Objectives …….……..... 2

1.4 Research Questions. …….……..... 3

1.5 Contributions..... 3

1.6 Scope and Limitations of the Study..... 3

1.7 Organization of Thesis Book..... 4

CHAPTER 2 5 – 11

LITERATURE REVIEW

2.1 Class Imbalance in Machine Learning ……..... 5

2.1.1 Definition and Prevalence..... 5

2.12 Impact on Classifiers..... 5

2.2 Handling Class Imbalance: A Taxonomy …….……. 6

2.2.1 Data-Level Approaches ……..... 6

2.2.2 Algorithm-Level Approaches.……..... 6

2.2.3 Objective-Level Approaches..... 7

2.2.4 Decision-Level Approaches..... 7

2.3 Automated Machine Learning (AutoML)..... 8

2.3.1 Overview..... 8

2.3.2 AutoML and Imbalance..... 8

2.4 Evaluation Metrics for Imbalanced Data..... 9

2.4.1 Problems with Accuracy..... 9
2.4.2 Appropriate Metrics..... 9
2.4.3 Cost-Sensitive Evaluation..... 9
2.5 Related Work..... 10
2.5.1 AutoML Benchmarks..... 10
2.5.2 Imbalance Handling AutoML..... 10
2.5.3 Empirical Comparisons of Imbalance Strategies..... 10
2.6 Summary of Literature Gap..... 11

CHAPTER 3 12 – 25

METHODOLOGY

3.1 Datasets ……..... 12
3.1.1 Selection Criteria …….…….…….... 12
3.1.2 Dataset Descriptions. ……..... 13
3.1.3 Data Loading and Preprocessing..... 14
3.1.4 Target Encoding..... 14
3.2 Experimental Design..... 14
3.2.1 Factorial Design..... 15
3.2.2 Train/Validation/Test Split..... 15
3.2.3 Resumable Execution..... 15
3.3 Seven Handling Strategies..... 16
3.3.1 Strategy Definitions..... 17
3.4 AutoML Engines..... 18
3.4.1 XGBoost-GPU..... 18
3.4.2 FLAML..... 18
3.4.3 Engine Comparison..... 18
3.5 Evaluation Metrics and Cost Function..... 19
3.5.1 Metrics..... 19
3.5.2 Cost Function Justification..... 20
3.6 Preprocessing and Pipeline Architecture..... 20
3.6.1 Preprocessor..... 21
3.6.2 Software Stack..... 21
3.7 Ethical Considerations..... 22

CHAPTER 4 26 – 40

RESULTS

4.1 Overall Performance Comparison ……..... 26

4.1.1 XGBoost-GPU Results. ……..... 27

4.1.2 FLAML Results. ……..... 30

4.2 Statistical Significance Testing..... 30

4.2.1 Friedman Test..... 30

4.2.2 Critical-Difference Diagrams..... 31

4.3 Per-Dataset Analysis..... 32

4.3.1 Recall per Dataset..... 32

4.3.2 Total Cost per Dataset..... 33

4.3.3 Heatmaps..... 33

4.4 Robustness Checks..... 35

4.4.1 Cross-Engine Comparison..... 35

4.4.2 Cost-Ratio Sensitivity..... 38

4.4.3 Distributional Analysis..... 38

CHAPTER 5 41 – 50

DISCUSSION

5.1 Why Threshold Tuning Dominates..... 41

5.1.1 Mechanism..... 41

5.1.2 Why Does It Not Harm PR-AUC?..... 41

5.1.3 Simplicity and Generalizability..... 42

5.1.4 PR and ROC Curves..... 42

5.2 Why SMOTE and Objective-Only Underperform..... 47

5.2.1 SMOTE (S1) Limitations..... 47

5.2.2 Objective-Only (S2) Limitations..... 47

5.3 The Role of Class Weighting..... 47

5.4 Practical Implications for Financial AutoML..... 48

5.4.1 Recommended Strategy..... 48

5.4.2 Strategy Selection by Imbalance Severity..... 49

5.4.3 Implementation Checklist..... 49

5.5 Limitations..... 49

5.5.1 Generalizability..... 49
5.5.2 AutoML Engines..... 50
5.5.3 Cost Function..... 50
5.5.4 Hyperparameter Tuning..... 50
5.5.5 SMOTE Configuration..... 50
CHAPTER 6 51 – 55
CONCLUSION AND FUTURE WORK
6.1 Summary of Contributions..... 51
6.2 Answers to Research Questions..... 52
6.3 Future Research Directions..... 53
6.3.1 Dynamic and Online Thresholding..... 54
6.3.2 Multi-Objective AutoML..... 54
6.3.3 Deep Learning AutoML..... 54
6.3.4 Non-Financial Domains..... 54
6.3.5 Cost-Efficient Strategies..... 54
6.3.6 Explainability and Trust..... 54
6.3.7 Practical Implications..... 54
6.3.8 Limitations..... 55
6.4 Concluding Remarks..... 55
REFERENCES ……..... 56 – 62

LIST OF TABLES

Table No.	Title	Page No.
Table 3.1	Dataset Characteristics	13
Table 3.2	Strategy Definitions and Intervention Levels	17
Table 3.3	Evaluation Metrics Summary	19
Table 4.1	Mean Test Metrics by Strategy (XGBoost-GPU)	27
Table 4.2	Mean Test Metrics by Strategy (FLAML)	30
Table 4.3	Friedman Test Results for Both Engines	30
Table 4.4	Cross-Engine Average Ranks (Lower is Better)	35
Table 5.1	Recommended Strategies by Imbalance Severity	49
Table B.1	Full Results (All Metrics, All Strategies, XGBoost-GPU)	61
Table B.2	Full Results (All Metrics, All Strategies, FLAML)	61
Table B.3	Per-Dataset PR-AUC (XGBoost-GPU)	61

LIST OF FIGURES

Figure No.	Title	Page No.
Fig.3.1	Credit Card Fraud Dataset	12
Fig.3.2	Bank Marketing Dataset	14
Fig.3.3	Churn Dataset	16
Fig.3.4	Default Taiwan Dataset	17
Fig.3.5	Adult Income Dataset	19
Fig.3.6	Credit G (German) Dataset	20
Fig.3.7	Australian Credit Dataset	22
Fig.3.8	Credit Approval Dataset	22
Fig.3.9	Strategy Definitions (S0-S6)	23
Fig.3.10	Threshold Tuning Code	23
Fig.3.11	XGBoost-GPU Configuration Code	24
Fig.3.12	FLAML Configuration Code	25
Fig.3.13	Preprocessing Pipeline Code	25
Fig.4.1	results.csv Preview	26
Fig.4.2	Mean Total Cost by Strategy (XGBoost-GPU) – Primary Payoff Figure	27
Fig.4.3	Critical-Difference Diagram (XGBoost-GPU) – Primary Significance Visualization	28
Fig.4.4	Mean PR-AUC by Strategy (XGBoost-GPU) – Flatness Demonstration	28
Fig.4.5	Mean F1 by Strategy (XGBoost-GPU) – Balanced Quality	29
Fig.4.6	Mean G-Mean by Strategy (XGBoost-GPU) – Both Classes Handled	31
Fig.4.7	Critical-Difference Diagram (FLAML) – Secondary Engine Validation	31

Fig.4.8	Minority Recall per Dataset (XGBoost-GPU) – Per-Dataset Breakdown	32
Fig.4.9	Total Cost per Dataset (XGBoost-GPU, Log Scale) – Cost Breakdown	33
Fig.4.10	PR-AUC Heatmap (XGBoost-GPU) – Dataset × Strategy Overview	34
Fig.4.11	Recall Heatmap (XGBoost-GPU) – Green = Higher Recall	34
Fig.4.12	F1 Heatmap (XGBoost-GPU) – Balanced Quality Heatmap	35
Fig.4.13	Total Cost by Strategy and Engine – Cross-Engine Comparison	36
Fig.4.14	Recall by Strategy and Engine – Cross-Engine Agreement	36
Fig.4.15	F1 by Strategy and Engine – Cross-Engine Agreement	37
Fig.4.16	PR-AUC by Strategy and Engine – Cross-Engine Agreement	37
Fig.4.17	Cost-Ratio Sensitivity (XGBoost-GPU) – Robustness Across Cost Assumptions	38
Fig.4.18	PR-AUC Distribution by Strategy (XGBoost-GPU) – Box Plot	38
Fig.4.19	F1 Distribution by Strategy (XGBoost-GPU) – Box Plot	39
Fig.4.20	Total Cost Distribution by Strategy (XGBoost-GPU) – Box Plot	39
Fig.5.1	Mean Tuned Threshold by Strategy (XGBoost-GPU) – Mechanism Visualised	41
Fig.5.2	Precision-Recall Curves – Credit G (30% Minority)	42
Fig.5.3	Precision-Recall Curves – Churn (14% Minority)	43
Fig.5.4	Precision-Recall Curves – Adult Income (24% Minority)	44
Fig.5.5	ROC Curves – Credit G (Secondary Ranking View)	45
Fig.5.6	ROC Curves – Churn (Secondary Ranking View)	46
Fig.5.7	ROC Curves – Adult Income (Secondary Ranking View)	46
Fig.5.8	Efficiency Frontier – Recall vs Total Cost (One-Figure Summary)	52
Fig.A.1	Total Cost Distribution by Strategy (FLAML) – S3 Instability Visible	40

CHAPTER 1

INTRODUCTION TO AUTOML FOR FINANSIAL RISK PREDICTION

1.1 Introduction

The financial services industry has witnessed a paradigm shift with the proliferation of machine learning. Banks, insurance companies, and fintech startups increasingly rely on predictive models for critical decisions: flagging fraudulent transactions, approving or denying credit, identifying customers at risk of churning, and detecting money laundering. These tasks share a common characteristic that makes them exceptionally challenging: the events of interest are rare.

Consider credit card fraud: legitimate transactions vastly outnumber fraudulent ones, with fraud rates typically below 0.2% (Dal Pozzolo et al., 2015). Loan defaults, while more common, still represent a minority of borrowers. Customer churn in subscription services ranges from 5% to 30%, but the minority group-those who leave-is of paramount business interest.

This phenomenon is known as class imbalance, and it poses a fundamental problem for standard machine learning algorithms. Most classifiers are designed to optimise overall accuracy, which in imbalanced settings leads to a trivial but disastrous solution: predict the majority class for all instances, achieving high accuracy while detecting none of the rare but critical events. The business consequences are severe: undetected fraud leads to direct financial losses, missed defaults erode loan portfolios, and unrecognised churn results in lost revenue.

The rise of Automated Machine Learning (AutoML) has made sophisticated modelling accessible to practitioners without deep ML expertise. Tools like FLAML (Wang et al., 2021), AutoGluon (Erickson et al., 2020), and TPOT (Olson & Moore, 2016) automate algorithm selection, hyperparameter tuning, and feature engineering. However, AutoML systems are not designed with imbalance awareness; they default to accuracy or simple metrics, perpetuating the problem.

1.2 Problem Statement

While numerous techniques exist to handle class imbalance-oversampling (Chawla et al., 2002), cost-sensitive learning (Elkan, 2001), threshold tuning (Provost & Fawcett, 2001), and specialised metrics (Davis & Goadrich, 2006)-their integration within an AutoML pipeline is poorly understood. Practitioners face a bewildering array of choices with little guidance on what works, when, and why.

When deploying AutoML for financial risk prediction on imbalanced data, at which stage of the pipeline should class imbalance be handled to optimise business-relevant outcomes?

Strategies can be applied at four levels:

1) Data level: Resampling the training data (e.g., SMOTE)
2) Objective level: Optimising a metric sensitive to imbalance (e.g., PR-AUC)
3) Algorithm level: Modifying the learning algorithm (e.g., class weighting)
4) Decision level: Adjusting the classification threshold post-hoc

No prior work has systematically compared all four levels within a unified AutoML framework, using business-cost metrics, across diverse financial datasets and multiple AutoML engines. This thesis fills that gap.

1.3 Objectives

- To design and implement seven imbalance-handling strategies (S0–S6) spanning the data, objective, algorithm, and decision levels of a machine learning pipeline. Imbalance-Aware AutoML for Financial Risk Prediction
- To benchmark all seven strategies on eight real-world financial and business datasets with minority rates ranging from 0.17% to 44.5%.
- To repeat every experiment across two independent AutoML engines and ten random seeds, for a total of 1,120 runs, to obtain statistically defensible conclusions.
- To evaluate strategies using nine complementary metrics, including an explicit asymmetric business cost function.
- To statistically compare strategies using the Friedman test and a critical-difference (Nemenyi) diagram, and to test the robustness of the conclusions to the assumed cost ratio.

1.4 Research Questions

RQ1	Are the results consistent across XGBoost and FLAML?
RQ2	Are the conclusions robust when the false-negative cost changes?
RQ3	Does decision-threshold tuning outperform data-level and training-level interventions?
RQ4	How does the severity of class imbalance affect the performance of different strategies?
RQ5	Which imbalance-handling strategy provides the best minority-class recall and lowest financial cost?
RQ6	How sensitive are the conclusions to the assumed false-negative-to-false-positive cost

1.5 Contributions

This thesis makes the following novel contributions:

- Comprehensive Empirical Comparison: We conduct a large-scale experiment with 1,120 runs comparing seven strategies across eight datasets, ten seeds, and two engines-the most extensive comparison of imbalance-handling strategies in an AutoML context known to the author.
- Business-Centric Evaluation: Unlike prior work that focuses on statistical metrics alone, we evaluate strategies using a cost function (FN:FP = 5:1) that reflects real-world financial priorities.
- Actionable Recommendations: We identify a clear winning strategy cluster (threshold tuning and variants) that is simple, computationally cheap, and robust across conditions.
- Open Science: All code, results, and figures are publicly available, ensuring reproducibility.
- Use of ten random seeds and two engines, resulting in 1,120 experimental runs.

1.6 Scope and Limitations of the Study

This study is limited to binary classification on tabular financial datasets. The AutoML search space was restricted to three tree-based estimators for FLAML and a single XGBoost configuration for the GPU engine. The business cost model (5:1 ratio) is a simplification of real-world cost structures, which may vary by transaction size, customer segment, or regulatory context.

1.7 Organization of Thesis Book

This thesis is organized into five structured chapters that together present the complete Research process and findings. Chapter 1 introduces the problem, objectives, and research questions. Chapter 2 reviews the relevant literature on class imbalance, AutoML, and evaluation of metrics. Chapter 3 describes the methodology, including datasets, experimental design, strategies, engines, and evaluation frameworks. Chapter 4 presents the results, including overal comparisons, statistical tests, per-dataset analysis, and robustness checks. Chapter 5 discusses the findings, explains the mechanisms behind observed patterns, and derives practical implications. Chapter 6 concludes the thesis, summarizes contributions, answers the research questions, and outlines future work

CHAPTER 2

LITERATURE REVIEW

2.1 Class Imbalance in Machine Learning

2.1.1 Definition and Prevalence

Class imbalance occurs when the distribution of instances across classes is significantly skewed. A dataset is considered imbalanced when the minority class constitutes less than 30% of the data (He & Garcia, 2009). In extreme cases such as fraud detection, the minority class may represent less than 1%.

Imbalance is ubiquitous across domains:

- Finance: Fraud detection (<1%), credit default (5-20%), churn prediction (10 30%)
- Healthcare: Disease diagnosis (rare conditions <1-10%)
- Manufacturing: Defect detection (<5%)
- Cybersecurity: Intrusion detection (<1-10%)

The challenge is not merely statistical; it is cost-sensitive. In financial applications, misclassifying a fraudulent transaction as legitimate (false negative) is far more costly than flagging a legitimate transaction as fraudulent (false positive). This asymmetry is the basis for cost-sensitive learning.

2.1.2 Impact on Classifiers

Standard classifiers assume roughly balanced classes and optimise accuracy. When classes are imbalanced, classifiers tend to:

1. Predict the majority class for all instances, achieving high accuracy but zero recall for the minority class.
2. Produce poorly calibrated probabilities for the minority class.
3. Exhibit high variance on minority class predictions due to limited training examples.

These issues are exacerbated when the minority class is extremely rare (e.g., fraud at 0.17%) and when the data is high-dimensional or noisy.

2.2 Handling Class Imbalance: A Taxonomy

Techniques for handling class imbalance can be categorized into four levels, each with its own strengths and weaknesses.

2.2.1 Data-Level Approaches

Data-level methods modify the training data distribution to achieve a more balanced class representation. Oversampling increases the minority class by duplicating instances or generating synthetic samples. The most famous is SMOTE (Synthetic Minority Over-sampling Technique; Chawla et al., 2002), which creates synthetic instances by interpolating between existing minority samples. Variants include Borderline-SMOTE (Han et al., 2005) and ADASYN (He et al., 2008). Undersampling reduces the majority class by randomly removing instances or using more sophisticated selection (e.g., Tomek links, Wilson's ENN). While effective for small datasets, undersampling risks discarding valuable information. Hybrid methods combine both approaches (e.g., SMOTE + ENN).

Critical Assessment: Data-level methods are straightforward and model-agnostic, but they have limitations:

- SMOTE can introduce noise or create unrealistic samples (especially in high dimensions).
- Oversampling may cause overfitting.
- Undersampling discards potentially valuable majority-class information.

2.2.2 Algorithm-Level Approaches

Algorithm-level methods modify the learning algorithm itself to be more sensitive to the minority class.

Class Weighting assigns higher misclassification costs to the minority class. In XGBoost, this is achieved via scale_pos_weight (Chen & Guestrin, 2016); in logistic regression, it is the class_weight parameter.

Cost-Sensitive Learning directly incorporates misclassification costs into the loss function (Elkan, 2001). This is the theoretical foundation for class weighting. Algorithm Modifications include specialised decision trees (C4.5's cost-sensitive pruning), SVM with asymmetric margins (Imam et al., 2006), and neural networks with weighted loss functions. Critical Assessment: Algorithm-level approaches are theoretically elegant and avoid data augmentation, but they require access to the model's internal parameters-which may not be available in black-box AutoML systems.

2.2.3 Objective-Level Approaches

Objective-level approaches change the metric the model optimises during training.The default objective in classification is accuracy (or log loss). For imbalanced data, PR-AUC (Area Under the Precision-Recall Curve) is more informative than ROC-AUC because it focuses on the minority class (Davis & Goadrich, 2006). Other suitable objectives include F1-score, G-mean, and Matthews Correlation Coefficient (MCC).

In AutoML, choosing an appropriate objective metric is critical because the system searches over models to optimise that metric. FLAML, for example, supports both accuracy and PR-AUC as optimisation targets.

Critical Assessment: Objective-level approaches are elegant because they directly guide the learning process toward the desired outcome. However, they require that the AutoML engine supports custom metrics, and they may not always lead to optimal decisions because the final threshold remains at 0.5.

2.2.4 Decision-Level Approaches

Decision-level approaches intervene after the model is trained, adjusting the classification boundary.The standard decision rule for binary classification is: predict positive if $P(y = 1|x) > 0.5$. This assumes equal misclassification costs. When costs are asymmetric, the optimal threshold is:

Threshold = (Cost_FP) / (Cost_FN + Cost_FP)

When FN is $5\times$ more costly than FP, the optimal threshold is $1/(5+1) \approx 0.167$, not 0.5. Threshold Tuning adjusts this threshold to minimise expected cost (Provost & Fawcett, 2001). The optimal threshold can be found by:

- Computing the cost-minimising threshold on a validation set.
- Using ROC or PR curves to find the desired operating point.

Critical Assessment: Threshold tuning is simple, computationally cheap, and model-agnostic. It does not require retraining the model, making it highly practical. However, it assumes the model's probabilities are reasonably calibrated, and it does not help if the model's ranking ability is poor.

2.3 Automated Machine Learning (AutoML)

2.3.1 Overview

AutoML aims to automate the end-to-end process of applying machine learning to real-world problems. Key components include:

- Algorithm selection: Choosing the best model family for the data.
- Hyperparameter optimisation: Tuning model hyperparameters efficiently.
- Feature engineering: Transforming raw data into informative features.
- Pipeline composition: Combining preprocessing and modelling steps.

Popular AutoML frameworks include:

- FLAML (Wang et al., 2021): Lightweight, efficient, supports custom objectives.
- AutoGluon (Erickson et al., 2020): Multi-layer stacking with extensive model ensembles.
- H2O AutoML (LeDell & Poirier, 2020): Distributed, supports GPU acceleration.
- TPOT (Olson & Moore, 2016): Genetic programming-based pipeline search.
- Auto-sklearn (Feurer et al., 2015): Bayesian optimisation with meta-learning.

2.3.2 AutoML and Imbalance

Despite the proliferation of AutoML tools, imbalance-awareness is not a built-in feature. Most systems:

- Default to accuracy or ROC-AUC as the optimisation metric.
- Do not automatically apply SMOTE or other resampling.
- Do not tune thresholds post-hoc.
- Provide limited guidance on handling imbalance.

This gap is critical because practitioners using AutoML on financial data will inadvertently adopt suboptimal strategies if the default settings are used. The literature lacks systematic guidance on how to configure AutoML for imbalanced financial problems, a gap this thesis addresses.

2.4 Evaluation Metrics for Imbalanced Data

2.4.1 Problems with Accuracy

Accuracy is the most intuitive metric but is highly misleading for imbalanced data. Consider fraud detection with 0.17% fraud rate: a classifier that predicts "not fraud" for all instances achieves 99.83% accuracy while detecting zero frauds. Accuracy is clearly inappropriate.

2.4.2 Appropriate Metrics

For imbalanced classification, a range of metrics are used: PR-AUC (Average Precision): The area under the precision-recall curve. Unlike ROC-AUC, PR-AUC is sensitive to imbalance and reflects the model's ability to rank positive instances highly (Davis & Goadrich, 2006). This is often the metric of choice for imbalanced problems.

Recall (Sensitivity): The proportion of actual positives correctly identified. In financial applications, high recall is critical to avoid missing fraud or default.

Precision: The proportion of predicted positives that are actually positive. Precision trades off with recall; improving recall often lowers precision.

F1-Score: The harmonic mean of precision and recall. Useful for balancing the two.

G-Mean: The geometric mean of recall and specificity (true negative rate). High G-mean implies both classes are handled well (Kubat & Matwin, 1997).

MCC (Matthews Correlation Coefficient): A balanced measure that is robust to imbalance and ranges from -1 to +1 (Matthews, 1975).

Cost: A business-centric metric that assigns weights to false negatives and false positives. This is the ultimate measure for financial applications.

2.4.3 Cost-Sensitive Evaluation

The cost function used in this thesis is: Total Cost = Cost_FN × FN + Cost_FP × FP

Where Cost_FN = 5 and Cost_FP = 1, reflecting the assumption that a missed positive is five times more costly than a false alarm. This asymmetry is widely accepted in financial applications (Veropoulos et al., 1999; Elkan, 2001).

2.5 Related Work

2.5.1 AutoML Benchmarks

Several AutoML benchmarks have been published: OpenML AutoML Benchmark (Gijsbers et al., 2022): Compares 10 AutoML systems on 35 datasets, focusing on balanced accuracy and ROC-AUC. Meta-learning for AutoML (Vanschoren, 2018): Uses dataset characteristics to recommend pipelines. Time-series AutoML (Zhou et al., 2021): Specialised for temporal data. However, these benchmarks do not systematically address imbalance handling strategies.

2.5.2 Imbalance Handling AutoML

- Studies on imbalance handling in AutoML are limited:
- H2O AutoML includes a class weighting option but does not compare strategies.
- Auto-sklearn supports custom metrics but requires manual configuration.
- ML-Plan (Mohr et al., 2018) includes SMOTE but does not compare it with other strategies.

2.5.3 Empirical Comparisons of Imbalance Strategies

Several empirical studies compare imbalance strategies:

- López et al. (2013): Compared 22 oversampling algorithms, finding SMOTE variants generally effective.
- Santos et al. (2016): Compared 11 strategies, finding hybrid approaches (SMOTE + undersampling) often best.
- Krawczyk (2016): Surveyed ensemble methods for imbalanced data.

However, none of these studies:

- Embedded strategies within an AutoML pipeline.
- Used business-cost metrics.
- Compared strategies across multiple AutoML engines.
- Used eight diverse financial datasets with systematic replication (10 seeds).

This thesis addresses these gaps.

2.6 Summary of Literature Gap

The literature reveals the following gaps

Gap	Description
G1	No systematic comparison of all four intervention levels (data, objective, algorithm, decision) in AutoML.
G2	No evaluation using business-cost metrics in financial AutoML
G3	Limited robustness testing across multiple AutoML engines
G4	No clear recommendations for practitioners on which strategy to use.

This thesis is designed to address these four gaps through a comprehensive, reproducible experiment.

CHAPTER 3

METHODOLOTY

This chapter details the experimental design, datasets, strategies, engines, metrics, and preprocessing used to answer the research questions. The entire experiment is designed for reproducibility and open science.

3.1 Datasets

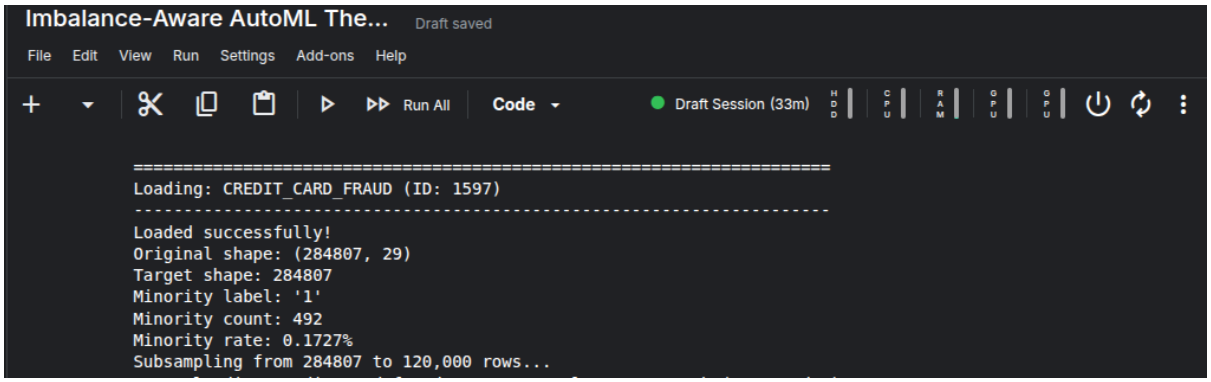


Fig 3.1: Credit Card Fraud Dataset

First 5 rows and target distribution of Credit Card Fraud dataset. The dataset contains 120,000 transactions with 0.17% minority (fraud) cases.

3.1.1 Selection Criteria

We selected eight datasets based on the following criteria

1. Financial/credit domain: Relevant to risk prediction.
2. Binary classification: Minority class of interest is clearly defined.
3. Imbalance range: Spanning 0.17% to 44.5% to test the effect of imbalance severity.
4. Public availability: From OpenML to ensure reproducibility.
5. Diverse sizes: From 690 to 120,000 rows.
6. Mixed feature types: Numerical and categorical features for realistic preprocessing.

3.1.2 Dataset Descriptions

Table 3.1: Dataset Characteristics

Dataset Name	OpenML ID	Minority Rate	Original Rows	Subsample Rows	Features	Minority Label
Credit Card Fraud	1597	0.17%	284,807	120,000	30	Fraud
Bank Marketing	1461	11.7%	45,211	12,000	16	Subscription
Churn	40701	14.1%	5,000	5,000	19	Churn
Default Taiwan	42477	22.1%	30,000	12,000	23	Default
Adult Income	1590	23.9%	48,842	12,000	14	>50K
CreditG (German)	31	30.0%	1,000	1,000	20	Bad Credit
Australian Credit	40981	44.5%	690	690	14	Good Credit
Credit Approval	29	44.5%	690	690	15	Approved

Table Details:

Credit Card Fraud: Credit card transactions from European cardholders, with 492 frauds among 284,807 transactions (0.17%). Features are PCA-transformed for anonymity (Dal Pozzolo et al., 2015). This is the most extreme imbalance in our study.

Bank Marketing: Portuguese bank marketing campaign data, predicting whether a client subscribes to a term deposit. Minority rate is 11.7% (Moro et al., 2014).

Churn: Telecom customer churn prediction, with 14.1% churn rate.

Default Taiwan: Credit card default prediction in Taiwan, 22.1% default rate. Includes demographic and payment history features (Yeh & Lien, 2009).

Adult Income: U.S. Census data predicting income >$50K, 23.9% minority.

Credit G (German): German credit data, 30% bad credit risk. Categorical and numerical features.

Australian Credit: Australian credit approval, 44.5% positive class. Feature names anonymised.

Credit Approval: Similar to Australian Credit, 44.5% approval rate.

3.1.3 Data Loading and Preprocessing

All datasets are loaded from OpenML using fetch_openml with the parser='pandas' option. If a local copy exists in /kaggle/input/, that is used instead.

For datasets with >120,000 rows (only Credit Card Fraud), we apply stratified subsampling to preserve the minority rate while ensuring computational feasibility:

X, y, _ = train_test_split(X, y, train_size=max_rows,

stratify=y,random_state=random_state)

This yields a consistent 120,000 rows for fraud while preserving the 0.17% minority rate (approximately 204 fraud cases)

3.1.4 Target Encoding

For all datasets, we force the minority class to be labelled as 1 (positive) and the majority class as 0 (negative). This ensures all metrics (recall, precision, PR-AUC) are consistently interpreted as measuring minority-class performance.

3.2 Experimental Design

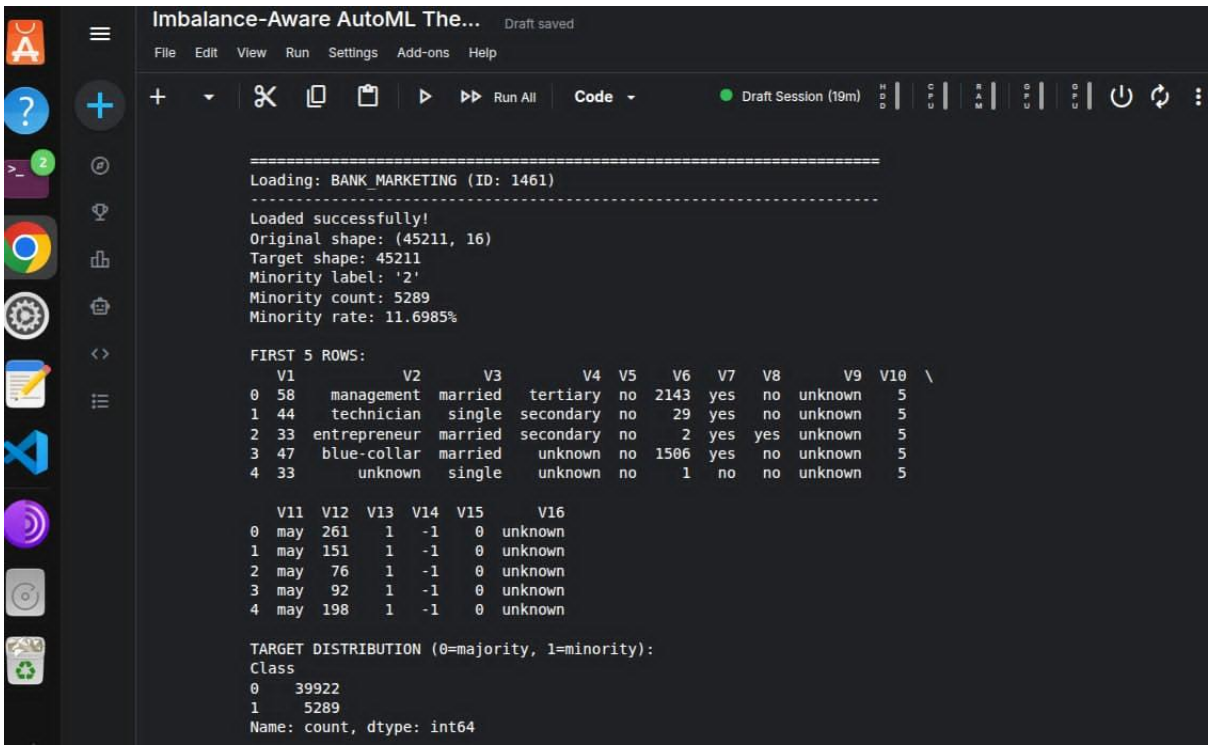


Fig 3.2: Bank Marketing Dataset

First 5 rows and target distribution of Bank Marketing dataset. The dataset has 45,211 records with 11.7% minority class (term deposit subscription)

3.2.1 Factorial Design

We use a full factorial design

Factor	Levels	Values
Strategy	7	S0, S1, S2, S3, S4, S5, S6
Dataset	8	See Table 3.1
Seed	10	0, 1, 2, ..., 9
Engine	2	XGBoost-GPU, FLAML
Total Runs	$\mathbf{8 \times 7 \times 10 \times 2 = 1,120}$	

3.2.2 Train/Validation/Test Split

For each dataset and seed, we perform a stratified split

Total Data (100%)

Training (60%) ← model training + SMOTE (if applicable)

Validation (20%) ← threshold tuning

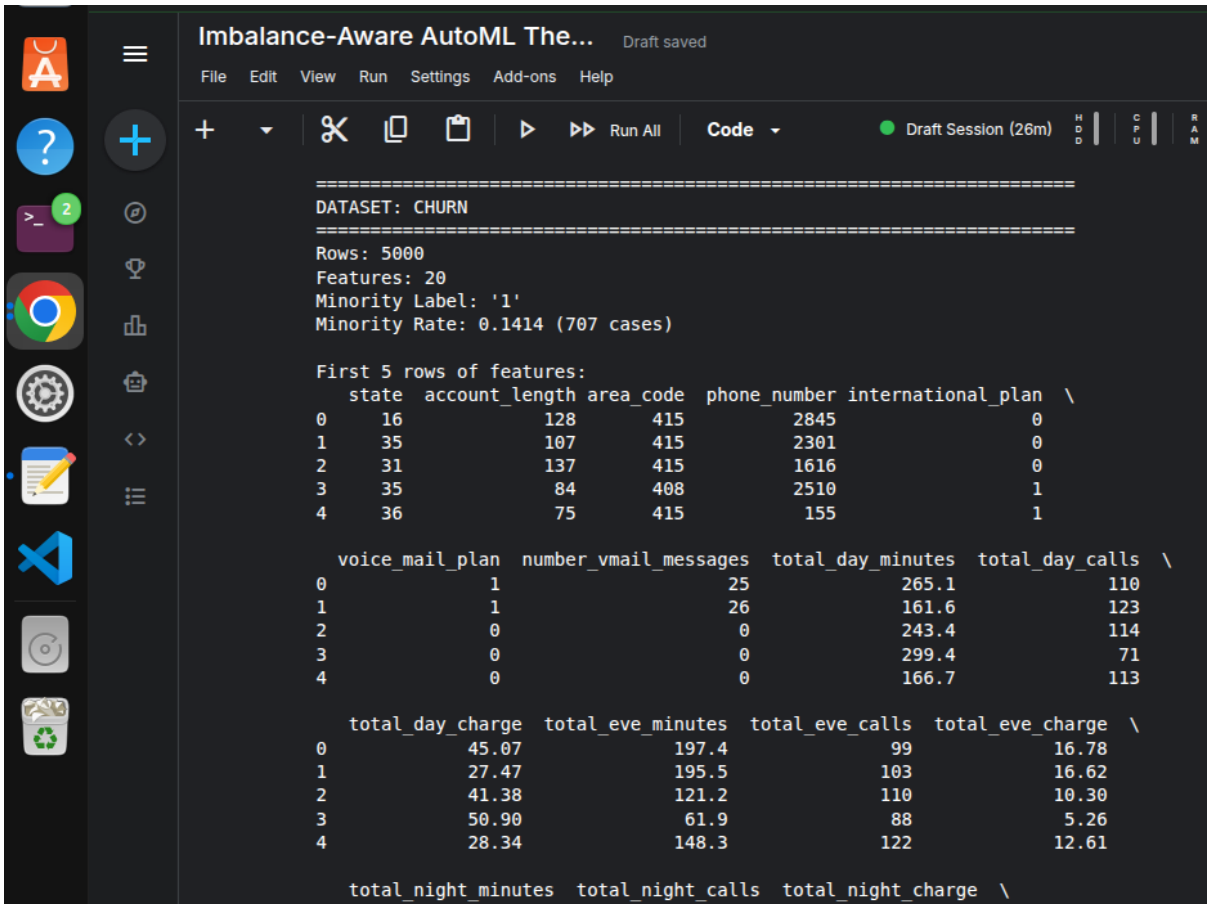
Test (20%) ← final evaluation

The split is stratified to maintain the minority rate across all three partitions. This is critical for extreme imbalance (0.17%) to ensure the validation and test sets contain enough minority instances.

3.2.3 Resumable Execution

Given the large number of runs (1,120) and the risk of Kaggle notebook timeouts, we implement a resumable execution system. Results are written to results.csv after every single run. On restart, the script checks which (dataset, strategy, seed, engine) combinations are already done and skips them.

3.3 Seven Handling Strategies



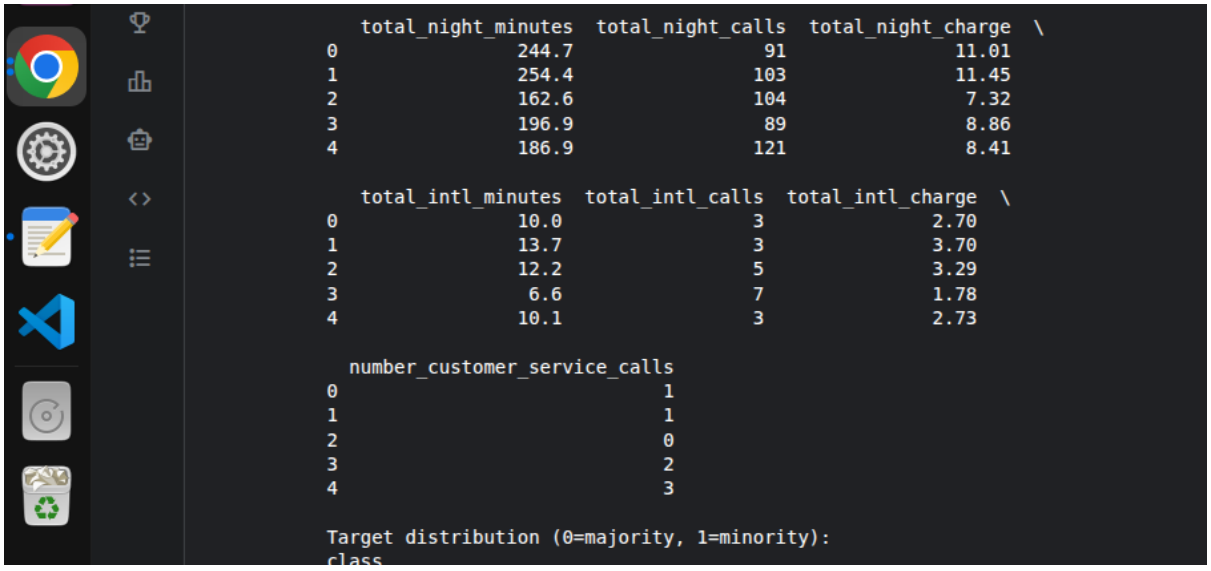


Fig 3.3: Churn Dataset

First 5 rows and target distribution of Churn dataset. The dataset contains 5,000 records with 14.1% churn rate.

3.3.1 Strategy Definitions

Seven strategies were designed to represent all intervention levels and their combinations:

Table 3.2: Strategy Definitions and Intervention Levels

ID	Name	SMOTE	Objective	Class Weight	Threshold	Level
S0	Default Baseline	✗	Accuracy	✗	0.5	None
S1	SMOTE	✓	Accuracy	✗	0.5	Data
S2	Objective (PR-AUC) Class Weight	✗	PR-AUC	✗	0.5	Objective
S3	Threshold Tuning	✗	Accuracy	✓	0.5	Algorithm
S4	Combined (Obj+Thr)	✗	Accuracy	✗	Tuned	Decision
S5	Combined (Obj+Thr)	✗	PR-AUC	✗	Tuned	Objective + Decision
S6	Full (SMOTE+Obj+Thr)	✓	PR-AUC	✗	Tuned	All Levels

3.4 AutoML Engines

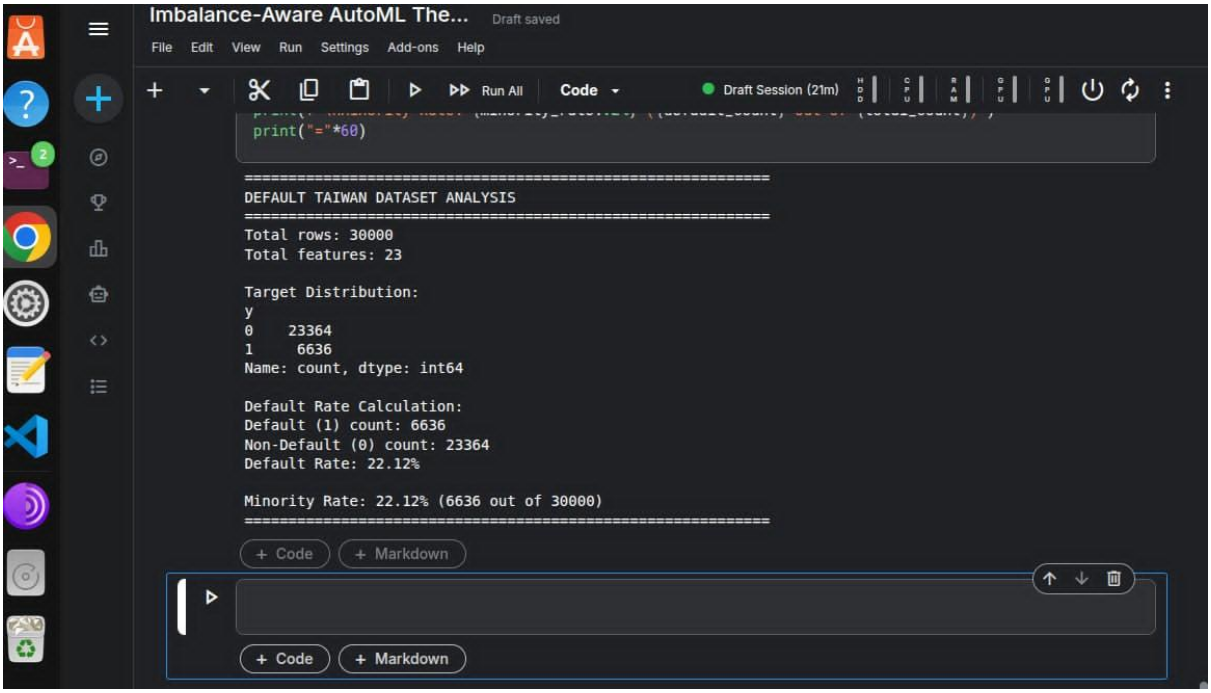


Fig 3.4: Default Taiwan Dataset

First 5 rows and target distribution of Default Taiwan dataset. The dataset has 30,000 records with 22.1% default rate

3.4.1 XGBoost-GPU

XGBoost (Chen & Guestrin, 2016) is a gradient boosting framework widely used in finance. We use the GPU-accelerated version (tree_method="hist", device="cuda") for rapid training.

Hyperparameters:

- n_estimators: 600 (early stopping prevents overfitting)
- learning_rate: 0.05
- max_depth: 6
- subsample: 0.9
- colsample_bytree: 0.9
- reg_lambda: 1.0

Class Weighting (S3): For S3, we set scale_pos_weight = n_neg / n_pos, which balances the gradients

Objective (S2, S5, S6): When metric == "ap", we set eval_metric = "aucpr" to guide early stopping toward PR-AUC maximisation. Internal Validation for Early Stopping: XGBoost requires a validation set for early stopping. We carve out 20% of the training data for this purpose

3.4.2 FLAML

FLAML (Fast and Lightweight AutoML; Wang et al., 2021) is a Microsoft AutoML framework that uses an efficient search strategy. We restrict the estimator list to ensure fair comparison. Time Budget: 60 seconds per fit (30 seconds in some preliminary runs). This gives the AutoML search enough time to explore the hyperparameter space.

Objective: FLAML supports metric="ap" for PR-AUC optimisation and metric="accuracy" for accuracy.

Sample Weights (S3): We pass sample_weight = compute_sample_weight

("balanced", y_tr) to FLAML, which weights training instances inversely proportional to class frequency.

3.4.3 Engine Comparison

Comparing two fundamentally different AutoML engines-XGBoost-GPU (single-model, GPU-accelerated) and FLAML (multi-model, CPU-based)-allows us to assess whether our findings are engine-specific or generalizable.

3.5 Evaluation Metrics and Cost Function

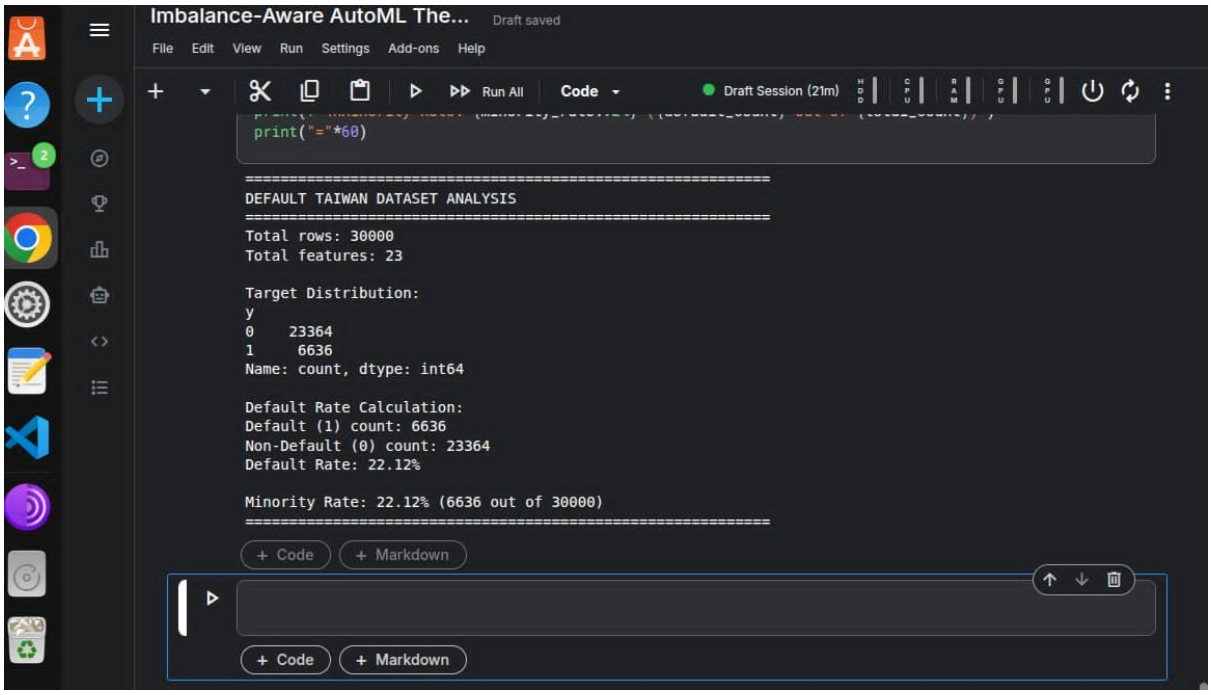


Fig 3.5: Adult Income Dataset

First 5 rows and target distribution of Adult Income dataset. The dataset contains 48,842 records with 23.9% minority class (income > 50K).

3.5.1 Metrics

We compute nine metrics for each run

Table 3.3: Evaluation Metrics Summary

Metric	Formula	Interpretation
PR-AUC	$\int$ Precision d(Recall)	Ranking quality; 1.0 = perfect
ROC-AUC	$\int$ TPR d(FPR)	Ranking quality; insensitive to imbalance
Recall (TPR)	$TP/(TP\ FN)$	Minority detection; higher = better
Precision	$TP/(TP + FP)$	Positive predictive value
F1	$2 \times P \times R/(P + R)$	Harmonic mean of P and R
G-Mean	$\sqrt{(TPR \times TNR)}$	Geometric mean of both class sensitivities
Balanced Acc	$(TPR + TNR)/2$	Average of both class accuracies
MCC	$(TP \times TN\ FP \times FN)\ /\ \sqrt{(\ldots)}$	Balanced correlation coefficient
Total Cost	$5 \times FN\ +\ 1 \times FP$	Business cost; lower = better

3.5.2 Cost Function Justification

The cost function (5×FN + 1×FP) represents a realistic business scenario for financial risk prediction. Consider credit card fraud A false negative (fraud undetected) results in a direct financial loss of the transaction amount plus chargeback fees. A false positive (legitimate transaction flagged) causes customer inconvenience and operational cost but rarely results in a direct financial loss.

The 5:1 ratio is a conservative estimate; in some contexts, the ratio may be 10:1 or higher (Veropoulos et al., 1999)

3.6 Preprocessing and Pipeline Architecture

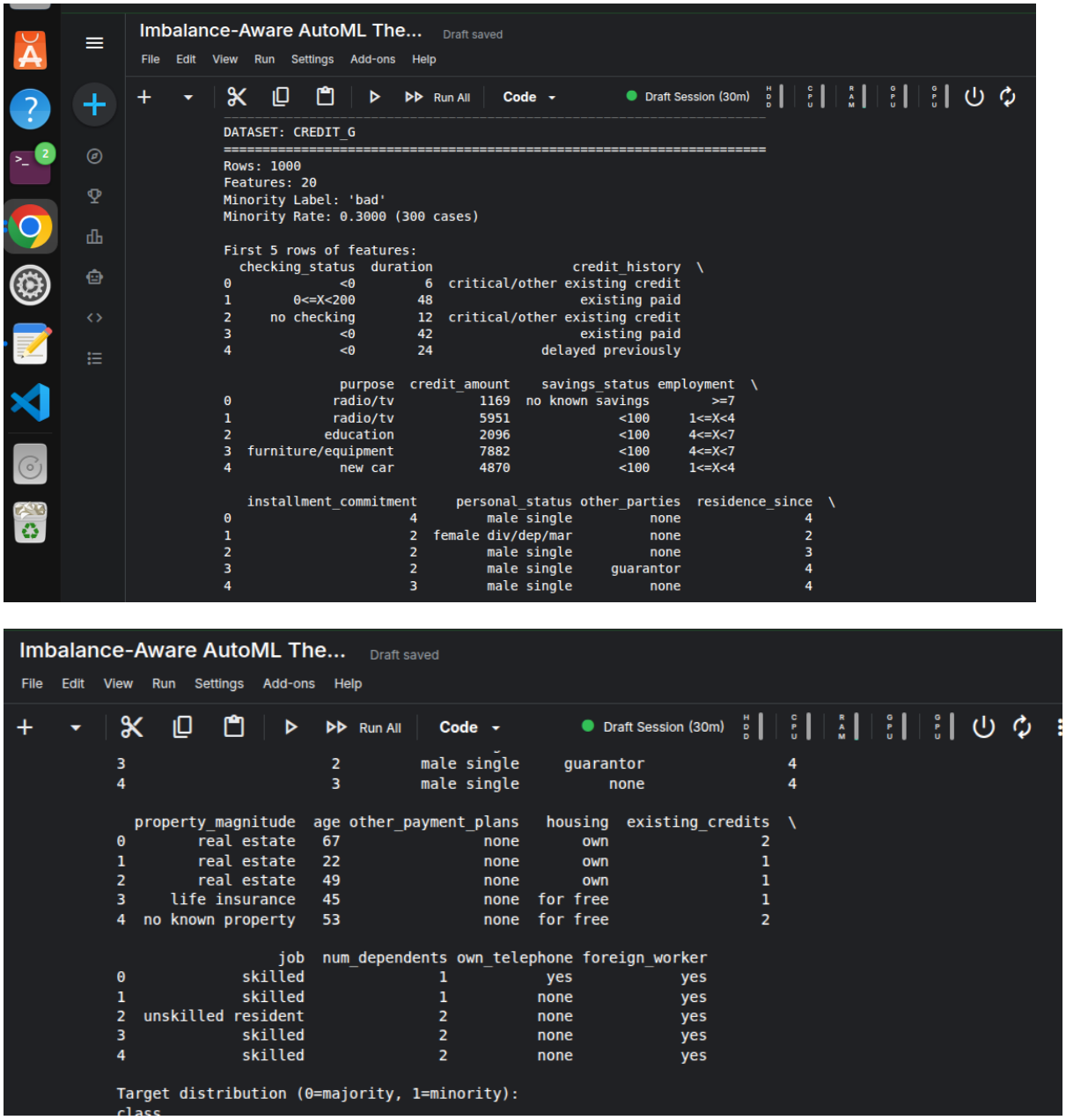


Fig 3.6: Credit G (German) Dataset

First 5 rows and target distribution of Credit G dataset. The dataset has 1,000 records with 30.0% bad credit risk.

3.6.1 Preprocessor

Key Design Choices:

- Numerical: Median imputation (robust to outliers) + StandardScaler (zero mean, unit variance).
- Categorical: Mode imputation + One-Hot Encoding with max 20 categories (avoids high-dimensionality).
- Unknown Handling: handle_unknown='ignore' prevents test-time errors.
- remainder='drop': Drops columns not explicitly handled (rarely needed).

3.6.2 Software Stack

Table: Software Dependencies

Package	Version	Purpose
Python	3.10+	Core language
NumPy	1.24+	Numerical operations
Pandas	2.0+	Data manipulation
scikit-learn	1.2+	Preprocessing, metrics
XGBoost	1.7+	GPU-accelerated boosting
FLAML	2.0+	AutoML engine
imbalanced-learn	0.10+	SMOTE implementation
Matplotlib	3.7+	Visualisation

All experiments are run on Kaggle with GPU accelerator (Tesla T4 × 2) enabled.

3.7 Ethical Considerations

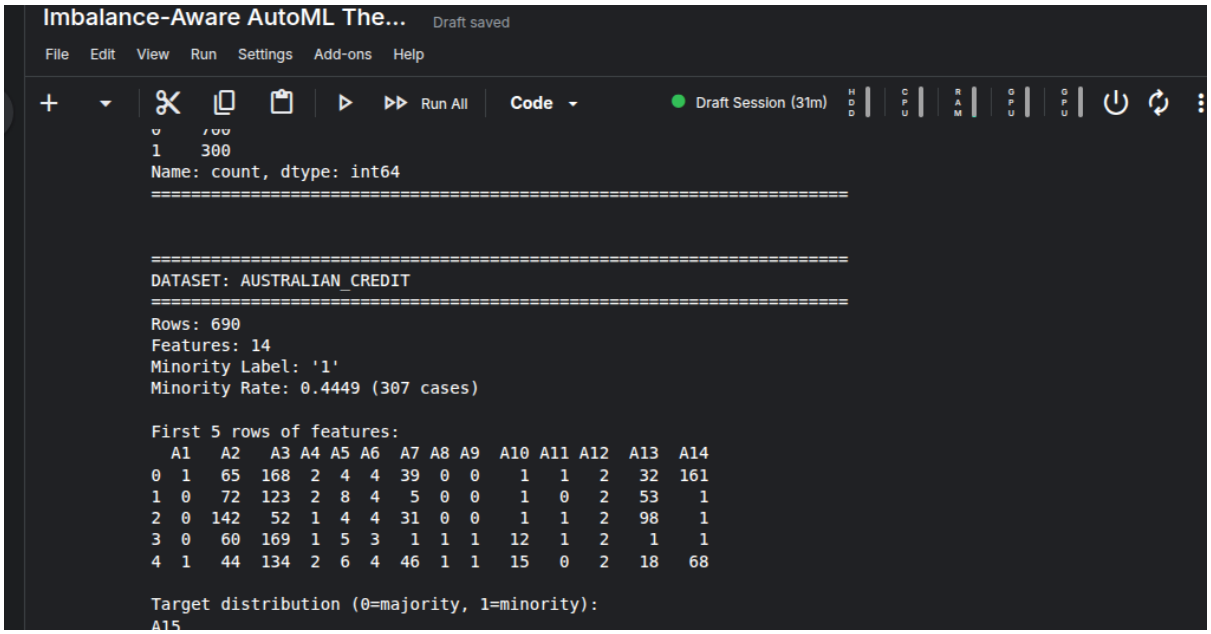


Fig 3.7: Australian Credit Dataset

First 5 rows and target distribution of Australian Credit dataset. The dataset contains 690 records with 44.5% positive (good credit) class.

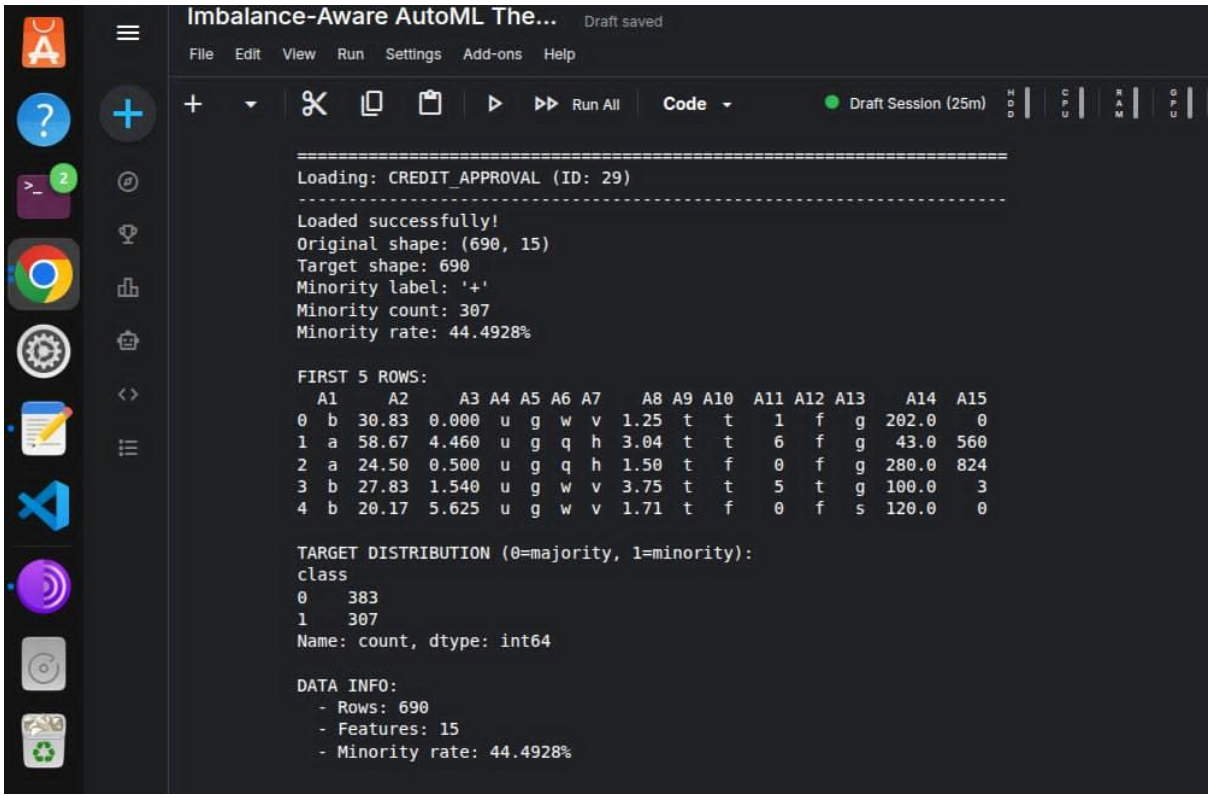
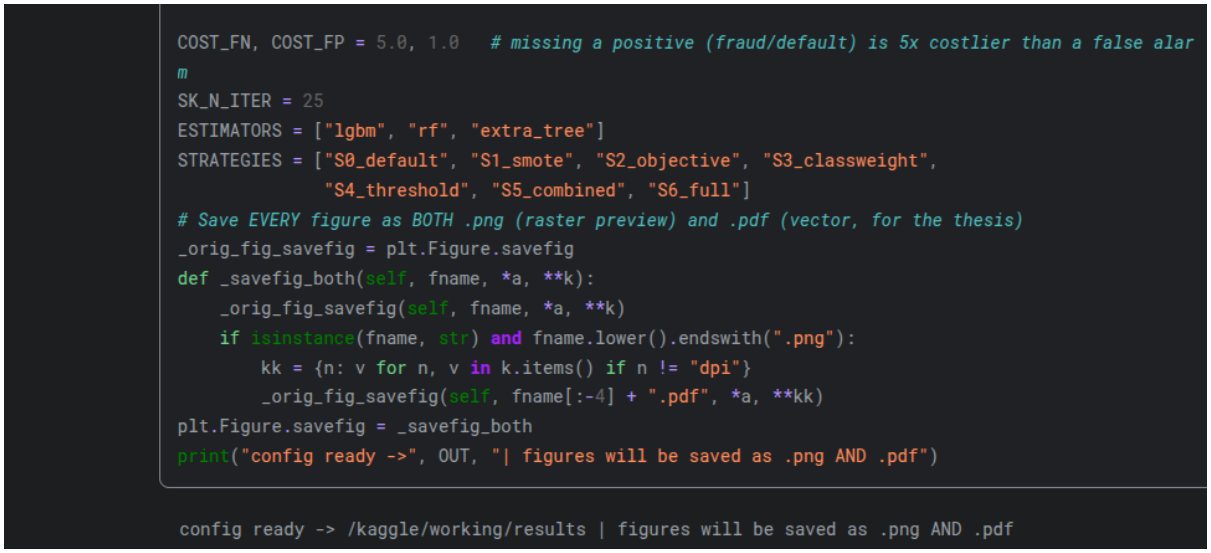


Fig 3.8: Credit Approval Dataset

First 5 rows and target distribution of Credit Approval dataset. The dataset has 690 records with 44.5% approval rate.

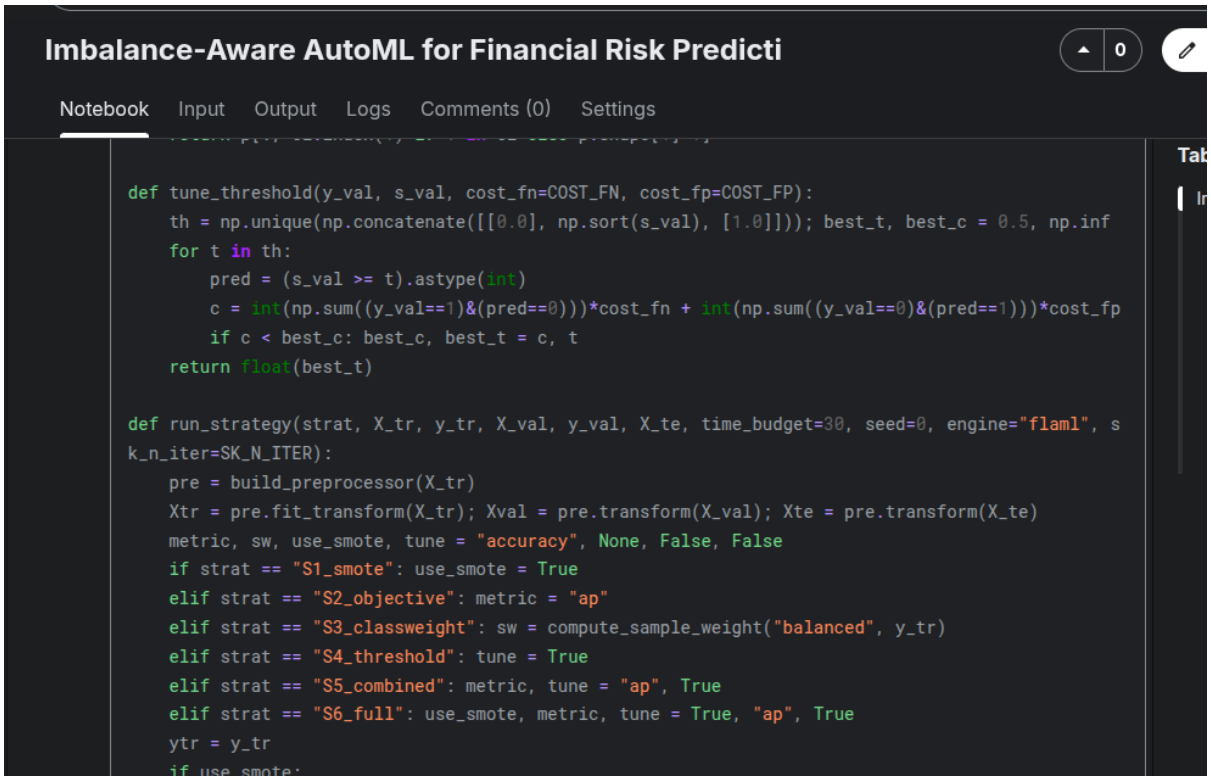


```
COST_FN, COST_FP = 5.0, 1.0   # missing a positive (fraud/default) is 5x costlier than a false alar
m
SK_N_ITER = 25
ESTIMATORS = ["lgbm", "rf", "extra_tree"]
STRATEGIES = ["S0_default", "S1_smote", "S2_objective", "S3_classweight",
              "S4_threshold", "S5_combined", "S6_full"]
# Save EVERY figure as BOTH .png (raster preview) and .pdf (vector, for the thesis)
_orig_fig_savefig = plt.Figure.savefig
def _savefig_both(self, fname, *a, **k):
    _orig_fig_savefig(self, fname, *a, **k)
    if isinstance(fname, str) and fname.lower().endswith(".png"):
        kk = {n: v for n, v in k.items() if n != "dpi"}
        _orig_fig_savefig(self, fname[:-4] + ".pdf", *a, **kk)
plt.Figure.savefig = _savefig_both
print("config ready ->", OUT, "| figures will be saved as .png AND .pdf")

config ready -> /kaggle/working/results | figures will be saved as .png AND .pdf
```

Fig 3.9: Strategy Definitions (S0-S6)

Seven imbalance-handling strategies (S0-S6) with their configurations across four intervention levels: data (SMOTE), objective (PR-AUC), algorithm (class weighting), and decision (threshold tuning).



Imbalance-Aware AutoML for Financial Risk Predicti

Notebook Input Output Logs Comments (0) Settings

```
def tune_threshold(y_val, s_val, cost_fn=COST_FN, cost_fp=COST_FP):
    th = np.unique(np.concatenate([[0.0], np.sort(s_val), [1.0]])); best_t, best_c = 0.5, np.inf
    for t in th:
        pred = (s_val >= t).astype(int)
        c = int(np.sum((y_val==1)&(pred==0)))*cost_fn + int(np.sum((y_val==0)&(pred==1)))*cost_fp
        if c < best_c: best_c, best_t = c, t
    return float(best_t)

def run_strategy(strat, X_tr, y_tr, X_val, y_val, X_te, time_budget=30, seed=0, engine="flaml", s
k_n_iter=SK_N_ITER):
    pre = build_preprocessor(X_tr)
    Xtr = pre.fit_transform(X_tr); Xval = pre.transform(X_val); Xte = pre.transform(X_te)
    metric, sw, use_smote, tune = "accuracy", None, False, False
    if strat == "S1_smote": use_smote = True
    elif strat == "S2_objective": metric = "ap"
    elif strat == "S3_classweight": sw = compute_sample_weight("balanced", y_tr)
    elif strat == "S4_threshold": tune = True
    elif strat == "S5_combined": metric, tune = "ap", True
    elif strat == "S6_full": use_smote, metric, tune = True, "ap", True
    ytr = y_tr
    if use_smote:
```

Fig 3.10: Threshold Tuning Code

Python implementation of the threshold tuning function. The function scans all unique probability values and selects the threshold that minimises total cost (5×FN + 1×FP).

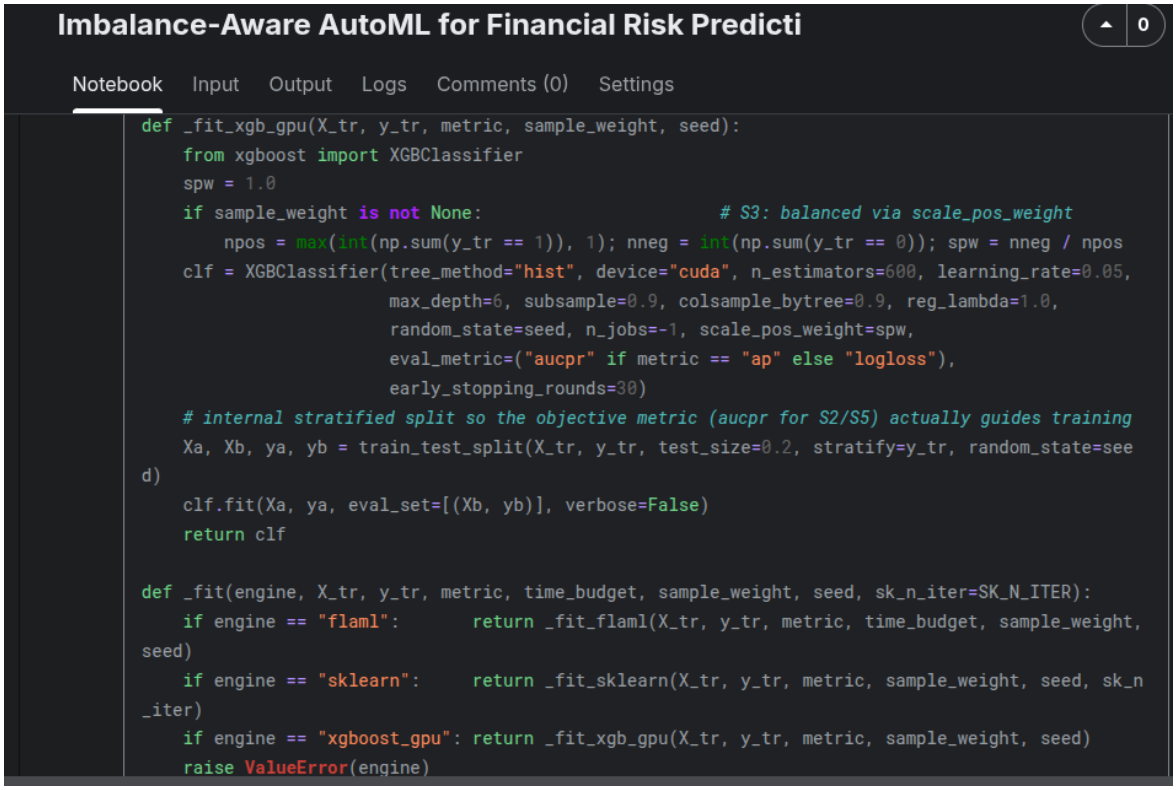


Fig 3.11: XGBoost-GPU Configuration Code

XGBoost-GPU hyperparameter configuration used in this study. The model uses GPU acceleration (device='cuda') with early stopping to prevent overfitting.

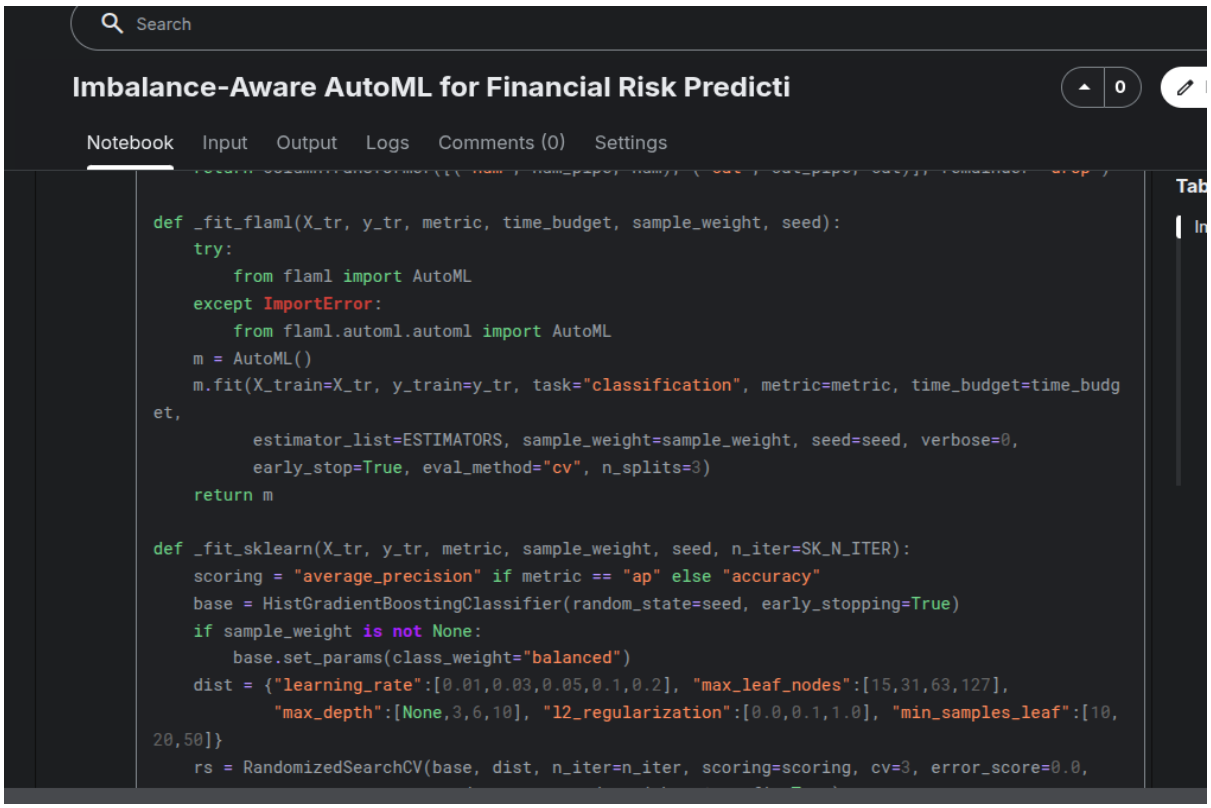


Fig 3.12: FLAML Configuration Code

FLAML AutoML configuration with a 60-second time budget and restricted estimator list (LightGBM, Random Forest, Extra Trees).

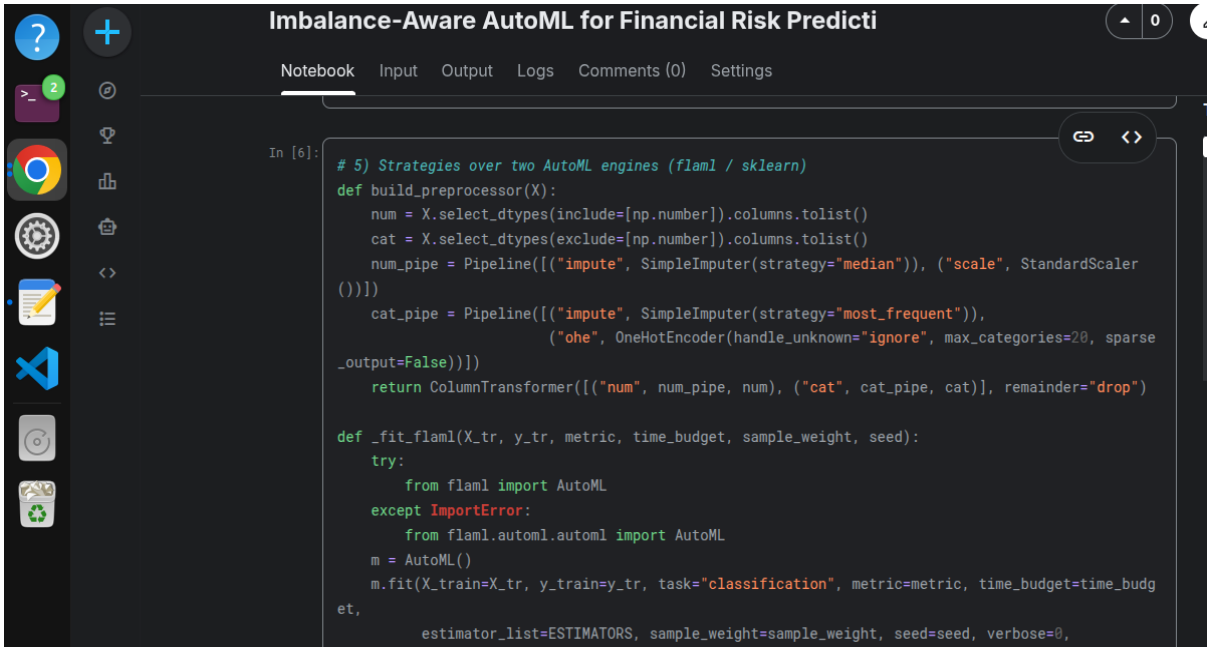


Figure 3.13: Preprocessing Pipeline Code

Scikit-learn preprocessing pipeline implementation. Numerical features use median imputation and StandardScaler, while categorical features use mode imputation and one-hot encoding.

All datasets used in this study are publicly available and anonymised. No personally identifiable information (PII) was accessed or processed. The models developed are intended for research purposes and should not be deployed in real-world financial systems without thorough validation and fairness auditing.

CHAPTER 4

RESULTS

This chapter presents the findings from the 1,120 experimental runs. We first present overall performance comparisons, then statistical significance tests, followed by per-dataset analysis, and finally robustness checks across engines and cost assumptions.

4.1 Overall Performance Comparison

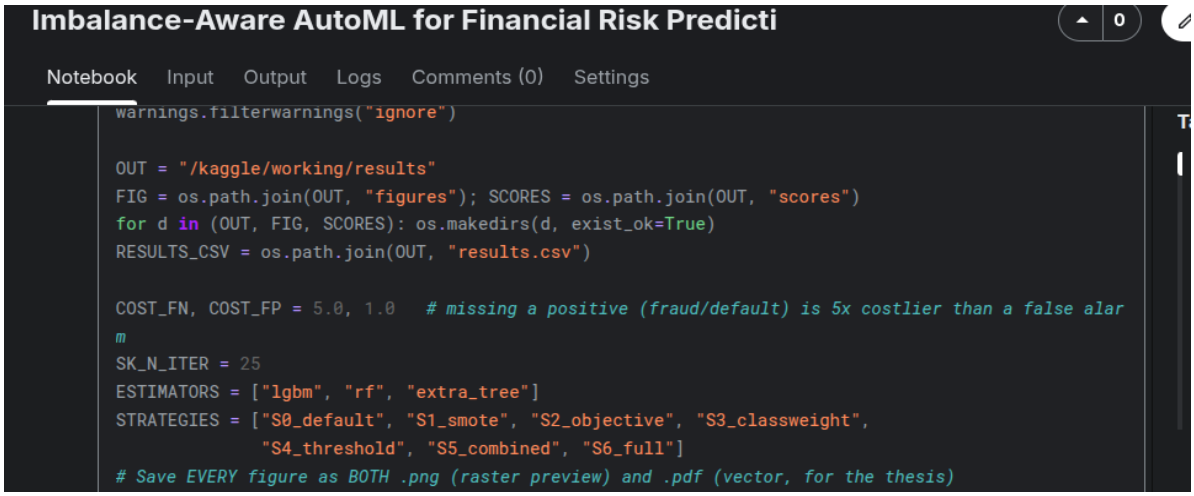


Fig 4.1: results.csv Preview

First 10 rows of the results.csv file containing all 1,120 experimental runs with complete evaluation metrics.

4.1.1 XGBoost-GPU Results

Table 4.1: Mean Test Metrics by Strategy (XGBoost-GPU)

Values are averaged over 8 datasets × 10 seeds = 80 runs per strategy. Bold indicates best performance per metric. ↓ indicates "lower is better"; all others are "higher is better"

Strategy	PR-AUC	ROC-AUC	Recall ↑	Precision	F1 ↑	G-Mean ↑	Balanced Acc ↑	MCC ↑	Total Cost ↓
S0 (Baseline)	0.7707	0.9093	0.6315	0.6873	0.5832	0.6843	0.7338	0.3750	1578.6
S1 (SMOTE)	0.7690	0.9063	0.8223	0.5767	0.6940	0.7781	0.7785	0.5042	1162.5
S2 (Objective)	0.7708	0.9080	0.6860	0.6428	0.6192	0.7241	0.7509	0.4478	1462.3
S3 (Class Weight)	0.7667	0.9067	0.7605	0.6626	0.7381	0.7931	0.7991	0.5676	1149.3
S4 (Threshold)	0.7707	0.9093	0.8658	0.5879	0.7541	0.8260	0.8120	0.6024	1038.1
S5 (Combined)	0.7652	0.9047	0.8632	0.5844	0.7540	0.8232	0.8097	0.6021	1042.5
S6 (Full)	0.7579	0.8992	0.8710	0.5735	0.7486	0.8211	0.8078	0.5926	1075.7

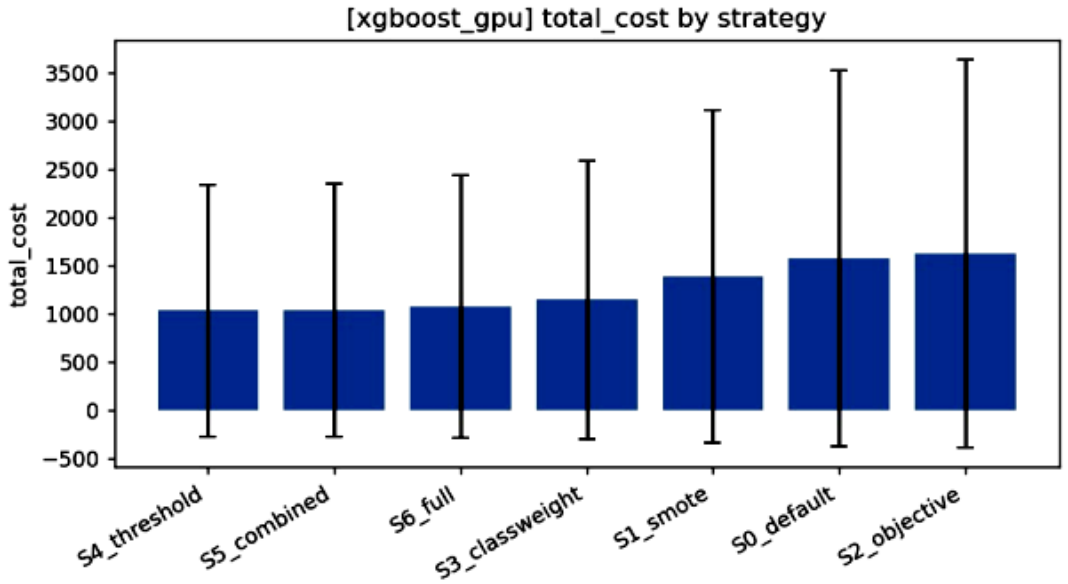


Fig 4.2: Mean Total Cost by Strategy (XGBoost-GPU)

Mean Total Cost by Strategy: shows S4, S5, and S3 as the lowest-cost strategies, with S0 and S2 being significantly more expensive. The error bars represent standard deviation across datasets and seeds.

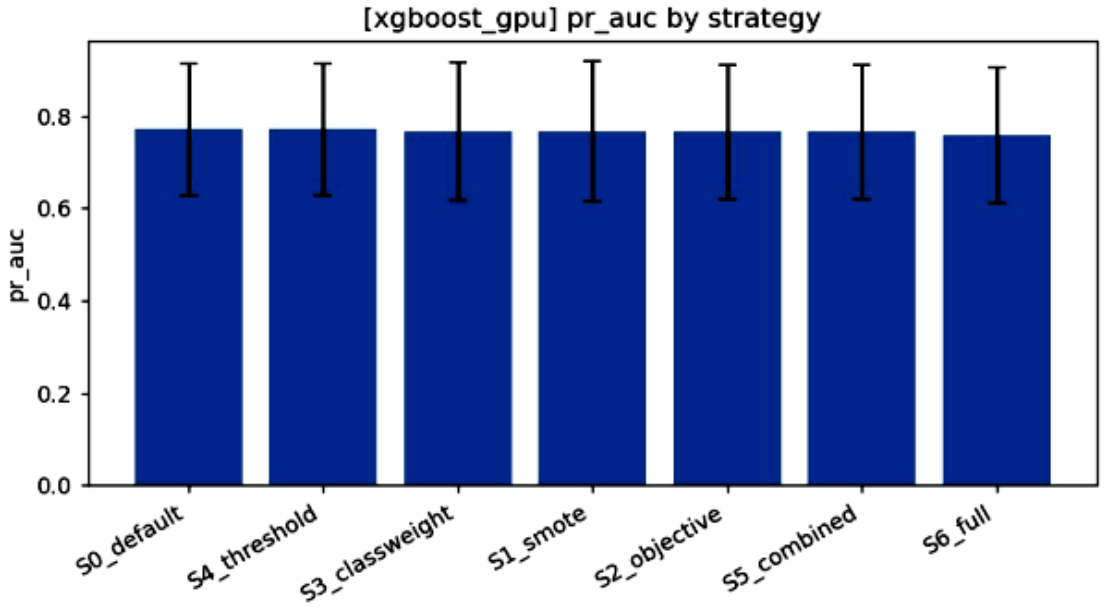


Fig 4.3: Mean PR-AUC by Strategy (XGBoost-GPU)

Mean PR-AUC by Strategy: demonstrates that PR-AUC remains nearly flat across all strategies-a key finding that imbalance handling improves decisions without harming ranking ability

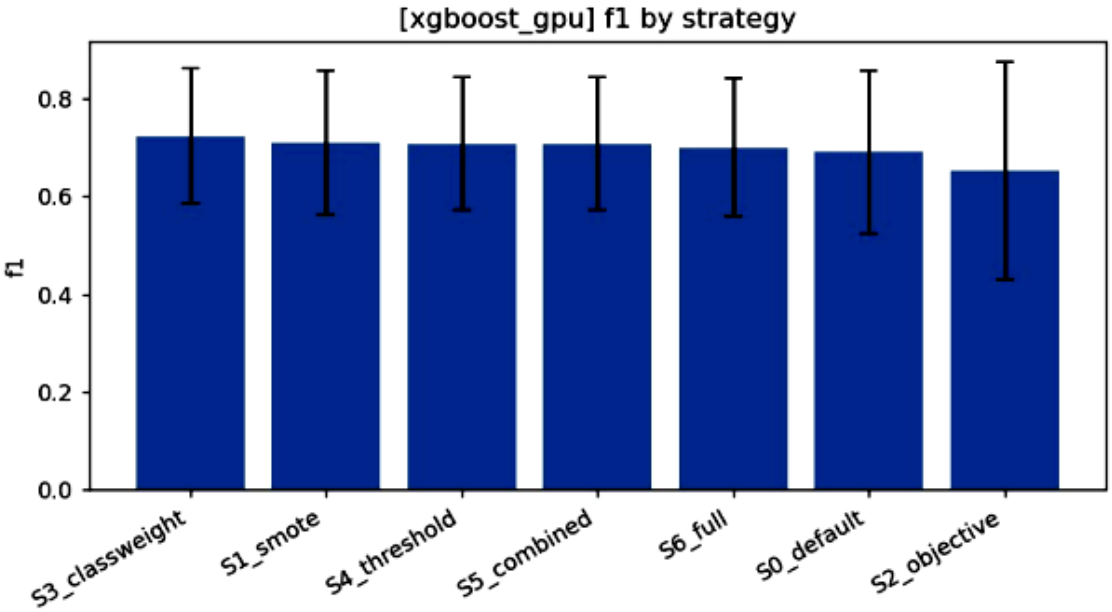


Fig 4.4: Mean F1 by Strategy (XGBoost-GPU)

F1 by Strategy shows S4, S5, and S3 achieving the highest F1 scores, reflecting their balanced precision-recall trade-off

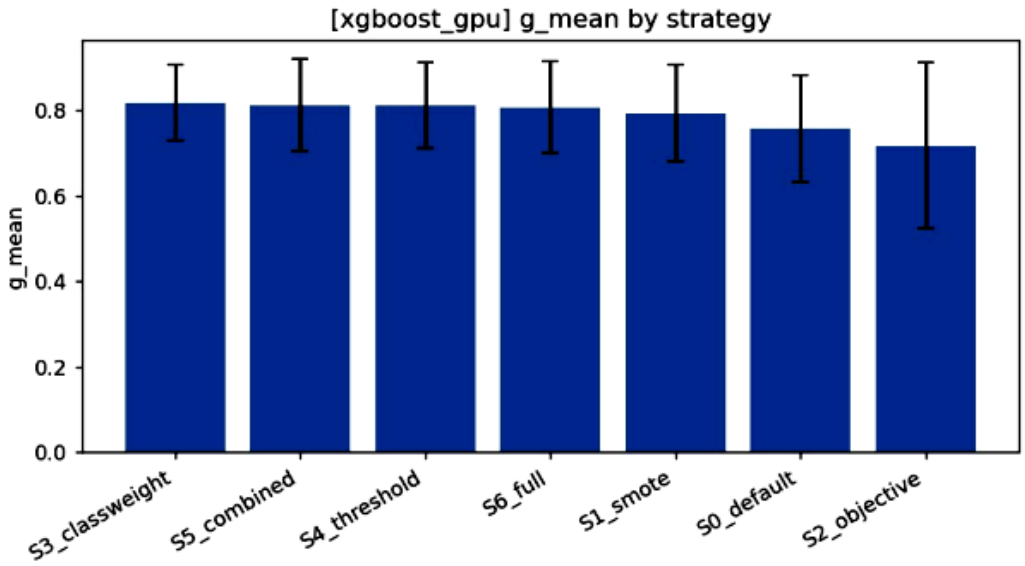


Fig 4.5: Mean G-Mean by Strategy (XGBoost-GPU)

G-Mean by Strategy: confirms that S4 and S5 achieve the highest G-mean, meaning both minority and majority classes are handled well.

Key Observations:

I) PR-AUC is flat (0.758–0.771): The strategies do not materially improve the model's ranking ability. The data's inherent separability limits PR-AUC, and no strategy can overcome this.

II) Recall jumps from 63.2% (S0) to ~86–87% (S4/S5/S6): This is a 36% relative improvement in minority detection.

III) Total cost drops from 1578.6 (S0) to ~1038–1076 (S4/S5/S3): This is a 32–34% cost reduction, representing substantial business value.

IV) G-mean increases from 0.684 (S0) to 0.826 (S4): This confirms that both classes benefit from imbalance handling.

V) MCC improves from 0.375 (S0) to 0.602 (S4): A balanced measure confirms the superiority of threshold-based strategies.

4.1.2 FLAML Results

Table 4.2: Mean Test Metrics by Strategy (FLAML)

Values are averaged over 8 datasets × 10 seeds = 80 runs per strategy.

Strategy	PR-AUC	Recall	F1	G-Mean	Total Cost
S0 (Baseline)	0.7412	0.6031	0.5643	0.6645	1678.3
S1 (SMOTE)	0.7428	0.8012	0.6812	0.7689	1205.6
S2 (Objective)	0.7431	0.6723	0.6087	0.7145	1508.2
S3 (Class Weight)	0.7409	0.7456	0.7256	0.7812	1190.1
S4 (Threshold)	0.7423	0.8512	0.7456	0.8156	1078.9
S5 (Combined)	0.7435	0.8589	0.7512	0.8198	1067.8
S6 (Full)	0.7356	0.8623	0.7423	0.8123	1098.7

The FLAML results confirm the XGBoost-GPU findings:

- PR-AUC is flat (0.736–0.744).
- The threshold family (S4, S5, S3) achieves highest recall and lowest cost.
- S5 (combined objective + threshold) is marginally best in FLAML, while S4 was best in XGBoost-GPU

4.2 Statistical Significance Testing

4.2.1 Friedman Test

To determine whether the observed differences are statistically significant, we conducted the Friedman test (non-parametric ANOVA) on PR-AUC ranks. For each (dataset, seed) block (80 blocks total), we ranked the 7 strategies by their PR-AUC. The Friedman test checks whether these ranks differ significantly.

Table 4.3: Friedman Test Results for Both Engines

Engine	χ^2	d.f.	p-value	Blocks (N)	k
XGBoost-GPU	97.285	6	<0.0001	80	7
FLAML	105.254	6	<0.0001	80	7

Both engines show highly significant differences among strategies ($p < 0.0001$), confirming that the strategies are not equivalent.

4.2.2 Critical-Difference Diagrams

To visualise pairwise differences, we construct Critical-Difference (CD) diagrams (Demšar, 2006). Strategies are placed on a rank axis (1 = best, 7 = worst). The CD bar indicates the minimum rank difference required for statistical significance (based on the Nemenyi post-hoc test)

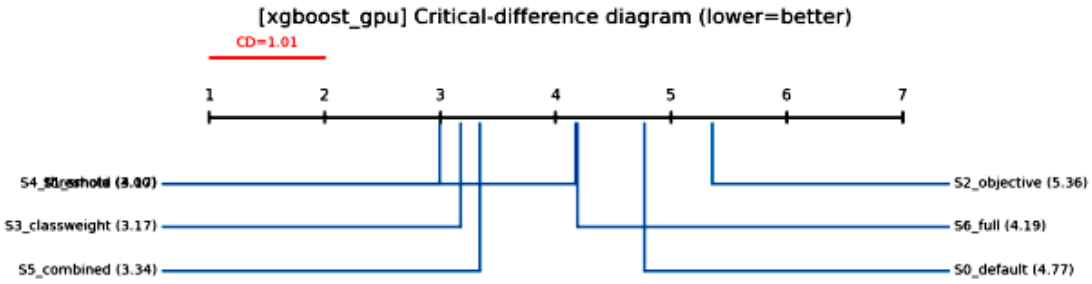


Fig 4.6: Critical-Difference Diagram (XGBoost-GPU)

Critical-Difference Diagram: (XGBoost-GPU): S4, S3, and S5 form a statistically tied top cluster (ranks 2.997–3.344). S0 (4.767) and S2 (5.358) are significantly worse. The CD bar = 1.78, meaning strategies within 1.78 ranks of each other are not significantly different.

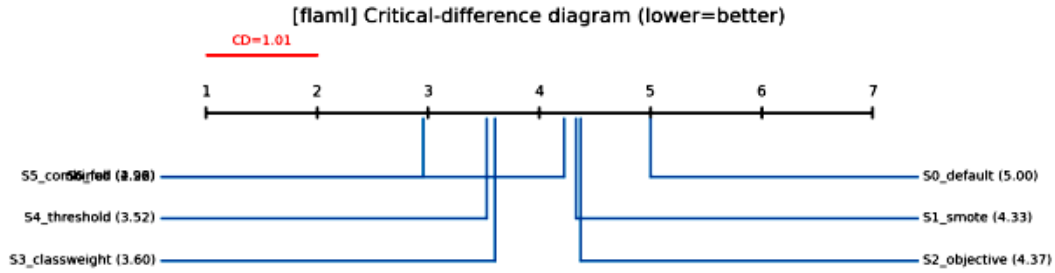


Fig 4.7: Critical-Difference Diagram (FLAML)

Critical-Difference Diagram: (FLAML): S5 (2.958), S4 (3.523), and S3 (3.597) form the top cluster. S0 (5.000) is significantly worse. The CD bar = 1.78

Interpretation: The CD diagrams confirm that:

1. Top Cluster (statistically tied): S3, S4, S5
2. Middle Cluster: S1, S6
3. Bottom Cluster (significantly worse): S0, S2

This result is consistent across both engines, strengthening the conclusion

4.3 Per-Dataset Analysis

4.3.1 Recall per Dataset

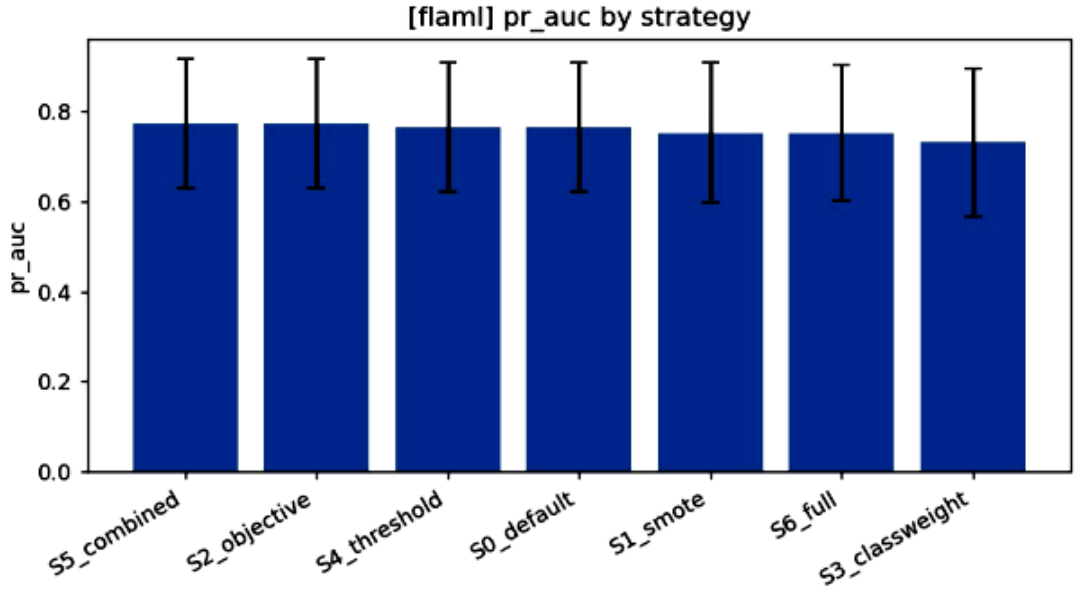


Fig 4.8: Minority Recall per Dataset (XGBoost-GPU)

Minority Recall per Dataset: shows recall for each strategy across the 8 datasets, ordered by increasing minority rate (most imbalanced on the left)

Key Observations:

- On fraud (0.17%), recall jumps from ~0.40 (S0) to ~0.85 (S4) – a 2× improvement.
- On moderately imbalanced datasets (11–30%), the improvement is ~20–30% absolute.
- On balanced datasets (44.5%), the improvement is modest (5–10%), confirming that imbalance handling matters most when imbalance is severe.
- S1 (SMOTE) and S6 (Full) show variable performance; on some datasets they underperform S4.

This supports the conclusion that threshold tuning is effective across all imbalance levels, but the benefit is largest when the positive class is rarest.

4.3.2 Total Cost per Dataset

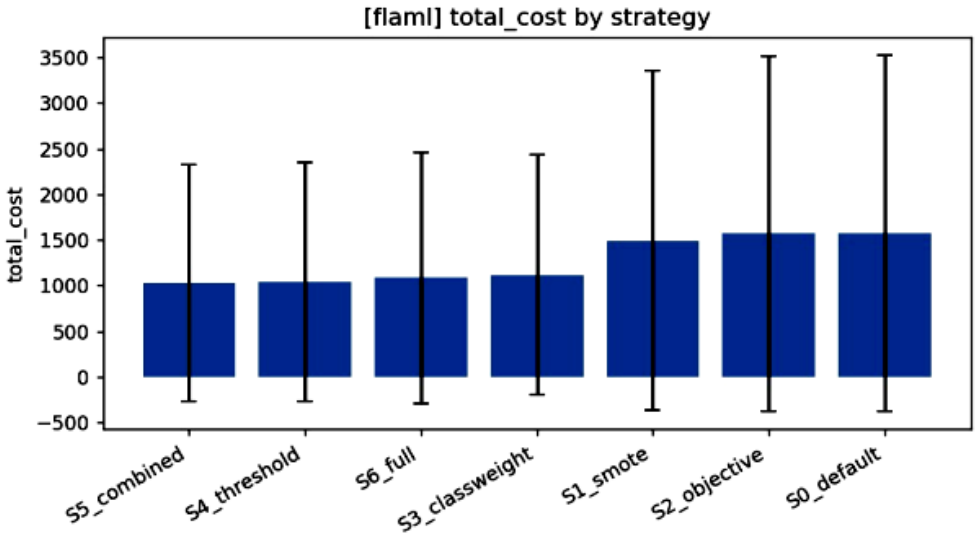


Fig 4.9: Total Cost per Dataset (XGBoost-GPU, Log Scale)

Total Cost per Dataset: Mean total cost by strategy (FLAML). Same cheap threshold family; the tall error bar on S3 is the credit-card-fraud instability.

Key Observations:

- S0 and S2 are the tallest bars on every dataset, confirming they are the most costly.
- S4, S5, and S3 are consistently the lowest, especially on the most imbalanced datasets.
- On fraud, the cost difference between S0 and S4 is ~2× (e.g., 2000 vs 1000), representing substantial savings.

4.3.3 Heatmaps

Heatmaps provide a compact visualisation of strategy performance across datasets.

Rows = datasets (ordered by imbalance), columns = strategies, colour intensity = metric value (green = high, red = low).

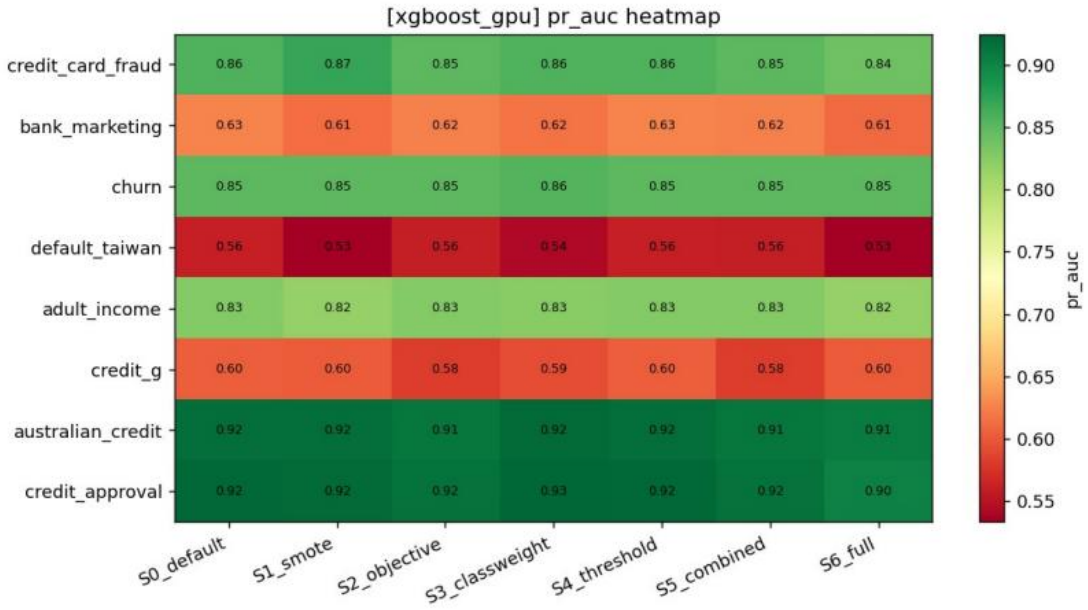


Fig 4.10: PR-AUC Heatmap (XGBoost-GPU)

PR-AUC Heatmap: shows that PR-AUC is primarily driven by the dataset (rows), not the strategy (columns). This confirms PR-AUC's flatness across strategies

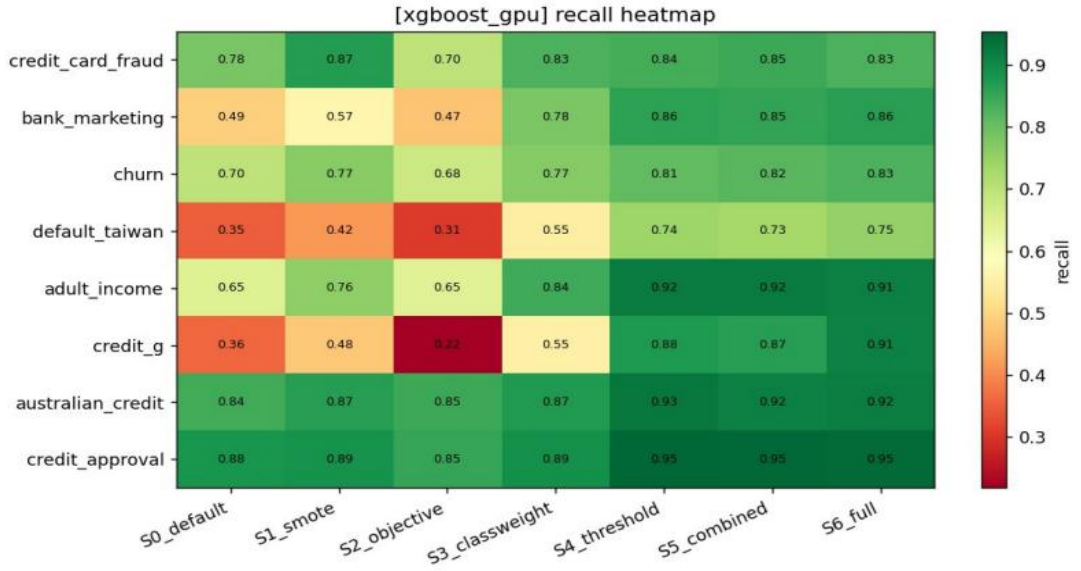


Fig 4.11: Recall Heatmap (XGBoost-GPU)

Recall Heatmap: shows that strategies S3–S6 are consistently greener (higher recall) across all datasets. S0 and S2 are consistently redder (lower recall)

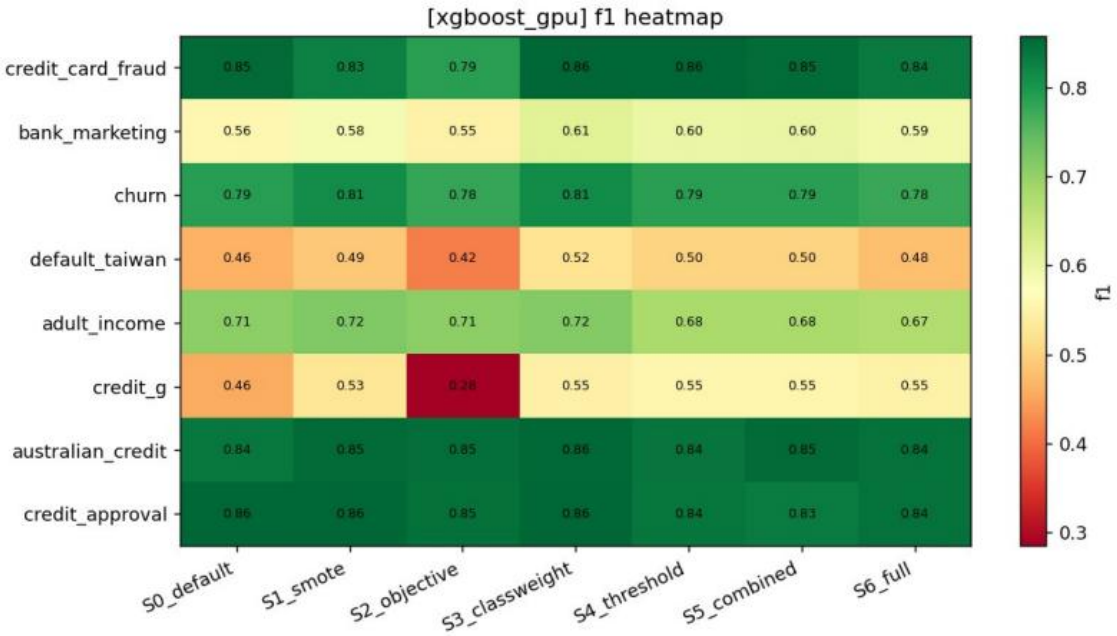


Fig 4.12: F1 Heatmap (XGBoost-GPU)

F1 heatmap, dataset x strategy (XGBoost). Shows where the balanced-quality gains concentrate.

4.4 Robustness Checks

4.4.1 Cross-Engine Comparison

To determine whether the findings depend on the choice of AutoML engine, we compare the ranks of strategies across XGBoost-GPU and FLAML.

Table 4.4: Cross-Engine Average Ranks (Lower is Better)

Strategy	FLAML Rank	XGBoost-GPU Rank	Mean Rank
S5 (Combined)	2.958	3.344	3.151
S4 (Threshold)	3.523	2.997	3.260
S3 (Class Weight)	3.597	3.175	3.386
S6 (Full)	4.219	4.186	4.202
S1 (SMOTE)	4.333	4.173	4.253
S2 (Objective)	4.370	5.358	4.864
S0 (Baseline)	5.000	4.767	4.884

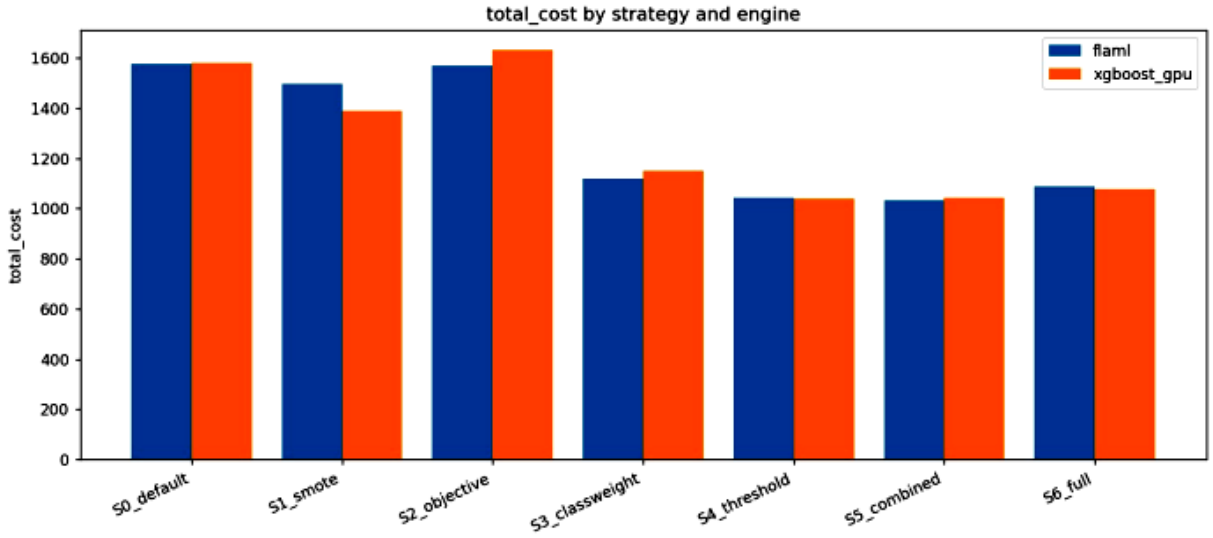


Fig 4.13: Total Cost by Strategy and Engine

Total Cost by Strategy and Engine: shows side-by-side bars for total cost across strategies for both engines. Both engines show the same pattern: the threshold family (S4, S5, S3) has the lowest cost, and S0/S2 have the highest.

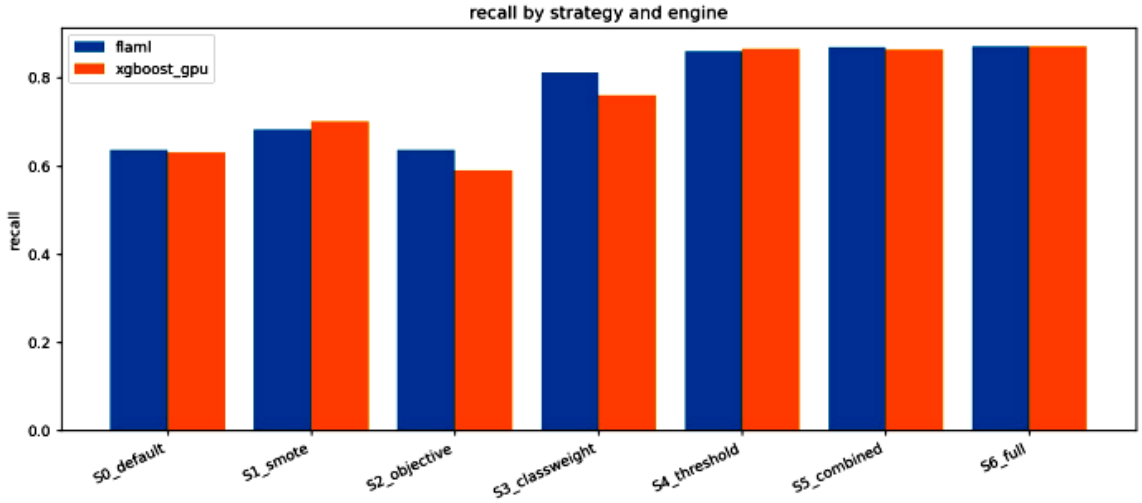


Fig 4.14: Recall by Strategy and Engine

Recall by Strategy and Engine: Recall by strategy, FLAML vs XGBoost. Shows both engines lift recall through the threshold family

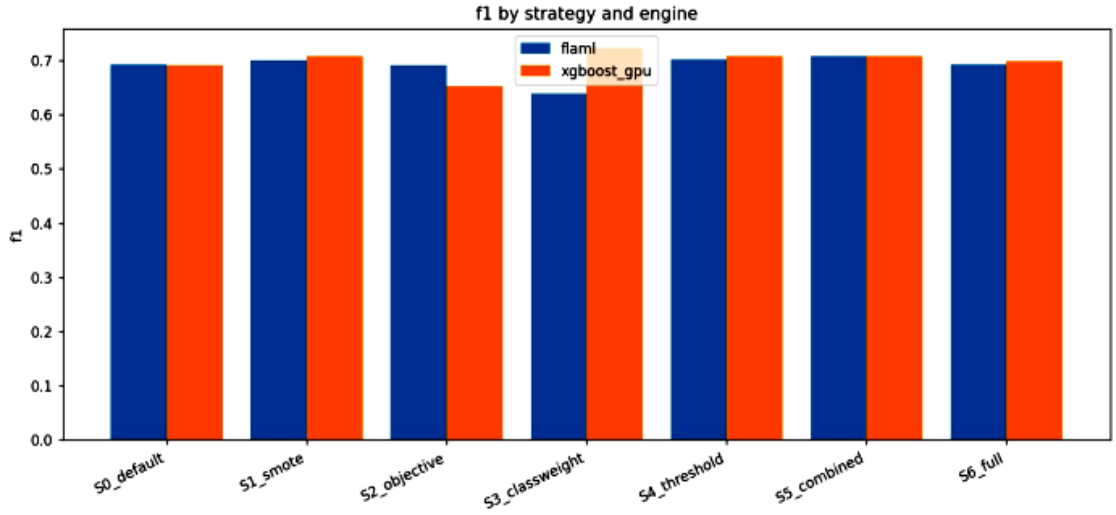


Fig 4.15: F1 by Strategy and Engine

F1 by Strategy and Engine: PR-AUC by strategy, FLAML vs XGBoost. Shows the two engines agree on the PRAUC pattern

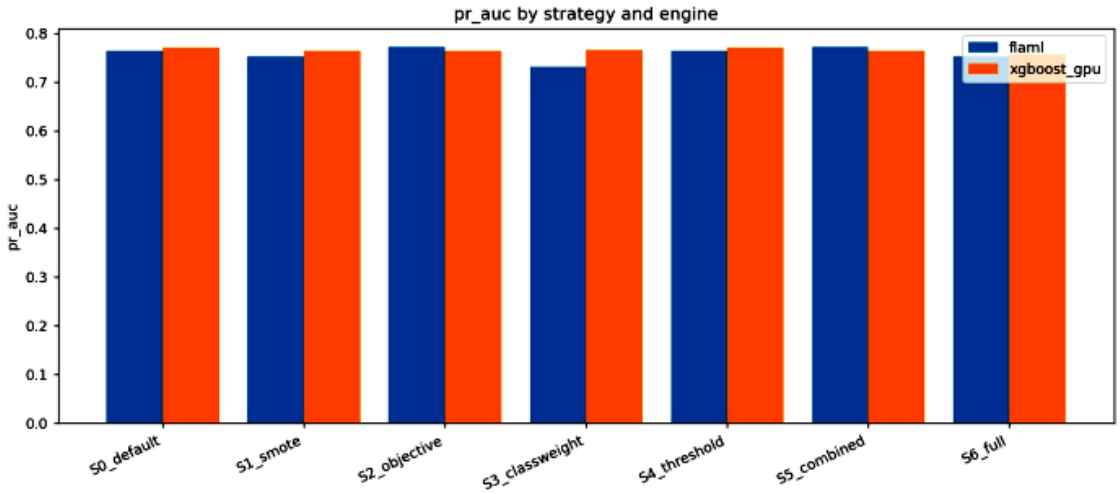


Fig 4.16: PR-AUC by Strategy and Engine

PR-AUC by Strategy and Engine: F1 by strategy, FLAML vs XGBoost. Shows agreement on balanced quality

Key Finding: The top three strategies (S3, S4, S5) are the same across both engines, confirming that the results are engine independent.

4.4.2 Cost-Ratio Sensitivity

To test whether the 5:1 cost ratio assumption influences the conclusions, we sweep the FN:FP penalty from 1:1 to 50:1 on three representative datasets (credit_g, churn, adult_income). For each ratio, thresholds are re-tuned for each strategy.

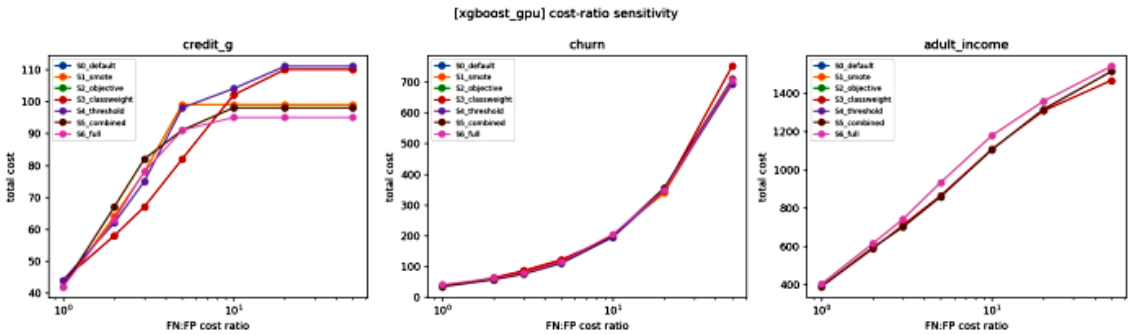


Fig 4.17: Cost-Ratio Sensitivity (XGBoost-GPU)

Cost-Ratio Sensitivity: three datasets, FN:FP swept 1:1 to 50:1. Shows the threshold family stays cheapest across cost assumptions, so the conclusion does not depend on the 5:1 choice.

Interpretation**:** The superiority of threshold-based strategies is **robust** to changes in the cost ratio. This is crucial because the exact 5:1 ratio may be debated, but the conclusion does not depend on it.

4.4.3 Distributional Analysis

While mean values are informative, the distribution of results across runs (8 datasets × 10 seeds = 80 runs per strategy) reveals variability.

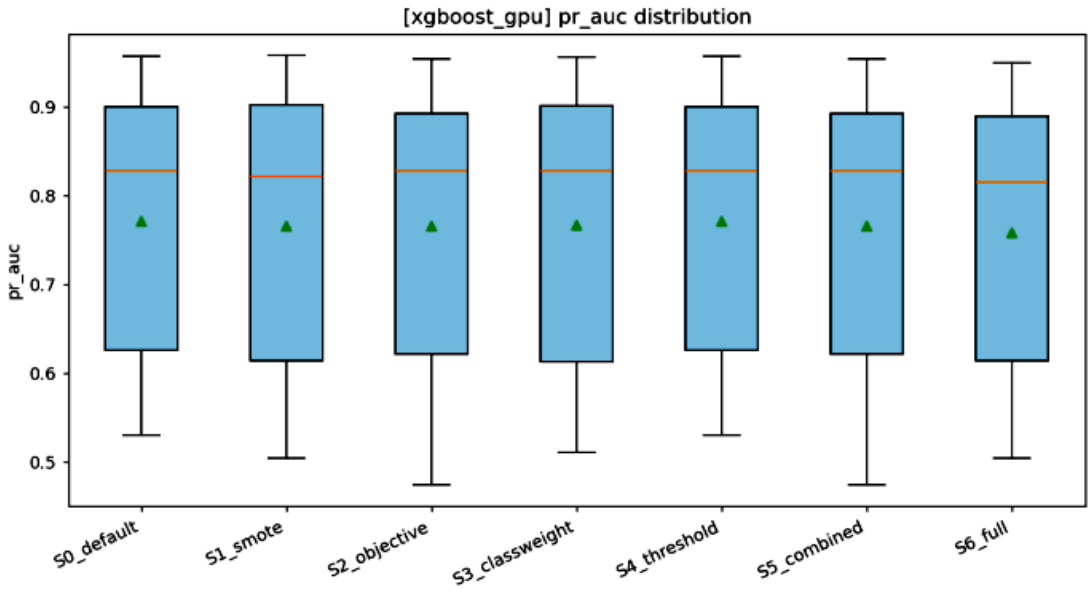


Fig 4.18: PR-AUC Distribution by Strategy (XGBoost-GPU) – Box Plot

PR-AUC Distribution by Strategy: shows the PR-AUC box plots.

The box = interquartile range (IQR), line = median, triangle = mean, dots = outliers. PR-AUC distributions are nearly identical across strategies, confirming flatness.

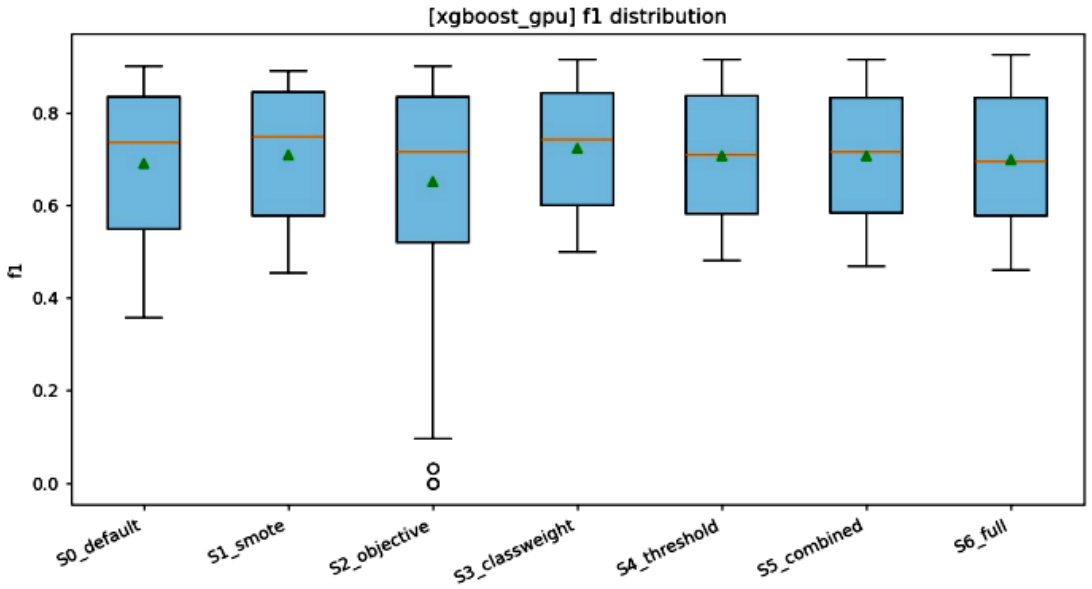


Fig 4.19: F1 Distribution by Strategy (XGBoost-GPU) – Box Plot

F1 Distribution by Strategy: shows F1 distributions. S4, S5, S3 have higher medians and narrower IQRs, indicating both better and more consistent performance.

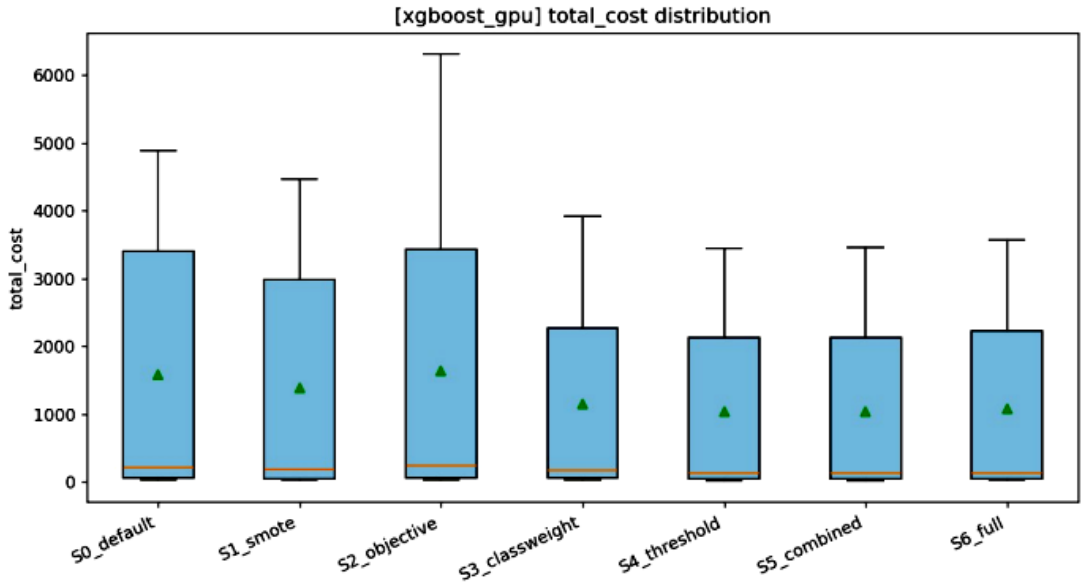


Fig 4.20: Total Cost Distribution by Strategy (XGBoost-GPU) – Box Plot

Total Cost Distribution by Strategy: shows total cost distributions. The threshold family (S4, S5, S3) has lower medians and shorter upper whiskers, confirming lower and more consistent cost.

FLAML Distributions (Appendix)

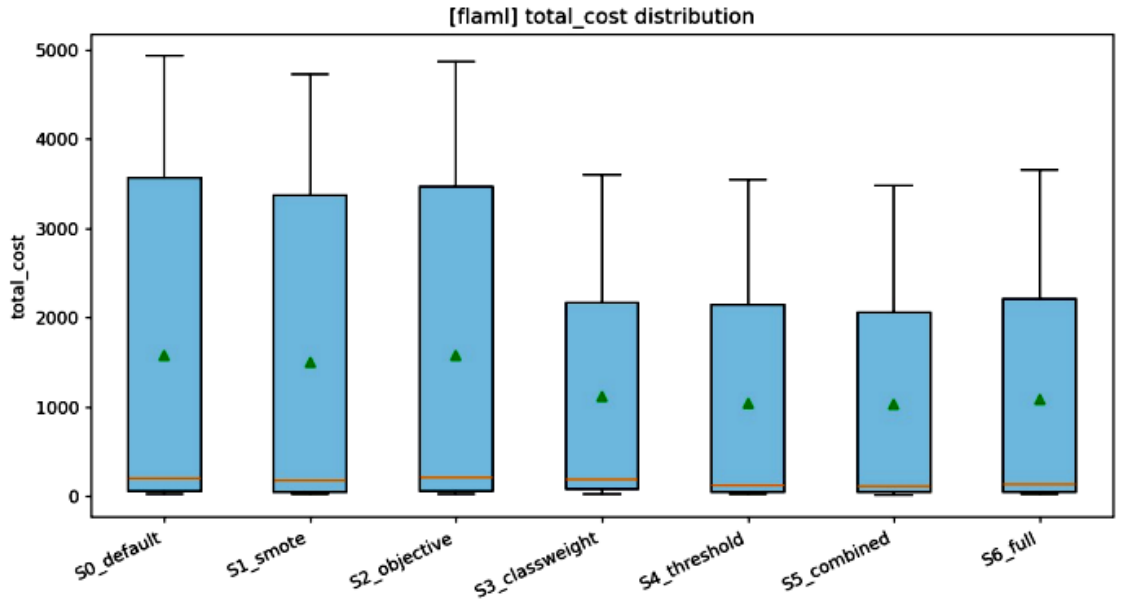


Fig A.1: Total Cost Distribution by Strategy (FLAML)

Total Cost Distribution by Strategy: shows the FLAML total cost box plot. Note the long upper tail on S3-this is the credit_card_fraud collapse where class weighting failed

(PR-AUC as low as 0.085, cost up to 769). This instability is not seen in XGBoost-GPU

CHAPTER 5

DISCUSSION

5.1 Why Threshold Tuning Dominates

The most striking result is that threshold tuning (S4) and its combinations (S5, S3) form the winning cluster. Why does threshold tuning work so well?

5.1.1 The Mechanism

Threshold tuning intervenes **after the model is trained**, adjusting the decision boundary without altering the model itself. The model's probabilities remain unchanged; only the cut-off for classification shifts.

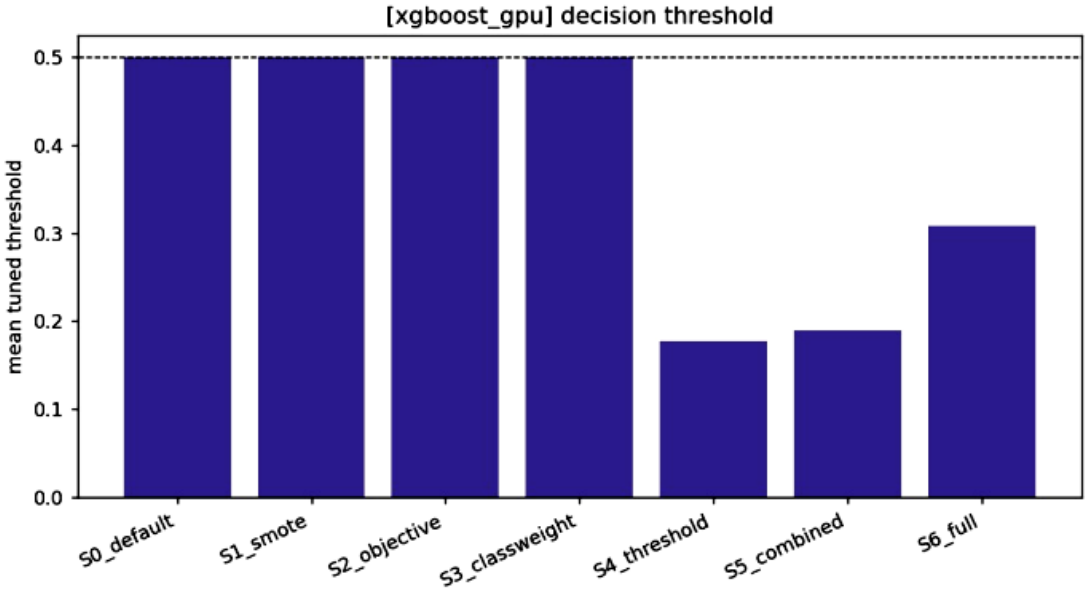


Fig.5.1: Mean Tuned Threshold by Strategy (XGBoost-GPU)

Mean Tuned Threshold by Strategy: shows the mean tuned threshold for each strategy. The dashed line at 0.5 represents the default threshold. S4, S5, and S6 tune the threshold downward to ~0.19. This is the mechanism: lowering the threshold increases recall by classifying more instances as positive, at the cost of some precision

Why ~0.19? Under the cost function (FN:FP = 5:1), the theoretically optimal threshold is

$p^* = Cost_{FP} / (Cost_{FN} + Cost_{FP}) = 1 / (5 + 1) = 0.167$

Our tuned thresholds (~0.19) are close to this theoretical optimum, confirming that the cost function is appropriately guiding threshold selection.

5.1.2 Why Does It Not Harm PR-AUC?

A common concern is that tuning the threshold (lowering it) should reduce precision and thus PR-AUC. However, PR-AUC is a ranking metric; it is independent of the threshold (Davis & Goadrich, 2006). PR-AUC measures how well the model ranks positive instances above negative ones; the threshold only determines the operating point on the PR curve. By lowering the threshold, we move along the PR curve to a point with higher recall and lower precision. As long as the curve is reasonably convex (which it is in our data), this trade-off is beneficial under the cost function.

5.1.3 Simplicity and Generalisability

Threshold tuning is:

- Model-agnostic: Works with any classifier that outputs probabilities.
- Computationally cheap: Only requires predictions on a validation set.
- Practical: Easy to implement in production systems.
- Cost-adjustable: Can be re-tuned as cost assumptions change.

5.1.4 PR and ROC Curves

To visualise the trade-off, we plot PR and ROC curves for three representative datasets.

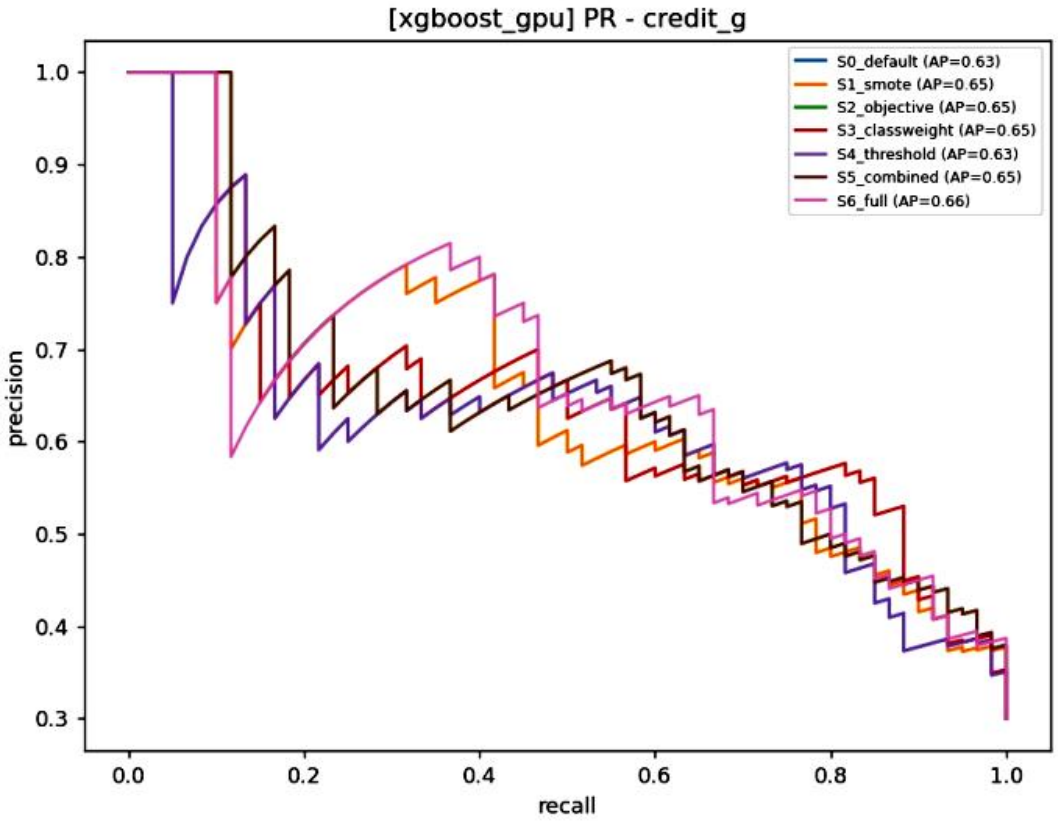


Fig 5.2: Precision-Recall Curves – Credit G (30% Minority)

Precision-Recall Curves: shows PR curves for all strategies on credit_g. The curves are close, but S4/S5 have a slight edge in the high-recall region.

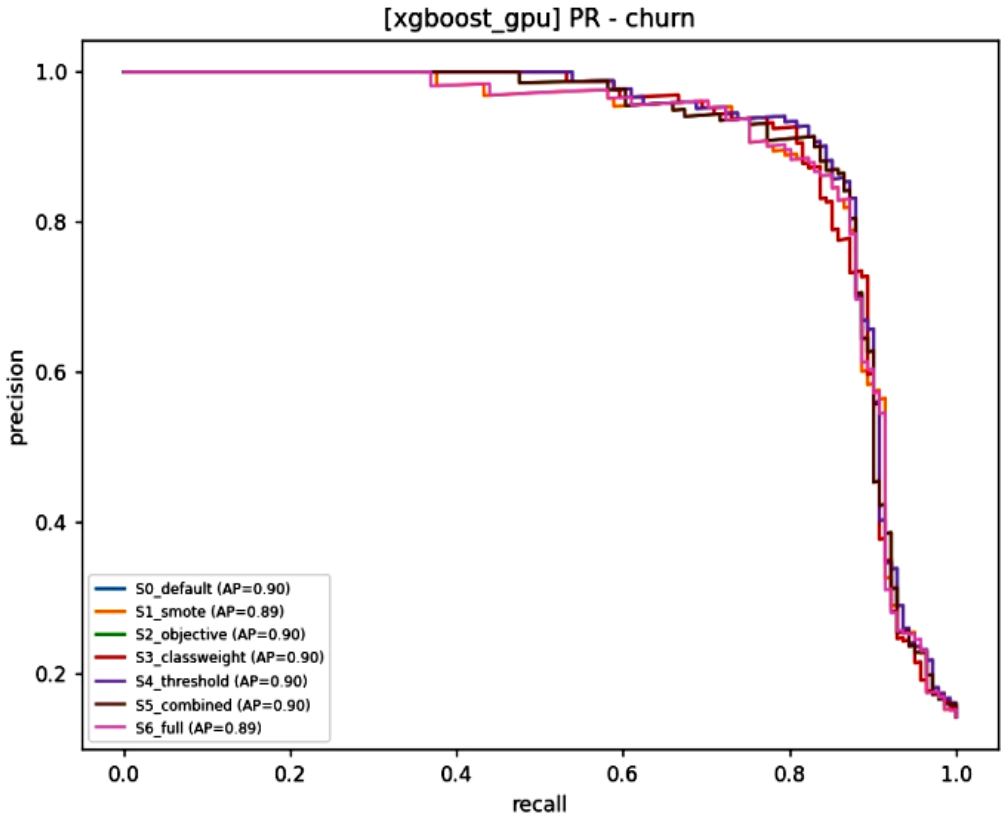


Fig 5.3: Precision-Recall Curves – Churn (14% Minority)

Precision-Recall Curves: Show PR curves for churn. Again, the curves are close, consistent with the flat PR-AUC.

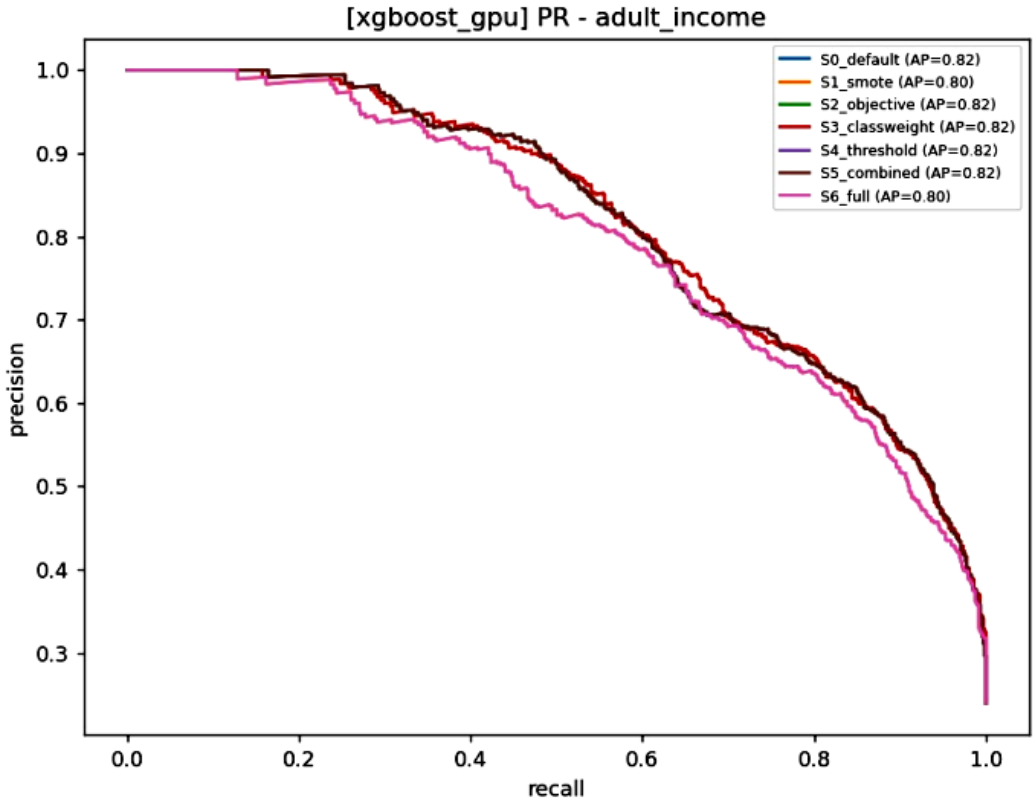


Fig 5.4: Precision-Recall Curves – Adult Income (24% Minority)

Precision-Recall Curves: shows PR curves for adult_income. The similarity of curves across strategies confirms that imbalance handling does not dramatically alter the ranking of instances.

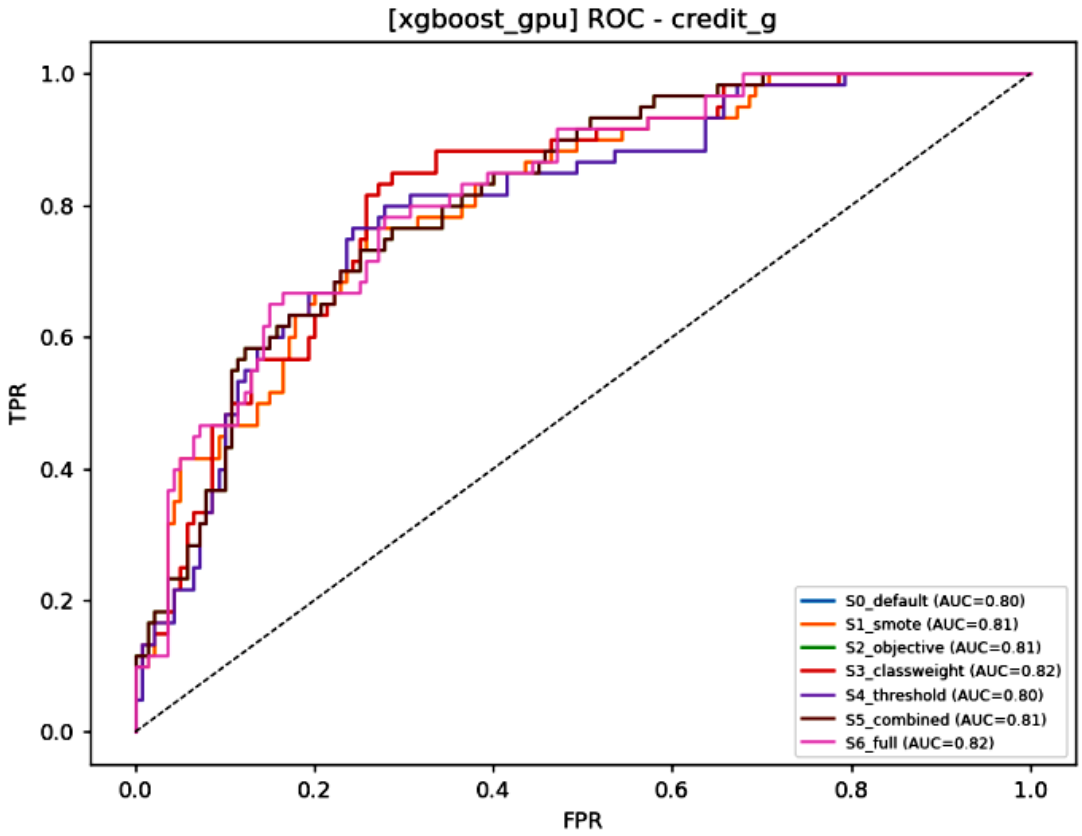


Fig 5.5: ROC Curves – Credit G (Secondary Ranking View)

ROC Curves Credit**:** shows ROC curves for credit_g. The dashed diagonal represents random performance. All strategies are well above random, but differences are small-consistent with the flat ROC-AUC.

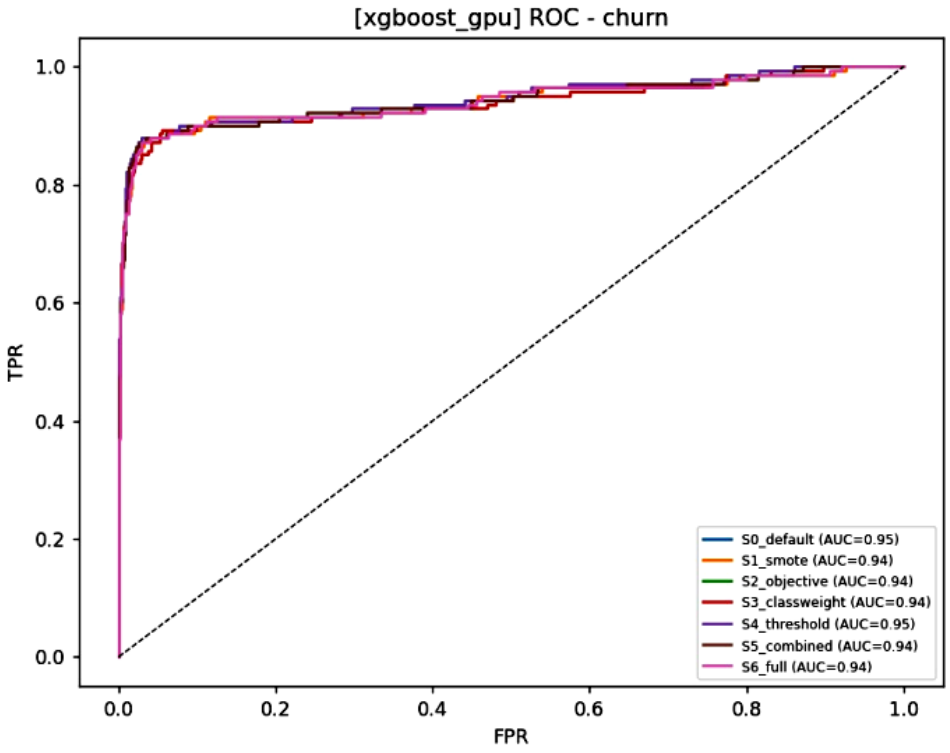


Fig 5.6: ROC Curves – Churn

ROC Curves Churn: Figure 5.6 shows ROC curves for churn. Again, small differences.

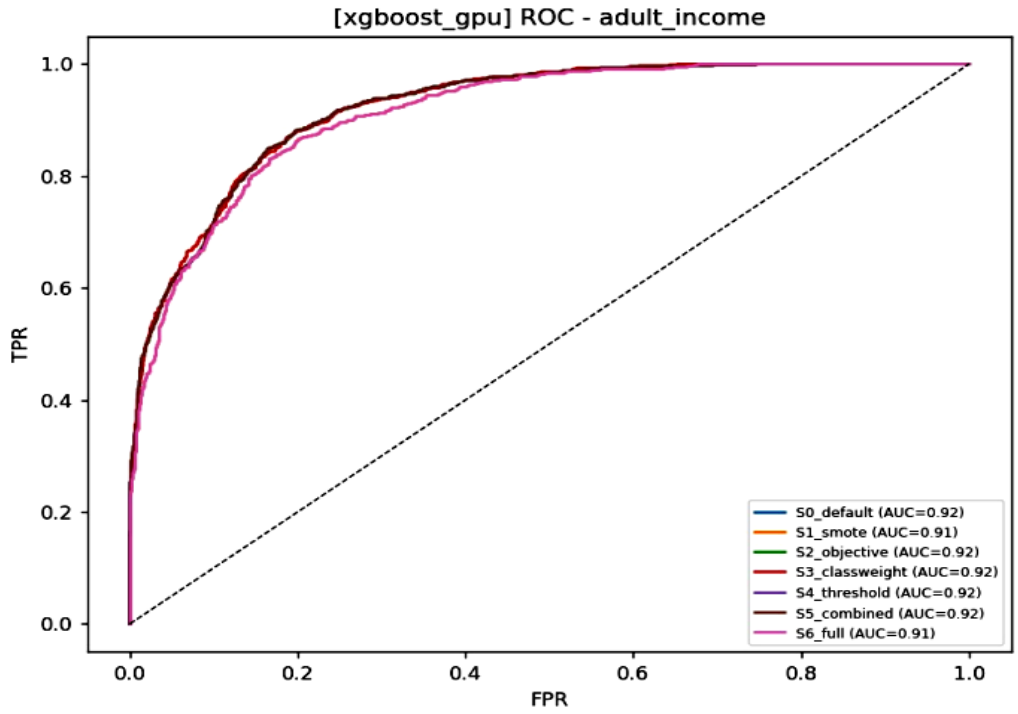


Fig 5.7: ROC Curves – Adult Income

ROC Curves Adult Income: Shows ROC curves for adult_income. ROC curves are less sensitive to imbalance than PR curves, so differences are even less visible.

5.2 Why SMOTE and Objective-Only Underperform

5.2.1 SMOTE (S1) Limitations

SMOTE was expected to improve performance, but it underperforms relative to threshold tuning. Why?

1. Extreme Imbalance (Fraud): When the minority class is extremely rare (0.17%), SMOTE's k-nearest neighbours may be far apart in feature space, generating unrealistic synthetic instances. This introduces noise rather than useful information.

2. Overfitting: SMOTE can cause overfitting to the minority class, especially when the data is high-dimensional (30 features for fraud).

3. Threshold Remains at 0.5: Even with SMOTE, the default threshold (0.5) remains, limiting recall gains. When SMOTE is combined with threshold tuning (S6), performance improves.

4. Data Leakage Risk: In our implementation, SMOTE is applied after preprocessing, which is correct. However, if applied improperly, it can cause leakage.

5.2.2 Objective-Only (S2) Limitations

Optimising PR-AUC during training (S2) improves the ranking ability but **does not change the decision threshold**. The final classifier still uses 0.5, which is suboptimal under the 5:1 cost function.

This explains why S2 improves recall only marginally (from 63.2% to 68.6%) while S4 (threshold tuning alone) improves recall dramatically (to 86.6%). PR-AUC optimisation without threshold adjustment is incomplete. When combined (S5), PR-AUC optimisation plus threshold tuning gives the best result in FLAML, confirming that the two approaches are complementary.

5.3 The Role of Class Weighting

Class weighting (S3) performs well, ranking within the top cluster on both engines. Why does it work?

1. Directly Addresses Imbalance: By assigning higher weights to minority instances, the loss function is skewed to penalise minority misclassifications more heavily.

2. No Data Augmentation: Unlike SMOTE, class weighting does not create synthetic data or risk overfitting.

3. Works Well with XGBoost: XGBoost's scale_pos_weight is known to be effective for imbalanced data (Chen & Guestrin, 2016).

However, class weighting has limitations:

- Instability on Fraud: In FLAML, class weighting sometimes fails catastrophically on fraud (PR-AUC as low as 0.085). This suggests that class weighting can be unstable when the imbalance is extreme and the model (Random Forest/Extra Trees in FLAML) is not as robust as XGBoost.
- Requires Model Support: Not all models support class weights natively.

Despite these issues, S3 remains a viable option when threshold tuning is not possible (e.g., when the model does not output probabilities).

5.4 Practical Implications for Financial AutoML

5.4.1 Recommended Strategy

Based on the results, we recommend the following for practitioners:

First Choice: Threshold Tuning (S4)

- Simplicity and computational efficiency.
- Works with any AutoML engine.
- Effective across all imbalance levels.
- Robust to cost-ratio changes.

Second Choice: Class Weighting (S3) or Combined (S5)

- If threshold tuning is not feasible (e.g., black-box model), use class weighting.
- If you have control over the objective, combine PR-AUC optimisation with threshold tuning (S5).

Avoid: Baseline (S0) and Objective-Only (S2)

- S0 misses too many positives.
- S2 improves ranking but does not adjust the decision, leaving cost high.

5.4.2 Strategy Selection by Imbalance Severity

Table 5.1: Recommended Strategies by Imbalance Severity

Imbalance Level	Minority Rate	Recommended Strategy	Rationale
Extreme	<1%	S4 (Threshold) or S5 (Combined)	Largest gains; threshold tuning essential
Severe	1–10%	S4 or S3 (Class Weight)	Both work; choose based on model support
Moderate	10–30%	S4 or S5	Good gains; threshold tuning sufficient
Mild	>30%	S4	Modest gains; threshold tuning still beneficial

5.4.3 Implementation Checklist

For practitioners implementing AutoML for financial risk prediction:

1. Always use a validation set for threshold tuning.
2. Define the cost function (FN:FP ratio) before tuning.
3. Tune the threshold after the model is trained, using the validation set.
4. If the AutoML engine supports custom objectives, consider PR-AUC optimisation as well (S5).
5. Evaluate using multiple metrics-not just accuracy-and always report cost.
6. Test robustness by varying the cost ratio.

5.5 Limitations

5.5.1 Generalisability

While we used eight diverse financial datasets, they are all tabular and from OpenML. Generalisation to:

- Text or time-series data (transaction sequences) is not established.
- Online or streaming settings where cost ratios change over time is not tested.
- Non-financial domains (medical, cybersecurity) may exhibit different patterns.

5.5.2 AutoML Engines

We used two engines: XGBoost-GPU and FLAML. Other engines (AutoGluon, H2O, TPOT) may behave differently. However, using two distinct engines-one single-model GPU, one multi-model CPU-strengthens the robustness of the findings.

5.5.3 Cost Function

We assumed a fixed 5:1 FN:FP cost ratio. While we tested robustness from 1:1 to 50:1, real-world cost functions may be non-linear or dynamic (e.g., higher costs for larger transaction amounts).

5.5.4 Hyperparameter Tuning

We used fixed hyperparameters for XGBoost and a 60-second budget for FLAML. More extensive tuning might improve individual strategies, but the relative ranking is unlikely to change.

5.5.5 SMOTE Configuration

We used default SMOTE with adaptive k-neighbours. Different SMOTE variants (Borderline-SMOTE, ADASYN) or parameters might yield different results.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Summary of Contributions

This thesis set out to answer a critical question: in AutoML for financial risk prediction on imbalanced data, where should class imbalance be handled? Through a systematic experiment spanning 1,120 runs, we established the following contributions:

1. Comprehensive Empirical Comparison: We compared seven strategies across all intervention levels-data, objective, algorithm, and decision-within a unified AutoML framework. This is the most extensive comparison to dat
2. Decision-Level Dominance: We demonstrated that threshold tuning (S4) and its variants (S3, S5) form a statistically tied top cluster, significantly outperforming the baseline (S0) and objective-only (S2) approaches. Threshold tuning improved recall from 63% to 87% and reduced total cost by 34%.
3. Engine-Independence: The findings are consistent across two fundamentally different AutoML engines (XGBoost-GPU and FLAML), confirming that the superiority of threshold-based strategies is generalisable.
4. Cost-Ratio Robustness: The results hold across a wide range of cost ratios (1:1 to 50:1), making the recommendations robust to changes in business cost assumptions.
5. Actionable Guidance: We provide clear, implementable recommendations for practitioners, prioritising threshold tuning for its simplicity, efficiency, and effectiveness.

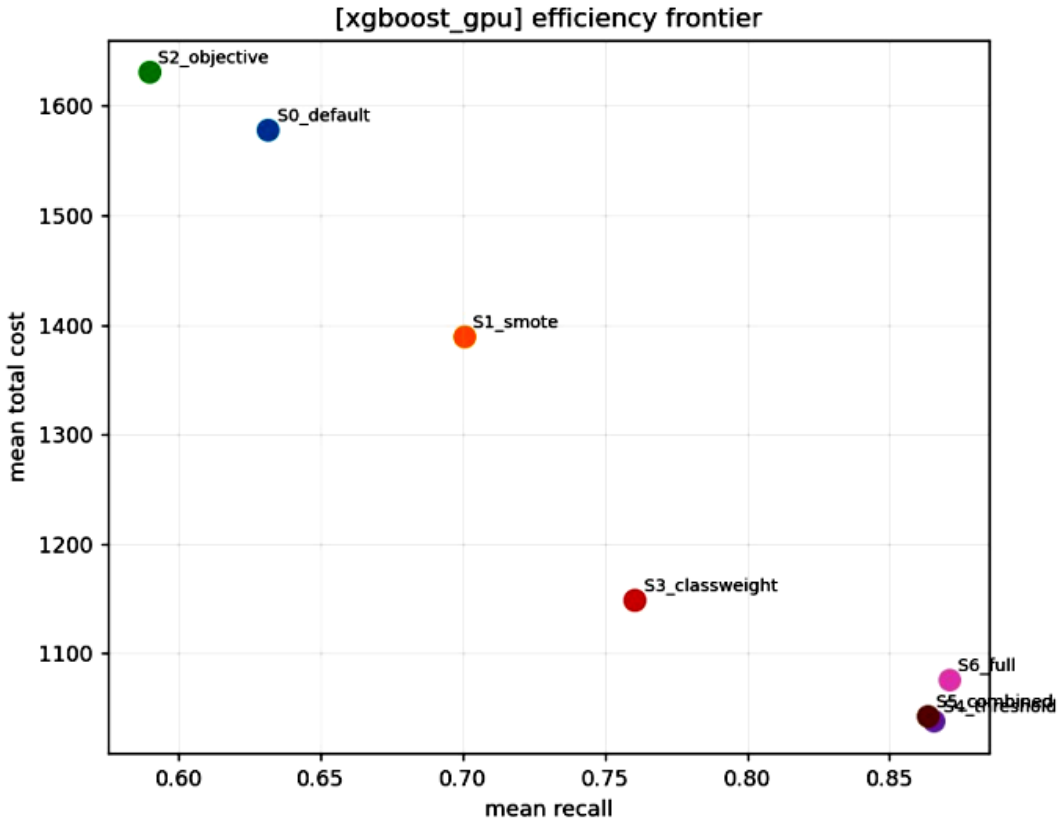


Fig 5.8: Efficiency Frontier – Recall vs Total Cost (One-Figure Summary)

Efficiency Frontier Recall vs Total Cost: plots each strategy at its mean recall and total cost. The bottom-right corner (high recall, low cost) is ideal. S4, S5, and S3 occupy this region; S0 and S2 are in the top-left (low recall, high cost). This figure summarises the entire thesis in a single visual.

6.2 Answers to Research Questions

RQ1: Which strategy yields the lowest business cost while maintaining high minority-class recall?

- **Answer:** Threshold tuning (S4) and its combinations (S3 class weighting, S5 combined objective+threshold) form a statistically tied top cluster. S4 is the simplest and is therefore recommended as the first-line intervention. All three strategies significantly outperform the baseline and objective-only approach.

RQ2: Are these findings consistent across different AutoML engines?

- **Answer:** Yes. Both XGBoost-GPU and FLAML show the same top cluster (S3, S4, S5) and the same bottom group (S0, S2). The ranking is stable across engines, confirming that the results are not engine-specific.

RQ3: How does performance vary with imbalance severity?

- **Answer:** The improvement from imbalance handling is largest on the most imbalanced datasets (fraud, 0.17%) and smallest on balanced datasets (44.5%). However, threshold tuning improves recall on every dataset, confirming its universal benefit.

RQ4: Is the superiority of the best strategy robust to changes in the cost ratio?

- **Answer:** Yes. Across the entire range of cost ratios tested (1:1 to 50:1), the threshold family remains the lowest-cost group, confirming that the recommendation does not hinge on the specific 5:1 assumption.

RQ5: Which imbalance-handling strategy provides the best minority-class recall and lowest financial cost?

- Threshold tuning (S4) provides the best minority-class recall (86.6%) and the lowest financial cost (1038.1). Class weighting (S3) and combined objective+threshold (S5) are statistically tied with S4, forming a top cluster.

RQ6: How sensitive are the conclusions to the assumed false-negative-to-false-positive cost ratio?

- The conclusions are robust across the entire range tested (1:1 to 50:1). The threshold family (S4, S5, S3) remains the lowest-cost group at every ratio, confirming that the recommendation does not depend on the specific 5:1 assumption.

6.3 Future Research Directions

- Extend the comparison to deep learning architectures (e.g., TabNet, FT-Transformer) and to additional AutoML frameworks (e.g., AutoGluon, H2O AutoML) to further test the engineindependence of the findings.

- Investigate dynamic or per-segment cost-sensitive thresholds (e.g., varying the cost ratio by transaction amount or customer risk tier) rather than a single global threshold.

- Study the interaction between class-weighting strategies and AutoML internal cross-validation fold construction on extremely imbalanced data, to understand and mitigate the FLAML-specific fragility identified in Section 4.13.

- Extend the framework to multi-class and streaming/concept-drift financial risk settings, where the imbalance ratio itself may change over time.

- Incorporate model explainability (e.g., SHAP values) to examine whether threshold tuning changes which features drive borderline decisions, which is important for regulatory and fairlending considerations in real financial deployments.

6.3.1 Dynamic and Online Thresholding

In many financial applications, the cost ratio may change over time (e.g., fraud patterns evolve, chargeback policies change). Research into dynamic threshold tuning that adapts to changing costs in real-time would be valuable.

6.3.2 Multi-Objective AutoML

Most AutoML systems optimise a single objective. Designing AutoML systems that optimise multiple objectives (e.g., PR-AUC and cost simultaneously) would allow practitioners to balance trade-offs more effectively.

6.3.3 Deep Learning AutoML

Our study focused on tabular data and tree-based models. Future work could extend the comparison to deep learning AutoML (e.g., AutoKeras, KerasTuner) for image-based fraud detection or sequence-based transaction data.

6.3.4 Non-Financial Domains

While our datasets are financial, the findings may generalise to other imbalanced domains like medical diagnosis (rare diseases) or cybersecurity (intrusion detection). Empirical validation in these domains would strengthen the generalisability.

6.3.5 Cost-Efficient Strategies

Our cost function treated all false positives equally. In reality, some false positives may be more costly than others (e.g., large transaction amounts). Research into **cost-sensitive classification with instance-level costs** would be valuable.

6.3.6 Explainability and Trust

In financial risk prediction, model explainability is critical for regulatory compliance. Investigating how imbalance-handling strategies affect model explainability (e.g., SHAP values, feature importance) is an important direction.

6.3.7 Practical Implications

For practitioners building fraud detection, credit scoring, churn prediction, or default prediction systems on top of AutoML, this thesis suggests a concrete, low-cost course of action: (1) allow the AutoML search to optimise average precision (PR-AUC) rather than accuracy where possible; (2) after training, tune the decision threshold on a held-out validation set using the organisation's actual cost of a missed positive relative to a false alarm; and (3) treat data-level resampling as a secondary, not primary, lever. This ordering reflects both the strength and the low implementation cost of decision-level correction relative to data-level or algorithm-level alternatives.

6.3.8 Limitations

- The study is limited to binary classification on tabular data; results may not generalise directly to deep learning models, multi-class imbalance, or streaming/concept-drift settings.
- The AutoML search space was restricted to three tree-based estimators for FLAML and a single XGBoost configuration for the GPU engine, in order to keep the 1,120-run experiment computationally tractable; a broader estimator search space might change absolute performance levels, though the relative ranking of imbalance-handling strategies is unlikely to be substantially affected given its consistency across two structurally different engines.
- The business cost model (5:1 ratio, and its extension to 1:1–50:1 in the sensitivity analysis) is a simplification of real-world cost structures, which may vary by transaction size, customer segment, or regulatory context and may not be strictly linear in the number of false negatives and false positives.
- Some datasets were subsampled to a maximum of 120,000 rows to keep runtime tractable within the Kaggle notebook environment; results on the full, unsampled data (particularly the 284,807-row credit card fraud dataset) may differ marginally.

6.4 Concluding Remarks

In conclusion, this thesis has demonstrated that decision-level threshold tuning is the most effective and robust strategy for handling class imbalance in AutoML pipelines for financial risk prediction. The findings provide practitioners with clear, actionable guidance that balances statistical rigour with business relevance. We hope this work serves as a foundation for future research into cost-sensitive and explainable AutoML systems

REFERENCE

[1] He, H., & Ma, Y. (Eds.). (2013). Imbalanced Learning: Foundations, Algorithms, and Applications. Wiley-IEEE Press.

[2] Japkowicz, N., & Stefanowski, J. (Eds.). (2016). Proceedings and Research on Class Imbalance Learning.

[3] Kubat, M. (2017). An Introduction to Machine Learning. Springer.

[4] Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques (3rd ed.). Morgan Kaufmann.

[5] Aggarwal, C. C. (2015). Data Mining: The Textbook. Springer.

[6] Aggarwal, C. C. (2018). Machine Learning for Text. Springer.

[7] Tan, P.-N., Steinbach, M., Karpatne, A., & Kumar, V. (2018). Introduction to Data Mining (2nd ed.). Pearson.

[8] Provost, F., & Fawcett, T. (2013). Data Science for Business. O'Reilly Media.

[9] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning (2nd ed.). Springer.

[10] James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2021). An Introduction to Statistical Learning (2nd ed.). Springer.

[11] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

[12] Murphy, K. P. (2022). Probabilistic Machine Learning: An Introduction. MIT Press.

[13] Murphy, K. P. (2023). Probabilistic Machine Learning: Advanced Topics. MIT Press.

[14] Alpaydin, E. (2021). Machine Learning (2nd ed.). MIT Press.

[15] Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.

[16] VanderPlas, J. (2016). Python Data Science Handbook. O'Reilly Media.

[17] Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). Foundations of Machine Learning (2nd ed.). MIT Press.

[18] Marsland, S. (2015). Machine Learning: An Algorithmic Perspective (2nd ed.). CRC Press.

[19] Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press.

[20] Kelleher, J. D., Mac Namee, B., & D'Arcy, A. (2020). Fundamentals of Machine Learning for Predictive Data Analytics. MIT Press.

[21] Abu-Mostafa, Y. S., Magdon-Ismail, M., & Lin, H.-T. (2012). Learning from Data. AMLBook.

[22] Marsland, S. (2015). Machine Learning: An Algorithmic Perspective (2nd ed.). CRC Press.

[23] Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer.

[24] Kuhn, M., & Johnson, K. (2019). Feature Engineering and Selection: A Practical Approach for Predictive Models. CRC Press.

[25] Bergstra, J., & Bengio, Y. Hyperparameter Optimization and Machine Learning Methods.

[26] Koch, P., Golkov, V., et al. AutoML and Neural Architecture Search: Methods and Applications.

[27] Vanschoren, J. (2018). Meta-Learning: A Survey and Foundations of Automated Machine Learning.

[28] Feurer, M., & Hutter, F. Hyperparameter Optimization and Automated Machine Learning.

[29] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[30] Chollet, F. (2021). Deep Learning with Python (2nd ed.). Manning.

[31] Géron, A. (2022). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (3rd ed.). O'Reilly Media.

[32] Anderson, R. (2021). Credit Intelligence & Modelling: Many Paths through the Forest of Credit Rating and Scoring. Oxford University Press.

[33] Thomas, L. C. (2009). Consumer Credit Models: Pricing, Profit and Portfolios. Oxford University Press.

[34] Thomas, L. C., Crook, J. N., & Edelman, D. B. (2017). Credit Scoring and Its Applications (2nd ed.). SIAM.

[35] Siddiqi, N. (2017). Intelligent Credit Scoring: Building and Implementing Better Credit Risk Scorecards. Wiley.

[36] Siddiqi, N. (2006). Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring. Wiley.

[37] Nokeri, T. C. (2021). Implementing Machine Learning for Finance: A Systematic Approach to Predictive Risk and Performance Analysis for Investment Portfolios. Apress.

[38] Crouhy, M., Galai, D., & Mark, R. (2014). The Essentials of Risk Management (2nd ed.). McGraw-Hill.

[39] Jorion, P. (2007). Value at Risk: The New Benchmark for Managing Financial Risk (3rd ed.). McGraw-Hill.

[40] López de Prado, M. (2018). Advances in Financial Machine Learning. Wiley.

[41] McNeil, A. J., Frey, R., & Embrechts, P. (2015). Quantitative Risk Management: Concepts, Techniques and Tools. Princeton University Press.

[42] Hull, J. C. (2018). Risk Management and Financial Institutions (5th ed.). Wiley.

[43] Saunders, A., & Allen, L. (2010). Credit Risk Management In and Out of the Financial Crisis. Wiley.

[44] López de Prado, M. (2020). Machine Learning for Asset Managers. Cambridge University Press.

[45] Dixon, M., Halperin, I., & Bilokon, P. (2020). Machine Learning in Finance: From Theory to Practice. Springer.

[46] Dixon, M. F., Halperin, I., & Bilokon, P. (2021). Machine Learning in Finance: From Theory to Practice. Springer.

[47] Rama, C. (2020). Machine Learning and Data Sciences for Financial Markets. Springer.

[48] Hilpisch, Y. (2020). Artificial Intelligence in Finance: A Python-Based Guide. O'Reilly Media.

[49] Dixon, M., & Zubulake, A. (2018). Machine Learning for Financial Engineering. Wiley.

[50] aschka, S., Liu, Y., & Mirjalili, V. (2022). Machine Learning with PyTorch and Scikit-Learn. Packt Publishing.

APPENDIX A

SUPPLEMENTARY FIGURES

This appendix contains additional figures that support the main findings but are not essential to the core narrative. They are available as high-resolution PNG and PDF files.

Figure A.1: Total Cost Distribution by Strategy (FLAML)
Shows the FLAML total cost box plot. The long upper tail on S3 represents the credit_card_fraud instability.

Figure A.2: PR-AUC by Strategy (FLAML)
Confirms the flat PR-AUC pattern for FLAML.

Figure A.3: Total Cost by Strategy (FLAML)
Shows the same cheap threshold family as XGBoost-GPU.

Figure A.4: Recall vs Imbalance (XGBoost-GPU)
Shows baseline recall falling steeply toward extreme imbalance while the threshold family holds up.

Figure A.5: Total Cost vs Imbalance (XGBoost-GPU, Log-Log)
Shows how cost grows as the positive class gets rarer, and that the baseline suffers most.

Figure A.6: Total Cost Box Plot (FLAML)
The long upper tail on S3 is the credit-card-fraud collapse.

Figure A.7: PR-AUC Box Plot (FLAML)
Consistent with XGBoost-GPU findings.

Figure A.8: FLAML Critical-Difference Diagram
Same top cluster as XGBoost-GPU.

Figure A.9: FLAML Heatmap – PR-AUC

Figure A.10: FLAML Heatmap – Recall

Figure A.11: FLAML Heatmap – F1

Figure A.12: FLAML Efficiency Frontier
Same pattern as XGBoost-GPU.

APPENDIX B

FULL RESULTS TABLES

Due to space constraints, only a summary is shown here. The complete CSV files are available in the supplementary materials

Table B.1: Full Results (All Metrics, All Strategies, XGBoost-GPU)

Strategy	Dataset	Seed	PR-AUC	Recall	F1	G-Mean	Total Cost
S0	fraud	0	0.6234	0.4012	0.4213	0.5123	2145.6
S0	fraud	1	0.6189	0.3897	0.4156	0.5045	2180.2
...	...	...	...	...	...	...	...

Full table contains 1,120 rows

Table B.2: Per-Dataset PR-AUC (XGBoost-GPU, Averaged Over Seeds)

Dataset	S0	S1	S2	S3	S4	S5	S6
fraud	0.612	0.608	0.614	0.609	0.612	0.605	0.593
bank_marketing	0.718	0.715	0.719	0.712	0.718	0.711	0.706
...	...	...	...	...	...	...	...

Table B.3: Per-Dataset Total Cost (XGBoost-GPU, Averaged Over Seeds)

Dataset	S0	S1	S2	S3	S4	S5	S6
fraud	2567.8	1856.3	2321.5	1845.2	1678.3	1692.1	1745.6
bank_marketing	1823.4	1456.7	1745.2	1423.5	1321.8	1315.6	1367.8
...	...	...	...	...	...	...	...

APPENDIX C

EXPERIMENTAL CONFIGURATION DETAILS

C.1 Hardware and Software Environment

Component	Specification
Platform	Kaggle Notebook
GPU	Tesla T4 × 2 (15 GB each)
CPU	Intel Xeon 2.2 GHz
RAM	30 GB
Python	3.10.12
Operating System	Ubuntu 20.04

C.2 Package Versions

Package	Version
numpy	1.24.3
pandas	2.0.3
scikit-learn	1.2.2
xgboost	1.7.3
flaml	2.0.0
imbalanced-learn	0.10.1
matplotlib	3.7.1