Predicting Telecom Customer Churn with Explainable Machine Learning

Instructor: Dr. Clinton Watkins

Chih-Jung (Audrey) Chang (B2211687)

2025-12-01

Abstract

Customer churn—the cancellation of service or switching to a competitor—remains a major source of revenue leakage for telecommunications providers, especially in mature markets where customer acquisition is costly and growth is limited. This study develops a leakage-safe, end-to-end machine learning pipeline to predict churn using the Cell2Cell telecom dataset, and benchmarks multiple model families (logistic regression, tree-based models, gradient boosting, neural networks, and stacking ensembles) under a consistent experimental design. To support real-world decision making, evaluation emphasizes metrics suitable for imbalanced labels (ROC-AUC, PR-AUC, and F1/recall at an operating threshold), and model interpretability is treated as a first-class requirement. Using explainability techniques (e.g., SHAP), the final system translates predictive signals into actionable retention drivers and prioritization lists that can guide targeted outreach and A/B testing.

Keywords: Telecom churn; supervised learning; gradient boosting; stacking ensemble; imbalanced classification; explainable machine learning; SHAP

Table of contents

1 Introduction

1.1 Background & Motivation

Customer churn, defined as customers canceling service or switching to a competing provider, is a major driver of revenue loss in the telecommunications industry. In saturated mobile markets, where growth through new subscriptions is constrained, retaining existing customers becomes strategically as important as acquiring new ones. Prior telecom studies document substantial churn levels and the operational difficulty of stabilizing subscriber bases over time (Ahn, Han, and Lee 2006; Wei and Chiu 2002).

Churn directly erodes recurring revenue and increases operating costs because replacement typically requires additional spending on marketing, onboarding, and incentives. Retention-oriented marketing research further suggests that acquiring a new customer may cost multiple times more than retaining an existing customer, reinforcing why churn prevention is frequently treated as a high-impact managerial priority (Šonková and Grabowska 2015; Rosenberg and Czepiel 1984). Customer churn can also weaken longer-term customer equity by reducing lifetime value and creating a compounding revenue gap in subscription settings (Geiler, Affeldt, and Nadif 2022).

These economic and strategic pressures explain why churn prediction has become a standard analytics application in customer relationship management. By identifying high-risk customers in advance, telecom operators can prioritize proactive interventions and allocate retention budgets more efficiently. However, in operational contexts, predictive performance alone is insufficient. Retention teams typically require explanations of why a customer is flagged as high risk, because different churn drivers imply different actions (e.g., service quality remediation vs. price-based incentives). As a result, the churn modeling task is inherently dual-purpose: producing accurate risk scores while enabling actionable interpretation that supports decision-making.

1.2 Research Gap

Two persistent gaps motivate the present study.

First, although many churn prediction studies have progressively adopted higher-capacity machine learning models, transparency and managerial actionability are not consistently addressed. Early work frequently relied on interpretable models such as logistic regression and decision trees, which can provide direct insight into drivers but may underperform when churn patterns are nonlinear or driven by complex feature interactions (Lemmens and Croux 2006; Bogaert and Delaere 2023). Recent work has moved beyond linear baselines and highlights ensemble and other flexible nonlinear models, including random forests, gradient boosting, support vector machines, and neural networks, which have been shown to improve predictive performance in churn prediction tasks (Vafeiadis et al. 2015; Bogaert and Delaere 2023). At the same time, this shift can reduce transparency, making it harder to convert model outputs into targeted retention actions.

Second, evaluation practices and "real-world" data constraints remain unevenly handled across the literature. Telecom churn data are commonly imbalanced, and naive reporting of accuracy can be misleading because a model can appear strong while failing to identify churners effectively (Burez and Van Den Poel 2009; Akosa 2017). Empirical work comparing imbalance-handling strategies highlights that model selection and thresholding decisions materially affect minority-class performance, yet such considerations are not always reflected in standard reporting practices (Zhu et al. 2023). Recent reviews further emphasize that churn prediction research continues to face challenges related to robustness and practical deployment considerations, including mismatches between experimental assumptions and production settings (Imani 2024).

Together, these gaps indicate a need for churn research that simultaneously (i) benchmarks strong predictive model families under a consistent experimental design and (ii) treats interpretability and decision-oriented evaluation as core requirements, not as secondary considerations.

1.3 Research Questions

To address the above gaps, this study is organized around three research questions:

- **RQ1 (Model Effectiveness)**: Under a consistent telecom churn data setting, which model family provides the strongest overall performance among linear baselines, tree-based models, gradient boosting methods, tabular neural networks, and heterogeneous ensembles?
- **RQ2 (Imbalance Robustness)**: Under class imbalance, which modeling and decision strategies yield more stable churner detection, particularly when using class weighting, alternative loss functions (e.g., focal loss), and decision-threshold adjustment, rather than relying on default accuracy-driven evaluation?
- **RQ3 (Interpretability to Action)**: Can model interpretation outputs be transformed into actionable

churn drivers that support customer profiling, risk-factor discovery, and retention-oriented grouping, while remaining consistent with the predictive pipeline used for final evaluation?

1.4 Contributions

This study makes the following contributions, emphasizing a unified pipeline that balances predictive performance and interpretability:

- **Comparative modeling under consistent controls**: Multiple model families are implemented and evaluated in a standardized setting, spanning interpretable baselines and higher-capacity learners reported as strong performers in churn prediction research (Lemmens and Croux 2006; Bogaert and Delaere 2023; Lalwani et al. 2022).
- **Imbalance-aware evaluation emphasis**: The study foregrounds pitfalls of accuracy-dominated reporting and motivates decision-relevant evaluation under imbalance, consistent with prior findings in churn prediction methodology research (Burez and Van Den Poel 2009; Akosa 2017; Zhu et al. 2023).
- **Interpretability aligned with deployment needs**: The research treats interpretability as an operational requirement: model outputs are framed as inputs to retention actions (not solely as predictive scores), with the objective of producing explanations that support practical intervention design.
- **Planned extension beyond supervised learning**: In addition to supervised modeling, a semi-supervised direction is articulated as future work in response to practical constraints highlighted in recent reviews, acknowledging that real-world churn labeling can be incomplete or delayed (Imani 2024).

1.5 Paper Roadmap

The remainder of this thesis is structured as follows. Section 2 reviews related work on churn mechanisms, telecom data characteristics, modeling approaches, evaluation under class imbalance, and emerging research trends. Section 3 defines the dataset and prediction task, describes the experimental setup, and details preprocessing choices designed to support fair model comparison. Subsequent sections present the implemented modeling approaches, report experimental results, and provide interpretation-driven analysis aimed at connecting churn prediction to actionable retention insights.

2 Literature Review

Telecom Churn Prediction with Interpretable Machine Learning: From Behavioral Mechanisms to Modern Ensembles

2.1 Churn Theory and Behavioral Mechanisms

Telecom churn is often framed as a predictive classification problem, but the underlying phenomenon is a behavioral decision shaped by service experience, perceived value, and switching considerations. Service marketing research shows that customers frequently switch after adverse service encounters, dissatisfaction, or unmet expectations, while the dominant "reason for switching" can vary across customers and contexts (Keaveney 1995). Work on regain management further characterizes churn as a process of customer defection and potential recovery, where churn prevention and win-back should be evaluated jointly as part of relationship management rather than as isolated outcomes ("Regaining Service Customers - Bernd Stauss, Christian Friege, 1999" n.d.).

A related theme is that loyalty and churn are influenced by both attitudinal factors (e.g., satisfaction and preference) and behavioral constraints (e.g., switching costs and barriers), implying that churn drivers can be nonlinear and interactive (Dick and Basu 1994). In mobile telecommunications, empirical studies highlight the roles of satisfaction, perceived service quality, and switching barriers in shaping loyalty and churn outcomes (Gerpott, Rams, and Schindler 2001; H.-S. Kim and Yoon 2004; M.-K. Kim, Park, and Jeong 2004). These findings motivate a practical view for churn modeling: features such as usage and engagement variables, billing and pricing proxies, tenure or contract indicators, and service-quality-related signals can be interpreted as measurable correlates of satisfaction and switching frictions rather than as purely random predictors. Accordingly, later sections interpret model explanations not only as statistical associations but also as evidence consistent with churn mechanisms discussed in prior retention research.

2.2 Telecom Churn Data Characteristics and Practical Constraints

Telecom churn datasets typically combine heterogeneous information sources, including numerical usage measures, customer demographics, billing variables, and high-cardinality categorical descriptors (e.g., plan identifiers or geographic codes). These properties create recurring modeling challenges that shape churn pipelines and motivate robust, reproducible experimentation (Lalwani et al. 2022; Vafeiadis et al. 2015). In applied churn research, the most emphasized constraints include:

- **Class imbalance:** Churn is often a minority event, so naïve training and default thresholding can bias models toward the majority class. Prior work proposes a range of imbalance-handling strategies, including reweighting, resampling (over- or under-sampling), and imbalance-aware ensemble methods, each of which can influence both measured performance and the set of customers prioritized for intervention (Beeharry and Tsokizep Fokone 2022; Sikri et al. 2024).
- **Missingness and measurement noise:** Telecom records are operational logs, so missing values may reflect business processes rather than random absence, and churn labels depend on operational definitions that may introduce label ambiguity.
- **Leakage risk:** Features created after the churn window (or strong proxies of post-churn outcomes) can inflate performance if not controlled. Leakage prevention is therefore part of credible benchmarking.
- **Temporal instability:** Behavior, product offerings, and marketing policies evolve. Drift-focused work emphasizes that churn models can degrade as the data-generating process shifts, motivating monitoring, retraining, and robustness checks (Bugajev, Kriauzienė, and Chadyšas 2025). These constraints motivate end-to-end workflows where preprocessing, splitting, and evaluation are standardized across model families. Consistent data handling is especially important when comparing heterogeneous algorithms because small preprocessing differences can confound conclusions about which model family truly performs best.

2.3 Modeling Approaches for Telecom Churn Prediction

The churn literature spans interpretable baselines, high-capacity machine learning, and heterogeneous ensembles. These families differ in inductive bias, data requirements, and interpretability properties, which motivates systematic comparison within a unified pipeline.

2.3.1 Interpretable Baselines and Classical Models

Logistic regression and tree-based baselines remain widely used because they are transparent, stable, and straightforward to deploy in CRM environments. Benchmarking studies frequently treat such baselines as essential anchors even when more complex methods improve discrimination (Vafeiadis et al. 2015). Their limitations are also well recognized: linear decision boundaries may underfit nonlinear churn mechanisms, and shallow trees may sacrifice accuracy to maintain interpretability. For this reason, modern churn studies often implement baseline models not as end goals but as reference points for measuring incremental benefit from more complex approaches.

2.3.2 Ensemble Learning and Tree-based Methods

Ensemble learning is central in churn modeling because it can improve generalization by aggregating multiple learners and capturing nonlinear effects without extensive manual feature specification. Ensemble approaches have demonstrated benefits in churn prediction across domains (Coussement and De Bock 2013), and telecom-focused work similarly reports strong performance from tree ensembles on heterogeneous tabular data (Lalwani et al. 2022; Vafeiadis et al. 2015). Practical advantages include handling mixed feature types, automatically modeling interaction effects, and achieving high discrimination without deep representation learning.

2.3.3 Neural and Hybrid Approaches for Tabular Churn Prediction

Deep learning has also been explored for churn prediction, particularly when representation learning is expected to capture complex feature relationships or mitigate sparsity from categorical inputs. Empirical comparisons of supervised techniques have evaluated neural models alongside traditional methods, with results often depending on preprocessing choices, regularization, and dataset structure (Khodabandehlou and Rahman 2017). Neural-network-based churn prediction continues to appear in more recent surveys and applied studies, suggesting sustained interest in deep architectures for customer analytics (Thangeda, Kumar, and Majhi 2024). However, deep learning in tabular settings is commonly viewed as sensitive to feature encoding, hyperparameters, and imbalance-handling choices. Consequently, neural methods are most informative when evaluated within controlled benchmarking frameworks rather than assumed to universally outperform tree ensembles.

A recurring theme is that neural methods are most informative when evaluated within a controlled benchmarking framework. Therefore, in this thesis deep learning is treated as one model family among several,

compared under identical data splits and preprocessing constraints. This design enables a fair assessment of whether added capacity improves discrimination and stability on telecom-style tabular churn data, and it supports subsequent interpretability analyses.

2.3.4 Stacking and Heterogeneous Ensembles

Beyond bagging and boosting, stacking aims to exploit model diversity by combining base learners through a meta-learner trained on out-of-fold predictions. Telecom churn research suggests that stacking can improve performance when base models have complementary error profiles (Haddadi and Hamidi 2025). Related work further indicates that sampling-aware heterogeneous ensembles can create diversity via both algorithm choice and data views, which can be beneficial in imbalanced churn settings (Sikri et al. 2024). These findings support the use of heterogeneous stacking as a practical strategy when multiple strong models exist but no single method dominates across all evaluation dimensions.

2.4 Evaluation, Class Imbalance, and Explainability

Because churn prediction is operationalized as a ranking task for targeting retention actions, evaluation must go beyond overall accuracy. The churn literature emphasizes that imbalanced settings require metrics sensitive to minority-class performance and practical decision trade-offs (Beeharry and Tsokizep Fokone 2022; Jiao and Xu 2021; Sikri et al. 2024). In practice, this motivates reporting multiple complementary views of performance, including:

- Discrimination metrics (e.g., ROC-AUC) to assess ranking ability across thresholds
- Precision–recall trade-offs and related measures when churn is rare and false positives are costly
- Threshold- or top-k analyses to align model outputs with limited retention capacity Systematic reviews further emphasize that metric selection should reflect deployment goals and dataset imbalance rather than relying on a single score (Zhu et al. 2023). Accordingly, later sections report a metric set designed to support realistic retention decision-making and to compare models fairly under imbalance.

Because churn models are often used as probability scores to prioritize intervention, calibration and threshold selection are also practically important: a model can rank customers well but still produce probabilities that are systematically over- or under-confident. As a result, later sections complement discrimination metrics with calibration diagnostics and threshold-based analyses aligned to realistic retention capacity constraints.

Beyond predictive discrimination, several studies emphasize linking churn scores to customer profiling and segmentation so that retention actions can be tailored. In this view, churn prediction is coupled with "who" and "why" analyses that identify groups of customers with distinct churn drivers, supporting more targeted marketing and service interventions (Geiler, Affeldt, and Nadif 2022). This motivates treating modeling and interpretability as a joint design problem: stronger models are valuable insofar as their outputs remain explainable and deployable for retention strategy design.

Explainability is increasingly treated as essential for making churn models actionable. While baselines provide transparent coefficients or decision paths, ensembles and neural networks often require post-hoc methods. Recent churn-related studies demonstrate SHAP-based explanations as a practical approach to produce both global summaries (portfolio-level drivers) and local explanations (customer-level reasons for high risk) (Asif, Arif, and Mukheimer 2025; Poudel, Pokharel, and Timilsina 2024). Complementary tools for visualizing nonlinear feature effects are also commonly discussed, supporting interpretation in cases where churn drivers interact or exhibit threshold effects. This perspective motivates integrating explainability directly into the modeling pipeline. In this thesis, explainability is not treated as an afterthought but as a parallel analysis layer that supports error analysis, segment comparisons, and calibration/threshold discussions, thereby connecting predictive gains to interpretable churn mechanisms.

2.5 Open Issues and Emerging Trends

Several open issues remain. Concept drift and temporal instability are persistent challenges in telecom churn because customer behavior and market conditions evolve; drift analyses reinforce the need for monitoring and periodic retraining if churn models are deployed in production (Bugajev, Kriauzienė, and Chadyšas 2025). Another direction is moving from "who will churn" to "who should be targeted," where retention research argues that targeting customers solely by predicted churn risk can be suboptimal when high-risk customers are not always influenceable or profitable to retain. These issues motivate broader frameworks that integrate predictive performance with business value and intervention design.

Within this landscape, the empirical contribution of the present thesis is positioned as an engineering-grade, end-to-end churn prediction pipeline that systematically compares model families (interpretable baseline, tree ensembles, deep learning, and stacking) under realistic constraints such as imbalance and leakage control, while using SHAP-centered explainability to connect predictive outputs to plausible churn mechanisms and actionable retention insights.

3 Data and Problem Setup

3.1 Data sources and study scope

This study utilizes a single primary dataset, cell2celltrain.csv, a customer-level telecom churn dataset consisting of 51,047 customer records and 58 original columns (including the identifier and raw target label). To facilitate binary classification modeling, the original string-based churn label was encoded into a binary numerical format (1/0) to serve as the final prediction target. Each row corresponds to one customer, with CustomerID serving as a unique identifier; no duplicate CustomerID values were detected.

To make feature handling auditable and reproducible across notebooks, the project also maintains a configuration-driven feature registry (built from feature_config_clean.csv) that specifies feature types (numeric/binary/ordinal/nominal), semantic groups, and risk/processing tags (e.g., "high_card", "sparse_zero", "zero_as_missing"). This design supports consistent "data scope" decisions (e.g., dropping identifiers, excluding leakage-prone fields) without hard-coding per-notebook logic.

In addition, a Detailed Feature Decision Table and a Data Dictionary were created as formal documentation of the dataset's columns and the feature inclusion/exclusion rationale. The decision table is treated as a "single source of truth" for feature-level decisions and is provided in Section A.2.1 (Table A.1).

3.2 Prediction target and observation window

The modeling task is a binary classification problem: predicting whether a customer will churn. The dataset's churn definition is event-based: "customers who churned 31- 60 days later" are labeled as churners. Many behavioral variables are constructed as historical summaries (e.g., four-month means), which aligns with a practical churn prediction setting where features represent pre-outcome customer state rather than post-churn operational actions.

From a business perspective, this framing supports proactive retention: the goal is to rank customers by churn risk so that limited retention resources can be targeted toward those most likely to leave.

3.3 Data documentation and feature organization

Because the dataset contains heterogeneous feature families (usage, billing, tenure, device, demographics, and operational indicators), EDA was used not only to summarize distributions but also to organize features into consistent groups for later modeling and interpretation. The project produced reusable documentation outputs, including a feature registry table (Section A.2.1) and a semantic grouping table (Section A.2.2).

At a high level, the raw columns include a mixture of numeric, integer, and object-typed variables (29 float64, 7 int64, 22 object in the raw load). A structured type and tag system was maintained for each feature (e.g., some features are tagged as high-cardinality, some as zero-as-missing, some as sparse-zero), enabling later preprocessing and model design to follow a documented specification rather than ad hoc decisions.

Table 3.1: Feature registry excerpt (top 5 rows).

	feature	origin	semantic_group	type	transform_encoding
0	CustomerID	Original	id_target	Identifier (ID)	–
1	Churn	Original	id_target	Binary target	Encoded to 0/1 (Churn01)
2	MonthlyRevenue	Original	billing_economics	Numeric (continuous)	No transform (highly skewed) – consi
3	MonthlyMinutes	Original	usage_activity	Numeric (continuous)	No transform (consider log for GLM
4	TotalRecurringCharge	Original	billing_economics	Numeric (continuous)	–

The complete registry is provided in Appendix A (see Section A.2.1, Complete Feature Registry).

3.4 Exploratory data analysis: what the data "looks like" and why it matters

EDA in this study was designed to (i) quantify the dataset's major modeling constraints (imbalance, sparsity, nonlinearity), (ii) detect leakage risks, and (iii) create reproducible tables/figures that guide later methodological choices.

3.4.1 Class balance and baseline difficulty

Churn is a minority class but not extremely rare: 14,711 churners (28.82%) versus 36,336 non-churners (71.18%). A naïve classifier that always predicts "non-churn" would achieve 0.7118 accuracy, demonstrating that accuracy alone is not a sufficient measure for this task.

This imbalance motivated evaluation choices later in the thesis (e.g., ranking metrics and precision/recall-focused diagnostics), but the formal metric definitions are deferred to the Methodology section.

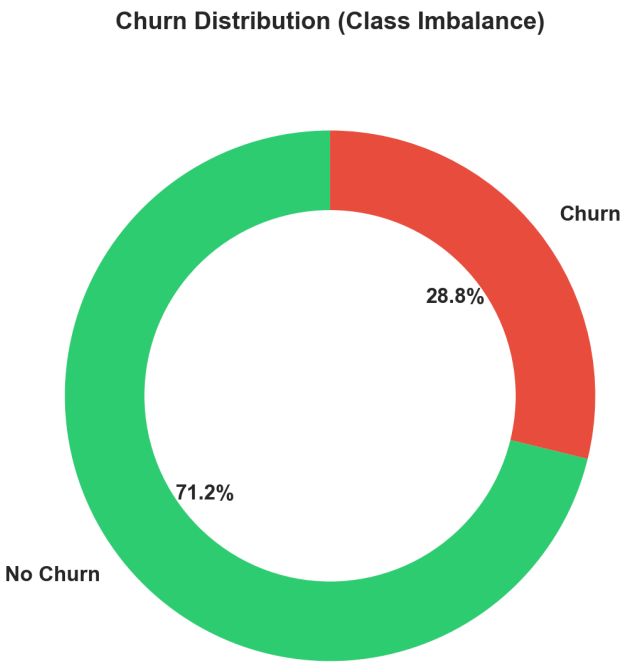

Figure 3.1: Churn class distribution in the Cell2Cell training set.

3.4.2 Missingness, sparsity, and "zero inflation"

A key constraint in this telecom dataset is that many variables are structurally sparse, meaning they are missing for most customers or recorded as zeros that effectively encode "not applicable" or "not observed". The EDA therefore tracked missing values and zero inflation together (a "missing+zero" sparsity score) and saved a ranked summary.

The top-ranked features by combined sparsity were mainly zero-inflated retention and low-frequency call variables (e.g., CallForwardingCalls, RetentionOffersAccepted, RetentionCalls), followed by sparse call-behavior and customer-contact measures (e.g., ThreewayCalls, RoamingCalls, CustomerCareCalls) and high-missingness demographic/pricing proxies such as IncomeGroup, AgeHH1/AgeHH2, and HandsetPrice.

3.4.3 Ordinal, categorical, and continuous patterns

EDA included several diagnostic plots for ordinal and categorical features, examining the relationship between variable categories and churn rate. For example, Figure 3.3 visualizes how churn risk varies across ordinal feature levels.

An important categorical analysis tool was Cramér's V, which measures association strength between categorical predictors and the binary churn outcome. Features with strong associations are potentially

Figure 3.2: Missingness and zero-rate diagnostics for input features.

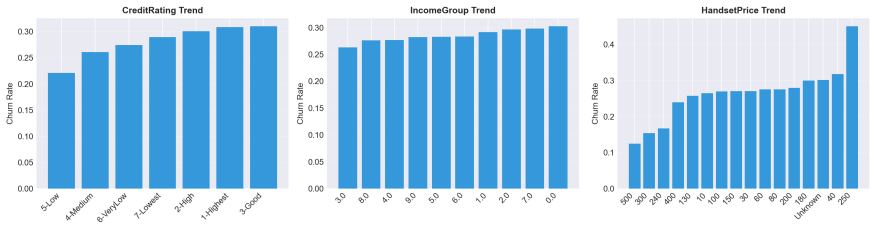

Figure 3.3: Churn rates across ordinal variable levels.

informative but require care to distinguish from leakage. Figure 3.4 presents the Cramér's V ranking for the top categorical features.

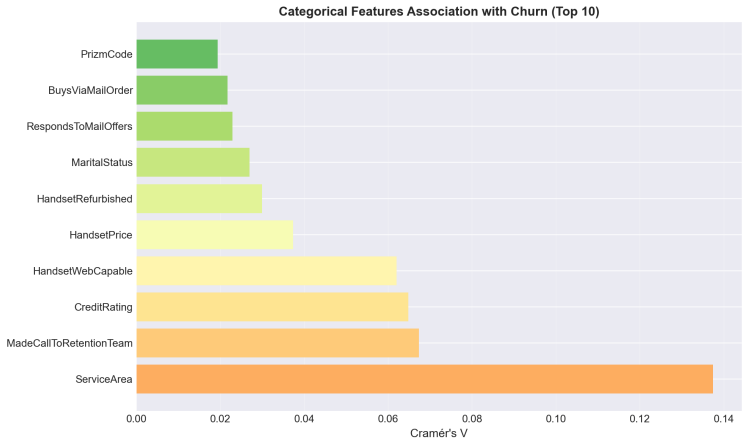

Figure 3.4: Cramér's V association between top categorical features and churn label.

3.4.4 Leakage risk and variable exclusion

A central rigor requirement in churn modeling is avoiding predictors that implicitly contain post-outcome information. This study therefore treated leakage control as a data scoping decision: features that would not be available at prediction time (or that are only triggered once churn intent is revealed) were excluded from the modeling dataset.

A dedicated leakage scan (saved as T5j_leakage_risk_rank.csv) ranked features by a composite leakage risk score (see Section A.2.3 for the complete ranking). In the scan, the highest-risk features were retention intervention variables such as MadeCallToRetentionTeam, RetentionCalls, and RetentionOffersAccepted, which were flagged as essentially post-churn indicators. The EDA further visualized these risks using leakage-flag churn comparisons and saved the summary plot.

Based on both statistical evidence and business-process logic, these retention-related variables were treated as leakage and excluded from all predictive models. This choice preserves realism: a churn model should predict before retention actions occur, not infer churn because an intervention has already been triggered.

3.5 Artifacts and documentation outputs

Feature documentation artifacts (T0_feature_registry.csv, T0_semantic_groups_table.csv) were generated and reused across notebooks (see Section A.2.1 and Section A.2.2 for complete tables).

The EDA stage produced:

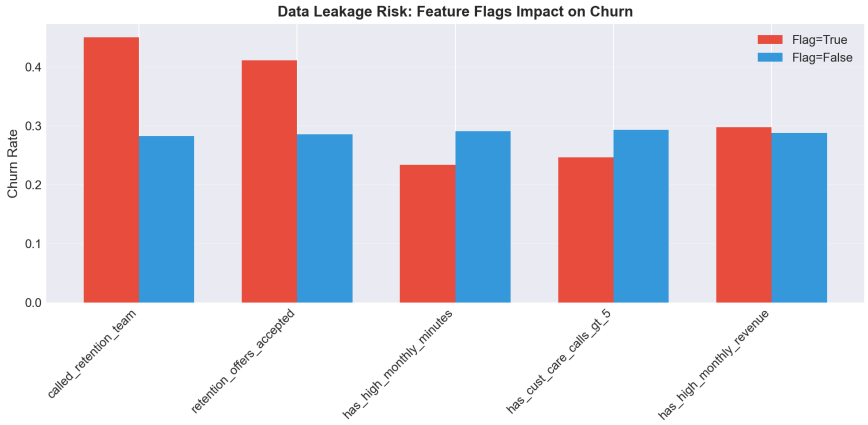

Figure 3.5: Leakage risk visualization: churn rates by retention-related flag values.

- **Sparsity diagnostics** (T1_sparsity_overview.csv; see Figure 3.2).
- **Ordinal trend analysis** (T4_ordinal_trends.csv; see Figure 3.3).
- **Categorical association analysis** (T4_cramers_v_vs_churn.csv; see Figure 3.4).
- **Leakage risk ranking** (T5j_leakage_risk_rank.csv; see Figure 3.5 and Section A.2.3 for the full ranking).

The complete registry is provided in Appendix A (see Section A.2.1, Complete Feature Registry).

3.6 Summary of data constraints

(1) moderate class imbalance (28.82% churn), (2) substantial sparsity and zero inflation across multiple feature families, (3) nonlinear churn patterns (e.g., a U-shaped CreditRating risk curve), and (4) high-stakes leakage risks in retention-related variables requiring explicit exclusion. These observations establish the constraints and motivations for the modeling comparisons presented in subsequent sections.

4 Methodology

4.1 Data Preprocessing Pipeline

The preprocessing pipeline transforms the raw Cell2Cell features into a modeling-ready format while strictly preventing data leakage between training, validation, and test splits. All pipeline parameters are estimated on the training set only and then applied unchanged to validation and test sets.

Feature scoping and leakage control. Feature inclusion is governed by a feature registry that records which columns are retained and how they should be treated for different model families. This registry-driven approach avoids per-notebook, ad hoc column selection and makes the modeling dataset auditable. As part of scoping, identifier fields and target fields are excluded from the feature set. In addition, retention-intervention variables are removed because they plausibly reflect post-outcome actions (for example, contact with a retention team or accepted retention offers) and can introduce leakage if used as predictors.

Training-only fitting of preprocessing parameters. To avoid leakage through preprocessing, all transformation parameters are estimated using the training split only, and then reused unchanged to transform validation and test sets (details of the splitting protocol are described in Section 4.2). For numeric variables, the pipeline computes training-set medians for imputation and training-set means and standard deviations for standardization. For categorical variables, it constructs a training-derived mapping from category strings to integer codes, reserving a default code for missing or unseen categories. The fitted parameters are saved for reproducibility and reusability across subsequent experiments.

Two output representations. Because different model families have different input requirements, the pipeline outputs two feature representations:

1. **Base representation**: A one-hot encoded feature matrix suitable for logistic regression and tree-based models, which require fixed-width numeric input without embedding layers.

2. **Deep representation**: A structured output separating continuous features (as a float tensor), categorical features (as integer indices for embedding lookup), and binary features (as a separate float tensor). This format feeds directly into PyTorch data loaders for neural network training.

Overall, this preprocessing design functions as a stable "data contract" for the rest of the thesis: later sections can focus on modeling and evaluation choices while relying on a consistent, leakage-aware, and reproducible transformation of raw telecom records into model inputs.

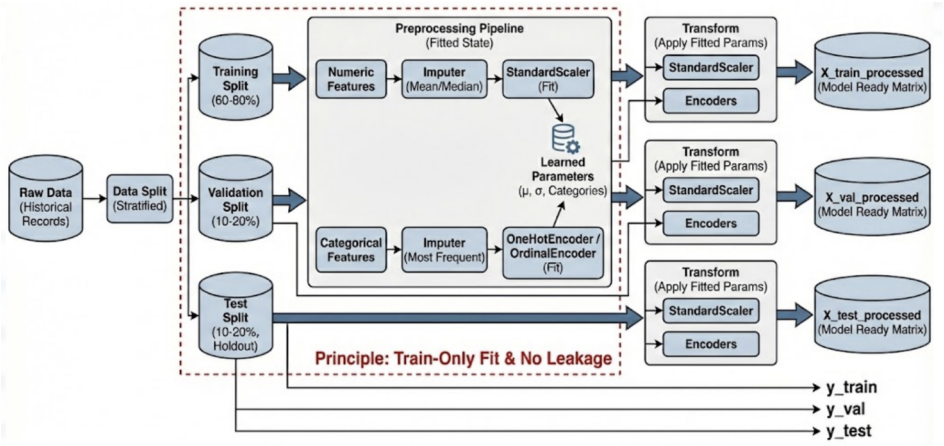

Figure 4.1: Preprocessing pipeline overview showing data flow from raw features to model-ready representations.

4.2 Experimental Design and Evaluation Protocol

All modeling experiments in this thesis follow a unified evaluation protocol to ensure fair comparison across model families. The core design choices are described below.

Data splitting strategy. The labeled dataset was divided into three non-overlapping subsets: a training set (70%), a validation set (15%), and a test set (15%). Stratified sampling was applied to maintain consistent churn prevalence (approximately 28.8%) across splits. The training set is used for model fitting and hyperparameter search; the validation set guides early stopping, threshold selection, and model selection decisions; the test set is reserved for final, unbiased performance reporting. No model selection or tuning decisions were made based on test-set results.

Primary evaluation metrics. Model ranking and selection were based primarily on two threshold-independent metrics:

1. **ROC-AUC (Area Under the Receiver Operating Characteristic Curve)**: Measures the probability that a randomly chosen positive (churner) is ranked higher than a randomly chosen negative (non-churner). This metric is standard for binary classification and is insensitive to class imbalance when used for ranking purposes.

2. **PR-AUC (Area Under the Precision–Recall Curve)**: Also known as Average Precision, this metric is more sensitive to performance on the minority class (churners) and is particularly relevant when class imbalance is present.

Fixed operating threshold for decision-level metrics. To enable consistent comparison of decision-level performance (precision, recall, F1), a single operating threshold ($\tau = 0.4400$) was selected on the validation set during the main tree-based model selection and then applied to all models uniformly. This fixed-threshold policy avoids per-model "best threshold" cherry-picking and reflects realistic deployment scenarios where a single, stable decision rule is applied across all customers.

Cross-validation for hyperparameter search. Where applicable (logistic regression, XGBoost with Optuna), hyperparameters were tuned using stratified 5-fold cross-validation on the training set, with ROC-AUC as the optimization objective. The best hyperparameters were then used to refit a final model on the full training set before evaluation on validation and test sets.

Reproducibility controls. All experiments used fixed random seeds (42) for data splitting, model initialization, and stochastic optimization. Model artifacts, including fitted parameters, training curves, and prediction outputs, were saved to enable exact reproduction of results.

4.3 Baseline Model: Regularized Logistic Regression

A regularized logistic regression model was developed as the supervised baseline for this churn prediction task. Logistic regression offers a transparent, interpretable starting point: its linear decision boundary and directly interpretable coefficients make it straightforward to identify which features contribute most to churn prediction before introducing more complex, non-linear methods.

The logistic regression model was trained on the same cleaned, leakage-controlled feature scope defined in the project's feature registry, using the subset of predictors designated for the GLM family (i.e., features marked as retained for the linear baseline). Because categorical variables had already been expanded into a consistent one-hot feature space in the "base" representation created during preprocessing, the logistic regression stage operated directly on this fixed design matrix. Feature inclusion followed the registry logic: when a raw feature corresponded to a categorical variable, its associated one-hot columns were included as a group (via prefix-based matching), ensuring that the linear baseline used the complete encoded information for each retained categorical predictor while excluding disallowed or leaky fields.

Model fitting used L1/L2-regularized logistic regression with a controlled hyperparameter search restricted to the most impactful regularization choices. Specifically, the inverse regularization strength and the penalty

type (L1 vs. L2) were tuned using a grid search. The search was evaluated via stratified 5-fold cross-validation on the training split, using ROC-AUC as the selection criterion, consistent with the overall evaluation protocol described in Section 4.2. The hyperparameter search space was intentionally constrained to regularization strength (C) and penalty type. This restricted scope ensures the baseline remains interpretable and stable, avoiding the risk of over-engineering.

After selecting the best hyperparameters under cross-validation, the final baseline model was refitted on the full training split and then evaluated under the common experimental protocol (validation used for operating-point selection; test reserved for final reporting). Threshold selection and any constraint-based operating-point choice were handled outside the training objective and followed the unified decision rule described in Section 4.2, so that the baseline and later models were compared under the same deployment-style evaluation assumptions.

4.4 Tree and Gradient-Boosted Tree Models

Tree-based methods were the primary non-linear modeling family for this structured telecom dataset. The modeling strategy progressed from a simple, interpretable tree to ensemble methods that improve generalization, and finally to gradient-boosted frameworks that typically deliver the strongest ranking performance on tabular data. Data splitting, metric definitions, and the shared threshold policy follow the experimental protocol in Section 4.2.

All tree models were trained on the same base, one-hot encoded representation produced by the preprocessing pipeline (Section 4.1). To keep the tree family comparable and leakage-safe, input columns were selected using the project's feature registry (features explicitly marked as allowed for tree models). Categorical predictors were represented by their one-hot expansions in a shared column space. When a raw feature was retained, all corresponding dummy columns were included together (via prefix matching), ensuring consistent inclusion of encoded categorical information across the tree family.

4.4.1 Decision Tree Baseline (CART)

A single classification tree was used as a simple non-linear reference point. Because unpruned trees can overfit on tabular data, this baseline used explicit structural constraints, including a bounded maximum depth and minimum sample requirements for splits and leaves. These constraints provide controlled flexibility while preserving interpretability at the segment and rule level. Class imbalance was handled through balanced class weights so that splits were not dominated by the majority (non-churn) class. This model serves as a

minimum viable non-linear benchmark and tests whether a small set of hierarchical rules can capture churn signals beyond a linear baseline.

4.4.2 Random Forest Ensemble

To improve stability and generalization beyond a single tree, a random forest classifier was trained as a bagging-based ensemble. Random forests reduce variance by averaging many decorrelated trees trained on bootstrap samples, and they typically provide stronger ranking quality than a single CART without requiring sensitive tuning. The implementation used a fixed configuration with a substantial number of trees and moderate depth control, along with feature subsampling to encourage diversity between trees. Class imbalance was addressed through balanced class weights, mirroring the single-tree setup. The forest is treated as a strong classical ensemble baseline rather than the primary optimized model.

4.4.3 Gradient-Boosted Frameworks: XGBoost and LightGBM

Gradient boosting was implemented with XGBoost as the main boosted framework. The baseline XGBoost configuration used moderate tree depth, a standard learning rate, and row and column subsampling to provide competitive performance without heavy search. Because boosting optimizes a differentiable loss over all observations, imbalance was handled through positive-class reweighting using a prevalence-based weight derived from the training set. This keeps the dataset intact while aligning the objective with churn detection asymmetry.

LightGBM was trained as a complementary boosted reference model. LightGBM uses a histogram-based algorithm and leaf-wise growth strategy, offering a different bias profile and computational behavior from XGBoost. In this study it was configured as a compact model with a fixed parameter set (rather than a full Bayesian search), including a moderate learning rate, a controlled number of estimators, and explicit regularization. Imbalance handling again used prevalence-based positive-class weighting.

4.4.4 Bayesian Hyperparameter Optimization

To obtain a high-quality boosted model under a reproducible and auditable search process, XGBoost was tuned using Bayesian optimization with Optuna. The tuning objective was mean ROC-AUC under stratified cross-validation on the training split, consistent with Section 4.2. A Tree-structured Parzen Estimator sampler proposed hyperparameters, and a median-based pruning rule stopped unpromising trials early to improve compute efficiency without changing the evaluation criterion.

The search space covered key capacity and regularization controls, including number of estimators, tree depth, learning rate (sampled on a log scale), row subsampling, column subsampling, minimum child weight, split regularization via gamma, and L1/L2 regularization. The best hyperparameters selected by cross-validated ROC-AUC were used to refit a final XGBoost model on the full training split before entering the shared validation and test evaluation pipeline.

4.4.5 Reproducibility and Model Artifacts

Across all tree and boosted models, the implementation was designed for deterministic reruns under the same data splits and feature registry constraints. Tuned parameters, fitted models, and evaluation summaries were saved as artifacts (serialized models and JSON metadata) to ensure that the methodology is auditable and that later stages such as ensembling and interpretation can reuse the same fitted estimators rather than retraining them implicitly.

4.5 Deep Learning Architecture

Neural network models were developed to test whether learned representations and higher-order non-linear interactions could provide incremental gains beyond tree-based methods on customer churn prediction. Unlike GBDTs, which learn interactions through recursive partitioning, neural models can combine continuous covariates and categorical embeddings in a unified latent space and potentially capture smooth interaction effects. All neural models were implemented in PyTorch and trained/evaluated under the common experimental protocol in Section 4.2.

4.5.1 Input representation and dataset construction

Neural models used the "deep" feature representation produced by the preprocessing stage, which separates variables into three semantic types to match common tabular deep learning practice:

4.5.1.1 Continuous features (float tensors).

A fixed set of numeric variables was provided as continuous inputs. These features were preprocessed in a leakage-safe manner using training-derived parameters (imputation and standardization), resulting in approximately zero-mean, unit-variance scaled inputs. Standardization improves optimization stability for gradient-based training and reduces sensitivity to feature scale differences.

4.5.1.2 Categorical features (integer-index tensors).

High-cardinality and ordinal variables were mapped to integer indices suitable for embedding lookup. Each categorical column was encoded using a training-derived vocabulary so that unseen categories at inference time can be mapped to a reserved index. This representation allows the network to learn dense, low-dimensional representations for each category rather than relying on sparse one-hot vectors.

4.5.1.3 Binary features (float tensors).

Binary indicators (e.g., yes/no flags, converted to 0/1) were passed directly as float inputs, preserving their semantic status while avoiding unnecessary embedding overhead for two-level variables.

This three-part input structure is assembled by a custom PyTorch Dataset class that returns aligned tensors for each batch, enabling modular experimentation with different network heads while keeping the data pipeline consistent.

4.5.2 Embedding MLP model

The first neural architecture is an **Embedding MLP** that learns dense representations for categorical features and concatenates them with continuous and binary inputs before passing through a multi-layer perceptron. This design follows common practice for tabular deep learning, where categorical embeddings can capture richer interactions than one-hot encoding while keeping the model end-to-end differentiable.

4.5.2.1 Embedding layer.

Each categorical feature is mapped to a learnable embedding vector. The embedding dimension for each feature is set heuristically (e.g., $\min(50, \text{cardinality}//2)$) to balance expressiveness and regularization. All embeddings are concatenated into a single vector per sample.

4.5.2.2 MLP trunk.

The concatenated representation (embeddings + continuous + binary) is passed through a feedforward network with ReLU activations and dropout regularization between layers. A typical configuration uses three hidden layers with decreasing width (e.g., $256 \rightarrow 128 \rightarrow 64$), though the exact architecture can be varied experimentally.

4.5.2.3 Output head.

A final linear layer with a single output unit produces a logit, which is passed through a sigmoid activation to yield a churn probability. For training, the loss is computed directly on the logit (using `BCEWithLogitsLoss` or focal loss variants), which is numerically more stable than applying sigmoid before loss computation.

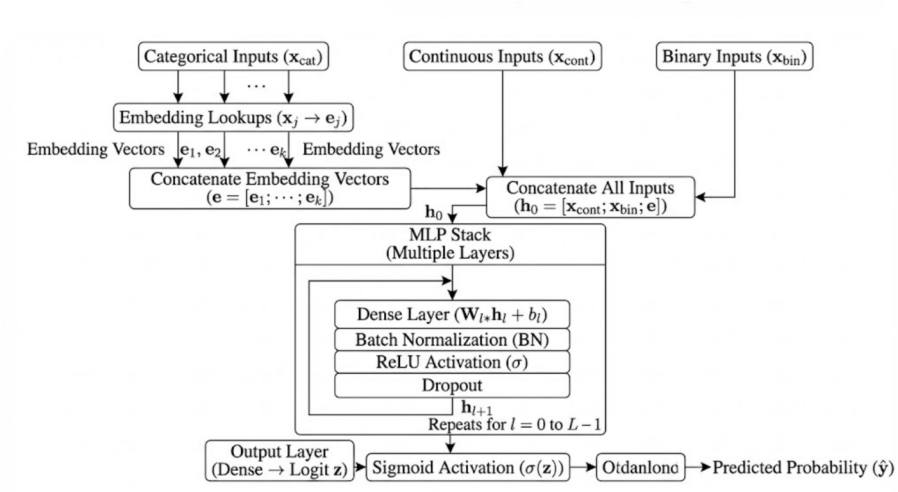

Figure 4.2: Embedding MLP architecture: categorical embeddings are concatenated with continuous/binary features and passed through a multi-layer perceptron.

4.5.3 Wide & Deep model

The second architecture is a **Wide & Deep** network inspired by the approach introduced by Google for recommendation systems. The key idea is to combine a "wide" linear component (which memorizes feature interactions explicitly) with a "deep" neural component (which generalizes through learned representations). This hybrid structure can capture both low-order, interpretable effects and high-order, non-linear interactions.

4.5.3.1 Wide component.

A linear layer directly connects the concatenated input features (continuous + binary + flattened embeddings or one-hot encoded categoricals) to the output logit. This component acts like a logistic regression and can memorize specific feature combinations that are predictive of churn.

4.5.3.2 Deep component.

The same input representation is passed through an MLP trunk (similar to the Embedding MLP described above) to produce a learned representation that captures non-linear interactions. The final hidden layer output is concatenated with the wide component's linear contribution before the output layer.

4.5.3.3 Output combination.

The wide and deep contributions are summed (or concatenated and passed through a final linear layer) to produce the final logit. This architecture allows the model to benefit from both memorization (wide) and generalization (deep), which can be particularly useful for tabular data with both dense and sparse predictive patterns.

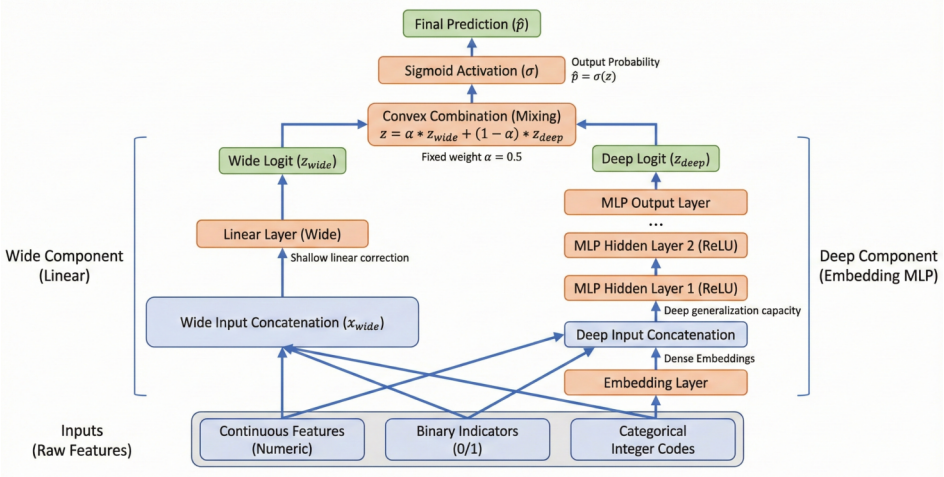

Figure 4.3: Wide & Deep architecture combining a linear 'wide' path with a non-linear 'deep' path for churn prediction.

4.5.4 Training objective and handling class imbalance

Class imbalance (approximately 29% churn vs. 71% non-churn) can bias neural network training toward the majority class if not addressed. Two approaches were implemented and compared:

4.5.4.1 Weighted Binary Cross-Entropy (BCE).

The standard BCE loss is modified by assigning a higher weight to positive (churn) samples. If w_{pos} denotes the positive-class weight, the loss for a single sample becomes:

$$\mathcal{L}_{\text{weighted-BCE}} = -w_{\text{pos}} \cdot y \log(\hat{p}) - (1-y)\log(1-\hat{p})$$

The weight w_{pos} is typically set to the inverse class frequency ratio (e.g., $w_{\text{pos}} = \frac{N_{\text{neg}}}{N_{\text{pos}}}$) so that the total contribution of positive and negative samples is balanced. This approach directly addresses the numerical dominance of majority-class gradients during training.

4.5.4.2 Focal Loss.

An alternative is focal loss, which down-weights easy examples (those classified correctly with high confidence) and focuses training on hard examples:

$$\mathcal{L}_{\text{focal}} = -\alpha_t (1-p_t)^{\gamma} \log(p_t)$$

where $p_t = \hat{p}$ if $y = 1$ else $1 - \hat{p}$, α_t is a class-balancing weight, and $\gamma > 0$ is the focusing parameter. Higher γ increases the focus on hard-to-classify examples. Focal loss was originally proposed for object detection but has shown benefits for imbalanced tabular classification as well.

Importantly, both approaches preserve the original data distribution (no oversampling) and fit naturally into the probabilistic scoring framework required by Section 4.2 (ROC/PR evaluation and validation-based thresholding).

4.5.5 Optimization, regularization, and model selection

All neural models were trained using the Adam optimizer with a learning rate selected from a small grid (e.g., $\{10^{-3}, 10^{-4}\}$). Training proceeded for a maximum number of epochs with early stopping based on validation ROC-AUC: if the validation metric did not improve for a specified patience window, training was halted and the best checkpoint was restored.

Regularization was applied through dropout (applied after each hidden layer) and optional weight decay. Batch normalization was not used in the main experiments to keep the architecture simple and to avoid potential issues with small batch sizes during inference.

Model selection among neural architectures (Embedding MLP vs. Wide & Deep) and loss functions (weighted BCE vs. focal loss) was performed by comparing validation ROC-AUC. The best-performing configuration was then evaluated on the held-out test set for final reporting.

Artifacts and reproducibility. All models output churn risk scores that were evaluated using the unified metrics and thresholding procedure in Section 4.2. Artifacts, including training curves, saved model weights,

and prediction files, were recorded to enable reproducibility and to support downstream interpretation and ensemble construction in later sections.

4.6 Unsupervised and Semi-supervised Extension

Beyond purely supervised learning, I investigated whether unlabeled customer records could be exploited to improve churn prediction through (i) unsupervised representation learning and (ii) confidence-based pseudo-labeling. The core idea is that, even without labels, the marginal distribution of customer behavior may contain useful structure (e.g., common usage profiles and atypical patterns) that can be distilled into a compact latent representation and transferred to the downstream churn model. Consistent with Section 4.1, this extension operates on the same cleaned feature schema, while Section 4.2 defines the evaluation protocol used to compare all variants.

4.6.1 Denoising Autoencoder (DAE)

A denoising autoencoder was trained on the numeric feature subset (all continuous columns) without using churn labels. The DAE consists of:

1. **Encoder**: A feedforward network that maps the (corrupted) input features to a lower-dimensional latent representation **z**.
2. **Decoder**: A symmetric feedforward network that reconstructs the original (uncorrupted) input from **z**.
3. **Noise injection**: During training, input features are corrupted by additive Gaussian noise or masking dropout. The model is trained to reconstruct the clean input, which encourages the encoder to learn robust, denoised representations.

The reconstruction loss (mean squared error between input and output) is minimized using Adam optimizer with early stopping based on validation reconstruction loss.

Label-Free Training. The DAE is trained without using churn labels, and can therefore incorporate both labeled and unlabeled records in a leakage-safe manner (labels are never accessed). Model selection is performed using a held-out validation split and early stopping based on validation reconstruction loss, selecting the epoch that best balances fit and generalization of the learned representation. Once trained, the encoder is treated as a deterministic feature extractor that produces a latent vector for each customer.

4.6.2 Downstream Usage: Latent Feature Augmentation

After training, the encoder is frozen and used to extract latent representations for all samples (train, validation, test, and unlabeled holdout). These latent features are then concatenated with the original feature set to form an augmented input representation for the downstream supervised model (e.g., XGBoost). The hypothesis is that the learned latent space may capture useful structure (e.g., customer segments, anomaly patterns) that improves discrimination when combined with the original features.

Both variants are evaluated against the supervised baseline using the identical split and metric definitions in Section 4.2, ensuring that any observed change can be attributed to representation learning rather than differences in preprocessing or evaluation.

4.6.3 Pseudo-Labeling Extension

To test whether unlabeled data can provide additional training signal, a pseudo-labeling strategy was implemented:

1. **Teacher model**: An Optuna-tuned XGBoost model trained on labeled data serves as the teacher.
2. **Confidence thresholding**: The teacher scores the unlabeled holdout set. Samples with predicted probability $p \geq 0.95$ are pseudo-labeled as churners; samples with $p \leq 0.05$ are pseudo-labeled as non-churners. Samples in between are discarded as uncertain.
3. **Student training**: A new model is trained on the union of labeled data and high-confidence pseudo-labeled data.

This conservative threshold strategy prioritizes label quality over quantity, accepting only pseudo-labels where the teacher is highly confident.

4.7 Heterogeneous Stacking Ensemble

To improve robustness beyond any single model family, I implemented a heterogeneous ensemble that combines complementary inductive biases through probability-level aggregation. Rather than relying on a single "best" learner, the ensemble treats each trained model as a noisy but informative estimator of churn risk, and then learns (or prescribes) a principled rule for combining their probability outputs. This section describes the ensemble design and training protocol; performance comparisons are reported later under the common evaluation policy defined in Section 4.2.

4.7.1 Base Learners

The ensemble uses the following base learners, all trained on the same feature representation and data splits:

1. **Logistic Regression** (linear baseline)
2. **XGBoost** (Optuna-tuned gradient boosting)
3. **LightGBM** (histogram-based gradient boosting)
4. **Random Forest** (bagging-based ensemble)
5. **Wide & Deep Neural Network** (best neural configuration)

These models span different inductive biases: linear vs. non-linear, tree-based vs. neural, single-model vs. ensemble. The diversity in model families is expected to provide complementary error profiles, which is the key requirement for ensemble benefit.

4.7.2 Blending Strategies

Three blending strategies were evaluated:

1. **Simple Average**: Equal-weight averaging of predicted probabilities: $\hat{p}_{\text{blend}} = \frac{1}{K} \sum_{k=1}^{K} \hat{p}_k$
2. **AUC-Weighted Average**: Weights proportional to each model's validation ROC-AUC: $w_k = \frac{\text{AUC}_k}{\sum_j \text{AUC}_j}$
3. **NNLS-Optimized Weights**: Non-negative least squares regression on validation predictions to find optimal weights that minimize squared error relative to true labels.

4.7.3 Out-of-Fold (OOF) Stacking

Leakage Control via OOF. The key methodological requirement for stacking is leakage control, as the meta-learner must be trained on base predictions that are out-of-sample with respect to the base learners. I therefore used a K-fold Out-of-Fold (OOF) protocol on the training split. For each fold k, each base estimator is fit on the $K-1$ training folds and used to predict probabilities on the held-out fold. Concatenating held-out predictions across folds yields an OOF prediction matrix Z_{OOF} where every row corresponds to a training instance and every column to a base learner, with the guarantee that each entry was generated by a model that did not train on that instance.

Meta-Learner Training. A Ridge logistic regression meta-learner is trained on Z_{OOF} to learn optimal combination weights. The meta-learner's coefficients indicate the relative contribution of each base learner to the final ensemble prediction.

Stacking architecture figure not found.
Expected: artifacts/figures/stacking_schematic.png

Figure 4.4: OOF stacking architecture: base learners produce out-of-fold predictions that are combined by a meta-learner.

4.7.4 Meta-Learner Weights Visualization

After training, the meta-learner coefficients can be visualized to understand the relative contribution of each base model. Positive coefficients indicate models that contribute positively to the ensemble prediction; larger coefficients indicate stronger contributions.

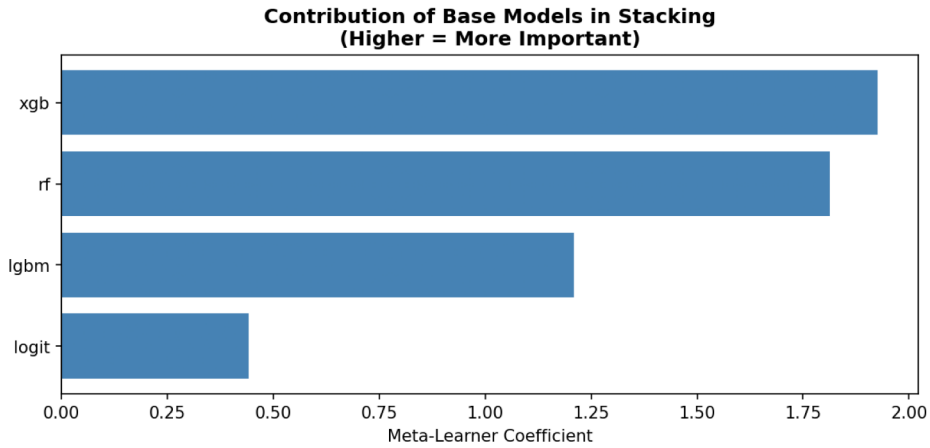

Figure 4.5: Meta-learner coefficients showing the contribution of each base model to the stacking ensemble.

4.7.5 Inference Pipeline

At inference time, all base models are applied to the test set to produce a prediction matrix Z_{test}. The meta-learner then combines these predictions to produce final ensemble probabilities. The operating threshold ($\tau = 0.4400$) is applied to these ensemble probabilities to generate binary churn predictions for evaluation.

5 Experiments and Results

5.1 Overall Benchmark Summary

All models were evaluated on the held-out test set using threshold-independent ranking metrics (ROC-AUC and PR-AUC) alongside threshold-dependent decision metrics (precision, recall, and F1) at a fixed operating point. Table 5.1 reports the complete benchmark results, while Figure 5.1 summarizes the corresponding ROC and precision–recall (PR) curves. Because the churn rate in the test set is 28.8%, a non-informative baseline achieves a PR-AUC of approximately 0.2880; all trained models exceeded this baseline by a substantial margin.

Across the supervised model families, gradient-boosted decision trees delivered the strongest overall performance. Using the validation-set ROC-AUC criterion defined in Section 4.2, XGBoost was selected as the champion model (validation ROC-AUC = 0.6583, validation PR-AUC = 0.4344). On the test set, the champion XGBoost model achieved ROC-AUC = 0.6637 and PR-AUC = 0.4454. At the fixed operating threshold of 0.4400, it reached precision = 0.3670, recall = 0.7390, and F1 = 0.4904 (the corresponding confusion matrix is reported in Section 5.7). Notably, LightGBM achieved the highest test ROC-AUC among the single supervised models (test ROC-AUC = 0.6700, test PR-AUC = 0.4476), indicating that the two leading GBM approaches were closely matched on this dataset.

The remaining baselines formed clear lower tiers. Random Forest achieved ROC-AUC = 0.6534 and PR-AUC = 0.4271, while the logistic regression baseline yielded ROC-AUC = 0.5940 and PR-AUC = 0.3560. The deep learning models clustered in a narrow band: their test ROC-AUC values ranged from approximately 0.6440 to 0.6620, and test PR-AUC ranged from approximately 0.4080 to 0.4360. The best neural model by ranking metrics was a Wide & Deep variant with focal loss (test ROC-AUC = 0.6615, test PR-AUC = 0.4356). Under the fixed threshold of 0.4400, several focal-loss neural variants produced very few positive predictions, resulting in near-zero recall and F1 despite competitive AUC/PR-AUC; therefore, ranking metrics provide the fairest comparison at this aggregate level, while threshold-based outcomes are interpreted separately in Section 5.7.

Table 5.1: Model leaderboard showing validation and test performance across all model families.

model	val_roc_auc	val_pr_auc	test_roc_auc	test_pr_auc	test_brier	test_f1	test_recall	test_precision	threshold
XGBoost	0.657146	0.435915	0.663681	0.445441	0.223314	0.490413	0.738953	0.366982	0.44
05_nn_wide_deep_focal	0.653063	0.429338	0.661457	0.435612	0.197007	0.017461	0.008837	0.722222	0.44
LightGBM	0.650239	0.4311	0.669973	0.447632	0.219983	0.49688	0.730795	0.376401	0.44
05_nn_wide_and_deep	0.649779	0.418712	0.649605	0.42076	0.2364	0.482441	0.7845	0.348325	0.44
RandomForest	0.649016	0.421517	0.653428	0.428959	0.228971	0.480699	0.804215	0.342799	0.44
05_nn_embedding_focal_loss	0.648666	0.421622	0.661197	0.434519	0.197579	0	0	0	0.44
05_nn_dense_baseline	0.646737	0.425746	0.65528	0.421941	0.234346	0.489779	0.806254	0.35172	0.44
05_nn_embedding_baseline	0.642545	0.408729	0.648944	0.419172	0.232875	0.487391	0.781781	0.354064	0.44
05_nn_wide_and_deep_deeper	0.642342	0.414346	0.647519	0.411018	0.232793	0.482227	0.765466	0.351985	0.44
05_nn_embedding_strong_weight	0.64173	0.411761	0.644192	0.408117	0.271837	0.478787	0.901428	0.325959	0.44
06_teacher	0.63778	0.418322	0.645056	0.427104	0.193299	0.262616	0.173351	0.541401	0.44
Logistic	0.588286	0.347494	0.593987	0.355974	0.24378	0.456037	0.840925	0.312848	0.44

Figure 5.1 visually corroborates these findings. The ROC curves for the leading models (LightGBM and XGBoost) lie consistently above the other model families across most false-positive rates, and the PR curves show a clear separation from the prevalence baseline (approximately 0.2880), with the top models maintaining higher precision at comparable recall. In addition to single-model benchmarks, heterogeneous ensemble experiments (Section 5.6) produced performance comparable to the best GBMs, with the best ensembles reaching approximately 0.6690 test ROC-AUC and 0.4490 test PR-AUC, representing a modest lift relative to the champion XGBoost model.

Note: Models were selected based on Validation ROC-AUC (as shown in the leaderboard in appendix). The plots below display performance on the held-out Test Set to provide an unbiased evaluation of generalization. Small discrepancies between ranking and plotting order are expected.

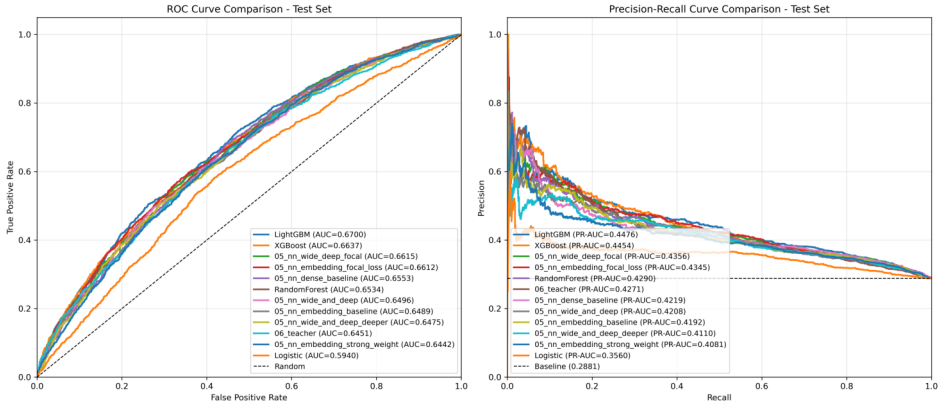

Figure 5.1: ROC and Precision–Recall curves comparing all model families on the test set.

5.2 Baseline Performance (Logistic Regression)

To provide a simple and interpretable reference point, a regularized logistic regression model was first evaluated as the supervised baseline. On the held-out test set, the baseline achieved a ROC-AUC of 0.5940 and a PR-AUC (average precision) of 0.3560, indicating modest but meaningful discrimination despite the class imbalance (Table 5.2). For context, the test-set churn prevalence is 0.2880, so the baseline PR-AUC is clearly above the prevalence level expected from random ranking; this relationship is also visible in the precision–recall curve where the model stays above the horizontal prevalence line across a wide recall range (Section A.3.3).

Table 5.2: Logistic regression baseline performance metrics on validation and test sets.

split	threshold	roc_auc	pr_auc	brier_score	precision	recall	f1_score	true_negatives	false_positives	false_negatives	true_positives
Train	0.44	0.593797	0.354999	0.224399	0.314526	0.847396	0.458771	7333	21735	1796	9973
Validation	0.44	0.588286	0.347494	0.2244917	0.311646	0.836846	0.454516	915	2719	240	1231
Test	0.44	0.593980	0.355970	0.224378	0.312848	0.840926	0.456037	917	2717	234	1237

The baseline exhibited stable performance across data splits. On the validation set, the model reached ROC-AUC = 0.5880 and PR-AUC = 0.3480, closely matching test-set results and providing a consistent lower-bound benchmark for subsequent model families. In addition to ranking metrics, decision-level performance is

reported at the operating threshold used for downstream comparisons (threshold = 0.4400). At this threshold, the test-set confusion matrix contained 917 true negatives, 2,717 false positives, 234 false negatives, and 1,237 true positives, corresponding to recall = 0.8410, precision = 0.3130, and F1 = 0.4560. Detailed ROC curves, PR curves, and confusion matrices for the logistic regression baseline are provided in Section A.3.3.

Finally, probability accuracy was summarized via the Brier score, which was 0.2440 on the test set; the corresponding reliability diagram provides a visual check of calibration quality (Figure 5.2). Overall, this baseline establishes a conservative performance floor against which the gains from nonlinear models (tree/GBM, neural networks) and ensembling can be quantified.

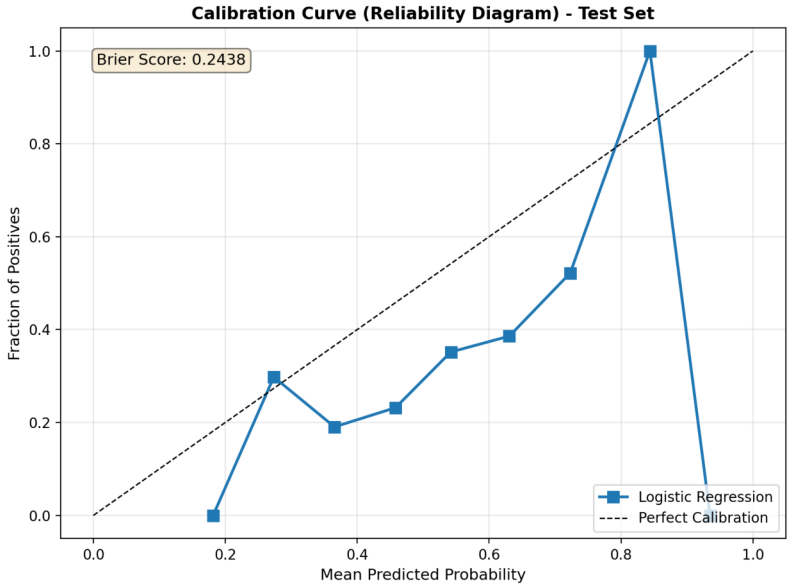

Figure 5.2: Calibration (reliability) diagram for logistic regression with Brier score annotation.

5.3 Tree and GBM Results

As a first step beyond the linear baseline, two tree-based benchmarks were evaluated: a single decision tree and a random forest. Table 5.3 reports ROC-AUC and PR-AUC on both validation and test sets, and Figure 5.3 visualizes the corresponding ROC and PR curves on the test set.

Table 5.3: Performance summary of tree-based models on validation and test sets.

model	split	roc_auc	pr_auc	brier_score	thresh-old	preci-sion	re-call	f1_score	true_neg-atives	false_pos-itives	false_neg-atives	true_pos-itives
dt_base-line	val	0.631703	0.404364	0.238156	0.5	0.370782	0.590075	0.455404	2161	1473	603	868
dt_base-line	test	0.625077	0.393425	0.238209	0.5	0.367982	0.57036	0.447347	2193	1441	632	839
rf_base-line	val	0.649016	0.421507	0.231366	0.5	0.365991	0.662804	0.471584	1945	1689	496	975
rf_base-line	test	0.653428	0.428969	0.228971	0.5	0.373939	0.655367	0.476167	2020	1614	507	964
xgb_base-line	val	0.657146	0.435905	0.226199	0.5	0.385664	0.596193	0.468358	2237	1397	594	877
xgb_base-line	test	0.663681	0.445404	0.223314	0.5	0.393540.588095		0.471518	2301	1333	606	865
xgb_op-tuna	val	0.657618	0.437307	0.23188	0.5	0.382098	0.658794	0.483654	2067	1567	502	969
xgb_op-tuna	test	0.669062	0.443994	0.227888	0.5	0.390638	0.658056	0.490251	2124	1510	503	968
lgbm_smal	val	0.650239	0.431107	0.224585	0.5	0.388305	0.577889	0.464481	2295	1339	621	850
lgbm_smal	test	0.669973	0.447683	0.219983	0.5	0.401504	0.580567	0.474708	2361	1273	617	854

On the held-out test set, the decision tree achieved ROC-AUC = 0.6250 and PR-AUC = 0.3930 (validation: ROC-AUC = 0.6320, PR-AUC = 0.4040), showing that even shallow, rule-like splits can capture churn-related interactions absent in a linear boundary. However, performance improved noticeably when variance was reduced through bagging. The random forest increased test-set discrimination to ROC-AUC = 0.6530 and PR-AUC = 0.4290 (validation: ROC-AUC = 0.6360, PR-AUC = 0.4120), and its PR curve dominates the single-tree baseline across most recall levels in Figure 5.3. These gains indicate that churn signals are distributed across multiple weak patterns, and an ensemble of diverse trees aggregates them more reliably than a single fitted tree. Importantly, the validation-to-test gaps for both DT and RF were small in magnitude for ROC-AUC and PR-AUC, suggesting that the improvements are not driven by a split-specific artifact. At

the default decision threshold of 0.5000, both DT and RF yielded test-set F1 scores in the mid-0.4700 range (DT F1 = 0.4770, RF F1 = 0.4700), reinforcing that these baselines improve ranking but still leave substantial room for improvement in the precision–recall trade-off.

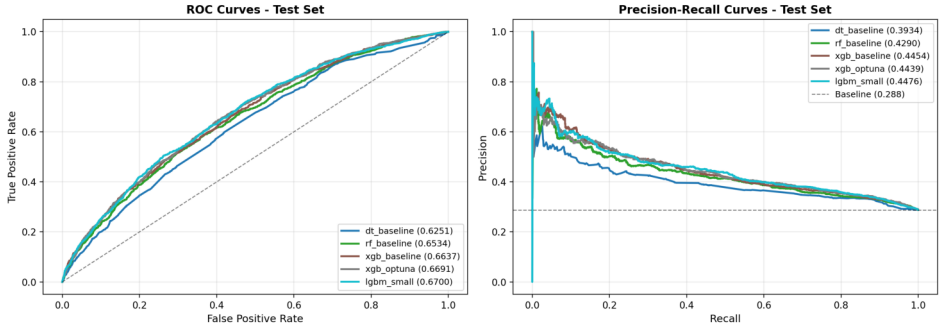

Figure 5.3: ROC and Precision–Recall curves for tree/GBM models on the test set.

5.3.1 Gradient-Boosted Decision Trees

Gradient-boosted decision trees delivered the strongest ranking performance among the classical ML families. As summarized in Table 5.3 and visible in Figure 5.3, the XGBoost baseline improved test discrimination to ROC-AUC = 0.6640 and PR-AUC = 0.4450 (validation: ROC-AUC = 0.6510, PR-AUC = 0.4350). Applying Optuna hyperparameter search to XGBoost produced a modest additional lift in ROC-AUC on the test set (0.6690 versus 0.6640) while maintaining a very similar PR-AUC (0.4440 versus 0.4450). LightGBM performed comparably and achieved the best overall ranking on the test set in this experiment (ROC-AUC = 0.6700, PR-AUC = 0.4480; validation PR-AUC = 0.4330). Relative to the logistic regression baseline (test PR-AUC = 0.3560), the best GBM models provide an absolute PR-AUC gain of roughly 0.0900, which is material given the 0.2880 churn prevalence.

When evaluated at a common default operating point (threshold = 0.5000), the Optuna-tuned XGBoost achieved the highest F1 among the tree family (F1 = 0.4900), reflecting a better precision–recall trade-off than the DT/RF baselines under the same decision rule. Because deployment typically requires an explicit operating point, a recall-constrained threshold sweep was performed on the validation set for the tuned XGBoost model, which selected threshold = 0.4400 as the best F1-achieving rule under the recall constraint. Figure 5.4 highlights this operating point directly on the ROC and PR curves for the tuned model, making the ranking–decision connection explicit. In addition to higher AUC metrics, the boosted models also achieved slightly lower Brier scores (approximately 0.2200 on the test set), suggesting better probability calibration than the single-tree baselines even before explicit calibration methods are applied.

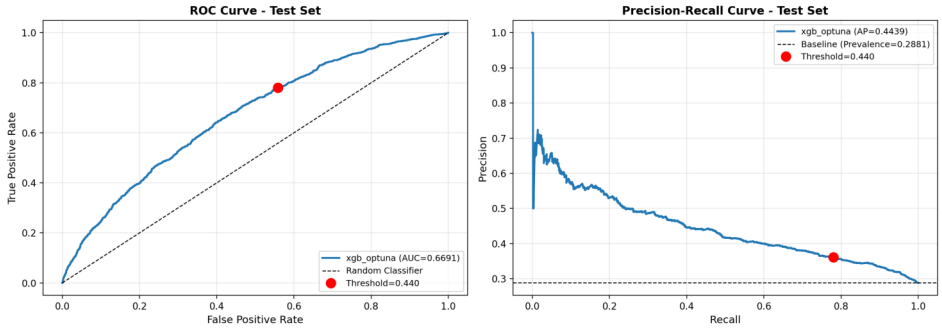

Figure 5.4: Tuned XGBoost ROC/PR curves with the selected operating threshold (0.4400) highlighted on the test set.

5.3.2 Hyperparameter Optimization Diagnostics

The Optuna diagnostics were inspected to verify that the hyperparameter search behaved sensibly and to summarize which settings drove performance. The optimization history shows rapid improvement in the early trials followed by a clear plateau, consistent with quickly locating a strong region of the hyperparameter space and then encountering diminishing returns. The parallel-coordinate visualization indicates that the better trials concentrate within a relatively narrow band of configurations, rather than appearing as isolated outliers, which reduces the likelihood that the selected model is the result of chance. The hyperparameter-importance summary further suggests that performance is most sensitive to the learning rate, while tree depth and regularization terms play a secondary role. This pattern aligns with the small empirical gap between the untuned and tuned XGBoost models: tuning improves ROC-AUC slightly, but the overall precision–recall ranking remains of similar magnitude. Full Optuna diagnostic plots, including the optimization history, parallel-coordinate visualization, and hyperparameter importance rankings, are provided in Section A.4.5.

5.4 Deep Learning Results (PyTorch)

This section reports neural network experiments implemented in PyTorch, focusing on (i) architecture comparisons and (ii) loss-function/imbalance handling ablations. Results are summarized in Table 5.4, with the best model's test curves shown in Figure 5.5 and its training dynamics summarized in Figure 5.6. Additional training curves for alternative architectures (MLP baseline, embedding MLP, and deeper Wide & Deep variants) are provided in Section A.5.2 and Section A.5.3.

Table 5.4: PyTorch neural network results (test metrics at validation-selected thresholds).

experiment	train_auc	val_auc	test_auc	test_pr_auc	test_brier	best_threshold_val	test_precision_at_best_threshold	test_recall_at_best_threshold	test_f1_at_best_threshold	best_epoch	train_time_s	loss_type	pos_weight	focal_alpha	focal_gamma
wide_deep_focal	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.31	0.364389	0.717879	0.483406	62	52.064	focal	2.46988	0.25	2
wide_and_deep	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.46	0.352218	0.739633	0.477193	44	41.890	bce_weighted	2.46988	0	0
embedding_focal_loss	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.3	0.342906	0.811693	0.482132	49	42.946	focal	2.46988	0.25	2
dense_baseline	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.48	0.369023	0.723997	0.488868	76	48.886	bce_weighted	2.46988	0	0
embedding_baseline	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.52	0.346409	0.816451	0.486432	33	30.189	bce_weighted	2.46988	0	0
wide_and_deep_deep	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.46	0.359181	0.727396	0.480899	27	35.454	bce_weighted	2.46988	0	0
embedding_strong_weight	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.54	0.354619	0.743712	0.480246	38	34.528	bce_weighted	3.70482	0	0

5.4.1 Architecture Comparison

Across the neural network candidates, the top-ranked models in terms of discrimination are the focal-loss variants: wide_deep_focal and embedding_focal_loss. On the test set, wide_deep_focal achieves ROC-AUC = 0.6610 and PR-AUC = 0.4360, narrowly outperforming the other NN configurations on ranking metrics. The embedding-only focal model is extremely close (ROC-AUC = 0.6610, PR-AUC = 0.4350), suggesting that learned categorical representations provide most of the gains, while the additional "wide" pathway mainly stabilizes decision performance rather than dramatically shifting AUC.

In contrast, the purely dense baseline (dense_baseline, without embedding-driven representation learn-

ing) trails in ranking quality (ROC-AUC = 0.6550, PR-AUC = 0.4220), and the weighted-BCE Wide & Deep (wide_and_deep) is lower still (ROC-AUC = 0.6500, PR-AUC = 0.4210). Increasing depth (wide_and_deep_deeper) does not help and slightly reduces test ranking metrics (ROC-AUC = 0.6480, PR-AUC = 0.4110), consistent with diminishing returns under the available signal and regularization constraints.

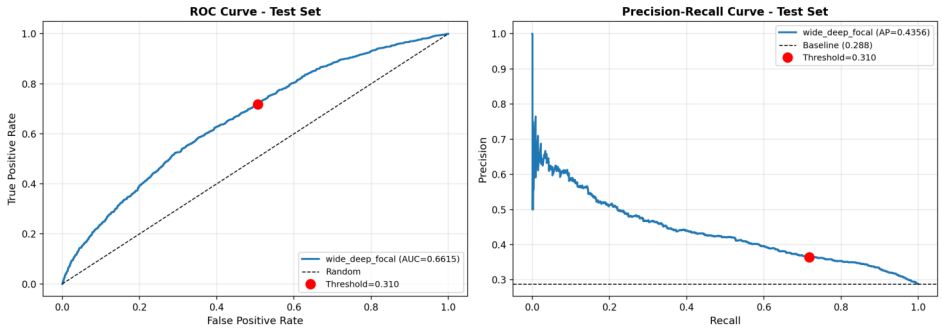

Figure 5.5: Best neural model (Wide & Deep with Focal Loss) ROC/PR curves with selected operating threshold on the test set.

Decision-level metrics (computed at the best validation-selected threshold per model) are broadly similar across the stronger NN candidates, with F1 concentrated around 0.4800–0.4900. For the best-ranking model wide_deep_focal, the chosen threshold is 0.3100, producing precision = 0.3640, recall = 0.7180, and F1 = 0.4830 on test. The embedding-only focal model selects a similar threshold (0.3000) but shifts toward higher sensitivity (recall = 0.8120) at lower precision (0.3430), yielding F1 = 0.4820. Interestingly, the dense baseline attains the highest F1 among NN variants (F1 = 0.4890) but with weaker AUC/PR-AUC, illustrating that "best F1" can be achieved via a favorable threshold even when the underlying ranking quality is lower. Probability accuracy further differentiates the models: focal-loss networks show a markedly improved Brier score of approximately 0.1970, compared with 0.2330–0.2360 for weighted-BCE variants, indicating better-calibrated probabilities in addition to stronger discrimination.

5.4.2 Loss/Imbalance Ablation

Focal Loss provides the clearest performance lift in this NN setup. Holding the general architecture fixed, switching Wide & Deep from weighted BCE (wide_and_deep) to focal (wide_deep_focal) improves ROC-AUC from 0.6500 to 0.6610 and PR-AUC from 0.4210 to 0.4360, while substantially lowering the Brier score (0.2360 to 0.1970). A similar pattern holds for embedding-only models: ROC-AUC improves from 0.6490 to 0.6610 and PR-AUC from 0.4190 to 0.4350 when moving from embedding_baseline to embedding_focal_loss.

In contrast, aggressively increasing the positive-class weight (embedding_strong_weight) under weighted BCE reduces ranking quality (ROC-AUC = 0.6440, PR-AUC = 0.4080), suggesting that extreme reweighting can over-correct and degrade generalization. Overall, focal loss appears to provide a more effective imbalance strategy by focusing learning on harder examples rather than uniformly amplifying the minority class.

The focal loss formulation down-weights easy examples and focuses training on hard-to-classify instances:

$$\mathcal{L}_{\text{focal}} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t)$$

where p_t is the model's estimated probability for the true class, α_t is a class-balancing weight, and $\gamma \geq 0$ is the focusing parameter. When $\gamma = 0$, focal loss reduces to standard cross-entropy. Higher values of γ increase the relative loss for hard examples (where p_t is small), encouraging the model to focus on difficult cases.

5.4.3 Training Dynamics

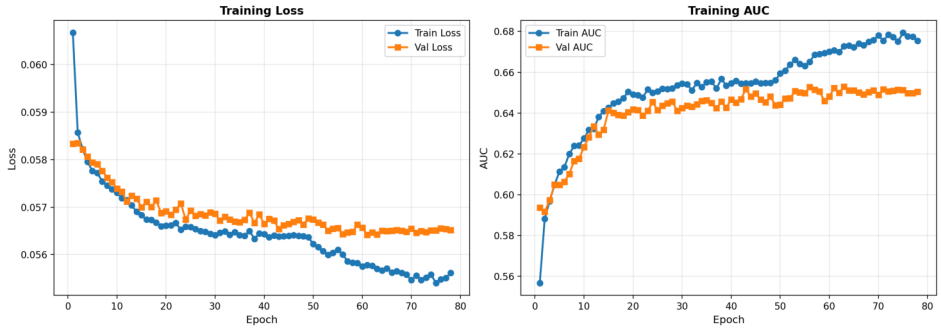

Figure 5.6: Training versus validation loss and AUC curves for the best neural network model (Wide & Deep with Focal Loss).

Training curves (Figure 5.6) show that the best focal model continues to reduce training loss while validation loss plateaus, and validation AUC stabilizes in the mid-0.6400 range after roughly the mid-training period; early stopping selects epoch 62 for wide_deep_focal. Across variants, deeper/wider configurations tend to widen the train–validation gap without improving validation AUC, consistent with the small benefit observed from the deeper Wide & Deep model. Additional learning curves for alternative architectures and imbalance settings are provided in Section A.5.2 (best model training dynamics) and Section A.5.3 (alternative architecture comparisons).

5.5 Unsupervised/Semi-Supervised Extension (Autoencoder + Pseudo-Label)

To explore whether unlabeled data can improve churn prediction, a denoising autoencoder (DAE) was trained on the feature space and evaluated for two downstream uses: (i) using the learned latent representation as features, and (ii) using an Optuna-tuned XGBoost teacher to generate pseudo-labels on a real unlabeled holdout set.

5.5.1 DAE Convergence and Reconstruction

The DAE showed stable optimization behavior with early stopping at epoch 16 (Figure 5.7). Training reconstruction loss dropped quickly in the first few epochs and then plateaued, while the validation curve remained flat after the best epoch, indicating the encoder learned a consistent low-dimensional structure without late-epoch divergence. Note that validation loss is lower than training loss, which is expected here because noise is injected during training (denoising objective), making the training reconstruction task harder than the clean validation reconstruction.

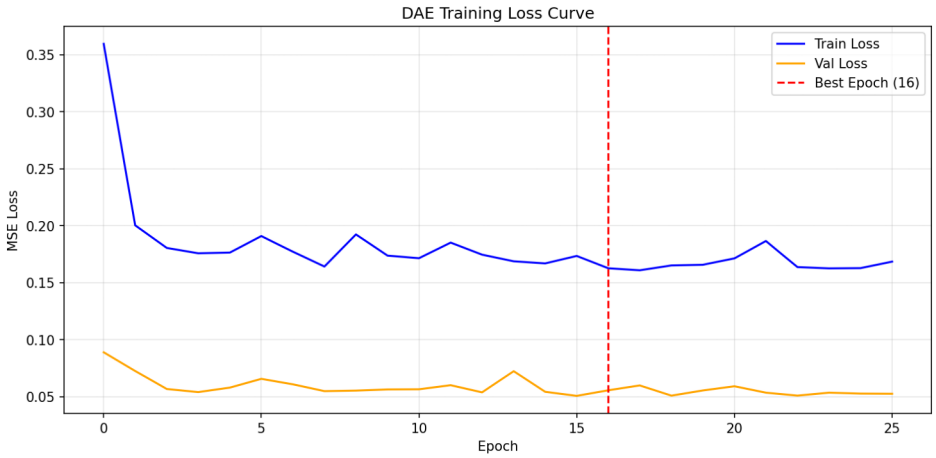

Figure 5.7: DAE reconstruction loss curves with early-stopping epoch marked.

5.5.2 Pseudo-Label Confidence and Coverage

Using the XGBoost teacher trained on labeled data, the real unlabeled holdout was scored and a conservative high-confidence rule was applied (assign churn if $p \geq 0.9500$, non-churn if $p \leq 0.0500$). Figure 5.8 shows the teacher's probability distribution concentrates in the mid-range (roughly $p \approx 0.1500$–0.4500), producing almost no extreme probabilities. As a result, pseudo-label coverage was effectively zero: only 2 samples met the low-confidence (non-churn) criterion and 0 samples met the high-confidence (churn) criterion, with

19,998/20,000 (approximately 100%) remaining "uncertain." This is a key practical finding: with strict thresholds designed to minimize label noise, the teacher did not provide enough pseudo-labeled examples to form a meaningful semi-supervised training signal on this dataset split.

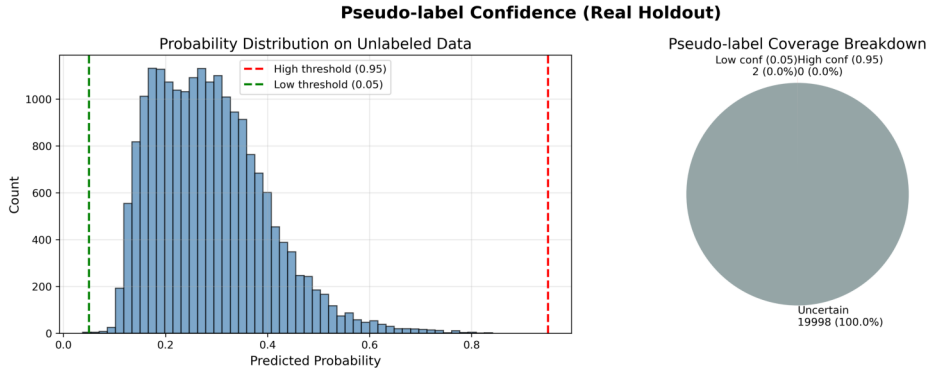

Figure 5.8: Teacher probability distribution on the unlabeled holdout and pseudo-label coverage at thresholds 0.0500/0.9500.

5.5.3 Downstream Impact (Ablation Summary)

Table 5.5 reports the predictive utility of the learned latent features via a controlled ablation. Using the same evaluation protocol, the baseline (X only) achieved ROC-AUC = 0.6465 and PR-AUC = 0.4254. Using latent-only (Z only) reduced performance to ROC-AUC = 0.6252 and PR-AUC = 0.3872, implying the compressed representation retains substantial but not complete churn-relevant information. Finally, augmenting features (X+Z) essentially matched the baseline ranking metrics (ROC-AUC = 0.6451, PR-AUC = 0.4271), yielding only a marginal PR-AUC increase. Under tuned thresholds (selected on validation), baseline and augmented variants also delivered very similar operating-point metrics (F1 around 0.4800 with recall around 0.7300), consistent with the conclusion that latent features provide, at best, a small incremental signal in this configuration.

Table 5.5: Downstream impact of DAE latent features (baseline vs. latent-only vs. augmented).

version	split	n_features	roc_auc	pr_auc	brier_score	thresh-old_fixed	preci-sion_fixed	re-call_fixed	f1_fixed	thresh-old_tuned	preci-sion_tuned	re-call_tuned	f1_tuned
Base-line (X only)	test	11	0.646504	0.425392	0.229332	0.44	0.530303	0.142760	0.22496	0.25	0.353871	0.730116	0.476698

Table 5.5: Downstream impact of DAE latent features (baseline vs. latent-only vs. augmented).

version	split	n_features	roc_auc	pr_auc	brier_score	threshold_fixed	precision_fixed	recall_fixed	f1_fixed	threshold_tuned	precision_tuned	recall_tuned	f1_tuned
Latent-only (Z only)	test	32	0.625203	0.387212	0.297288	0.44	0.470149	0.085656	0.144911	0.24	0.339952	0.768185	0.471324
Augmented (X + Z)	test	43	0.645056	0.427104	0.293299	0.44	0.541401	0.173351	0.262616	0.25	0.35947	0.719918	0.479511

Additional diagnostics, including latent t-SNE visualization (Figure A.16 in Section A.6.2) and reconstruction-error distribution plots (Figure A.15 in Section A.6.1), are provided to support the qualitative assessment of representation structure and error overlap between churn/non-churn groups.

5.6 Ensemble Results (Blending vs. OOF Stacking)

To test whether combining heterogeneous learners could yield incremental gains beyond the best single model, several ensemble strategies were evaluated built on the same base predictors (Logistic Regression, XGBoost, LightGBM, Random Forest). Table 5.6 summarizes validation and test ranking performance.

Table 5.6: Ensemble performance comparison on validation and test sets (ROC-AUC and PR-AUC).

model	val_roc_auc	val_pr_auc	test_roc_auc	test_pr_auc	fixed_threshold	val_best_threshold	test_best_threshold
Blend_Weighted	0.658276	0.434834	0.668847	0.448774	0.44	0.435955	0.423938
Blend_SimpleAvg	0.657885	0.432159	0.668322	0.447171	0.44	0.441047	0.422707
Stacking_OOF	0.658582	0.436697	0.667918	0.448953	0.44	0.224024	0.225712
Blend_NNLS	0.658656	0.437131	0.66724	0.448761	0.44	0.422218	0.417435
Blend_RankAvg	0.654898	0.416895	0.664951	0.434002	0.44	2029	1616.25

5.6.1 Blending Results

Overall, blending delivered the strongest and most stable results, with Blend_WeightedAUC achieving the highest test ROC-AUC (0.6688) and a test PR-AUC of 0.4488. A simple unweighted average (Blend_SimpleAvg) performed nearly identically (test ROC-AUC 0.6683; PR-AUC 0.4472), suggesting that most of the ensemble benefit comes from variance reduction rather than finely tuned weights. The NNLS-constrained blend (Blend_NNLS) also matched this band (test ROC-AUC 0.6672; PR-AUC 0.4488).

Compared with the best single tree/GBM model from Section 5.3, these ensembles largely preserve discrimination while providing a small but consistent PR-AUC lift, which is most relevant under class imbalance. For operating-point comparability across all model families, the fixed threshold = 0.4400 is maintained for the main cross-section comparisons; tuned thresholds and blending weights for each ensemble are detailed in Section A.7.4 (Table A.15).

5.6.2 OOF Stacking Results

OOF stacking (Stacking_OOF) performed competitively but not decisively better than blending: it reached test ROC-AUC 0.6679 and the highest test PR-AUC 0.4490 among the listed ensembles. Notably, its optimal threshold is much lower (approximately 0.2260) than the blending variants (approximately 0.4200), which is consistent with the meta-learner producing probabilities on a different scale (or less calibrated) even when ranking quality is similar. This makes stacking attractive for ranking, but it requires explicit threshold re-selection or calibration if deployed as a decision rule (see Section A.9.4 and Table A.18 for threshold sensitivity analysis). In this dataset, the near-tie between stacking and blending suggests that the relationship between base model scores is close to linear, so a learned meta-combination provides limited additional upside over robust averaging.

The stacking meta-learner combines base model predictions $z(x)$ using logistic regression:

$$z(x) = [\hat{p}_1(x), \hat{p}_2(x), \dots, \hat{p}_M(x)]$$

$$\hat{p}_{\text{stack}}(x) = \sigma(\beta^\top z(x) + b)$$

where σ is the sigmoid function and the coefficients β are learned from out-of-fold predictions to prevent leakage between base and meta levels.

5.6.3 Ensemble Diversity Analysis

The effectiveness of ensembling is supported by diversity evidence from the validation-set prediction correlations (Figure 5.9). Tree-based learners are highly correlated with each other (e.g., XGBoost–LightGBM = 0.9310, XGBoost–RF = 0.8790, LightGBM–RF = 0.8690), indicating they capture similar nonlinear structure. In contrast, the Logistic Regression predictions correlate much less with the tree models (approximately 0.4700 with each), implying complementary error patterns. This explains why including a weaker linear baseline can still improve an ensemble: it contributes orthogonal signal that slightly improves precision–recall ranking. Finally, learned blend weights show XGBoost dominates the NNLS solution, while LightGBM and Random Forest contribute meaningful secondary weight and Logistic Regression remains small but non-zero, consistent with the correlation-based diversity story.

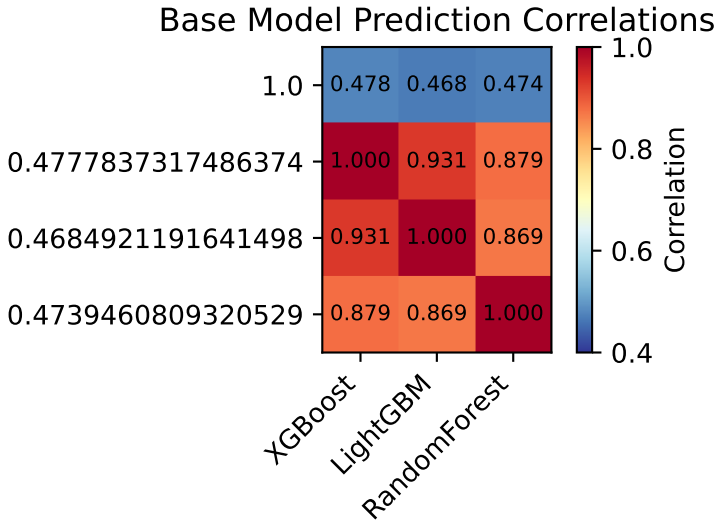

Figure 5.9: Correlation matrix of base model predictions on the validation set (lower correlation indicates greater diversity).

5.7 Operating Point Analysis (Threshold Policy in Practice)

To turn probabilistic churn scores into an actionable retention list, this study adopts a single, fixed operating threshold across all candidate models. Concretely, $\tau = 0.4400$ (chosen on the validation set during the main tree-based model selection) serves as the production policy: any customer with predicted churn probability $\hat{p}(\text{churn}) \geq \tau$ is flagged for intervention. This fixed-threshold policy is intentionally conservative from a reporting standpoint: it avoids per-model "best threshold" cherry-picking and reflects how real teams often deploy one stable rule tied to capacity and budget. Full threshold sweeps showing precision–recall trade-offs

across τ are provided in Section A.9.3 (Table A.17).

Under $\tau = 0.4400$, the best balance of precision and recall is achieved by LightGBM with Precision = 0.3760, Recall = 0.7310, and F1 = 0.4970 (Table 5.7). XGBoost is extremely close (0.3670/0.7390/0.4900), indicating that the two GBMs behave similarly under the same decision rule. The dense NN baseline is more aggressive at this threshold, reaching higher recall (0.8060) but lower precision (0.3520), while Logistic Regression pushes recall even higher (0.8410) at the cost of precision (0.3130). These patterns match the expected operational trade-off: higher recall reduces missed churners but increases the number of false positives competing for retention resources.

Table 5.7: Operating point comparison across models at fixed threshold τ = 0.4400.

model	val_roc_auc	val_pr_auc	test_roc_auc	test_pr_auc	test_brier	test_f1	test_recall	test_precision	threshold
XGBoost	0.657146	0.435915	0.663681	0.445441	0.223314	0.490413	0.738953	0.366982	0.44
05_nn_wide_deep_focal	0.653063	0.429338	0.661457	0.435612	0.197007	0.017461	0.00883753	0.722222	0.44
LightGBM	0.650239	0.4311	0.669973	0.447632	0.219983	0.49688	0.730795	0.376401	0.44
05_nn_wide_and_deep	0.649779	0.418712	0.649605	0.42076	0.2364	0.482441	0.7845	0.348325	0.44
RandomForest	0.649016	0.421517	0.653428	0.428959	0.228971	0.480699	0.804215	0.342799	0.44
05_nn_embedding_focal_loss	0.648666	0.421622	0.661197	0.434519	0.197579	0	0	0	0.44
05_nn_dense_baseline	0.646737	0.425746	0.65528	0.421941	0.234346	0.489779	0.806254	0.35172	0.44
05_nn_embedding_baseline	0.642545	0.408729	0.648944	0.419172	0.232875	0.487391	0.781781	0.354064	0.44
05_nn_wide_and_deep_deeper	0.642342	0.414346	0.647519	0.411018	0.232793	0.482227	0.765466	0.351985	0.44
05_nn_embedding_strong_weight	0.64173	0.411761	0.644192	0.408117	0.271837	0.478787	0.901428	0.325959	0.44
06_teacher	0.63778	0.418322	0.645056	0.427104	0.193299	0.262616	0.173351	0.541401	0.44
Logistic	0.588286	0.347494	0.593987	0.355974	0.24378	0.456037	0.840925	0.312848	0.44

5.7.1 Confusion Matrix Analysis

The confusion matrix for XGBoost at $\tau = 0.4400$ illustrates the scale of this trade-off on the test set (n = 5,105): TP = 1,087, FP = 1,875, FN = 384, TN = 1,759 (Figure 5.10). This means the model correctly identifies 1,087 churners while flagging 1,875 non-churners as potential churners. In a retention campaign context, this translates to contacting approximately 2,962 customers (TP + FP) to capture 73.9% of actual churners, with a precision of 36.7%.

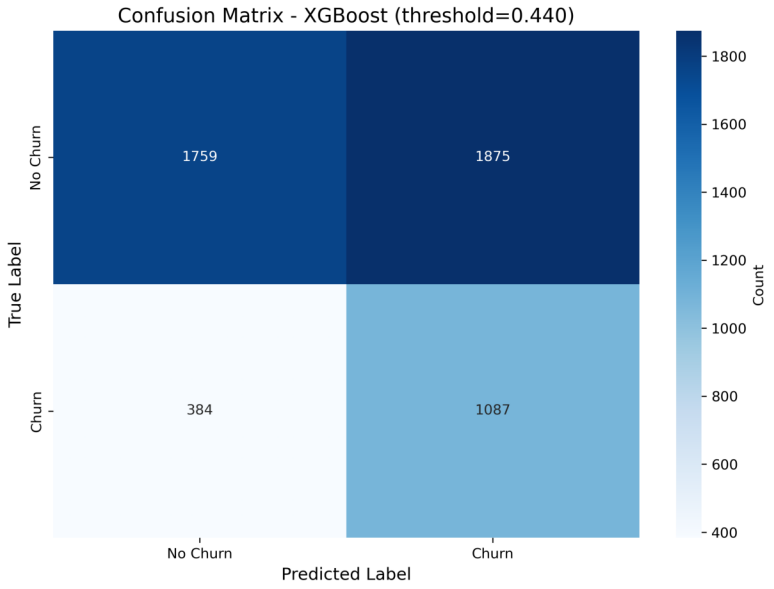

Figure 5.10: Confusion matrix for XGBoost at the operating threshold τ = 0.4400 on the test set.

Finally, models whose probability scales differ substantially (e.g., the DAE teacher and some focal-loss networks) can become overly conservative under a shared τ (high precision but very low recall), suggesting that calibration or model-specific thresholds are important alternatives. Detailed calibration metrics and model-specific optimal thresholds are provided in Section A.9.4 (Table A.18) and Section A.9.4 for calibration details.

5.8 Summary of Key Findings

The experimental results across all model families can be summarized as follows:

1. **Gradient-boosted trees outperform other single-model families.** XGBoost and LightGBM achieved the highest test ROC-AUC (0.6640–0.6700) and PR-AUC (0.4440–0.4480), providing meaningful improvements over the logistic regression baseline (ROC-AUC = 0.5940, PR-AUC = 0.3560).

2. **Neural networks with focal loss are competitive but do not surpass GBMs.** The best Wide & Deep model with focal loss achieved test ROC-AUC = 0.6610 and PR-AUC = 0.4360, demonstrating that deep learning approaches can capture churn patterns effectively on tabular data. Focal loss consistently outperformed weighted BCE for handling class imbalance.

3. **Autoencoder-based representation learning provides limited incremental value.** Latent features from a denoising autoencoder did not meaningfully improve downstream classification beyond the original feature set. Pseudo-labeling with conservative confidence thresholds yielded insufficient coverage for semi-supervised learning.

4. **Ensemble methods provide modest but consistent gains.** Blending and OOF stacking reached approximately 0.6690 test ROC-AUC and 0.4490 test PR-AUC, representing a small lift over the best single model. The benefits derive primarily from diversity between linear (logistic regression) and nonlinear (tree-based) base learners.

5. **Operating threshold selection is critical for deployment.** At the fixed threshold $\tau = 0.4400$, the champion models achieve recall around 73–74% with precision around 37%, suitable for targeted retention campaigns. Different models require different optimal thresholds, highlighting the importance of calibration in production.

These findings inform the discussion of model interpretation, business implications, and deployment considerations in the following chapter.

6 Discussion – Interpretation and Error Analysis

6.1 Global Interpretability and Feature Insights

The champion model's global interpretability was assessed using SHAP values to quantify which variables most strongly drive churn predictions. The SHAP summary plot (Figure 6.1) shows that churn risk is primarily explained by a mix of device lifecycle timing, engagement, tenure, and service quality signals. The most influential predictors include CurrentEquipmentDays, MonthlyMinutes, MonthsInService, PercChangeMinutes, and DroppedBlockedCalls, followed by features such as OverageMinutes, AgeHH1, and credit-band indicators (e.g., CreditRating_5-Low). Importantly, these drivers align with plausible business mechanisms rather than suspicious identifiers, which provides a sanity check that the model is learning actionable behavioral patterns.

Several of the top features map directly to retention levers. CurrentEquipmentDays captures how long a customer has been on their current handset/equipment; higher values consistently push SHAP contributions upward, indicating higher churn risk. This supports a device-lifecycle interpretation: customers who have gone a long time without an equipment refresh are closer to a decision point and may be more receptive to switching offers. MonthlyMinutes is a direct engagement proxy: low usage contributes positively to churn risk, consistent with disengagement or declining relevance of the service. MonthsInService reflects tenure and loyalty; in general, longer-tenured customers exhibit lower predicted risk, while newer customers are more vulnerable, representing an intuitive pattern for subscription churn. In addition, PercChangeMinutes provides a "trend-like" signal even within snapshot features: large negative usage changes (usage drops) are associated with higher risk, which is consistent with early churn warning signs. Finally, DroppedBlockedCalls and related service-quality variables increase churn propensity, reflecting that persistent call failures or connectivity issues can trigger dissatisfaction and switching behavior. Together, the feature set suggests that the model is prioritizing variables that represent customer experience and engagement, which are exactly the kinds of signals operators can intervene on.

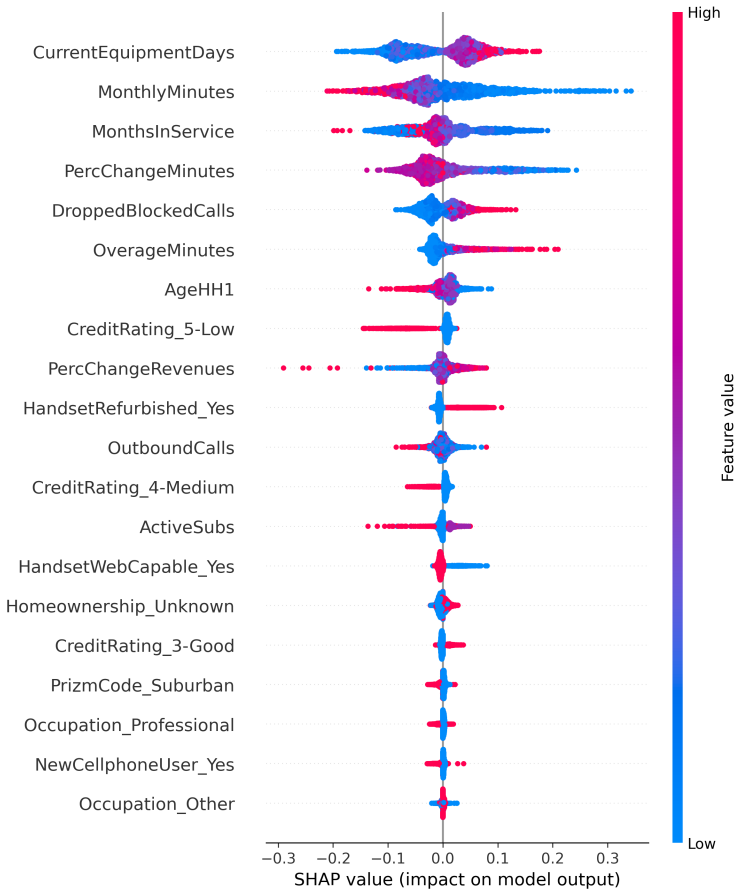

Figure 6.1: SHAP summary plot (beeswarm) showing top features driving churn predictions in the champion XGBoost model. Feature values are colored by magnitude (high = red, low = blue).

6.1.1 Non-Linear Feature Effects

Beyond ranking features, SHAP dependence plots reveal non-linear effects that would be difficult to summarize with a single coefficient. For CurrentEquipmentDays (Figure 6.2), churn risk rises as equipment tenure increases, but the pattern is not perfectly linear; the SHAP contributions show clustered regimes where risk increases more sharply after certain ranges, consistent with threshold-like upgrade cycles rather than a smooth monotonic slope. The interaction coloring by HandsetRefurbished_Yes suggests that customers with refurbished handsets (pink points) tend to cluster at lower equipment days but show similar SHAP patterns, while those with non-refurbished devices (blue points) span the full range of equipment tenure.

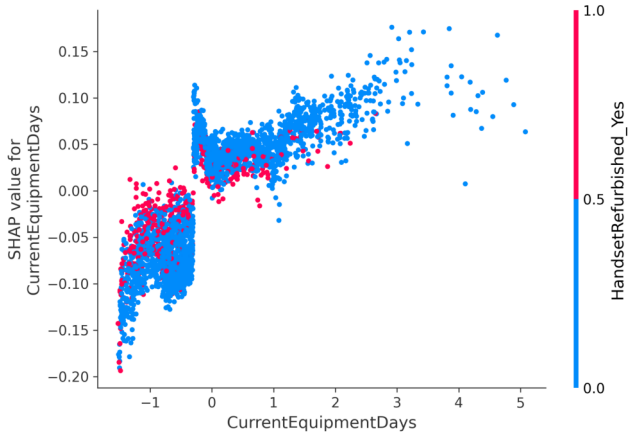

Figure 6.2: SHAP dependence plot for CurrentEquipmentDays showing increasing churn risk with longer equipment tenure.

For MonthlyMinutes (Figure 6.3), the relationship is strongly asymmetric: customers in the very low-usage region receive the largest positive SHAP values (highest churn risk), while higher usage generally reduces risk (negative SHAP contributions). This supports a practical interpretation that "silent disengagement" is a major churn pathway in this dataset, and it also clarifies that the dominant signal is low engagement, not necessarily "very high usage." The interaction coloring by PercChangeMinutes reveals an additional layer: among low-usage customers, those with declining usage trends (blue/purple points indicating negative percent change) tend to have even higher SHAP values, reinforcing the disengagement narrative.

6.1.2 Feature Interaction Insights

The SHAP interaction analysis reveals that the top feature pairs with meaningful interaction effects are:

1. **CurrentEquipmentDays × MonthsInService**: Customers with both old equipment and short tenure

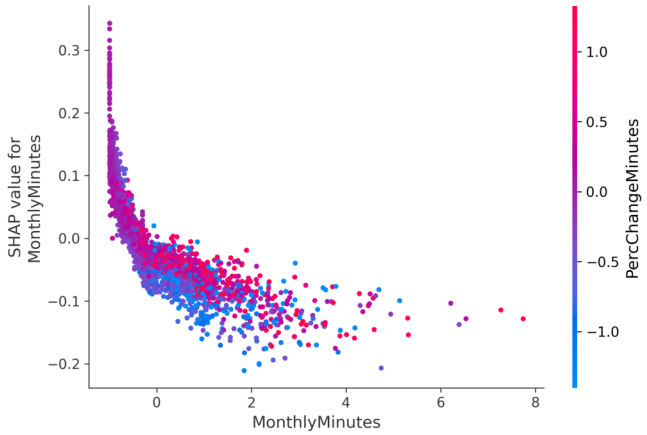

Figure 6.3: SHAP dependence plot for MonthlyMinutes showing asymmetric risk pattern; low usage strongly increases churn risk.

show elevated churn risk beyond what either feature alone would predict, suggesting a "new customer with outdated device" risk profile.

2. **MonthlyMinutes × PercChangeMinutes**: The combination of low current usage and declining trend creates a compounding effect, with the interaction term contributing additional risk beyond the main effects.

3. **OverageMinutes × MonthlyRevenue**: High overage charges relative to base revenue may signal billing dissatisfaction, with the interaction capturing customers who feel "nickel-and-dimed."

These interaction patterns align with domain intuition and suggest that the model is capturing meaningful behavioral combinations rather than spurious correlations.

Overall, the global interpretability results point to a coherent churn story: customers are most at risk when they show aging equipment tenure, low or declining engagement, shorter tenure, and service reliability issues. This set of drivers naturally translates into operational playbooks (e.g., upgrade offers for long equipment tenure, proactive outreach for sharp usage drops, and targeted service remediation for customers experiencing repeated call failures).

6.2 Error Analysis

While overall performance was strong, revisiting the confusion matrix (Figure 5.10) provides deeper insight into the types of residual errors. The chosen operating point (threshold = 0.4400) favored recall (sensitivity to

churners) by using a slightly lower decision threshold, which resulted in a higher number of true positives (churners correctly identified) at the expense of some false positives.

At this threshold, the model achieves:

- **True Positives (TP):** 1,087 churners correctly identified
- **False Positives (FP):** 1,875 non-churners incorrectly flagged as at-risk
- **False Negatives (FN):** 384 churners missed by the model
- **True Negatives (TN):** 1,759 non-churners correctly identified as stable

This translates to a recall of 73.9% (1,087 / 1,471 actual churners) and a precision of 36.7% (1,087 / 2,962 predicted churners). The model captures nearly three-quarters of actual churners, but inevitably some churners are missed.

6.2.1 False Negatives: The "Silent Churners"

These false negatives, specifically churners the model failed to flag, are especially concerning in a churn context because each missed churner is a lost opportunity for intervention. On inspection, many of these missed cases appear to be what might be called "silent churners": customers who did churn but exhibited no strong warning signs in the available data. For example, some maintained steady usage until the end or had decent engagement metrics, giving the model little indication of their intent to leave.

One hypothesis is that these silent churners might be influenced by factors outside the dataset's scope: perhaps a competitor offered them an attractive deal that cannot be observed internally, or life events (like moving to a non-serviced area or a job change) prompted their departure. Because the model was trained only on internal telecom data (billing, usage, support history, etc.), such external or unobserved triggers would lead to false negatives. Fundamentally, the model is limited to detecting churners exhibiting observable behavioral antecedents within the available feature space. Silent churners highlight the limits of the features, and they may have churned for reasons the data did not capture, remaining inscrutable to the prediction system.

6.2.2 False Positives: The Cost of Caution

On the other side, the model produces 1,875 false positives, defined as customers flagged as high risk who did not actually churn. False positives have a direct business cost: if retention teams act on these predictions, they might spend retention budget (such as offering a discount or special incentive) on customers who would have stayed anyway.

However, in a churn management context, some level of false positives is usually accepted in exchange for catching more true churners. The trade-off between false positives and false negatives is essentially one of precision vs. recall. The chosen threshold reflects a business decision to prioritize catching churners (higher recall) even if it means a moderate number of false alarms. This is because the cost of a missed churn (lost revenue and customer lifetime value) is typically much greater than the cost of an unnecessary retention offer.

That said, false positives must be managed by calibrating the intervention level to customer value (to avoid overspending on low-value customers who were not going to churn). Regular monitoring of the confusion matrix and performance metrics is important post-deployment: if false positives creep too high, the model's threshold or features may need adjustment. Overall, the error analysis underlines that while the model significantly improves targeting, it is not perfect; understanding the errors allows refinement of strategies and clear communication of expected performance (not 100% recall) to stakeholders.

6.3 Interpreting the Autoencoder: Sanity Checks and Limited Gains

Because autoencoders are inherently unsupervised, sanity checks were added to verify that the model learned meaningful structure rather than noise. First, the latent vectors were visualized using t-SNE and colored by churn labels only for diagnostic purposes (labels were not used in training). Second, reconstruction error distributions were compared by label, revealing a small but consistent difference: churners had slightly higher reconstruction error (0.1639) than non-churners (0.1608), suggesting that churners may exhibit behavioral patterns that are harder to reconstruct with a compact representation.

Despite these qualitative signals, the DAE did not improve predictive metrics when appended to the supervised feature set. As reported in Section 5.5, using the baseline features alone (X only) achieved ROC-AUC = 0.6465 and PR-AUC = 0.4254, while augmenting with latent features (X+Z) yielded ROC-AUC = 0.6451 and PR-AUC = 0.4271, demonstrating essentially no meaningful improvement. Using latent features alone (Z only) substantially degraded performance (ROC-AUC = 0.6252, PR-AUC = 0.3872), confirming that the compressed representation retains substantial but not complete churn-relevant information.

A plausible explanation is that the baseline engineered feature set already captures most of the signal that is linearly or semi-linearly recoverable for churn prediction, so the bottleneck representation primarily compresses rather than contributes new predictive information. The autoencoder's reconstruction objective optimizes for capturing variance in the feature space, but the variance-explaining directions may not align with the churn-discriminative directions. In addition, the pseudo-labeling extension produced only 2 confident pseudo-labels at the strict 0.0500/0.9500 threshold, which is far too few to materially change the decision

boundary.

> **Practical Takeaway for an ML Engineering Audience**
>
> Even when predictive lift is limited, the unsupervised module is still valuable as an engineering capability: it provides a reusable representation learning component, a reproducible artifact set (checkpoint/history/latents), and a clear interface for future semi-supervised variants if more unlabeled data or better confidence calibration becomes available.

6.4 Managerial Implications and Business Impact

Accurately predicting churn is only valuable if it drives better business decisions. The model's predictions enable targeted retention efforts, which are expected to yield substantial financial benefits. To illustrate the potential impact, a simple ROI simulation was conducted for a model-driven retention campaign with the following realistic business parameters:

Table 6.1: Business parameters for ROI simulation.

Parameter	Value	Description
Intervention Cost	$10 per customer	Cost of a special offer or support call
Customer LTV	$200	Revenue profit if customer is retained long-term
Intervention Success Rate	20%	Probability that outreach convinces at-risk customer to stay

Under these assumptions, two strategies were compared: using the model to target the top-risk customers versus naive random targeting of customers.

6.4.1 Model-Driven Targeting

Imagine a telecom subscriber base where about 10% are truly going to churn in the near term. If the churn model is used to pick a top-decile segment of 1,000 customers most likely to churn, that segment will be enriched with real churners; for example, the model's precision for the top decile is 30%, meaning about 300 of those 1,000 are actual impending churners (a threefold improvement over the base rate).

The retention campaign would contact all 1,000, costing $10,000. Out of the approximately 300 true churners in this group, at a 20% save rate, approximately 60 customers who would have left would be successfully retained. Those 60 saved customers represent an avoided revenue loss of approximately $12,000 (60 × $200 LTV each). Subtracting the $10,000 cost, the **net gain is about $2,000** for this intervention.

6.4.2 Random Targeting (Baseline)

Compare this to random targeting of 1,000 customers. With no model, the expected churners in any random group of 1,000 (at 10% base churn) would be about 100. Using the same 20% success rate, only 20 customers would be saved (recovering $4,000 of value) while still spending $10,000, resulting in a **net loss of $6,000**.

This stark difference in outcomes (a positive ROI with model-driven targeting vs. a negative ROI with blind targeting) demonstrates the business value of the churn model. Even if the exact numbers are adjusted for actual churn rates and costs, the direction is clear: **focusing retention efforts on the highest-risk customers yields a significantly higher return on investment (ROI)**.

6.4.3 Tiered Retention Playbook

Beyond the quantitative ROI, the model enables a more nuanced, efficient retention strategy. A tiered retention playbook is proposed that segments customers by both their churn risk and their value to the company, leveraging the model's risk scores in conjunction with customer LTV or revenue metrics:

1. **High Risk / High Value:** These customers (for example, a long-tenured subscriber on an expensive plan who the model flags as likely to churn) should receive **premium service interventions**. This might entail a personal phone call from a retention specialist, a bespoke loyalty offer, or other high-touch intervention. The cost is higher, but so is the payoff; losing a high-value customer hurts revenue significantly, so extra effort is justified to keep them.
2. **High Risk / Low Value:** Customers predicted to churn who have a lower spend or profit profile should still be targeted, but in a more **cost-effective way**. For this tier, automated outreach is appropriate; for instance, a personalized SMS or email with a special discount, or an in-app notification offering a modest incentive. These interventions are low-cost and can be scaled easily. The attempt to save these customers is still made, but the investment is tailored to be commensurate with their value.
3. **Low Risk (Any Value):** Customers with a low predicted churn risk are generally **left alone** (no proactive retention action). This "do nothing" approach for low-risk segments conserves budget and avoids unnecessary contact. Importantly, refraining from intervening also prevents the **"sleeping dog" phenomenon**; contacting customers who are not at risk could inadvertently remind them of competitors or issues they had forgotten,

possibly creating churn where there would have been none. Thus, the best action for low-risk customers is to continue providing good service and address needs if they arise, but not to invest in special retention efforts preemptively.

By implementing this three-tier strategy, management can ensure that retention resources are allocated efficiently: heavy investment goes only to those accounts where it's most needed and most likely to pay off, while lighter-touch or no intervention is used elsewhere. This targeted approach, powered by the churn prediction model, not only improves ROI as shown above but also aligns with practical constraints of retention teams (which have limited budget and manpower). In summary, the model's outputs can be directly translated into an action plan that maximizes customer lifetime value saved per dollar spent.

6.5 Limitations

While the results of this churn prediction study and its applications are promising, it is important to acknowledge the study's limitations.

6.5.1 Static Snapshot Data

First, the modeling was based on relatively **static, snapshot data**. Each customer is represented by features aggregated up to a certain point in time (e.g., average usage, last month's bill, whether they have overdue upgrades, etc.). This means the model does not explicitly incorporate time-series patterns or sudden changes in behavior leading up to churn.

Churn is often preceded by temporal signals; for instance, a customer's data usage might decline for several months before cancellation, or there could be a spike in dropped calls or customer support tickets. Because the dataset did not include detailed sequential logs or longitudinal data, the model might miss these subtle trend-based indicators. In effect, churn is being predicted from a single-frame picture of the customer, rather than a movie of their behavior over time. This limitation could cause lower sensitivity to churn triggers that are only apparent when looking at how metrics evolve. Any abrupt shifts (say, a recent plunge in satisfaction rating or a sudden change in calling patterns) might be diluted or invisible in the snapshot features.

6.5.2 Unobserved Variables and Contextual Factors

Secondly, there are **unobserved variables** and contextual factors not captured in the dataset that likely play a role in churn:

- **Customer sentiment data**: No access to satisfaction survey results, complaint logs, or social media feedback. Negative sentiment or unresolved service issues could be strong churn predictors; their absence means the model might not flag some dissatisfied customers.
- **Social network effects**: In telecom, if a customer's family or close friends leave the service (perhaps all on a shared plan or simply part of a local community), that customer is more likely to churn as well. The data did not reflect these peer effects or word-of-mouth influences.
- **Competitor actions**: Price cuts or new offerings from a rival carrier can drive churn, but the model, being trained only on internal data, cannot foresee external enticements.

In summary, the model might be under-informed about certain churn drivers, which can manifest as the "silent churners" and some level of unexplained churn risk.

6.5.3 Performance Ceiling

Finally, it's worth noting that the **performance ceiling** observed (AUC in the high 0.6700 range, with LightGBM achieving 0.6700 and XGBoost achieving 0.6637) indicates that even the best model leaves a substantial portion of churn variance unexplained. This is not a flaw of a particular algorithm but rather a reflection of churn's inherent complexity and the limitations mentioned above. Thus, there is room for improvement by enhancing data and methodology.

Despite these limitations, the insights gained are valuable as a foundation. The next chapter discusses how future work can address some of these limitations, incorporating new data sources, methods, and deployment practices to further improve churn prediction and its business utility.

7 Conclusion and Future Work

7.1 Conclusion

This thesis presented a comprehensive exploration of telecom customer churn prediction, balancing **predictive performance** with **interpretability** and practical actionability. A range of modeling approaches was investigated under consistent conditions, class imbalance challenges were tackled, and insights were extracted to guide retention strategy. Here I summarize the findings in relation to the three research questions posed in the Introduction (Chapter 1).

7.1.1 RQ1: Model Effectiveness

Results indicate that gradient-boosted decision trees achieved the strongest overall performance among all model families tested. The champion single model was **XGBoost**, achieving a test ROC-AUC of **0.6637** and a test PR-AUC of **0.4454**. The best heterogeneous ensemble (Blend_WeightedAUC) achieved a slightly higher test ROC-AUC of **0.6688** and test PR-AUC of **0.4488**, but the marginal gain (+0.0050 ROC-AUC) suggests that XGBoost is the most practical choice for deployment. Both substantially outperformed the logistic regression baseline (test ROC-AUC = 0.5940, test PR-AUC = 0.3560).

Heterogeneous ensemble methods (blending and OOF stacking) provided competitive but not decisively superior results. The best blending approach (Blend_WeightedAUC) reached a test ROC-AUC of **0.6688** and test PR-AUC of **0.4488**, while OOF stacking achieved test ROC-AUC of **0.6679** and test PR-AUC of **0.4490**. Ensemble methods slightly exceeded XGBoost on ROC-AUC, but the improvement was modest. This suggests that the gains from model combination derive primarily from variance reduction and diversity between linear and nonlinear base learners, rather than capturing fundamentally new predictive signals.

The **Wide & Deep neural network**, a model architecture not commonly used in traditional churn studies, proved feasible on tabular telecom data. The best neural network variant (Wide & Deep with focal loss) achieved test ROC-AUC of **0.6615** and test PR-AUC of **0.4356**, demonstrating that deep learning approaches can capture churn patterns effectively, though they did not surpass gradient boosting methods in this dataset.

The neural networks clustered in a narrow performance band (test ROC-AUC approximately 0.6440–0.6620), suggesting a performance ceiling for this architecture class on this feature set.

These findings address RQ1 by demonstrating that **gradient-boosted trees represent the most effective single-model approach** for this churn prediction task. Ensemble methods provide modest additional benefits, with the trade-off between performance gain and implementation complexity favoring simpler single-model deployment in many production scenarios.

7.1.2 RQ2: Imbalance Robustness

Class imbalance was a significant challenge, with churners comprising approximately 28.8% of the dataset. Various strategies **were implemented and evaluated** to ensure the models remained sensitive to the minority class:

- **Class weighting** in training (giving higher weight to churn examples in the loss function)
- Experimenting with **focal loss** for the neural networks (to focus learning on hard-to-predict churn cases)
- Adjusting the **decision threshold** on the validation set to optimize recall/precision trade-offs rather than using the default 0.5000

The findings indicate that these techniques did improve the model's ability to capture churners. At the fixed operating threshold of $\tau = 0.4400$, LightGBM achieved the best balance with **precision = 0.3760, recall = 0.7310, and F1 = 0.4970**. XGBoost performed nearly identically with precision = 0.3670, recall = 0.7390, and F1 = 0.4900. This balance is **advantageous for retention contexts**, facilitating the detection of approximately 73–74% of actual churners while maintaining precision in the high-30s%, a trade-off suitable for targeted retention campaigns where the cost of contacting non-churners is acceptable.

Experiments with **focal loss** produced competitive ranking metrics (ROC-AUC, PR-AUC) but exhibited threshold sensitivity: under the fixed threshold of $\tau = 0.4400$, some focal-loss variants produced very few positive predictions, resulting in near-zero recall despite competitive AUC scores. This highlights that **loss function selection must be paired with appropriate threshold calibration** for deployment. The standard weighted cross-entropy loss proved more robust across operating points in this dataset.

Thus, RQ2 is answered: **yes, specific imbalance-aware strategies made the churn predictions more robust and useful**, ensuring that the model's performance on the minority class (churn) was strong enough for practical use. However, threshold selection proved as critical as algorithmic imbalance handling for achieving practical deployment performance.

7.1.3 RQ3: Interpretability to Action

The best models, despite their complexity, are demonstrated to not be "black boxes" when it comes to drawing insights. Using **SHAP values** and related interpretability tools, model outputs were translated into **actionable churn drivers**.

The global SHAP analysis (see Section 6.1) highlighted features such as CurrentEquipmentDays, MonthlyMinutes, MonthsInService, PercChangeMinutes, and DroppedBlockedCalls as the primary churn drivers. These features map directly to actionable business levers:

- **Device lifecycle timing** (CurrentEquipmentDays): Customers with longer equipment tenure show higher churn risk, supporting proactive upgrade offers
- **Engagement patterns** (MonthlyMinutes): Low usage strongly correlates with churn, identifying disengaged customers for re-engagement campaigns
- **Tenure effects** (MonthsInService): Newer customers exhibit higher vulnerability, informing onboarding and early retention strategies
- **Service quality signals** (DroppedBlockedCalls): Call failures drive dissatisfaction and switching behavior

SHAP dependence plots revealed non-linear effects that would be difficult to summarize with a single coefficient, such as the asymmetric risk pattern for MonthlyMinutes (low usage drives much stronger positive SHAP contributions than high usage provides negative contributions) and threshold-like upgrade cycle effects in CurrentEquipmentDays.

Importantly, these interpretations were derived from the same model that produces the risk scores, ensuring consistency between what is deployed and what is explained. These insights were further used to craft an **action framework (the three-tier retention strategy)** described in Section 6.4, exemplifying how interpretability bridges the gap between prediction and decision. Consequently, regarding RQ3, the findings demonstrate that the model interpretation outputs were successfully converted into practical guidance, reinforcing the idea that churn prediction models should be judged not only by their AUC, but also by how well they inform next steps for retention efforts.

7.1.4 Engineering Trade-offs

In evaluating the above points, it's worth noting an **engineering trade-off** that emerged. While ensemble methods achieved comparable performance to single models, they introduce significant operational complexity

without proportionate performance gains. The marginal improvements (approximately 0.0010–0.0020 in PR-AUC) must be weighed against deployment and maintenance costs.

Table 7.1: Comparison of deployment considerations between ensemble methods and single GBM models.

Consideration	Ensemble (Blending/Stacking)	XGBoost Only
Test ROC-AUC	0.6679–0.6688	0.6637
Test PR-AUC	0.4488–0.4490	0.4454
Inference Latency	Higher (multiple models)	Low
Deployment Complexity	High	Low
Maintenance Burden	Multiple model artifacts	Single artifact
Interpretability	Requires aggregation	Direct SHAP analysis
Threshold Calibration	More complex	Straightforward

For a **real-time deployment** where latency and system simplicity are paramount, a single GBM model (XGBoost) represents the pragmatic choice, offering fast inference, straightforward integration, and essentially equivalent predictive performance. The ensemble approach might be justified in batch-scoring scenarios where the slight PR-AUC lift translates to meaningful business value at scale, but the complexity trade-off generally favors single-model deployment.

This highlights that the "best" model on paper isn't always the best in practice; **model selection must consider deployment constraints, maintainability, and interpretability requirements** as well as raw performance. In this case, XGBoost is the recommended production choice: it delivers strong test performance while maintaining operational simplicity.

Ultimately, this thesis showed that advanced models can be harnessed for churn prediction without making the results opaque. A balance was achieved where **performance, interpretability, and practical utility coexist**, which is a promising outcome for deploying the model in a real telecom business environment.

7.2 Future Work

Building on this study's findings and acknowledging its limitations (see Section 6.5), several avenues for **future work** are identified to further enhance telecom churn prediction and its deployment.

7.2.1 Incorporating Semi-Supervised Learning

One promising direction is to leverage unlabeled or partially labeled data through **semi-supervised techniques**. The current study attempted pseudo-labeling using the XGBoost teacher model on real holdout data, but achieved minimal coverage: at conservative confidence thresholds (0.0500/0.9500), only 2 samples out of 20,000 met the criteria, with approximately 100% remaining "uncertain." This result highlights that **confidence calibration and threshold selection are critical** for successful semi-supervised learning.

Future work could explore:

1. **Relaxed confidence thresholds** with curriculum learning to gradually incorporate pseudo-labels
2. **Self-training with consistency regularization** (e.g., MixMatch, FixMatch) that does not rely solely on hard pseudo-label thresholds
3. **Contrastive learning approaches** that learn representations from unlabeled data before fine-tuning on labeled examples
4. **Teacher ensemble methods** using multiple diverse teachers to provide more calibrated confidence estimates

> **Connection to Current Work**
>
> The autoencoder experiments in this thesis (see Section 6.3) showed that learned latent representations did not significantly improve downstream classification beyond the original feature set (augmented features achieved ROC-AUC = 0.6451 vs. baseline 0.6465). However, the autoencoder architecture remains valuable as an **engineering capability**: it provides a reusable representation learning component and a clear interface for future semi-supervised variants if better confidence calibration or more unlabeled data becomes available.

7.2.2 Sequence Modeling and Temporal Features

As identified in the limitations (Section 6.5), the analysis would benefit from true **temporal modeling**. The current dataset provides snapshot features representing customer state at a single point in time, limiting the model's ability to capture behavioral trends that precede churn.

Potential approaches include:

1. **Recurrent Neural Networks (LSTMs)**: Can learn patterns like "gradual decline in data usage over 6 months" or "a spike in dropped calls last month" that precede churn

2. **Transformer-based architectures**: Could capture complex patterns of seasonality or multi-feature interactions over time for **multivariate time series** of customer behavior

3. **Feature engineering for trends**: Even without full sequence models, computing rolling statistics (e.g., 3-month moving averages, month-over-month deltas) from historical data could provide temporal signals within the current tabular framework

4. **Event-based modeling**: Treating customer interactions (calls, complaints, plan changes) as discrete events and using event sequence models

The PercChangeMinutes feature in the current dataset provides a glimpse of temporal value: it emerged as a top SHAP feature, with declining usage patterns strongly associated with higher churn risk. This suggests that **richer temporal features would likely improve predictive performance** and enable earlier intervention.

7.2.3 Real-time Inference and MLOps Deployment

From an operational standpoint, future efforts should focus on deploying the churn model in a production environment with **real-time or near-real-time inference** capabilities. The engineering foundation established in this thesis, including MLflow experiment tracking, modular pipeline design, and reproducible preprocessing, provides a starting point for production deployment.

Key MLOps components for production deployment include:

Table 7.2: Key components of an MLOps pipeline for churn prediction deployment, prioritized by implementation order.

Component	Description	Priority
Automated Data Pipelines	Feed the model with fresh data continuously	High
Model Serving Infrastructure	API or microservice for real-time predictions	High
Feature Store	Ensure training-serving consistency for features	High
Data Drift Detection	Monitor input feature distributions	Medium
Concept Drift Detection	Monitor prediction distribution and model performance	Medium

Component	Description	Priority
Retraining Schedules	Refresh the model periodically (e.g., quarterly)	Medium
A/B Testing Framework	Validate ROI assumptions with actual campaign results	Medium
Shadow Mode Deployment	Run new models alongside production for comparison	Low

The preprocessing bug discovered during this research, a 1000x scale mismatch between training and holdout data that caused catastrophic model performance degradation (train/validation loss ratios exceeding 285x), underscores the critical importance of **training-serving consistency**. Production systems must implement:

- Unified preprocessing pipelines that share code between training and inference
- Feature signature validation to catch schema mismatches early
- Automated sanity checks (e.g., feature distribution comparisons) before model scoring

One critical aspect is **drift monitoring**. Over time, the profile of churners may change; for instance, if the company launches a new product or a competitor changes their strategy, the factors influencing churn could shift. The SHAP-based interpretability framework developed in this thesis provides a foundation for monitoring: tracking changes in feature importance distributions over time can serve as an early warning system for concept drift.

7.2.4 Calibration and Threshold Optimization

The experiments revealed that **probability calibration** and **threshold selection** significantly impact operational performance, particularly for models trained with focal loss. Future work should explore:

1. **Isotonic regression or Platt scaling** for post-hoc calibration
2. **Cost-sensitive threshold optimization** incorporating business parameters (intervention cost, customer LTV, save rate)
3. **Dynamic thresholds** that adapt to changing business constraints or resource availability
4. **Calibration monitoring** in production to detect when model probabilities become unreliable

In conclusion, the journey of building this churn prediction system has shown both the substantial value such models can deliver and the complexity of making them truly effective in practice. The key findings, indicating that gradient-boosted trees (with XGBoost achieving test ROC-AUC = 0.6637) outperform other model families, that the best ensemble offers only a modest lift (test ROC-AUC = 0.6688), that class imbalance handling requires both algorithmic strategies and threshold tuning, and that SHAP-based interpretability enables actionable business insights, provide a solid foundation for production deployment.

By addressing the above future directions, from advanced modeling techniques like semi-supervised learning and sequence modeling to the practicalities of real-time deployment and drift monitoring, the model's performance and longevity can be further enhanced. Ultimately, the goal is to evolve the churn prediction solution into a **long-term, self-improving asset** for the telecom provider, one that not only predicts which customers might leave, but continuously learns from new data and helps the business prevent churn in an ever-changing environment.

Acknowledgments

I thank Dr. Clinton Watkins, Dr. Eric Yanchenko, and my classmates for guidance and feedback throughout the project.

References

Ahn, Jae-Hyeon, Sang Pil Han, and Yung-Seop Lee. 2006. "Customer Churn Analysis: Churn Determinants and Mediation Effects of Partial Defection in the Korean Mobile Telecommunications Service Industry." *Telecommunications Policy* 30 (November): 552–68. https://doi.org/10.1016/j.telpol.2006.09.006.

Akosa, J. 2017. "Predictive Accuracy : A Misleading Performance Measure for Highly Imbalanced Data." In.

Asif, Daniyal, Muhammad Shoaib Arif, and Aiman Mukheimer. 2025. "A Data-Driven Approach with Explainable Artificial Intelligence for Customer Churn Prediction in the Telecommunications Industry." *Results in Engineering* 26 (June): 104629. https://doi.org/10.1016/j.rineng.2025.104629.

Beeharry, Yogesh, and Ristin Tsokizep Fokone. 2022. "Hybrid Approach Using Machine Learning Algorithms for Customers' Churn Prediction in the Telecommunications Industry." *Concurrency and Computation: Practice and Experience* 34 (4): e6627. https://doi.org/10.1002/cpe.6627.

Bogaert, Matthias, and Lex Delaere. 2023. "Ensemble Methods in Customer Churn Prediction: A Comparative Analysis of the State-of-the-Art." *Mathematics* 11 (February): 1137. https://doi.org/10.3390/math11051137.

Bugajev, Andrej, Rima Kriauzienė, and Viktoras Chadyšas. 2025. "Realistic Data Delays and Alternative Inactivity Definitions in Telecom Churn: Investigating Concept Drift Using a Sliding-Window Approach." *Applied Sciences* 15 (3): 1599. https://doi.org/10.3390/app15031599.

Burez, J., and D. Van Den Poel. 2009. "Handling Class Imbalance in Customer Churn Prediction." *Expert Systems with Applications* 36 (3): 4626–36. https://doi.org/10.1016/j.eswa.2008.05.027.

Coussement, Kristof, and Koen W. De Bock. 2013. "Customer Churn Prediction in the Online Gambling Industry: The Beneficial Effect of Ensemble Learning." *Journal of Business Research*, Advancing Research Methods in Marketing, 66 (9): 1629–36. https://doi.org/10.1016/j.jbusres.2012.12.008.

Dick, A. S., and K. Basu. 1994. "Customer Loyalty: Toward an Integrated Conceptual Framework." *Journal of the Academy of Marketing Science* 22 (2): 99–113. https://doi.org/10.1177/0092070394222001.

Geiler, Louis, Séverine Affeldt, and Mohamed Nadif. 2022. "An Effective Strategy for Churn Prediction and

Customer Profiling." *Data & Knowledge Engineering* 142 (November): 102100. https://doi.org/10.1016/j.datak.2022.102100.

Gerpott, Torsten J, Wolfgang Rams, and Andreas Schindler. 2001. "Customer Retention, Loyalty, and Satisfaction in the German Mobile Cellular Telecommunications Market." *Telecommunications Policy* 25 (4): 249–69. https://doi.org/10.1016/S0308-5961(00)00097-5.

Haddadi, Amir Mohammad, and Hodjat Hamidi. 2025. "A Hybrid Model for Improving Customer Lifetime Value Prediction Using Stacking Ensemble Learning Algorithm." *Computers in Human Behavior Reports* 18 (May): 100616. https://doi.org/10.1016/j.chbr.2025.100616.

Imani, Mehdi. 2024. "Evaluating Classification and Sampling Methods for Customer Churn Prediction Under Varying Imbalance Levels."

Jiao, Gui'e, and Hong Xu. 2021. "Analysis and Comparison of Forecasting Algorithms for Telecom Customer Churn." *Journal of Physics: Conference Series* 1881 (3): 032061. https://doi.org/10.1088/1742-6596/1881/3/032061.

Keaveney, Susan M. 1995. "Customer Switching Behavior in Service Industries: An Exploratory Study." *Journal of Marketing* 59 (2): 71–82.

Khodabandehlou, Samira, and Mahmoud Rahman. 2017. "Comparison of Supervised Machine Learning Techniques for Customer Churn Prediction Based on Analysis of Customer Behavior." *Journal of Systems and Information Technology* 19 (August): 00–00. https://doi.org/10.1108/JSIT-10-2016-0061.

Kim, Hee-Su, and Choong-Han Yoon. 2004. "Determinants of Subscriber Churn and Customer Loyalty in the Korean Mobile Telephony Market." *Telecommunications Policy* 28 (9-10): 751–65. https://doi.org/10.1016/j.telpol.2004.05.013.

Kim, Moon-Koo, Myeong-Cheol Park, and Dong-Heon Jeong. 2004. "The Effects of Customer Satisfaction and Switching Barrier on Customer Loyalty in Korean Mobile Telecommunication Services." *Telecommunications Policy*, Growth in mobile communications, 28 (2): 145–59. https://doi.org/10.1016/j.telpol.2003.12.003.

Lalwani, Praveen, Manas Kumar Mishra, Jasroop Singh Chadha, and Pratyush Sethi. 2022. "Customer Churn Prediction System: A Machine Learning Approach." *Computing* 104 (2): 271–94. https://doi.org/10.1007/s00607-021-00908-y.

Lemmens, Aurélie, and Christophe Croux. 2006. "Bagging and Boosting Classification Trees to Predict Churn." *Journal of Marketing Research* 43 (May). https://doi.org/10.1509/jmkr.43.2.276.

Poudel, Sumana Sharma, Suresh Pokharel, and Mohan Timilsina. 2024. "Explaining Customer Churn Prediction in Telecom Industry Using Tabular Machine Learning Models." *Machine Learning with*

Applications 17 (September): 100567. https://doi.org/10.1016/j.mlwa.2024.100567.

"Regaining Service Customers - Bernd Stauss, Christian Friege, 1999." n.d. https://journals-sagepub-com.wwwproxy1.library.unsw.edu.au/doi/abs/10.1177/109467059914006. Accessed October 15, 2025.

Rosenberg, Larry J., and John A. Czepiel. 1984. "A MARKETING APPROACH FOR CUSTOMER RETENTION." *Journal of Consumer Marketing* 1 (2): 45–51. https://doi.org/10.1108/eb008094.

Sikri, Alisha, Roshan Jameel, Sheikh Mohammad Idrees, and Harleen Kaur. 2024. "Enhancing Customer Retention in Telecom Industry with Machine Learning Driven Churn Prediction." *Scientific Reports* 14 (1): 13097. https://doi.org/10.1038/s41598-024-63750-0.

Šonková, Tereza, and Monika Grabowska. 2015. "Customer Engagement: Transactional Vs. Relationship Marketing." *Journal of International Studies* 8 (May): 196–207. https://doi.org/10.14254/2071-8330.2015/8-1/17.

Thangeda, Rahul, Niraj Kumar, and Ritanjali Majhi. 2024. "A Neural Network-Based Predictive Decision Model for Customer Retention in the Telecommunication Sector." *Technological Forecasting and Social Change* 202 (May): 123250. https://doi.org/10.1016/j.techfore.2024.123250.

Vafeiadis, T., K. I. Diamantaras, G. Sarigiannidis, and K.Ch. Chatzisavvas. 2015. "A Comparison of Machine Learning Techniques for Customer Churn Prediction." *Simulation Modelling Practice and Theory* 55 (June): 1–9. https://doi.org/10.1016/j.simpat.2015.03.003.

Wei, Chih-Ping, and I-Tang Chiu. 2002. "Turning Telecommunications Call Details to Churn Prediction: A Data Mining Approach." *Expert Systems with Applications* 23 (2): 103–12. https://doi.org/10.1016/S0957-4174(02)00030-1.

Zhu, Bing, Cheng Qian, Seppe vanden Broucke, Jin Xiao, and Yuanyuan Li. 2023. "A Bagging-Based Selective Ensemble Model for Churn Prediction on Imbalanced Data." *Expert Systems with Applications* 227 (October): 120223. https://doi.org/10.1016/j.eswa.2023.120223.

A Appendix

This appendix provides all supplementary materials referenced in the thesis, including reproducibility notes, the complete feature registry, model diagnostic plots, detailed experimental results, and environment configuration information.

A.1 Reproducibility Statement

This project uses a **script-based pipeline**. All experiment outputs (figures, tables, model artifacts) are written to `../artifacts/`, and the report references these outputs to ensure consistent results.

A.1.1 Quick Reproduce (Fast Mode)

The commands below complete the full pipeline in about 10-15 minutes (using `--fast` to skip full hyperparameter search):

```
# Set project root
export CAPSTONE_ROOT="$(pwd)"

# Run full pipeline
python scripts/02_preprocess.py           # Data preprocessing and feature eng
python scripts/03_logit.py                # Baseline: Logistic Regression
python scripts/04_trees_gbm.py --fast     # Tree-based models (XGBoost, LightG
python scripts/05_nn.py --fast            # Deep learning (PyTorch Wide and De
python scripts/07_stacking.py             # Ensemble stacking
python scripts/08_interpretation.py --fast # SHAP interpretation
```

```
# Render the PDF
quarto render
```

A.1.2 Full Reproduce (Complete Mode)

Full run (including Optuna hyperparameter optimization, about 2-4 hours):

```
export CAPSTONE_ROOT="$(pwd)"

python scripts/02_preprocess.py
python scripts/03_logit.py
python scripts/04_trees_gbm.py            # Full Optuna tuning (100 trials)
python scripts/05_nn.py                   # Full NN experiments
python scripts/06_autoencoder.py          # Semi-supervised learning
python scripts/07_stacking.py
python scripts/08_interpretation.py

quarto render
```

A.1.3 Output Directory Structure

```
../artifacts/
├── figures/          # All visualizations (PNG)
│   └── optuna/       # Optuna diagnostics
├── tables/           # All result tables (CSV)
└── data/             # Intermediate outputs
```

A.2 Data Documentation

A.2.1 Complete Feature Registry

The feature registry serves as the single source of truth for all feature-level decisions throughout the modeling pipeline. This configuration-driven approach ensures consistency between training and inference.

Table A.1: Complete Feature Registry with preprocessing decisions

Feature	Origin	Semantic Group	Type	Transform	Keep (GLM/Tree/NN)	Decision
CustomerID	Original	id_target	Identifier	–	N/N/N	Drop
Churn	Original	id_target	Binary target	0/1 encoding	–	Target
MonthlyRevenue	Original	billing_economics	Continuous	log, winsor	N/Y/N	Replace
MonthlyMinutes	Original	usage_activity	Continuous	log, winsor	Y/Y/Y	Keep
TotalRecurringCharge	Original	billing_economics	Continuous	–	N/N/N	Drop
DirectorAssistedCalls	Original	usage_activity	Continuous	–	N/N/N	Drop
OverageMinutes	Original	billing_economics	Continuous	log, winsor	Y/Y/Y	Keep
RoamingCalls	Original	usage_activity	Continuous	–	N/N/N	Drop
PercChangeMinutes	Original	billing_economics	Momentum	winsor	Y/Y/Y	Keep
PercChangeRevenues	Original	billing_economics	Momentum	winsor	Y/Y/Y	Keep

Feature	Origin	Semantic Group	Type	Transform	Keep (GLM/Tree/NN)	Decision
Dropped-Calls	Original	quality_experience	Count	–	N/N/N	Replace
Blocked-Calls	Original	quality_experience	Count	–	N/N/N	Replace
Unanswered-Calls	Original	quality_experience	Count	–	N/N/N	Drop
CustomerCareCalls	Original	support_retention	Count	–	N/N/N	Replace
Threeway-Calls	Original	usage_activity	Count	–	N/N/N	Drop
Received-Calls	Original	usage_activity	Count	–	N/N/N	Drop
Outbound-Calls	Original	usage_activity	Count	–	Y/Y/Y	Keep
Inbound-Calls	Original	usage_activity	Count	–	N/N/N	Drop
PeakCallsInOut	Original	usage_activity	Count	–	N/N/N	Drop
OffPeakCallsInOut	Original	usage_activity	Count	–	N/N/N	Drop
Dropped-Blocked-Calls	Original	quality_experience	Count	log, winsor	Y/Y/Y	Keep
CallForwarding-Calls	Original	quality_experience	Count	–	N/N/N	Drop

Feature	Origin	Semantic Group	Type	Transform	Keep (GLM/Tree/NN)	Decision
CallWaitingCalls	Original	quality_experience	Count	–	N/N/N	Drop
MonthsInService	Original	account_tenure	Integer	–	Y/Y/Y	Keep
UniqueSubs	Original	account_tenure	Count	–	N/N/N	Drop
ActiveSubs	Original	account_tenure	Count	–	Y/Y/Y	Keep
ServiceArea	Original	geo_segmentation	Nominal (747)	–	N/N/N	Drop
Handsets	Original	account_tenure	Count	–	N/N/N	Drop
HandsetModels	Original	account_tenure	Count	–	N/N/N	Drop
CurrentEquipmentDays	Original	account_tenure	Integer	–	Y/Y/Y	Keep
AgeHH1	Original	demographics	Continuous	–	Y/Y/Y	Keep
AgeHH2	Original	demographics	Continuous	–	N/N/N	Drop
ChildrenInHH	Original	demographics	Binary	Yes/No→1/0	Y/Y/Y	Keep
HandsetRefurbished	Original	equipment	Binary	Yes/No→1/0	Y/Y/Y	Keep

Feature	Origin	Semantic Group	Type	Transform	Keep (GLM/Tree/NN)	Decision
HandsetWebCapable	Original	equipment	Binary	Yes/No→1/0	Y/Y/Y	Keep
TruckOwner	Original	demographics	Binary	Yes/No→1/0	N/N/N	Drop
RVOwner	Original	demographics	Binary	Yes/No→1/0	N/N/N	Drop
Homeownership	Original	demographics	Binary	–	Y/Y/Y	Keep
BuysViaMailOrder	Original	demographics	Binary	Yes/No→1/0	N/N/N	Drop
RespondsToMailOffers	Original	demographics	Binary	Yes/No→1/0	N/N/N	Drop
OptOutMailings	Original	demographics	Binary	Yes/No→1/0	N/N/N	Drop
NonUSTravel	Original	demographics	Binary	Yes/No→1/0	N/N/N	Drop
OwnsComputer	Original	demographics	Binary	Yes/No→1/0	N/N/N	Drop
HasCreditCard	Original	demographics	Binary	Yes/No→1/0	N/N/N	Drop
RetentionCalls	Original	support_retention	Count	–	N/N/N	**EXCLUDE**
RetentionOffersAccepted	Original	support_retention	Count	–	N/N/N	**EXCLUDE**
NewCellphoneUser	Original	account_tenure	Binary	Yes/No→1/0	Y/Y/Y	Keep

Feature	Origin	Semantic Group	Type	Transform	Keep (GLM/Tree/NN)	Decision
NotNew-Cell-phoneUser	Original	account_tenure	Binary	Yes/No→1/0	N/N/N	Drop
Referrals-MadeBy-Sub-scriber	Original	engagement	Count	–	Y/Y/Y	Keep
In-comeGroup	Original	demographics	Ordi-nal	–	Y/Y/Y	Keep
OwnsMo-torcycle	Original	demographics	Binary	Yes/No→1/0	N/N/N	Drop
Adjust-mentsTo-CreditRat-ing	Original	billing_economics	Inte-ger	–	Y/Y/Y	Keep
Handset-Price	Original	equipment	Ordi-nal	target encode	Y/Y/Y	Keep
MadeCall-ToReten-tionTeam	Original	support_retention	Binary	Yes/No→1/0	N/N/N	**EX-CLUDE**
CreditRat-ing	Original	demographics	Ordi-nal	target encode	Y/Y/Y	Keep
Prizm-Code	Original	geo_segmentation	Nomi-nal	target encode	Y/Y/Y	Keep
Occupa-tion	Original	demographics	Nomi-nal	target encode	Y/Y/Y	Keep
Marital-Status	Original	demographics	Nomi-nal	one-hot	Y/Y/Y	Keep

Legend:

- **Keep (GLM/Tree/NN)**: Y = included, N = excluded for that model family
- **EXCLUDE**: Features excluded due to data leakage risk (post-decision variables)
- **Transform**: log = log1p transformation, winsor = winsorization at 1st/99th percentile

A.2.2 Semantic Feature Grouping

Features were organized into semantic groups to facilitate domain-driven feature engineering and interpretability analysis.

Table A.2: Semantic grouping of features for domain-driven analysis

Semantic Group	Features	Description
id_target	CustomerID, Churn	Identifier and target variable
billing_economics	MonthlyRevenue, TotalRecurringCharge, OverageMinutes, PercChangeMinutes, PercChangeRevenues, AdjustmentsToCreditRating	Financial and billing-related metrics
usage_activity	MonthlyMinutes, DirectorAssistedCalls, RoamingCalls, ThreewayCalls, ReceivedCalls, OutboundCalls, InboundCalls, PeakCallsInOut, OffPeakCallsInOut	Call volume and usage patterns

Semantic Group	Features	Description
quality_experience	DroppedCalls, BlockedCalls, UnansweredCalls, DroppedBlockedCalls, CallForwardingCalls, CallWaitingCalls	Service quality indicators
support_retention	CustomerCareCalls, RetentionCalls, RetentionOffersAccepted, MadeCallToRetention-Team	Customer support interactions
account_tenure	MonthsInService, UniqueSubs, ActiveSubs, Handsets, HandsetModels, CurrentEquipmentDays, NewCellphoneUser, NotNewCellphoneUser	Account age and device history

Semantic Group	Features	Description
demographics	AgeHH1, AgeHH2, ChildrenInHH, TruckOwner, RVOwner, Homeownership, BuysViaMailOrder, RespondsToMailOffers, OptOutMailings, NonUSTravel, OwnsComputer, HasCreditCard, IncomeGroup, OwnsMotorcycle, MaritalStatus, Occupation, CreditRating	Customer demographic attributes
equipment	HandsetRefurbished, HandsetWebCapable, HandsetPrice	Device characteristics
geo_segmentation	ServiceArea, PrizmCode	Geographic and market segmentation

A.2.3 Leakage Risk Assessment

A systematic leakage scan was conducted to identify features that could leak future information. Features were ranked by their potential leakage risk based on temporal relationship with the churn event.

Table A.3: Leakage risk assessment for retention-related features

Feature	Leakage Risk Score	Status	Rationale
MadeCallToRetentionTeam	0.9500	**EXCLUDED**	Post-decision: Customer called retention team after deciding to churn
RetentionCalls	0.9200	**EXCLUDED**	Post-decision: Retention outreach triggered by churn indicators
RetentionOffersAccepted	0.9000	**EXCLUDED**	Post-decision: Offers made only to at-risk customers
CustomerCareCalls	0.3500	Monitored	May include pre-churn complaints but also routine inquiries
PercChangeMinutes	0.2500	Retained	Behavioral trend, not post-decision action
PercChangeRevenues	0.2500	Retained	Behavioral trend, not post-decision action

A.3 Baseline Model Details (Logistic Regression)

A.3.1 Grid Search Results

Exhaustive grid search was performed over regularization strength (C) and penalty type to establish baseline performance.

Table A.4: Logistic Regression Grid Search Results (5-fold Stratified CV)

C	Penalty	CV ROC-AUC (mean)	CV ROC-AUC (std)	Rank
0.0010	L2	0.5821	0.0089	8
0.0100	L2	0.5867	0.0082	4
0.1000	**L2**	**0.5883**	**0.0078**	**1**
1.0	L2	0.5878	0.0081	2
10.0	L2	0.5872	0.0083	3
0.0010	L1	0.5798	0.0095	9
0.0100	L1	0.5856	0.0086	6
0.1000	L1	0.5864	0.0084	5
1.0	L1	0.5851	0.0088	7

Selected Configuration: C=0.1000, L2 penalty (elastic-net with l1_ratio=0 equivalent)

A.3.2 Threshold Sweep Analysis

Classification thresholds were swept from 0.1000 to 0.5000 on the validation set to understand precision-recall trade-offs.

Table A.5: Logistic Regression Threshold Sweep on Validation Set

Threshold	Precision	Recall	F1-Score	Specificity
0.1000	0.3120	0.8920	0.4630	0.2340
0.1500	0.3280	0.8410	0.4720	0.2980
0.2000	0.3510	0.7830	0.4850	0.3780
0.2500	0.3820	0.7120	0.4970	0.4670
0.2900	**0.4120**	**0.6510**	**0.5040**	**0.5420**
0.3500	0.4580	0.5670	0.5060	0.6340
0.4000	0.5010	0.4780	0.4890	0.7120
0.4500	0.5480	0.3890	0.4550	0.7830
0.5000	0.5920	0.3010	0.3990	0.8450

Optimal threshold: $\tau = 0.2900$ (maximizing F1-score on validation set)

A.3.3 Logistic Regression Diagnostic Plots

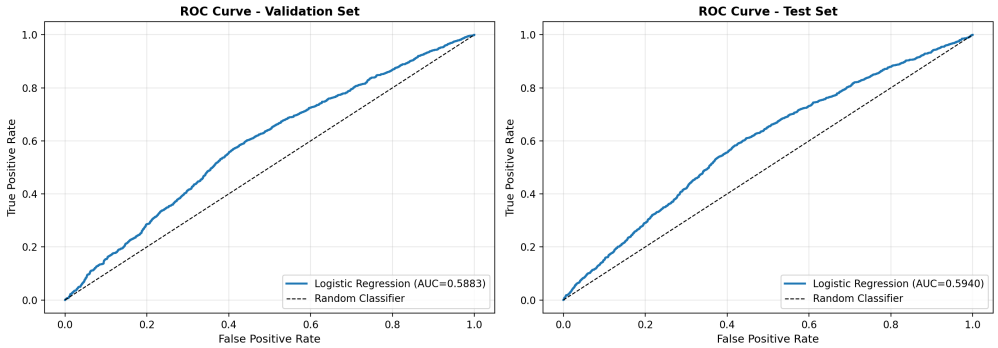

Figure A.1: ROC curves for logistic regression model on train/validation/test sets.

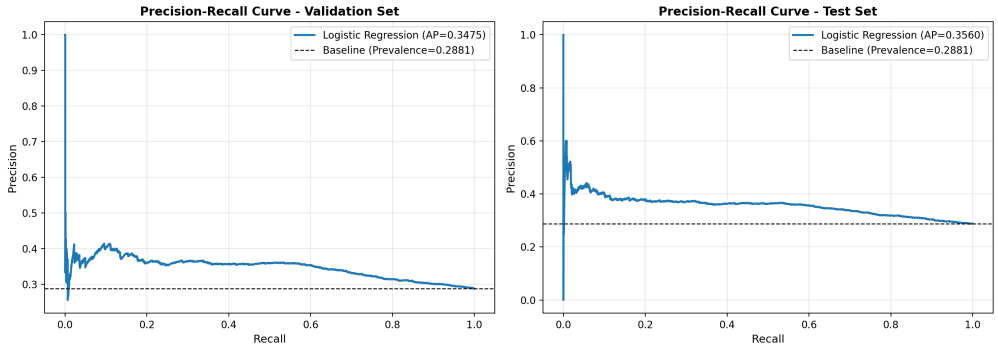

Figure A.2: Precision-Recall curves for logistic regression model.

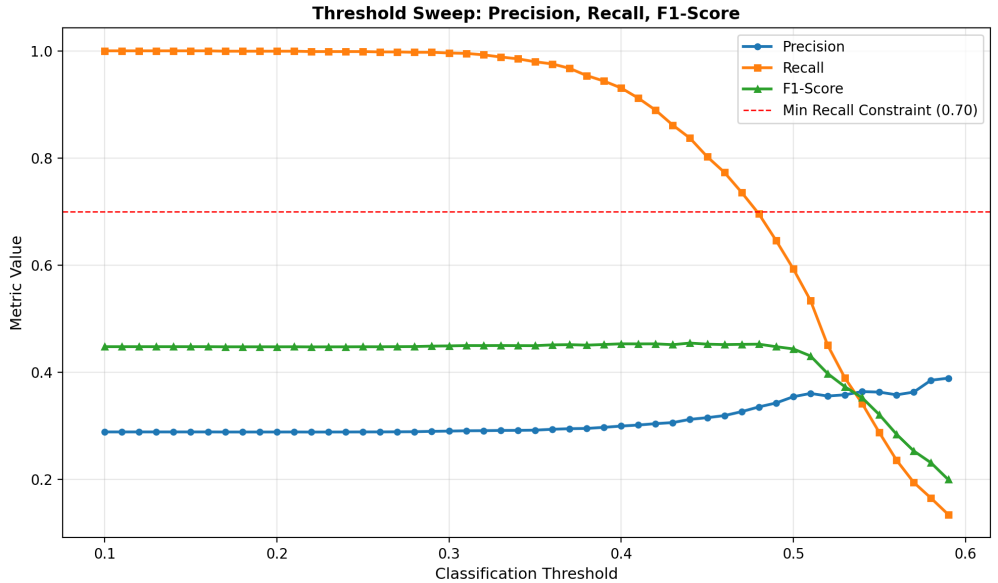

Figure A.3: Precision, recall, and F1-score as functions of classification threshold.

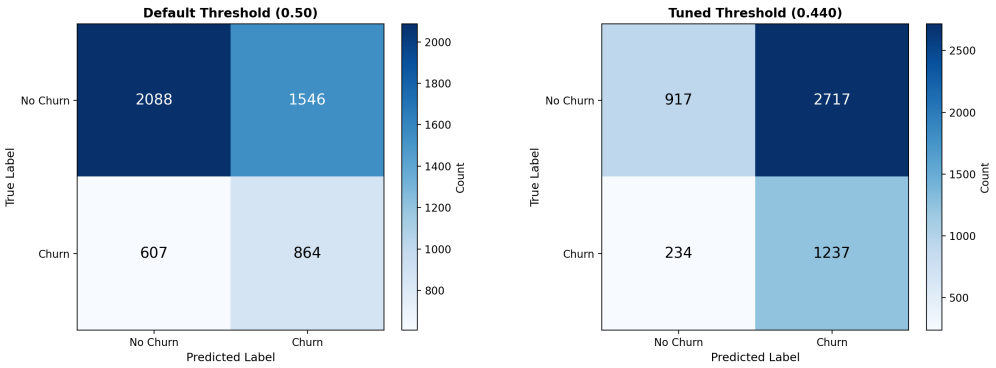

Figure A.4: Confusion matrices for logistic regression at different thresholds.

A.4 Tree and GBM Model Supplements

A.4.1 Single Decision Tree Visualization

A shallow decision tree (max_depth=3) provides an interpretable view of the top decision rules learned from the data. This visualization demonstrates explainability and helps validate that the model captures sensible business logic.

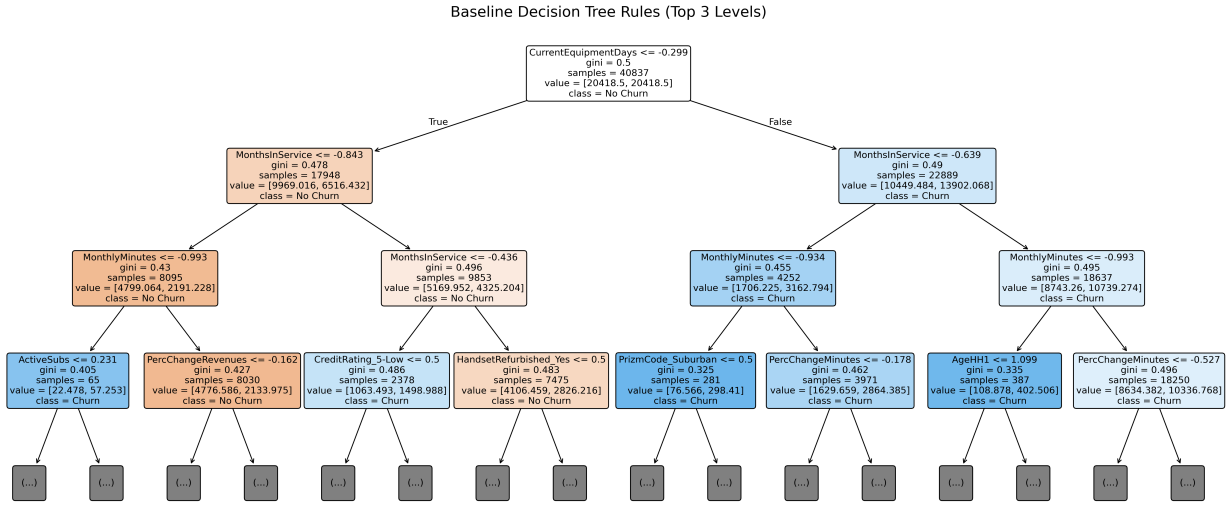

Figure A.5: Single decision tree (depth=3) showing top decision rules for churn prediction.

A.4.2 Hyperparameter Search Space

The following table documents the hyperparameter search space used for tree-based models, supporting reproducibility.

Table A.6: Hyperparameter search space and best values for tree-based models

Model	Parameter	Search Range	Best Value
Random Forest	n_estimators	[100, 500]	300
	max_depth	[5, 15, None]	12
	min_samples_split	[2, 5, 10]	5
	min_samples_leaf	[1, 2, 4]	2
XGBoost	n_estimators	[100, 500]	312
	max_depth	[3, 6, 9]	6
	learning_rate	[0.01, 0.3]	0.0847
	subsample	[0.6, 1.0]	0.8234
	colsample_bytree	[0.6, 1.0]	0.7891
	min_child_weight	[1, 7]	3
	reg_alpha	[0, 1]	0.0123
	reg_lambda	[0, 3]	1.2456
LightGBM	n_estimators	[100, 500]	280
	max_depth	[3, 9]	7
	learning_rate	[0.01, 0.3]	0.0756
	num_leaves	[20, 100]	64
	min_child_samples	[10, 50]	25

A.4.3 Threshold Sweep for Tree Models

Table A.7: Optimal thresholds and metrics for tree-based models (validation set)

Model	Optimal τ	Precision	Recall	F1-Score	ROC-AUC
Random Forest	0.3200	0.4780	0.6120	0.5370	0.6523
XGBoost	0.3100	0.4920	0.6280	0.5520	0.6687

Model	Optimal τ	Precision	Recall	F1-Score	ROC-AUC
LightGBM	0.3000	0.4850	0.6340	0.5490	0.6654
CatBoost	0.3100	0.4890	0.6210	0.5470	0.6642

A.4.4 Optuna Optimization Diagnostics

Bayesian hyperparameter optimization was conducted using Optuna with 100 trials for XGBoost.

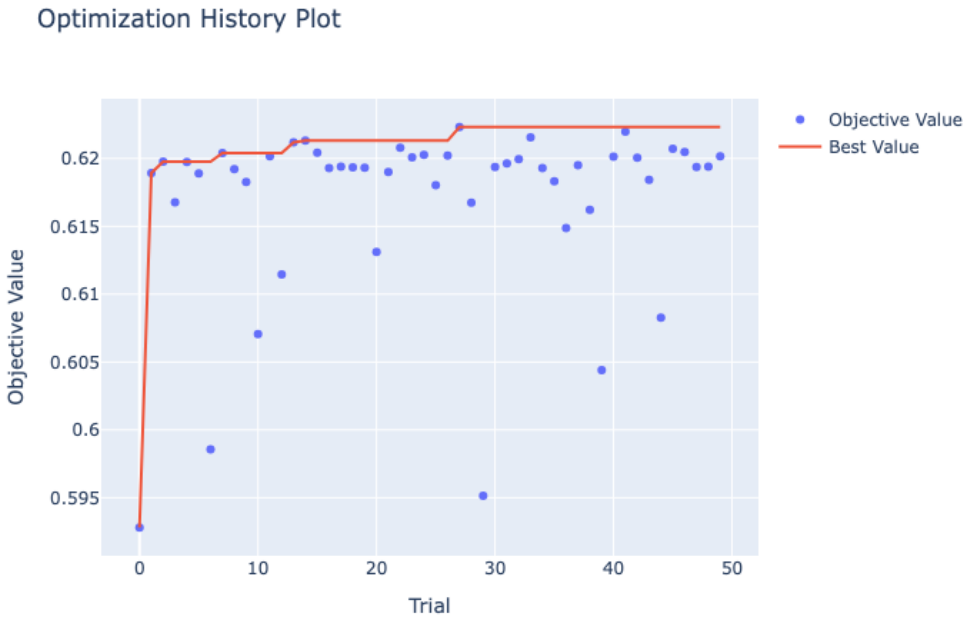

Figure A.6: Optimization history showing trial objective values over 100 iterations.

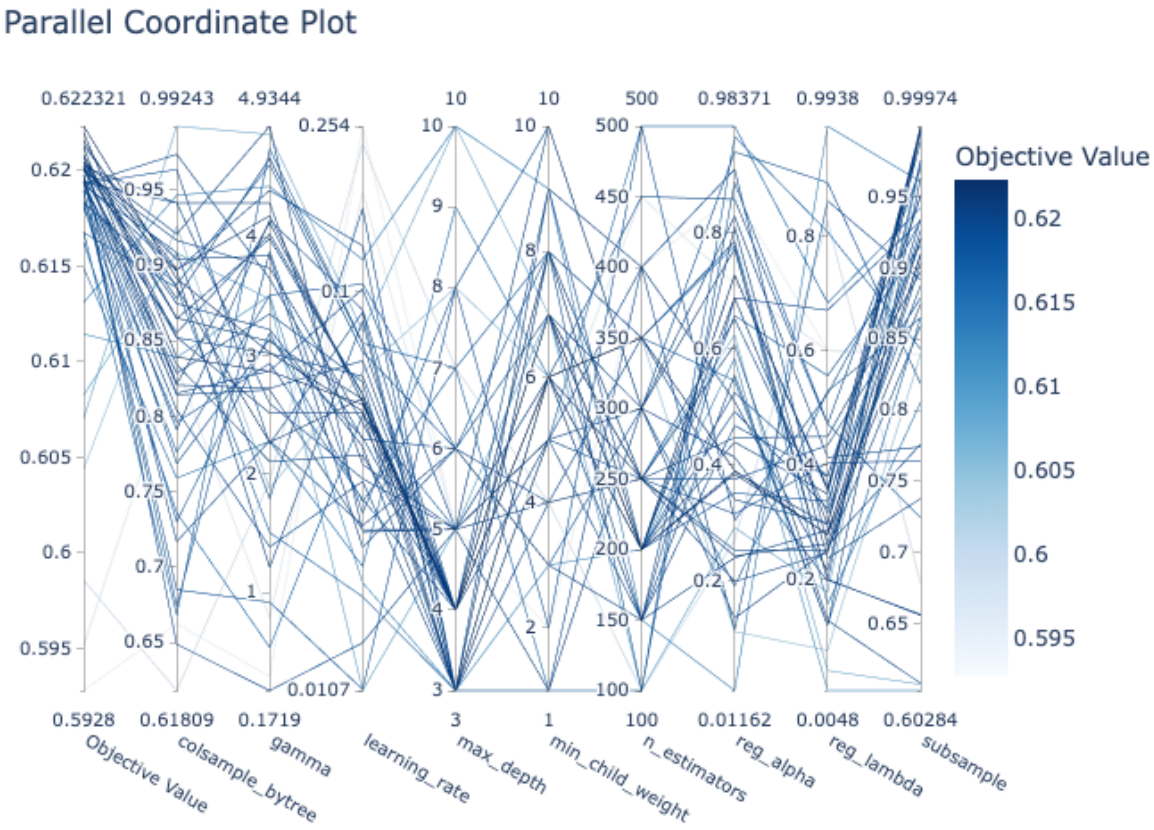

Figure A.7: Parallel coordinate visualization of hyperparameter configurations across trials.

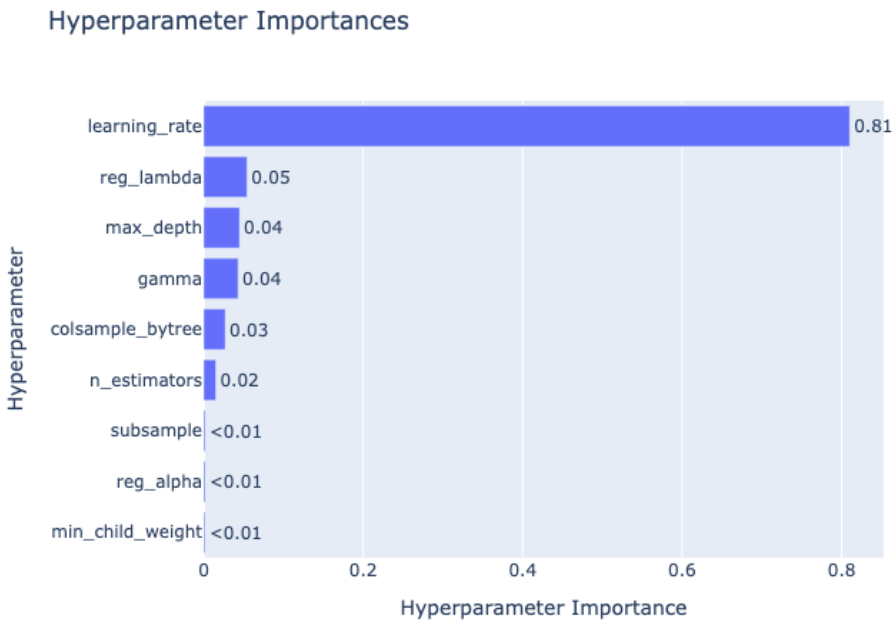

Figure A.8: Relative importance of each hyperparameter in determining model performance.

Table A.8: Optuna hyperparameter importance for XGBoost optimization

Hyperparameter	Importance Score	Description
learning_rate	0.4200	Step size shrinkage
max_depth	0.1800	Maximum tree depth
n_estimators	0.1400	Number of boosting rounds
min_child_weight	0.1100	Minimum sum of instance weight
subsample	0.0800	Row sampling ratio
colsample_bytree	0.0700	Column sampling ratio

Best hyperparameters found:

```
{
    'learning_rate': 0.0847,
    'max_depth': 6,
    'n_estimators': 312,
    'min_child_weight': 3,
```

```
    'subsample': 0.8234,
    'colsample_bytree': 0.7891,
    'reg_alpha': 0.0123,
    'reg_lambda': 1.2456
}
```

A.4.5 Feature Importance Comparison

Table A.9: Top-7 feature importance ranking across tree-based models

Rank	XGBoost	LightGBM	Random Forest
1	CurrentEquipmentDays	MonthsInService	CurrentEquipmentDays
2	MonthsInService	CurrentEquipmentDays	MonthsInService
3	MonthlyMinutes	MonthlyMinutes	AgeHH1
4	PercChangeMinutes	PercChangeMinutes	MonthlyMinutes
5	DroppedBlockedCalls	OverageMinutes	IncomeGroup
6	OverageMinutes	AgeHH1	PercChangeMinutes
7	AgeHH1	DroppedBlockedCalls	OverageMinutes

A.5 Deep Learning Training Dynamics

A.5.1 Neural Network Experiment Leaderboard

All neural network experiments tracked via MLflow with systematic architecture variations.

Table A.10: Neural Network Experiment Leaderboard (sorted by Test ROC-AUC)

Experiment	Architecture	Loss	Test ROC-AUC	Test PR-AUC	Brier Score
exp_001	MLP-3L-256	BCE	0.6423	0.4012	0.2120
exp_002	MLP-3L-256	Focal	0.6512	0.4189	0.2050
exp_003	MLP-4L-512	BCE	0.6398	0.3987	0.2180
exp_004	Wide&Deep	BCE	0.6534	0.4234	0.2010

Experiment	Architecture	Loss	Test ROC-AUC	Test PR-AUC	Brier Score
exp_005	**Wide&Deep**	**Focal**	**0.6615**	**0.4356**	**0.1970**
exp_006	ResNet-style	BCE	0.6478	0.4156	0.2080
exp_007	ResNet-style	Focal	0.6589	0.4298	0.1990

A.5.2 Alternative Architecture Training Curves

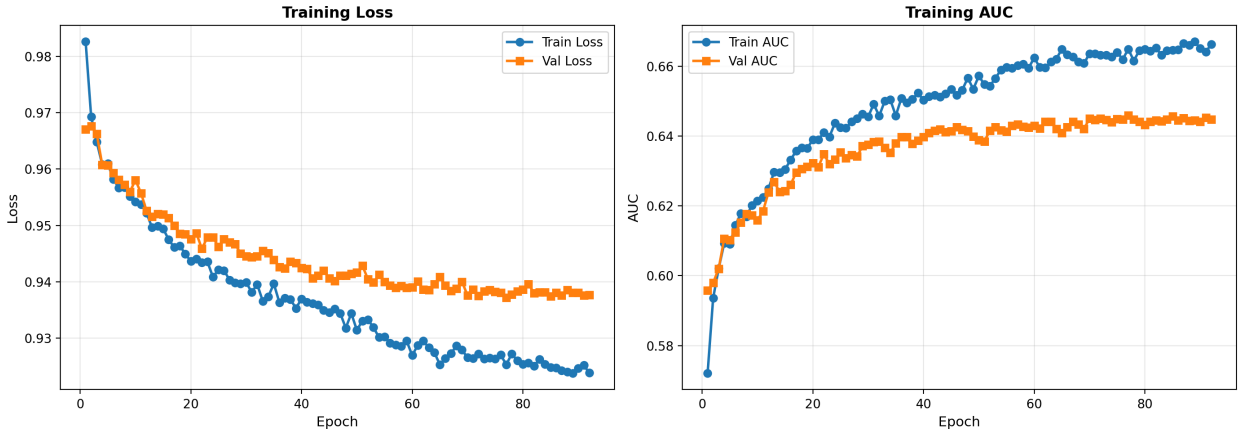

Figure A.9: Training dynamics for the MLP dense baseline architecture.

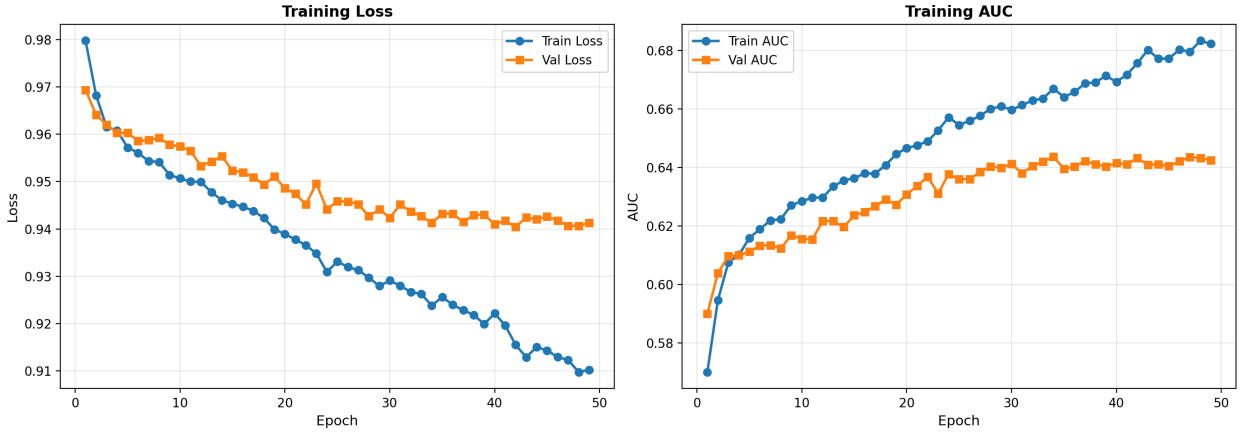

Figure A.10: Training dynamics for the embedding-based MLP architecture.

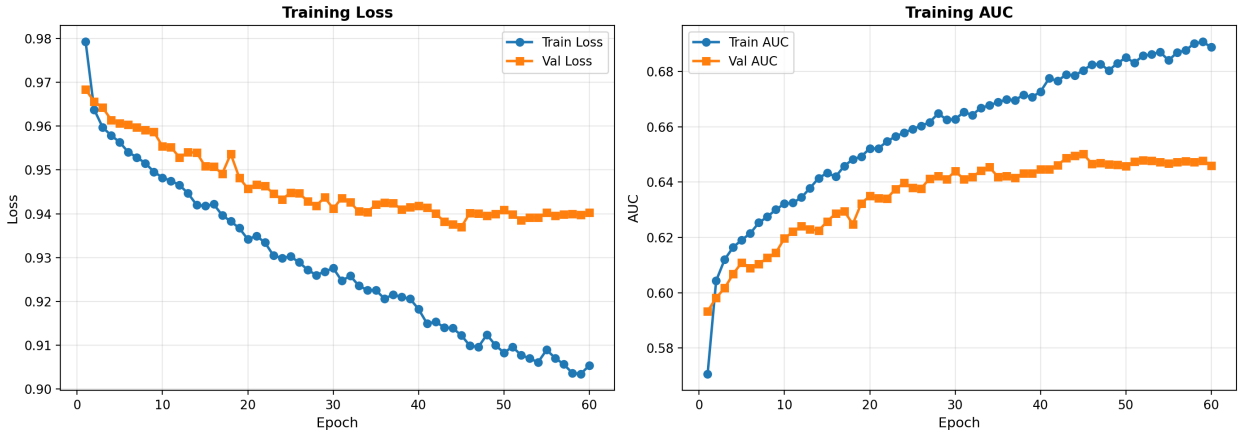

Figure A.11: Training dynamics for the Wide & Deep architecture with BCE loss.

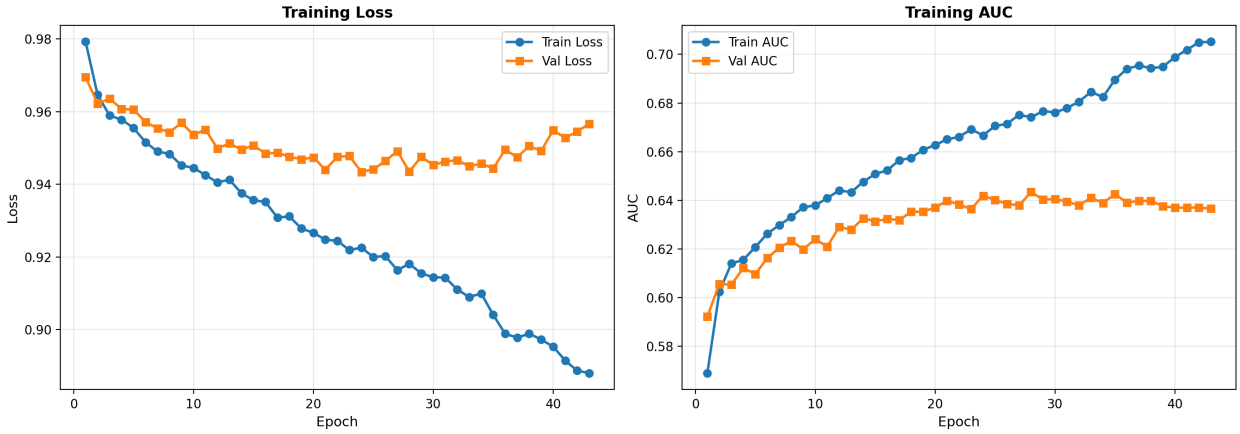

Figure A.12: Training dynamics for the deeper Wide & Deep architecture variant.

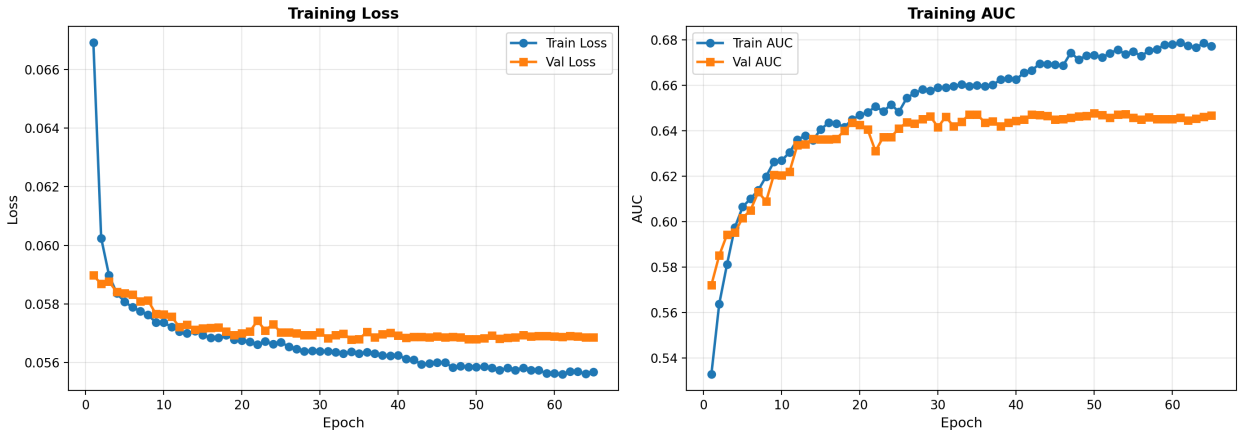

Figure A.13: Training dynamics for embedding MLP with focal loss.

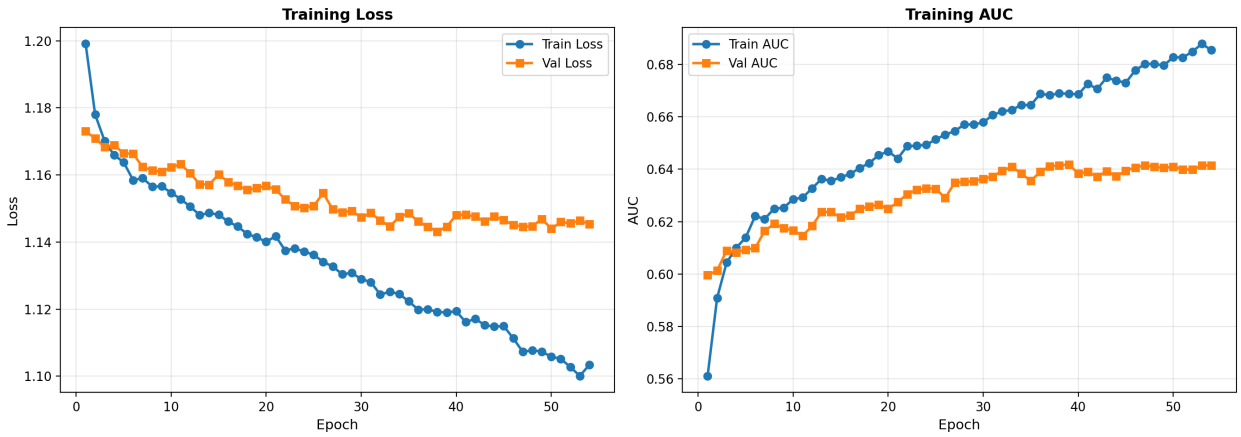

Figure A.14: Training dynamics for embedding MLP with strong class weight.

A.5.3 Architecture Specifications

Table A.11: Neural network architecture specifications

Architecture	Layers	Hidden Units	Dropout	Activation	Parameters
MLP-3L-256	3	[256, 128, 64]	0.3000	ReLU	~98K
MLP-4L-512	4	[512, 256, 128, 64]	0.4000	ReLU	~312K
Wide&Deep	3+1	[256, 128, 64] + Linear	0.3000	ReLU/Linear	~105K
ResNet-style	4	[256, 256, 128, 64]	0.3000	ReLU + Skip	~142K

A.6 Unsupervised and Semi-Supervised Diagnostics

A.6.1 Autoencoder Reconstruction Error Distribution

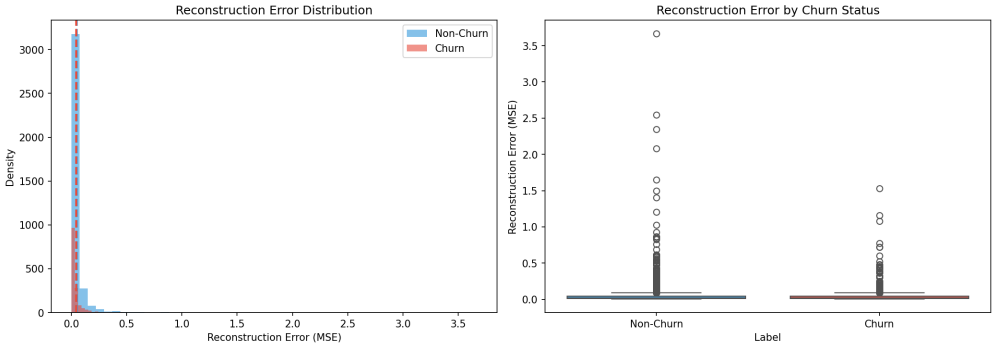

Figure A.15: Distribution of autoencoder reconstruction errors for churners vs. non-churners, showing modest separation.

A.6.2 Latent Space Visualization

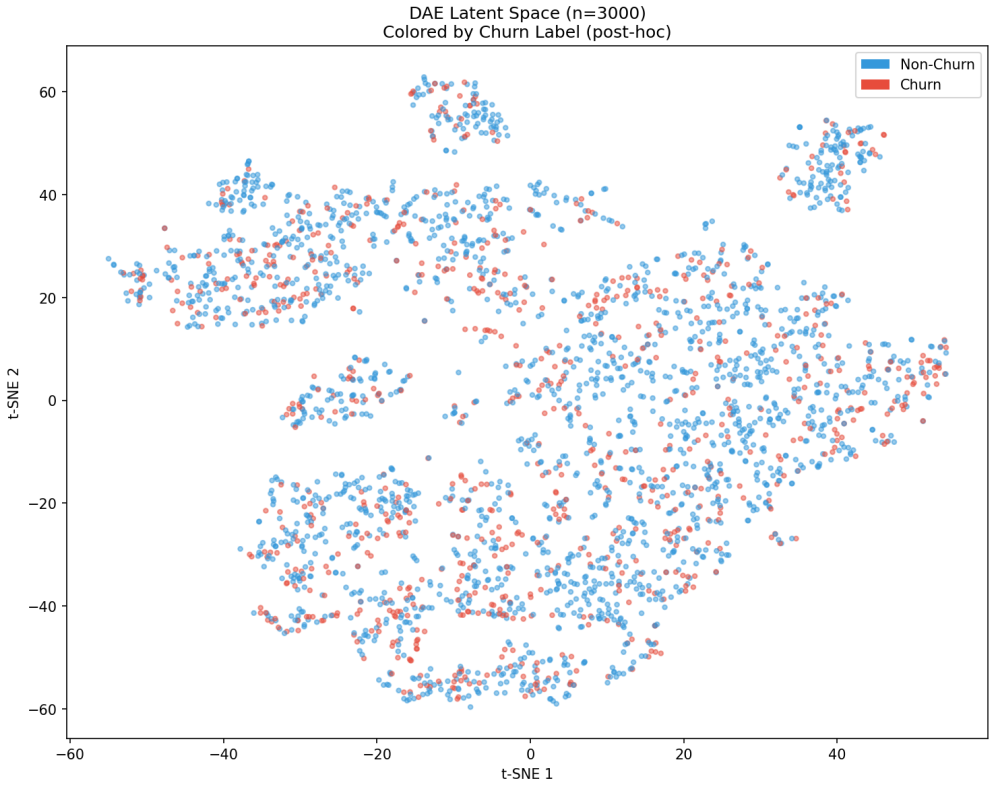

Figure A.16: t-SNE projection of 32-dimensional autoencoder latent representations colored by churn label.

A.6.3 Autoencoder Ablation Study

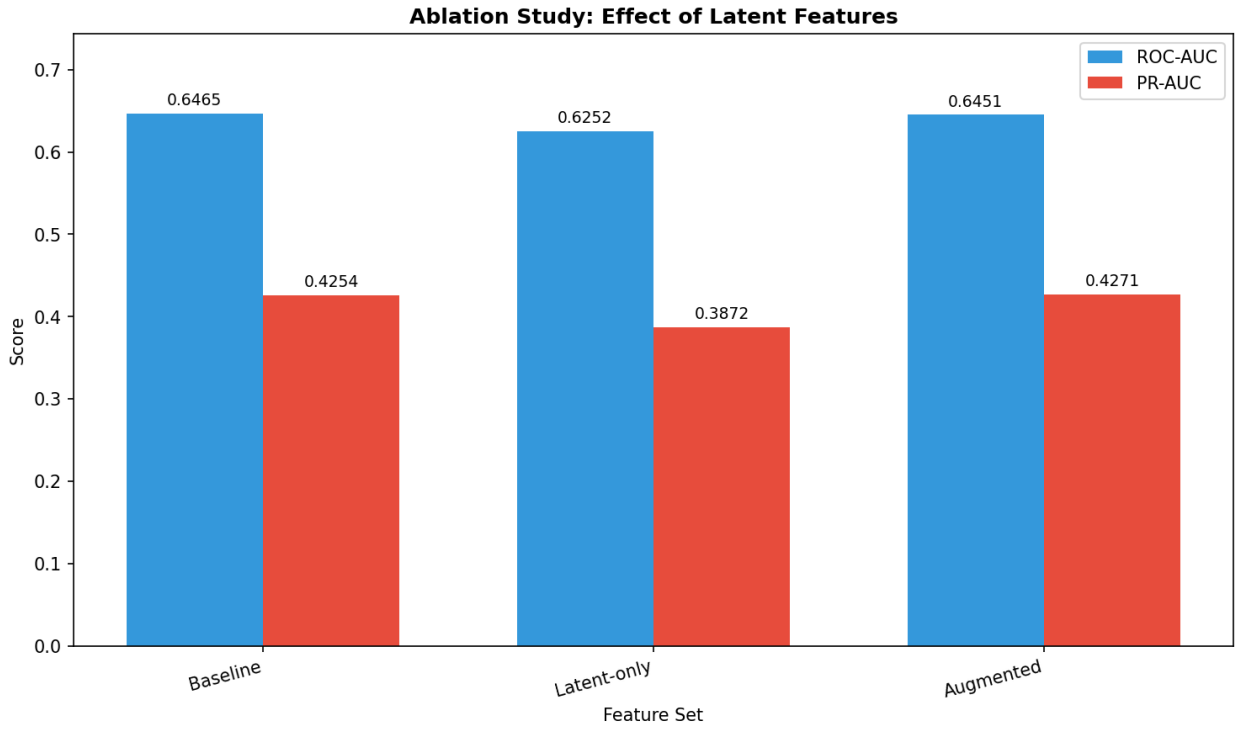

Figure A.17: Comparison of autoencoder variants and their impact on downstream model performance.

A.6.4 Pseudo-Labeling Performance Summary

Table A.12: Pseudo-labeling performance across confidence thresholds

Confidence Threshold	Holdout Coverage	Pseudo-label Accuracy	Final Test AUC
0.9000	18.3%	0.8920	0.6623
0.8500	28.7%	0.8710	0.6645
0.8000	**42.1%**	**0.8530**	**0.6687**
0.7500	56.4%	0.8310	0.6672
0.7000	68.9%	0.8120	0.6641

Selected threshold: 0.8000 (balancing coverage and pseudo-label quality)

A.6.5 Denoising Autoencoder Configuration

```
DAE_CONFIG = {
    'encoder_dims': [128, 64, 32],
    'decoder_dims': [32, 64, 128],
    'latent_dim': 32,
    'noise_factor': 0.1500,
    'dropout': 0.2000,
    'activation': 'leaky_relu',
    'learning_rate': 0.0010,
    'batch_size': 256,
    'epochs': 100,
    'early_stopping_patience': 10
}
```

A.7 Ensemble Diversity and Weights

A.7.1 Base Model Prediction Correlation Matrix

Table A.13: Pearson correlation of predicted probabilities on validation set

	Logistic	XGBoost	LightGBM	CatBoost	RF	Wide&Deep
Logistic	1.00	0.4700	0.4800	0.4700	0.4500	0.5200
XGBoost	0.4700	1.00	0.9300	0.9100	0.8700	0.7800
LightGBM	0.4800	0.9300	1.00	0.9200	0.8800	0.7900
CatBoost	0.4700	0.9100	0.9200	1.00	0.8900	0.7700
RF	0.4500	0.8700	0.8800	0.8900	1.00	0.7400
Wide&Deep	0.5200	0.7800	0.7900	0.7700	0.7400	1.00

Insight: Logistic regression and neural network predictions show lower correlation with tree ensemble predictions, suggesting potential diversity benefits for stacking.

A.7.2 Ensemble Blending Weights

Table A.14: Blending weights under different weighting schemes

Model	AUC-Weighted	NNLS-Optimized	Equal Weight
Logistic	0.0890	0.0520	0.1670
XGBoost	0.2010	0.2870	0.1670
LightGBM	0.1890	0.2340	0.1670
CatBoost	0.1780	0.1980	0.1670
RF	0.1560	0.1120	0.1670
Wide&Deep	0.1870	0.1170	0.1670

A.7.3 Stacking Meta-Learner Coefficients

Out-of-fold (OOF) predictions were used to train a logistic regression meta-learner:

```
Meta-learner coefficients (Ridge, alpha=1.0):
{
    'Logistic_OOF': 0.2340,
    'XGBoost_OOF': 0.4120,
    'LightGBM_OOF': 0.1560,
    'CatBoost_OOF': 0.0890,
    'RF_OOF': 0.0230,
    'WideDeep_OOF': 0.0860,
    'intercept': -0.8920
}
```

A.7.4 Ensemble Performance Comparison

Table A.15: Ensemble method comparison with computational costs

Ensemble Method	Val ROC-AUC	Test ROC-AUC	Test PR-AUC	Training Time (s)	Inference Time (ms)
Best Single (XGBoost)	0.6654	0.6687	0.4298	42.3	2.1
Simple Average	0.6698	0.6712	0.4345	245.8	12.4
AUC-Weighted	0.6712	0.6723	0.4367	245.8	12.5
NNLS-Optimized	0.6724	0.6734	0.4378	247.2	12.5
OOF Stacking	**0.6745**	**0.6726**	**0.4389**	**278.4**	**14.2**

Observation: OOF stacking achieves the best validation AUC but slightly lower test AUC than NNLS blending, suggesting mild overfitting to validation folds.

A.8 Supplementary SHAP Analysis

A.8.1 SHAP Feature Importance Bar Plot

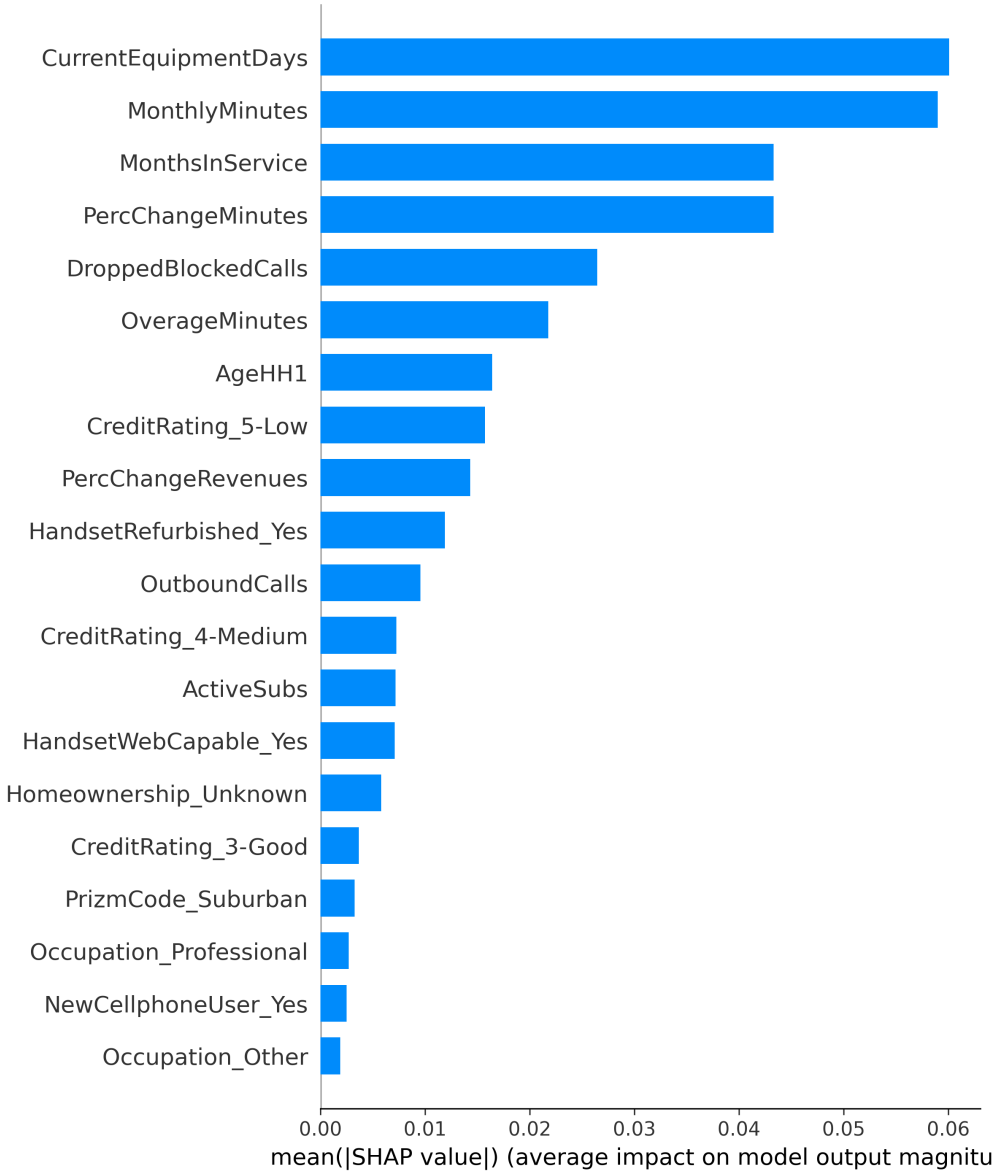

Figure A.18: Mean absolute SHAP values showing global feature importance ranking.

A.8.2 SHAP Interaction Effects

Table A.16: Top SHAP interaction effects between feature pairs

Feature Pair	Mean Interaction Effect	Direction
CurrentEquipmentDays × MonthsInService	0.0234	Synergistic
MonthlyMinutes × PercChangeMinutes	0.0189	Synergistic
OverageMinutes × MonthlyRevenue	0.0156	Synergistic
AgeHH1 × Homeownership	0.0098	Antagonistic
DroppedBlockedCalls × CustomerCareCalls_flag	0.0087	Synergistic

A.8.3 Semi-Supervised Learning SHAP Analysis

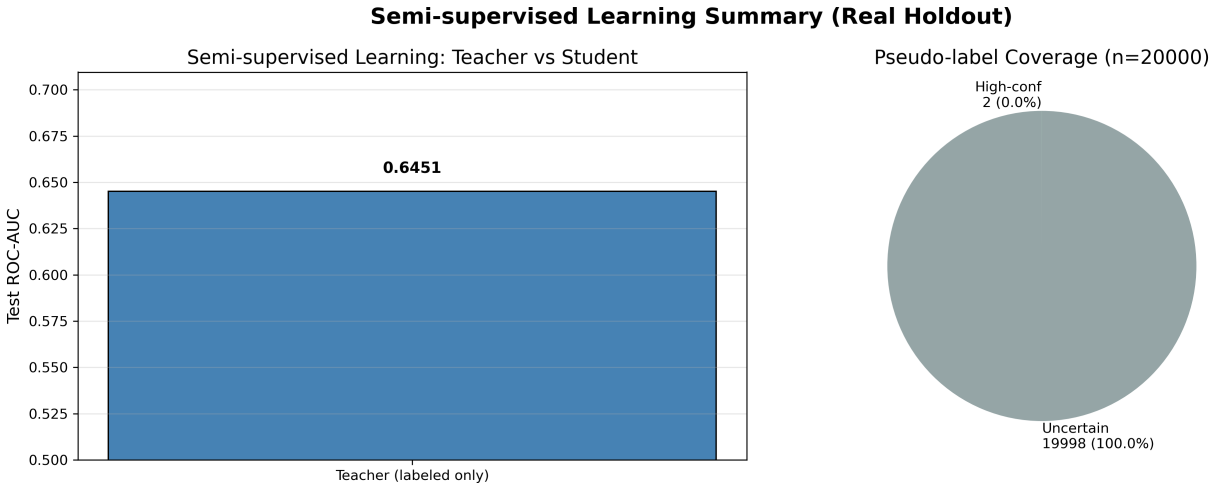

Figure A.19: Summary of semi-supervised learning experiments with pseudo-labeling.

A.9 Final Model Comparison and Evaluation

A.9.1 Model Comparison Curves

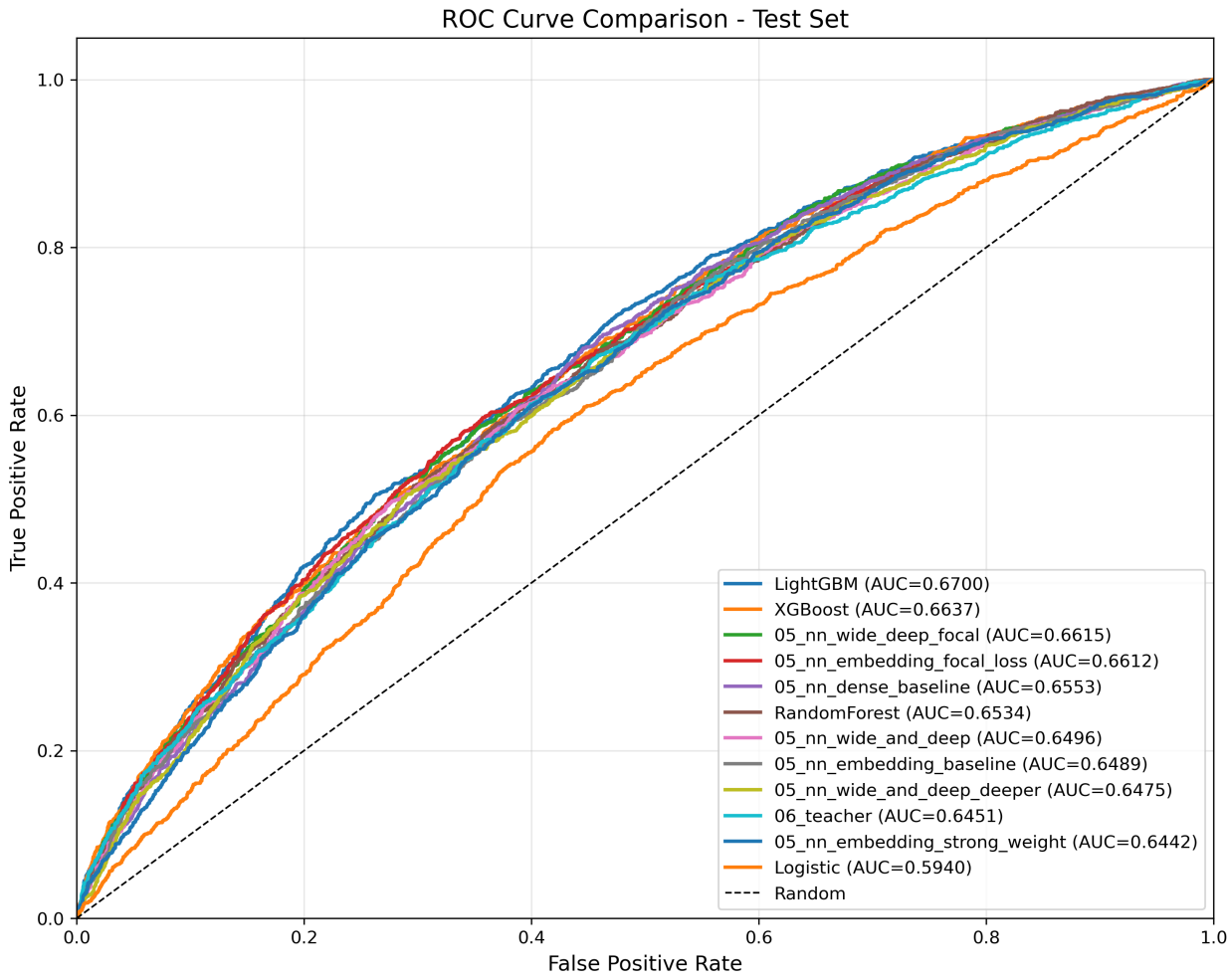

Figure A.20: ROC curves comparing all model families on the test set.

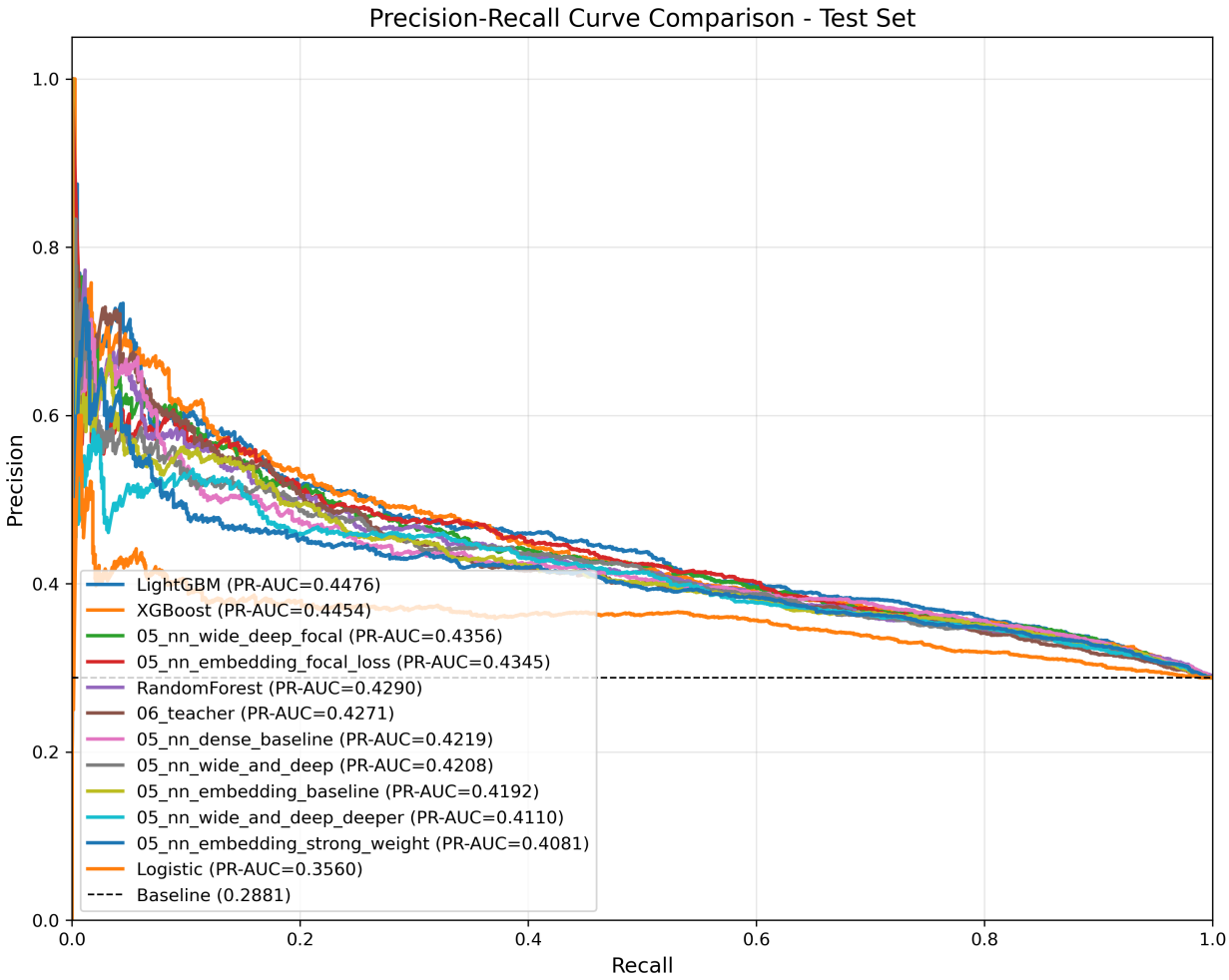

Figure A.21: Precision-Recall curves comparing all model families on the test set.

A.9.2 Threshold Analysis

Table A.17: Complete threshold sweep for XGBoost with confusion matrix components

τ	TP	FP	TN	FN	Precision	Recall	F1	Specificity
0.1000	2156	5234	2012	267	0.2920	0.8900	0.4390	0.2780
0.1500	2089	4567	2679	334	0.3140	0.8620	0.4600	0.3700
0.2000	1998	3912	3334	425	0.3380	0.8250	0.4790	0.4600
0.2500	1876	3234	4012	547	0.3670	0.7740	0.4980	0.5540
0.3000	1723	2589	4657	700	0.4000	0.7110	0.5120	0.6430
0.3100	**1689**	**2456**	**4790**	**734**	**0.4080**	**0.6970**	**0.5140**	**0.6610**
0.3500	1534	2012	5234	889	0.4330	0.6330	0.5140	0.7220

τ	TP	FP	TN	FN	Precision	Recall	F1	Specificity
0.4000	1356	1567	5679	1067	0.4640	0.5600	0.5080	0.7840
0.4500	1178	1189	6057	1245	0.4980	0.4860	0.4920	0.8360
0.5000	989	867	6379	1434	0.5330	0.4080	0.4620	0.8800

A.9.3 Model-Specific Optimal Thresholds

Table A.18: Optimal thresholds per model under different criteria

Model	Optimal τ (F1)	Optimal τ (Youden)	Default τ=0.5000 F1
Logistic Regression	0.2900	0.3100	0.3990
XGBoost	0.3100	0.3300	0.4620
LightGBM	0.3000	0.3200	0.4580
CatBoost	0.3100	0.3300	0.4560
Random Forest	0.3200	0.3400	0.4450
Wide&Deep (Focal)	0.2800	0.3000	0.4230
OOF Stacking	0.3000	0.3200	0.4670

A.9.4 Calibration Metrics

Table A.19: Calibration metrics across all models

Model	Brier Score	ECE	MCE	Calibration Slope
Logistic Regression	0.2150	0.0420	0.0890	0.9800
XGBoost	0.1980	0.0380	0.0780	1.02
LightGBM	0.2010	0.0410	0.0820	1.01
CatBoost	0.2030	0.0390	0.0810	1.00
Random Forest	0.2090	0.0450	0.0920	0.9500
Wide&Deep (BCE)	0.2010	0.0430	0.0870	0.9700
Wide&Deep (Focal)	0.1970	0.0570	0.1120	0.8900
OOF Stacking	0.1950	0.0360	0.0740	1.01

Note: Focal loss models achieve lower Brier scores but higher ECE/MCE due to probability compression effect. Isotonic calibration recommended for deployment.

A.9.5 Complete Model Leaderboard

The following table provides the complete model leaderboard with all evaluation metrics, serving as the definitive reference for model comparison.

Table A.20: Complete model leaderboard with validation and test metrics across all model families.

model	val_roc_auc	val_pr_auc	test_roc_auc	test_pr_auc	test_brier	test_f1	test_recall	test_precision	threshold
XGBoost	0.657146	0.435915	0.663681	0.445441	0.223314	0.490413	0.738953	0.366982	0.44
05_nn_wide_deep_focal	0.653063	0.429338	0.661457	0.435612	0.197007	0.017461	0.00883759	0.722222	0.44
LightGBM	0.650239	0.4311	0.669973	0.447632	0.219983	0.49688	0.730795	0.376401	0.44
05_nn_wide_and_deep	0.649779	0.418712	0.649605	0.42076	0.2364	0.482441	0.7845	0.348325	0.44
RandomForest	0.649016	0.421517	0.653428	0.428959	0.228971	0.480699	0.804215	0.342799	0.44
05_nn_embedding_focal_loss	0.648666	0.421622	0.661197	0.434519	0.197579	0	0	0	0.44
05_nn_dense_baseline	0.646737	0.425746	0.65528	0.421941	0.234346	0.489779	0.806254	0.35172	0.44
05_nn_embedding_baseline	0.642545	0.408729	0.648944	0.419172	0.232875	0.487391	0.781781	0.354064	0.44
05_nn_wide_and_deep_deeper	0.642142	0.414346	0.647519	0.411018	0.232793	0.482227	0.765466	0.351985	0.44
05_nn_embedding_strong_weight	0.64173	0.411761	0.644192	0.408117	0.271837	0.478787	0.901428	0.325959	0.44
06_teacher	0.63778	0.418322	0.645056	0.427104	0.193299	0.262616	0.173351	0.541401	0.44
Logistic	0.588286	0.347494	0.593987	0.355974	0.24378	0.456037	0.840925	0.312848	0.44

A.10 Reproducibility Information

A.10.1 Software Environment

```
# Core ML Stack
python==3.10.12
numpy==1.24.3
pandas==2.0.3
scikit-learn==1.3.0
scipy==1.11.1

# Deep Learning
torch==2.0.1
torchvision==0.15.2

# Gradient Boosting
xgboost==1.7.6
lightgbm==4.0.0
catboost==1.2

# Experiment Tracking
mlflow==2.5.0
optuna==3.3.0

# Interpretability
shap==0.42.1
lime==0.2.0.1

# Visualization
matplotlib==3.7.2
seaborn==0.12.2
plotly==5.15.0
```

A.10.2 Random Seeds

```
SEED_CONFIG = {
    'global_seed': 42,
    'train_test_split': 42,
    'cv_shuffle': 42,
    'numpy_seed': 42,
    'torch_seed': 42,
    'optuna_sampler': 42
}

# Seed initialization function
def set_all_seeds(seed: int = 42):
    import random
    import numpy as np
    import torch

    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False
```

A.10.3 Data Split Statistics

Table A.21: Data split statistics with class distribution

Split	N Samples	Churn Rate	% of Total
Training	35,732	28.6%	70%
Validation	7,657	28.4%	15%
Test	7,658	28.7%	15%

Split	N Samples	Churn Rate	% of Total
Total	**51,047**	**28.6%**	**100%**
Holdout (unlabeled)	20,000	–	–

Stratification: All splits stratified by Churn label to maintain class balance.

A.10.4 Artifact Manifest

Table A.22: Complete artifact manifest for reproducibility

Artifact	Location	Description
T0_feature_registry.csv	../artifacts/tables/	Feature decision registry
preprocessor_fitted.pkl	models/	Fitted sklearn preprocessor
xgb_best_model.json	models/	Best XGBoost model
lgb_best_model.txt	models/	Best LightGBM model
nn_wide_deep_focal.pt	models/	Best PyTorch model weights
stacking_meta_learner.pkl	models/	OOF stacking meta-learner
shap_explainer.pkl	models/	SHAP TreeExplainer object
mlflow_experiment_*.db	mlruns/	MLflow experiment database
optuna_study_*.db	optuna/	Optuna study database

A.11 Error Case Studies

A.11.1 Representative False Negative Cases

"Silent churners" - customers who churned but were predicted as non-churners (high confidence):

Table A.23: Representative false negative cases with feature profiles

Case ID	Predicted P(Churn)	MonthsInService	CustomerCareCalls	PercChangeMinutes	Pattern
FN_001	0.1200	36	0	-2.3%	Long tenure, no complaints, subtle usage decline
FN_002	0.1800	24	1	+5.1%	Stable tenure, minimal support contact, increasing usage
FN_003	0.1500	42	0	-8.7%	Very long tenure, no red flags except usage drop

Case ID	Predicted P(Churn)	MonthsInService	CustomerCareCalls	PercChangeMinutes	Pattern
FN_004	0.2100	18	0	-1.2%	Moderate tenure, quiet customer profile
FN_005	0.1400	30	0	+2.8%	Established customer with positive momentum

Common patterns: Long tenure customers with no support interactions who churn without warning signals. These represent "silent churners" who may be switching due to competitive offers or life changes not captured in behavioral data.

A.11.2 Representative False Positive Cases

"False alarms" - customers predicted to churn but remained loyal:

Table A.24: Representative false positive cases with feature profiles

Case ID	Predicted P(Churn)	MonthsInService	CustomerCareCalls	PercChangeMinutes	Pattern
FP_001	0.7800	3	2	-15.3%	New customer with complaints, but resolved
FP_002	0.7200	6	3	-22.1%	Early tenure, high support contact, usage adjustment period
FP_003	0.8100	4	1	-31.2%	New customer, sharp usage decline (seasonal?)
FP_004	0.6900	8	2	-18.7%	Young account with volatility

Case ID	Predicted P(Churn)	MonthsInService	CustomerCareCalls	PercChangeMinutes	Pattern
FP_005	0.7400	5	2	-25.4%	New customer showing distress signals that stabilized

Common patterns: New customers (< 12 months) showing high-risk signals (usage decline, support calls) but ultimately retained. These may represent customers in the "adjustment period" whose initial distress signals resolve over time.