Evaluating Machine Learning Techniques for Telecom Customer Churn Prediction

Author:

Dadakidis Giorgos

Supervisor:

Eftychios Protopapadakis

-June 2025-

This research was originally submitted as a thesis in Applied Informatics at the University of Macedonia

Abstract

Customer churn is a persistent and costly challenge for companies in the telecommunications sector, where maintaining existing subscribers is often more profitable than acquiring new ones. Accurately identifying customers who are likely to leave is critical for enabling targeted retention strategies. However, churn prediction is complicated by significant class imbalance, as the number of churners typically represents a small fraction of the overall customer base.

This thesis explores the application of machine learning techniques to the churn prediction problem using a structured experimental approach. Five experimental settings were designed to evaluate and improve model performance under imbalanced data conditions: a baseline scenario using the original dataset, a cost-sensitive learning setup with class weighting, a recall-optimized configuration through hyperparameter tuning, an experiment incorporating synthetic oversampling (SMOTE) and a final experiment using the top 20 important features . A variety of classification models were assessed, including both traditional machine learning algorithms and neural networks.

The study aims to investigate how different learning strategies and evaluation criteria affect model behavior and performance in the context of churn prediction. Emphasis is placed on addressing the imbalance issue, optimizing recall of the minority class, and comparing the effectiveness of algorithmic and data-driven solutions. The findings provide insights into the trade-offs and considerations involved in developing fair and practical predictive models for real-world customer churn scenarios.

Table of Contents

List of Figures:

List of Tables:

1. Introduction

In today’s competitive business landscape, retaining existing customers is crucial for any company. Acquiring a new customer is typically 5 to 10 times more costly than retaining an existing one. Given these facts, businesses are increasingly focusing on customer retention and preventing customer churn. As a result, preventing customer departure is crucial for reducing organizational costs, increasing market share, and improving the company's overall position (Shabankareh, Mohammad Javad, et al., 2022).

By leveraging customer data through machine learning algorithms, businesses can uncover hidden patterns, associations, and trends that serve as indicators of customer behavior prior to churn. Furthermore, machine learning enables the development of predictive models that utilize historical data to effectively forecast signs of customer departure (Lukita, Chandra, et al., 2023).

To mitigate churn, companies should actively seek customer feedback, conduct market research, and stay informed about industry trends. Building a competitive advantage requires identifying the unique strengths of their products or services and consistently communicating these to their customer base. Implementing effective customer retention strategies is essential for minimizing customer attrition and maintaining long-term business growth (Lukita, Chandra, et al., 2023).

This study aims to investigate the effectiveness of various machine learning techniques in predicting customer churn within the telecommunications sector. It focuses on the application of models such as Random Forest, Support Vector Machines (SVM), Naïve Bayes, neural networks, and K-Nearest Neighbors (KNN) to both identify the key factors contributing to customer attrition and evaluate the predictive performance of these classifiers in the binary task of distinguishing between customers who are likely to churn and those who are expected to remain. The analysis is conducted using the Cell2Cell telecommunications churn dataset, obtained from Kaggle, which consists of 51,047 instances and 58 features representing customer behavior, demographics, and service usage.

1.1 Problem Definition

In today's highly competitive environment, consumers are presented with an abundance of choices, and switching from one service or provider to another often incurs little to no cost (Keramati, Abbas, et al., 2014). In contrast, businesses expend significant resources not only to acquire new customers but also to retain existing ones (Shabankareh, Mohammad Javad, et al., 2022). Even after acquiring a new customer, considerable time and investment are required before that customer becomes profitable and demonstrates long-term loyalty (Keramati, Abbas, et al., 2014).

Customer churn prediction has emerged as a widely recognized challenge. However, most prior research has primarily relied on statistical approaches and regression analyses, leaving a significant gap in the application of machine learning methodologies (Lukita, Chandra, et al., 2023).

Moreover, existing customers tend to be more profitable, as they are generally less responsive to competitors' promotional activities (Shabankareh, Mohammad Javad, et al., 2022). To gain a sustainable competitive advantage, companies must understand the factors that drive customer attrition, the variables that influence customer lifetime value, and the strategies that enhance customer loyalty. Therefore, it is both logical and advantageous for organizations to invest in satisfying their current customers in order to improve retention and ensure long-term success (Shabankareh, Mohammad Javad, et al., 2022).

1.2 Purpose-Objectives

This research aims to explore machine learning techniques for predicting customer churn within a business context, with a particular focus on improving customer retention strategies. The primary objective is to assist business decision-makers in reducing customer attrition, thereby enhancing the long-term sustainability and profitability of their organizations.

The study also seeks to identify the key factors that lead to customer churn, including demographic characteristics, behavioral patterns, and customer interactions with the company. By understanding these factors, businesses can develop more targeted and personalized retention strategies.

Additionally, the research investigates methods for handling imbalanced datasets, such as synthetic oversampling or undersampling which are particularly relevant in churn prediction scenarios. The performance of various machine learning algorithms—such as Random Forest, AdaBoost, and neural networks—will be evaluated to determine the most effective models for this binary classification task.

Finally, the study emphasizes the importance of using appropriate evaluation metrics, including AUC-ROC, precision, recall, accuracy, and F1-score, to accurately assess model performance and support the optimization of retention strategies. The insights derived from this research aim to contribute practical value to the telecommunications industry.

1.3 Structure of the Thesis

This thesis is organized into five main chapters. Chapter 1 provides an introduction to the topic, presenting the research problem, the objectives of the study, and the structure of the thesis itself. Chapter 2 outlines the methodology of the literature review, with a particular focus on the machine learning algorithms examined, the evaluation metrics used, and the outcomes reported in previous studies. Chapter 3 presents the proposed experimental methodology, derived through analysis of the existing literature. It also includes a flowchart summarizing the approach as well as an analysis about the used classifiers. Chapter 4 details the experimental setup as well as the results of the study, highlights the limitations encountered during the research process, and presents visualizations of the model outcomes. Finally, Chapter 5 summarizes the main conclusions of the study. The thesis concludes with a comprehensive list of references used throughout the work.

2. Related work

Although no existing studies specifically address customer churn prediction using the Cell2Cell dataset, a substantial body of research has been conducted within the telecommunications industry using alternative datasets. These studies have explored a wide range of machine learning techniques and customer-related features to predict churn behavior, evaluate model performance, and improve retention strategies. Despite the dataset differences, the methodologies and challenges remain closely aligned, providing valuable insights applicable to the current study. The following review presents a summary of recent works focusing on churn prediction in the telecom sector, emphasizing the datasets used, the algorithms applied, and the outcomes most relevant to this thesis.

Kirgiz et al. (2024) conducted a study using real-world data from a Turkish telecom operator to assess the impact of branded apps, gamified loyalty giveaways, and in-house OTT services on customer churn. The dataset included over 100,000 postpaid mobile subscribers with 19 variables spanning demographic, usage, and billing information. The authors applied logistic regression and random forest algorithms, finding that random forest performed significantly better (F1 score: 0.96 vs. 0.69). Feature importance analysis revealed that gamified loyalty giveaways, remaining contract duration, and app

login frequency were strong predictors of churn, whereas OTT service usage had little effect. The study highlights the value of personalized digital engagement strategies while questioning the immediate churn-reducing benefits of telecom-owned OTT platforms.

In their study on improving customer churn prediction, Sahni et al. examined the effects of various resampling techniques on model performance using a real-world telecommunications dataset. The dataset consisted of approximately 20,000 customer records, including demographic details, usage behavior, and account status, with a significant class imbalance between churned and retained customers. The authors implemented several machine learning classifiers, including decision trees, random forest, gradient boosting, and SVM, in conjunction with resampling methods such as SMOTE, ADASYN, and random undersampling. Their findings demonstrated that applying SMOTE with ensemble models—particularly gradient boosting—substantially improved prediction performance, especially in terms of F1-score and recall. The research underscores the importance of handling class imbalance effectively in churn prediction tasks and confirms the utility of synthetic oversampling methods in enhancing classifier robustness.

Alweshah et al. developed a customer churn prediction model based on data from a Jordanian telecommunications provider, focusing on customer behavior and service usage patterns. The dataset included 2,915 customer records with attributes such as service type, contract length, payment history, and call duration. The study employed several machine learning classifiers, including logistic regression, decision trees, SVM, and random forest, to assess their effectiveness in predicting churn. Among these, the random forest algorithm achieved the highest accuracy, demonstrating its suitability for handling complex feature interactions. The researchers highlighted the roleth of data preprocessing, including normalization and feature selection, which significantly enhanced model performance. Their findings support the use of supervised learning approaches in telecom churn analysis and highlight the potential of ensemble methods for improving classification results in real-world settings.

Pradeep and Arya explored the effectiveness of hybrid and ensemble learning approaches for customer churn prediction using a telecom dataset comprising 7,043 customer records, which included variables such as tenure, service subscriptions, contract type, and customer demographics. In this paper, the authors evaluated multiple models, including logistic regression, decision trees, and random forest, and extended the analysis with hybrid combinations like stacking and bagging. Their results indicated that ensemble methods, particularly random forest and the stacked model combining logistic regression and gradient boosting, outperformed individual classifiers in terms of accuracy and AUC-ROC. The study also emphasized the interpretability of models through feature importance analysis, identifying tenure, contract type, and monthly charges as the most influential churn indicators. This work reinforces the effectiveness of ensemble approaches in telecom churn prediction and supports the inclusion of hybrid techniques in comparative model assessments.

Another notable contribution to churn prediction in the telecom sector is the work of Rani and Kant (2020), who explore the potential of semi-supervised learning in improving classification accuracy. They utilize a pseudo-labeling approach that combines labeled and unlabeled data to train six traditional classifiers—SVM, Random Forest, Logistic Regression, AdaBoost, Gradient Boosting, and XGBoost. Their model was tested on a hybrid dataset derived from two publicly available telecom datasets, incorporating demographic, billing, service usage, and contract information. The semi-supervised approach led to substantial improvements in performance metrics across all classifiers, with XGBoost and Gradient Boosting achieving the highest accuracies of 99.62% and 99.24%, respectively. These results suggest that integrating unlabeled data can significantly enhance predictive performance, a relevant insight for handling real-world telecom datasets that often lack comprehensive labels.

Several studies have explored the application of neural networks for churn prediction in the telecom industry. One notable work by Khan et al. (2019) implemented a multilayer perceptron (MLP) artificial neural network using demographic, billing, and service usage data from Pakistani telecom providers. The model utilized backpropagation with hyperbolic tangent and softmax activation functions and achieved an accuracy of 79%. The researchers emphasized the model's ability to capture complex relationships and provide actionable insights into churn behavior. Their findings demonstrate that neural networks can effectively process high-dimensional telecom datasets and outperform several traditional classification methods in identifying customers at risk of churning.

In summary, the reviewed literature highlights a wide array of machine learning applications for customer churn prediction within the telecommunications industry. Despite variations in datasets, model architectures, and preprocessing techniques, common trends emerge—such as the consistent effectiveness of ensemble methods (e.g., Random Forest, Gradient Boosting), the value of feature selection and resampling strategies, and the increasing interest in hybrid and semi-supervised approaches. These studies provide a valuable foundation for the present research, which seeks to apply and compare similar techniques on the underexplored Cell2Cell dataset. By building on these prior insights, the current thesis aims to contribute to the existing body of knowledge by validating established methods in a new context and identifying the most efficient models for telecom churn prediction.

3. Proposed Methodology

This study addresses the problem of customer churn prediction as a binary classification task using the Cell2Cell dataset. The dataset contains approximately 51,000 customer records and 58 features, encompassing demographic characteristics behavioral patterns, and service usage information. For this research, a variety of machine learning algorithms have been used to predict whether a customer will churn.

The proposed methodology follows a structured data science pipeline consisting of data acquisition, exploratory data analysis (EDA), preprocessing, model development, testing and evaluation. Each step was designed to ensure data quality, algorithm robustness, and statistical reliability of results. Special attention was given to addressing class imbalance, optimizing hyperparameters and selecting appropriate features.

Ten classifiers were applied: **Knn, Linear Discriminant Analysis, Logistic Regression, Decision Trees, Random Forest, Naïve Bayes, AdaBoost, Gradient Boosting, XGBoost and a neural network-based classifier.** These classifiers were selected to provide a balanced comparison between traditional models, probabilistic approaches, and advanced ensemble techniques such as boosting, which are known for their ability to improve predictive accuracy by combining multiple weak learners and handling complex, non-linear patterns in the data.

To assess the effectiveness of different modeling strategies, six experiments were conducted. These included a baseline run with default settings, the introduction of class weighting to implement cost-sensitive learning, the adjustment of hyperparameter tuning with recall as the optimization metric, and the application of SMOTE for synthetic oversampling. Additional experiments combined SMOTE with class weighting and incorporated weighted loss functions in gradient-based models. These experimental variations aimed to evaluate the impact of imbalance handling and cost sensitivity on model performance, particularly in terms of recall and F1-score.

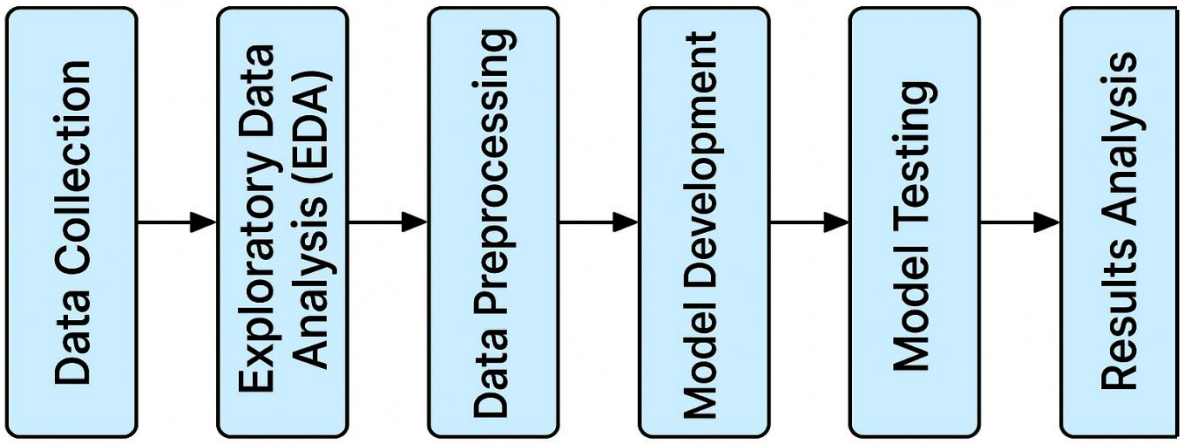

Figure 1: Proposed Methodology Workflow

3.1 Machine Learning Classifiers

In the context of customer churn prediction in the telecom industry, machine learning classifiers provide a robust framework for analyzing structured data and making predictive inferences. This study explores a diverse set of classifiers ranging from linear models to ensemble techniques, all of which are commonly applied in academic and industrial research. The selection of classifiers was influenced by their theoretical diversity, practical interpretability, and performance across classification tasks involving imbalanced datasets, as demonstrated in relevant literature. The goal is to assess their individual strengths and suitability for modeling customer churn behavior, especially under different experimental modifications such as cost-sensitive learning and synthetic sampling strategies.

Logistic Regression (LR)

Logistic Regression is a linear classification model widely utilized for binary outcome prediction tasks like churn detection. It estimates the probability that a given input belongs to a particular category by fitting data to a logistic function, mapping any real-valued input into the (0,1) interval. LR is particularly valued for its simplicity and interpretability, making it a reliable baseline model for churn prediction. It is also computationally efficient and performs well when the relationship between features and target variable is approximately linear. As emphasized by Bhatnagar and Srivastava (2025), logistic regression has proven effective in telecom churn contexts when accompanied by proper preprocessing and balancing techniques.

k-Nearest Neighbors (KNN)

KNN is a non-parametric, instance-based learning algorithm that classifies a sample based on the majority class among its k-nearest neighbors in the feature space. It does not assume any underlying data distribution, making it versatile for a range of applications. However, its performance can degrade with high-dimensional data or imbalanced class distributions, which is a common challenge in churn datasets. KNN's sensitivity to feature scaling and choice of k also requires careful tuning (Bhatnagar & Srivastava, 2025).

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a statistical classification method based on the assumption that the different classes generate data based on Gaussian distributions with shared covariance matrices. LDA seeks to reduce dimensionality while preserving as much class-discriminatory information as possible by projecting data onto a lower-dimensional space where class separability is maximized. It achieves this by finding a linear combination of features that best separates the classes. In the context of churn prediction, LDA is advantageous when class distributions are relatively well-behaved and linearly separable. Although it is less flexible than non-linear models, LDA remains computationally efficient and interpretable, often serving as a strong baseline in binary classification problems.

Naive Bayes

Naive Bayes is a probabilistic classifier grounded in Bayes' Theorem, which assumes independence between predictors. Despite its simplicity and strong assumption of feature independence, it has proven effective in various classification problems, including churn prediction. Its main strength lies in its efficiency, requiring relatively little training data and offering fast computation, which makes it ideal for large datasets. However, in churn datasets where correlations between customer features are common, the independence assumption often leads to lower predictive performance compared to other models. As highlighted in comparative studies, Naive Bayes often underperforms when dealing with imbalanced datasets or non-linear feature interactions (Majhi et al., 2011).

Decision Tree (DT)

Decision Trees classify instances by recursively splitting data based on feature values, forming a tree structure where internal nodes represent features and leaves represent outcomes. This algorithm is intuitive and interpretable, capable of handling both categorical and numerical features. However, it is prone to overfitting, particularly in the absence of pruning or regularization. Decision Trees are foundational in ensemble methods and serve as effective base learners for boosting and bagging strategies, especially when dealing with class imbalance issues common in churn prediction.

Random Forest (RF)

Random Forest is an ensemble method that builds multiple decision trees on different subsets of the data and averages their predictions to improve generalization. It reduces overfitting and improves accuracy by introducing randomness in both sample and feature selection. In churn prediction tasks, Random Forest often emerges as a top performer due to its robustness to noise and ability to capture non-linear interactions. The literature consistently identifies it as one of the best models in telecom churn scenarios, achieving accuracies exceeding 90% with proper feature selection and resampling strategies.

AdaBoost

AdaBoost (Adaptive Boosting) constructs a strong classifier by combining multiple weak learners, typically decision stumps, in a sequential manner. Each model focuses on correcting the errors of its predecessors by adjusting the weights of misclassified instances. Although sensitive to noisy data and outliers, AdaBoost can yield competitive results in moderately imbalanced datasets and has been shown to adapt well under cost-sensitive and SMOTE-based conditions, as reported in recent research on telecom churn prediction (Bhatnagar & Srivastava, 2025).

Gradient Boosting

Gradient Boosting is a sequential ensemble learning technique that constructs a strong classifier by iteratively combining the outputs of multiple weak learners, typically decision trees. Each tree is trained to correct the errors made by the previous ensemble, focusing more on the misclassified instances in each round. This approach allows the model to progressively improve its performance by emphasizing difficult cases, making it particularly effective for tasks involving class imbalance, such as churn prediction. Its ability to optimize a chosen loss function directly and adaptively has made it a robust choice for classification problems in real-world datasets. Despite requiring careful parameter tuning to avoid overfitting, Gradient Boosting has shown strong predictive power in churn modeling contexts (Burez and Van den Poel, 2009).

XGBoost (Extreme Gradient Boosting)

XGBoost is a highly optimized gradient boosting framework that incorporates regularization, parallel computation, and efficient handling of missing data. It iteratively builds trees that minimize a specified loss function and adjusts for overfitting via shrinkage and subsampling. In churn prediction, XGBoost consistently delivers high accuracy and F1 scores, particularly in imbalanced and high-dimensional datasets. The effectiveness of XGBoost in telecom applications is well-documented, demonstrating its capacity to identify nuanced patterns in customer behavior.

Neural Networks

Artificial Neural Networks (ANNs) are powerful machine learning models inspired by the structure and functioning of the human brain. They are composed of layers of

interconnected nodes (neurons) that transform input data through activation functions, allowing the model to learn complex, non-linear relationships. In the context of customer churn prediction, neural networks have been successfully applied to telecom datasets, demonstrating strong performance due to their capacity to automatically learn abstract feature representations. However, their effectiveness heavily depends on the network architecture, proper regularization, and balanced training data. Studies have shown that when appropriately tuned, neural networks can outperform traditional models, although they may require more computational resources and are sensitive to overfitting if not carefully regularized (Majhi et al., 2011).

These classifiers collectively provide a balanced mix of simplicity, interpretability, computational efficiency, and predictive power, making them well-suited for experimentation in churn prediction frameworks.

4. Experimental Setup

This chapter presents the detailed implementation of the proposed methodology, outlining each stage of the experimental workflow used to predict customer churn. It includes a systematic explanation of the dataset used, preprocessing techniques applied, feature selection methods, model development, hyperparameter tuning, and evaluation procedures. All experiments were executed in the Google Colab environment using CPU runtime, leveraging a variety of Python libraries to support the end-to-end machine learning pipeline. The purpose of this chapter is to provide transparency and reproducibility in the experimental design, ensuring that the results are both interpretable and grounded in rigorous methodological practices.

For this purpose, the main libraries of the python language, that were used in this process are mentioned:

- **Pandas**: Used for data manipulation and preprocessing tasks, such as loading the dataset, handling missing values, filtering records, and preparing feature matrices and labels.
- **Numpy:** Provided efficient numerical operations and array handling, supporting matrix computations, statistical functions, and seamless integration with other libraries like scikit-learn.
- **Seaborn:** Utilized for exploratory data analysis (EDA) and advanced data visualization, including feature distribution plots, correlation heatmaps, and class imbalance insights.
- **Sklearn:** Served as the core library for building, training, and evaluating machine learning models. It was also used for preprocessing (e.g., encoding, scaling), applying cross-validation, hyperparameter tuning, and computing classification metrics.

- **Matplotlib:** Used alongside Seaborn for generating detailed visualizations, such as confusion matrices and ROC curves, to better interpret and present the performance of the models.

4.1 Data acquisition

The dataset used in this study is the Cell2Cell Churn Dataset, originally published by a U.S.-based telecommunications company and made publicly available through the Kaggle platform. It was specifically created for churn prediction tasks and has been widely used in academic and applied machine learning research. The features include demographic attributes, billing details, service plan attributes and customer behavior metrics such as call frequency, tenure and usage patterns. The dataset was obtained in CSV format and imported into the Python environment via Google Colab using Pandas for structured data manipulation.

4.2 Exploratory Data Analysis

Exploratory Data Analysis (EDA) serves as a foundational step for revealing the dataset's structure, patterns, and potential irregularities. Prior to deploying machine learning models, it is necessary to assess data integrity, detect anomalies, and identify issues such as imbalance or noise. The findings from this stage directly inform preprocessing decisions and contribute to selecting features most likely to affect churn prediction.

4.2.1 Feature Overview

A comprehensive table summarizing all 58 features was constructed to outline their roles and formats within the dataset including a brief description and classification of each feature as either continuous or symbolic. Classifying features as either continuous or symbolic allows for tailored preprocessing, ensuring that each variable is transformed appropriately for modeling.

- **Continuous features** represent numeric values that can take on a wide range of magnitudes, often used in quantitative analysis. Examples include variables like monthly charges, minutes of use, and account tenure. These features typically require scaling and may capture patterns or trends in churn behavior.

- **Symbolic features**, also referred to as categorical, represent non-numeric values or labels, such as gender, state, or contract type. These features are often encoded into numerical form (e.g., via one-hot or label encoding) so they can be used effectively by machine learning algorithms.

Table 1: Description of Dataset Features Classified by Type

No	Feature Name	Feature Type	Description
1	CustomerID	Continuous	Unique identifier for each customer.
2	Churn	Symbolic	Indicates if the customer has churned: Yes or No.
3	MonthlyRevenue	Continuous	Monthly revenue charged to the customer.
4	MonthlyMinutes	Continuous	Total minutes used by the customer per month.
5	TotalRecurringCharge	Continuous	Monthly recurring charges.
6	DirectorAssistedCalls	Continuous	Calls assisted by a director.
7	OverageMinutes	Continuous	Minutes used above plan allowance.
8	RoamingCalls	Continuous	Number of roaming calls.
9	PercChangeMinutes	Continuous	Percentage change in minutes used.
10	PercChangeRevenues	Continuous	Percentage change in revenue.
11	DroppedCalls	Continuous	Number of dropped calls.
12	BlockedCalls	Continuous	Number of blocked calls.
13	UnansweredCalls	Continuous	Number of unanswered calls.
14	CustomerCareCalls	Continuous	Calls made to customer care.
15	ThreewayCalls	Continuous	Three-way calls made.
16	ReceivedCalls	Continuous	Number of calls received.
17	OutboundCalls	Continuous	Number of outbound calls.
18	InboundCalls	Continuous	Number of inbound calls.
19	PeakCallsInOut	Continuous	In/out calls during peak hours.
20	OffPeakCallsInOut	Continuous	In/out calls during off-peak hours.
21	DroppedBlockedCalls	Continuous	Sum of dropped and blocked calls.
22	CallForwardingCalls	Continuous	Calls using call forwarding.
23	CallWaitingCalls	Continuous	Calls received with call waiting.
24	MonthsInService	Continuous	Number of months in service.
25	UniqueSubs	Continuous	Unique subscribers under this account.
26	ActiveSubs	Continuous	Currently active subscribers.
27	ServiceArea	Symbolic	Geographical service area.
28	Handsets	Continuous	Number of handsets on the account.
29	HandsetModels	Continuous	Different handset models used.
30	CurrentEquipmentDays	Continuous	Days since current equipment was activated.
31	AgeHH1	Continuous	Age of first household member.
32	AgeHH2	Continuous	Age of second household member.
33	ChildrenInHH	Symbolic	Indicates children in the household.
34	HandsetRefurbished	Symbolic	Refurbished handset used.
35	HandsetWebCapable	Symbolic	Whether handset supports web access.
36	TruckOwner	Symbolic	Indicates if customer owns a truck.
37	RVOwner	Symbolic	Indicates if customer owns a recreational vehicle.

38	Homeownership	Symbolic	Indicates if customer owns their home.
39	BuysViaMailOrder	Symbolic	Purchases made through mail order.
40	RespondsToMailOffers	Symbolic	Responds to promotional mail.
41	OptOutMailings	Symbolic	Opted out of marketing mailings.
42	NonUSTravel	Symbolic	Travelled outside the U.S.
43	OwnsComputer	Symbolic	Owns a personal computer.
44	HasCreditCard	Symbolic	Owns a credit card.
45	RetentionCalls	Continuous	Calls to retention department.
46	RetentionOffersAccepted	Continuous	Retention offers accepted.
47	NewCellphoneUser	Symbolic	New to using a cellphone.
48	NotNewCellphoneUser	Symbolic	Not new to using a cellphone.
49	ReferralsMadeBySubscriber	Continuous	Number of referrals made.
50	IncomeGroup	Continuous	Income bracket of the household.
51	OwnsMotorcycle	Symbolic	Indicates motorcycle ownership.
52	AdjustmentsToCreditRating	Continuous	Modifications to credit score.
53	HandsetPrice	Symbolic	Price of the handset.
54	MadeCallToRetentionTeam	Symbolic	Whether a retention call was made.
55	CreditRating	Symbolic	Credit rating category.
56	PrizmCode	Symbolic	PRIZM customer segmentation code.
57	Occupation	Symbolic	Customer's occupation.
58	MaritalStatus	Symbolic	Marital status of the customer.

4.2.2 Churn Class Distribution

Following the feature overview, it is important to examine the distribution of the target variable, Churn, to better understand the class balance within the dataset. The bar chart below illustrates the number of customers who have churned compared to those who have not.

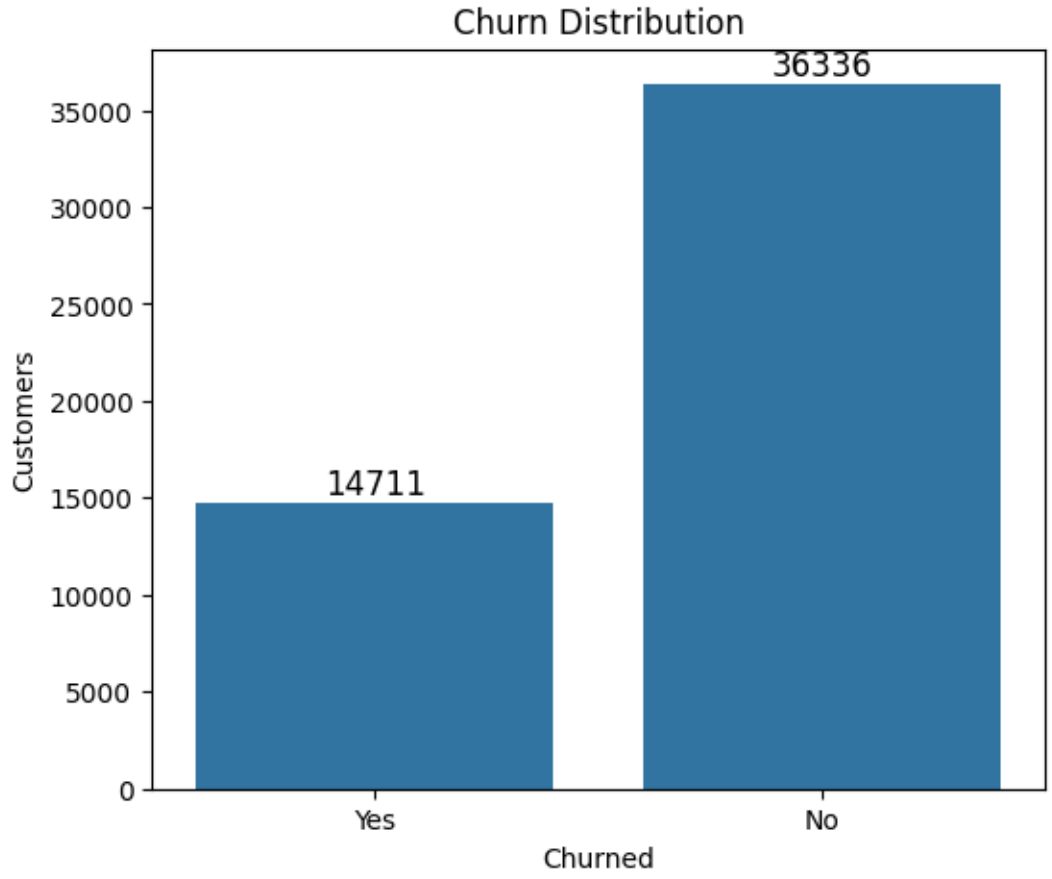

Figure 2: Dataset's Churn Distribution

As shown in Figure 2, the dataset is highly imbalanced, with 14,711 customers labeled as churned and 36,336 as non-churned. This means that approximately 29% of the customers in the dataset have churned. Such imbalance can lead to biased machine learning models that favor the majority class, reducing the model's ability to accurately detect churners. Addressing this imbalance is therefore a key focus in later stages of model development and evaluation.

4.2.3 Feature Importance Analysis

To highlight the variables with the greatest predictive contribution, a feature importance analysis was performed using a Random Forest classifier. This approach evaluates the

relative contribution of each feature to the predictive performance of the model. By fitting the model on the full dataset, encoded using one-hot encoding for categorical variables, the algorithm assigns importance scores to each feature based on how frequently and effectively it is used to split nodes across the ensemble of trees. The top 20 features with the highest importance values are presented in the chart below.

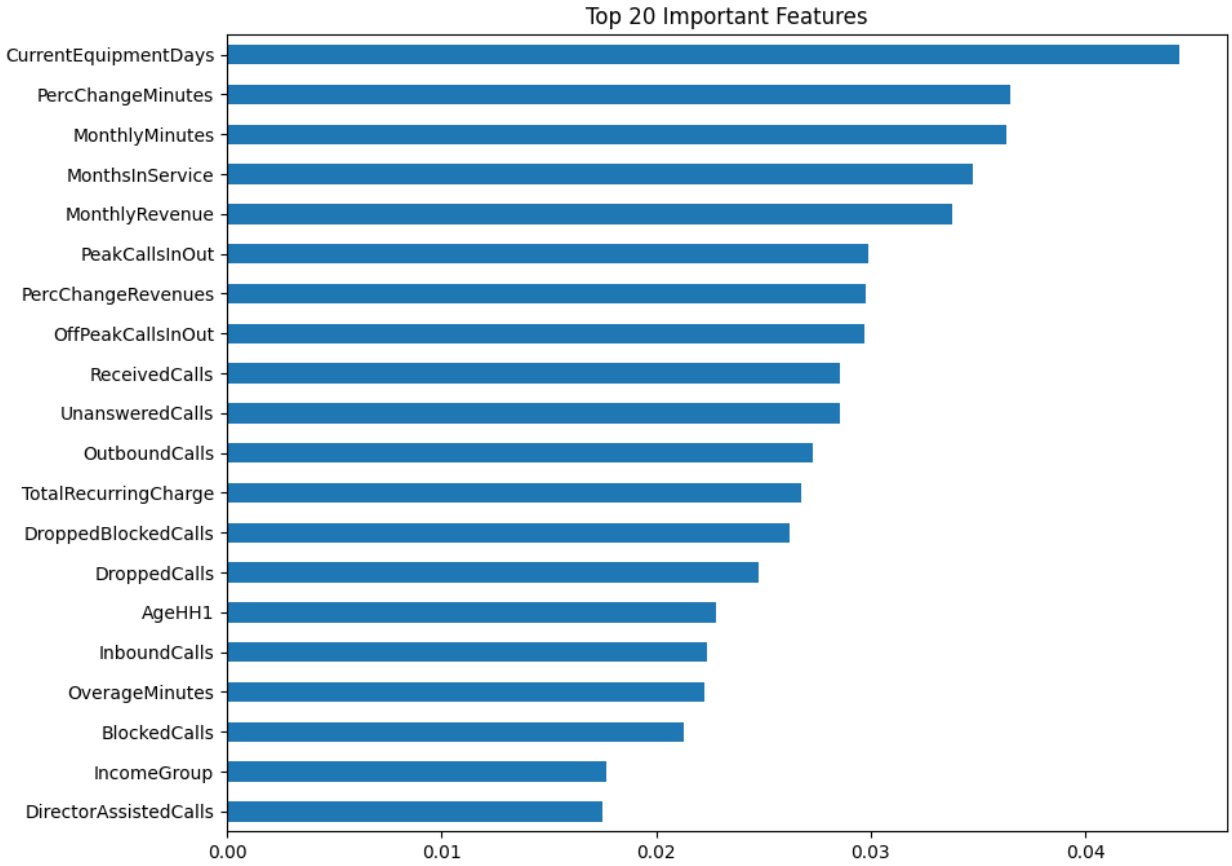

Figure 3: Most important features

As shown in Figure 3, the most influential feature in predicting customer churn is *CurrentEquipmentDays*, followed by *PercChangeMinutes*, *MonthlyMinutes*, and *MonthsInService*. While these features rank highest in relative importance according to the Random Forest model, it is worth noting that their absolute importance values remain relatively low, with the top feature reaching just over 0.04. This suggests that no single variable dominates the prediction, and churn behavior may be influenced by a combination of subtle factors rather than a few highly decisive ones. Nevertheless, the leading features are largely related to service usage and customer lifecycle, indicating that shifts in behavior and long-term engagement play meaningful roles. Additional important variables include call activity metrics such as *ReceivedCalls*, *UnansweredCalls*, and *DroppedBlockedCalls*, which may signal user dissatisfaction or reduced interaction. These insights are useful for guiding both feature selection in model optimization and the development of more targeted customer retention strategies.

4.2.4 Visual Analysis of Feature Distributions

To further explore the behavior of key features identified by the Random Forest classifier, boxplots and histograms were generated for the top-ranked predictors. These visualizations aim to highlight potential distribution differences between churned and non-churned customers and to support the model's feature importance rankings through visual examination.

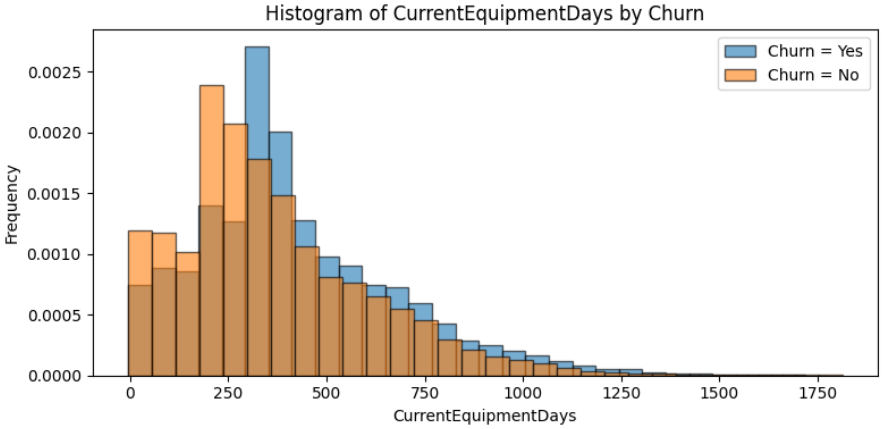

Figure 4: Histogram of CurrentEquipmentDays

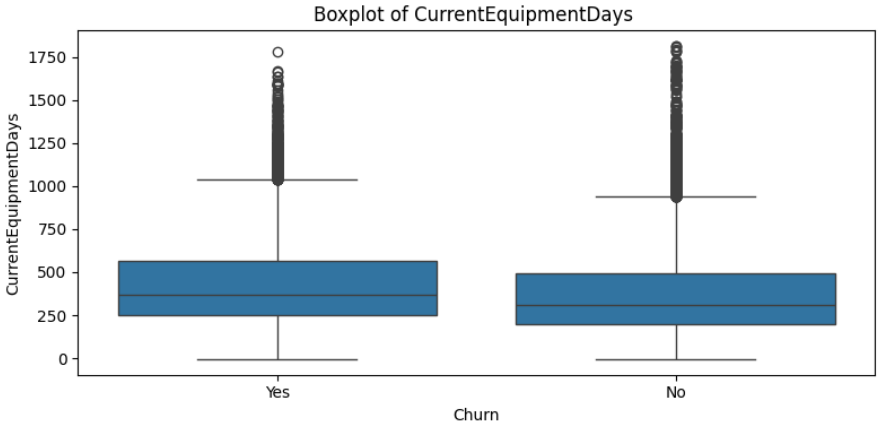

Figure 5: Boxplot of CurrentEquipmentDays

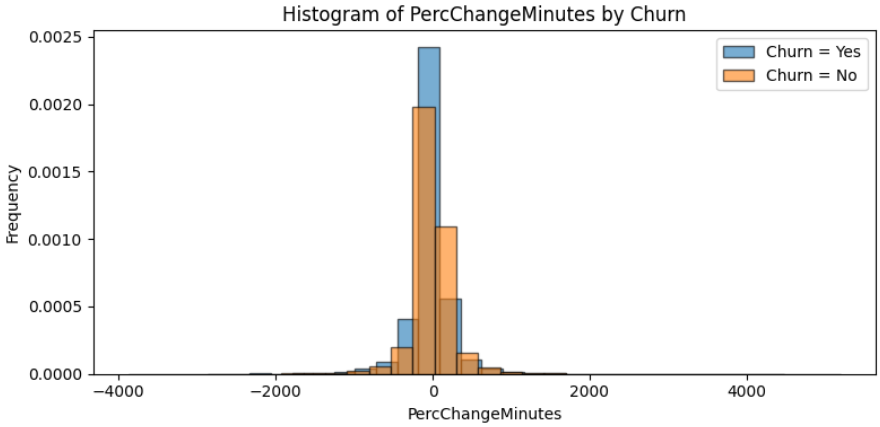

Figure 6:Histogram of PercChangeMinutes

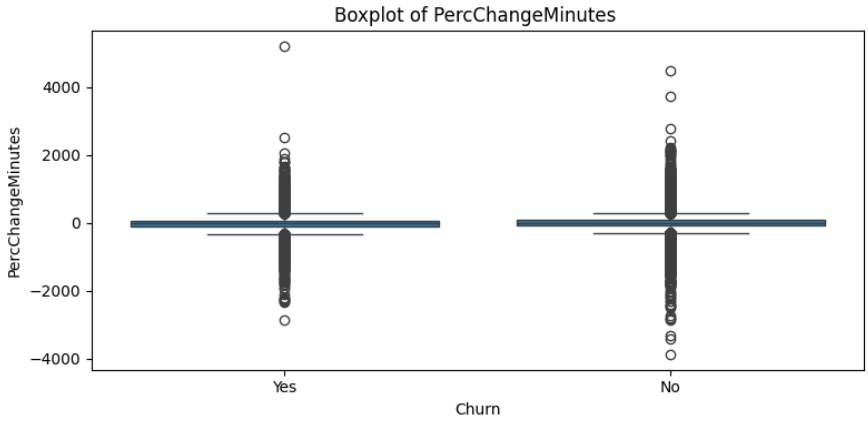

Figure 7:Boxplot of PercChangeMinutes

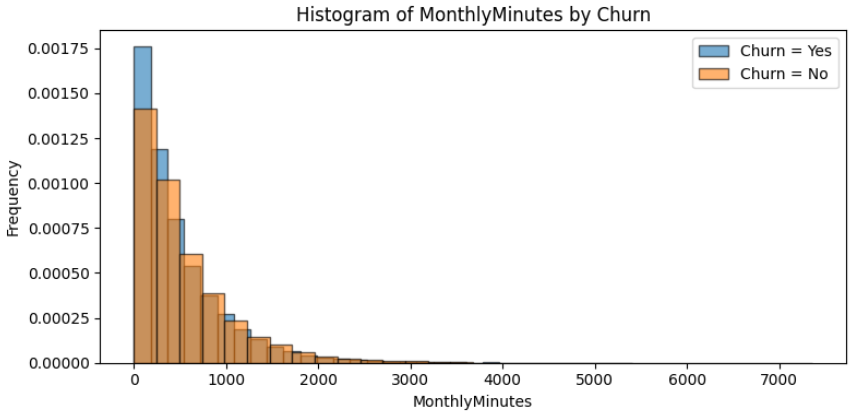

Figure 8: HIstogram of MonthlyMinutes

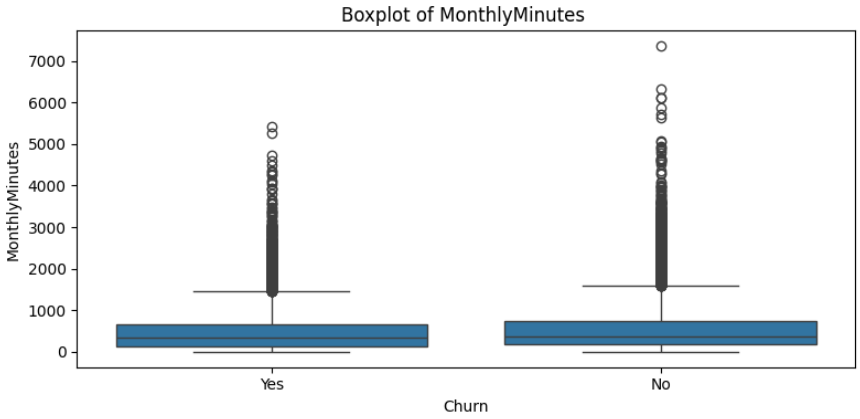

Figure 9: Boxplot of MonthlyMinutes

Despite being ranked as the most important features by the Random Forest model, the visualizations of *CurrentEquipmentDays*, *MonthlyMinutes*, and *PercChangeMinutes* reveal substantial overlap between the churned and non-churned classes. Boxplots show similar interquartile ranges and medians, while histograms confirm highly comparable distribution shapes and central tendencies. This lack of visible distinction can be attributed to several factors. First, the dataset is notably imbalanced, with churned customers representing a much smaller proportion, making their influence less pronounced in visual summaries. Second, the dataset's size and complexity introduce noise, subtle patterns, and non-linear interactions that are not readily apparent in univariate plots. Third, churn is often the result of multifactorial behavior; a single feature rarely offers strong separation without considering contextual or combined effects. These findings demonstrate that while visual analysis is informative, its limitations must be acknowledged, particularly in classification problems where relationships are not linearly or independently expressed.

The boxplots across all features also reveal a high number of extreme values. These outliers may reflect atypical usage patterns, such as business accounts, billing anomalies, or data entry errors. Although some outliers are expected in large datasets, their presence can distort measures of central tendency and complicate interpretation based on visualizations. Certain tree-based algorithms employed in this study—such as Random Forest, Decision Trees, and XGBoost—are relatively robust to such anomalies due to their threshold-based splitting mechanisms. In contrast, models like k-Nearest Neighbors, Logistic Regression, LDA, and Neural Networks are more sensitive, as they depend on distance calculations or gradient-based optimization. Consequently, outliers may affect model performance unevenly and warrant careful consideration during preprocessing and evaluation.

4.3 Data Preprocessing

The data preprocessing phase involved a series of quality assurance steps aimed at ensuring the reliability and consistency of the dataset prior to modeling. The dataset was first examined for duplicates, of which none were found—indicating that each row

represented a unique customer instance. Subsequently, the dataset was assessed for missing values, outliers, and irrelevant features.

4.3.1 Missing Values

Table 2 summarizes the features that contain missing values, along with the respective number and percentage of missing entries.

Table 2: Missing values for each feature:

Feature Name	Missing Values	Percent Missing
MonthlyRevenue	156	0.31%
MonthlyMinutes	156	0.31%
TotalRecurringCharge	156	0.31%
OverageMinutes	156	0.31%
RoamingCalls	156	0.31%
PercChangeMinutes	367	0.72%
PercChangeRevenues	367	0.72%
ServiceArea	24	0.05%
Handsets	1	0%
HandsetModels	1	0%
CurrentEquipmentDays	1	0%
AgeHH1	909	1.78%
AgeHH2	909	1.78%

To address the missing data, appropriate imputation strategies were applied based on the nature and distribution of each feature. For numerical variables such as *MonthlyRevenue, MonthlyMinutes,* and *CurrentEquipmentDays*, missing values were filled using the median, ensuring robustness against outliers and skewed distributions. This approach was consistently applied to all continuous features with missing entries, including *PercChangeMinutes*, *OverageMinutes*, and *AgeHH1*. For categorical features like *ServiceArea*, the most frequent value (mode) was used to preserve consistency in class representation. These imputations ensured that no information was lost due to row removal while maintaining statistical validity for subsequent modeling.

4.3.2 Outliers Clipping

To address the potential influence of extreme values on model performance and statistical integrity, outlier detection and clipping were applied to all numerical features in the dataset. The interquartile range (IQR) method was used, with thresholds set at three times the IQR below the first quartile (Q1) and above the third quartile (Q3), to identify and cap extreme values. This approach was selected for its balance between robustness and simplicity in large datasets.

Table 3:Outliers clipped for each feature:

Feature Name	Outliers Clipped	Percentage Clipped
MonthlyRevenue	1018	1.99%
MonthlyMinutes	546	1.07%
TotalRecurringCharge	249	0.49%
DirectorAssistedCalls	2616	5.12%
OverageMinutes	3298	6.46%
RoamingCalls	8361	16.38%
PercChangeMinutes	2718	5.32%
PercChangeRevenues	8707	17.06%
DroppedCalls	1406	2.75%
BlockedCalls	3093	6.06%
UnansweredCalls	1326	2.60%
CustomerCareCalls	3994	7.82%
ThreewayCalls	3234	6.34%
ReceivedCalls	1209	2.37%
OutboundCalls	1055	2.07%
InboundCalls	2358	4.62%
PeakCallsInOut	872	1.71%
OffPeakCallsInOut	1246	2.44%
DroppedBlockedCalls	1564	3.06%
CallForwardingCalls	234	0.46%
CallWaitingCalls	4711	9.23%
UniqueSubs	234	0.46%
ActiveSubs	23	0.05%
Handsets	1149	2.25%
HandsetModels	281	0.55%
CurrentEquipmentDays	101	0.20%
RetentionCalls	1745	3.42%
RetentionOffersAccepted	881	1.73%
ReferralsMadeBySubscriber	2384	4.67%
AdjustmentsToCreditRating	1838	3.60%

The results in Table 3 shows substantial numbers of outliers in several features, such as *RoamingCalls*, *PercChangeRevenues*, and *CustomerCareCalls*, each with thousands of entries outside the defined range. These values were clipped to the calculated bounds to reduce skewness and stabilize model inputs. This preprocessing step is especially important for algorithms sensitive to numerical range and scale (e.g., logistic regression, k-nearest neighbors, and neural networks), where extreme values can disproportionately affect distance measures and gradient updates. While tree-based models like random forests and boosting algorithms are inherently more robust to outliers, maintaining consistency across the pipeline enhances the overall comparability and reliability of the experimental outcomes.

4.3.3 Feature Selection

Feature selection plays a vital role in the preprocessing stage of machine learning workflows. It helps reduce dimensionality, enhance model interpretability, and improve generalization by eliminating redundant, irrelevant, or noisy features. By focusing on the most informative predictors, feature selection not only enhances model performance but also reduces computational complexity, which is particularly important when working with

large-scale datasets, such as the one used in this study. Moreover, reducing the number of input variables can help mitigate overfitting and contribute to more stable and explainable models.

In this study, two complementary techniques were applied for feature selection. The first approach was based on variance thresholding. Features with extremely low variance are unlikely to provide meaningful signals for distinguishing between classes. A threshold of 0.1 was set, and five features were identified and removed: *'CallForwardingCalls', 'RetentionCalls', 'RetentionOffersAccepted', 'ReferralsMadeBySubscriber', and 'AdjustmentsToCreditRating'*. These attributes exhibited minimal variation across the dataset and were considered uninformative for the prediction task.

The second method involved analyzing the correlation matrix of all numeric features to detect and address multicollinearity. When two features were highly correlated (threshold set at Pearson correlation coefficient > 0.8), one of the pairs was removed to reduce redundancy. As a result, the features *'DroppedBlockedCalls'*, *'ActiveSubs'*, and *'HandsetModels'* were excluded from further analysis. This two-step selection process ensured a more compact, non-redundant feature set to support more effective model learning.

After applying both variance thresholding and correlation-based filtering, a total of 8 features were excluded from the original 58. As a result, the final dataset consists of 50 selected features, which will be used for model training and evaluation in the subsequent experiments.

4.3.4 Feature Encoding

Feature encoding is a crucial step in the data preprocessing pipeline, especially when working with machine learning models that require numerical input. Many features in the dataset were categorical or binary in nature, and these needed to be transformed into a numerical format to be properly interpreted by algorithms such as logistic regression, support vector machines, or neural networks. Encoding also helps preserve the semantics of categorical variables while ensuring that they contribute meaningfully to model learning.

Before encoding, every categorical feature was checked for unique values in order to understand each feature category. The first type of encoding applied was binary encoding, where features with two possible outcomes—typically “Yes” or “No”—were converted into values of 1 and 0 respectively. These included attributes like *OwnsComputer, HandsetWebCapable, TruckOwner,* and *BuysViaMailOrder.* Binary encoding is a simple yet effective way to represent dichotomous variables, especially for models that can learn from discrete inputs.

The second approach was ordinal encoding, which was applied to features that carry a natural order or hierarchy in their values. The *CreditRating* variable, for example, included levels such as '1-Highest' to '7-Lowest', which were mapped to integers from 1 to 7, preserving the intended ranking. This method ensures that the ordinal relationships are maintained during model training. Similarly, Homeownership was encoded as 1 for "Known" and 0 for "Unknown", while *MaritalStatus* was represented with 1 for "Yes", 0 for "No", and 2 for "Unknown" to reflect distinct categories with increasing uncertainty.

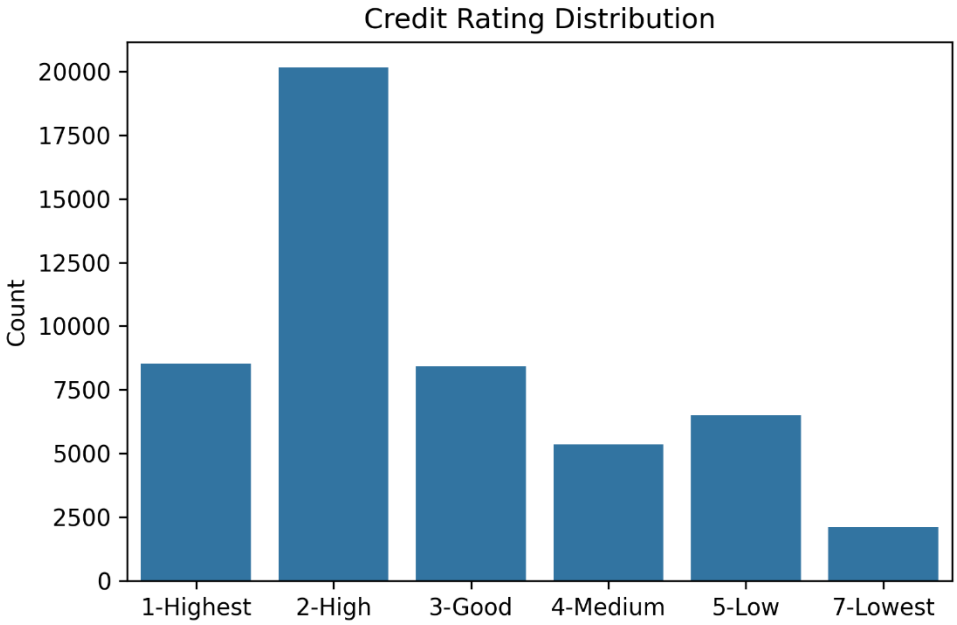

Figure 10: Credit Rating Distribution

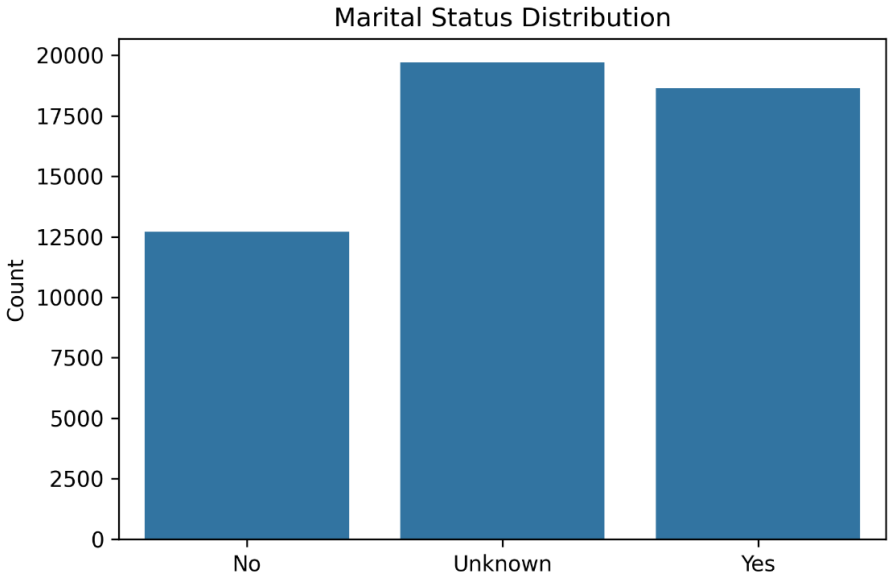

Figure 11: Marital Status Distribution

Some nominal categorical variables, such as *PrizmCode*, *Occupation*, and *ServiceArea*, were encoded using label mapping. These variables do not have a natural order, but their values were mapped to integers for efficient storage and compatibility with algorithms. For example, *PrizmCode* was encoded with a scheme mapping 'Suburban' to 0, 'Town' to 1, and so forth. Similarly, *Occupation* categories like 'Professional', 'Crafts', and 'Retired' were converted to integers from 0 to 7. To avoid high cardinality in *ServiceArea*, the dataset was grouped so that only regions with more than 100 instances retained their individual category, and the rest were assigned the label "Other". This was then mapped into integers. Grouping infrequent classes helps reduce noise and prevent overfitting.

One special case was the *HandsetPrice* variable, which was stored as strings and sometimes had non-numeric entries such as “unknown”. This column was converted to numeric format using pandas' to_numeric function with coercion, and missing values were imputed with the median. Overall, this stage ensured that the final dataset was cleanly encoded with all features in numerical format—ready for model training.

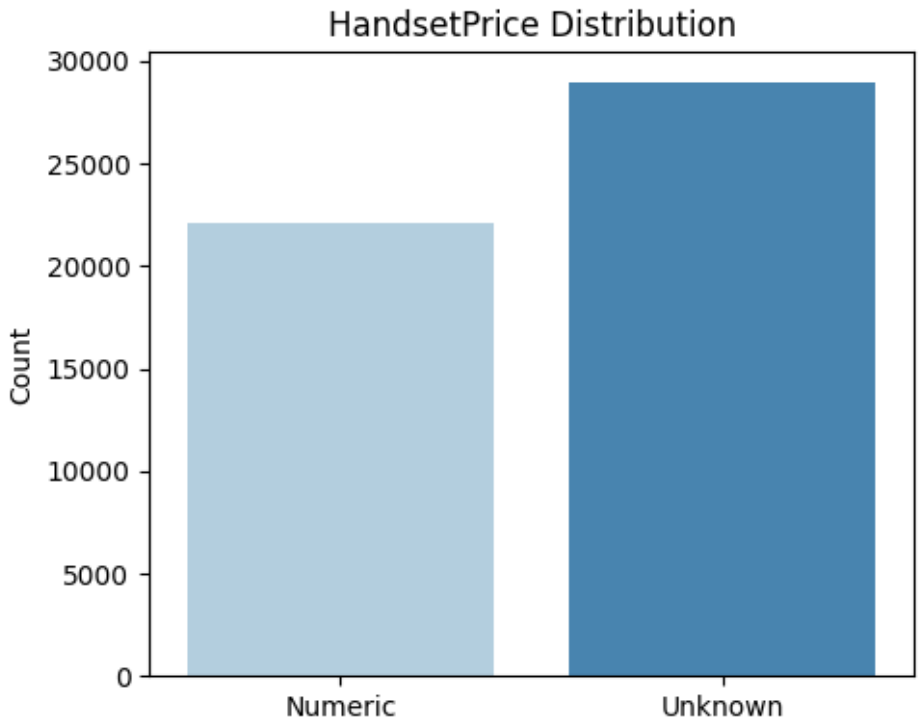

Figure 12: HandsetPrice Distribution between numeric and unknown

4.3.5 Feature Scaling

Feature scaling is another fundamental preprocessing step, particularly necessary for machine learning models that are sensitive to the scale or magnitude of input features. This includes algorithms that use distance-based metrics (e.g., KNN), rely on gradient descent optimization (e.g., neural networks), or are regularized through penalties (e.g., logistic regression with L1/L2 regularization). Without scaling, features with larger numeric ranges can dominate others during model training, potentially skewing the results.

In this study, **standardization** was chosen as the scaling technique. This method transforms the features to have a mean of zero and a standard deviation of one. Standardization is preferred over min-max normalization in this context because it does not compress the range of data and is less sensitive to outliers—although the extreme values had already been clipped during earlier preprocessing.

Only numeric columns were standardized. The StandardScaler from scikit-learn was then applied to each of these columns. As a result, all continuous and discrete numerical variables now reside on the same scale, ensuring equal contribution during model training.

After encoding and scaling, a final integrity check was performed to confirm that:

- No missing values remained.
- All features were converted to numeric format.
- The dataset was clean and properly structured for input into machine learning algorithms.

This encoding and scaling pipeline not only ensures compatibility with various models but also improves learning efficiency and predictive performance. Together with previous preprocessing steps such as outlier clipping and feature selection, the dataset has now been fully prepared for experimentation.

4.4 Model Development

This section presents the model development phase, where a variety of classification algorithms were implemented to predict customer churn. The goal was to explore and compare the performance of multiple models under consistent experimental conditions. To ensure reliable evaluation and reduce bias from imbalanced class distributions, stratified K-fold cross-validation was applied. Additionally, randomized hyperparameter tuning was used to optimize each model's performance. This stage also includes the development of a neural network to provide a deep learning baseline against traditional classifiers. All models were trained and tested using the preprocessed dataset, with consistent metric evaluation across experiments.

4.4.1 Data Splitting Strategy

To ensure reliable model evaluation while accounting for the class imbalance in the dataset, a stratified K-fold cross-validation approach was employed. Specifically, the data was split into four folds, preserving the original proportion of churned and non-churned customers in each subset. The decision to use 4 folds represents a practical trade-off between computational efficiency and comprehensive evaluation. Compared to leave-one-out or higher K values like 10, which would offer marginally more generalization insight at a higher computational cost, 4-fold cross-validation allows each data point to be used for training three times and for testing once. This is especially important for imbalanced classification problems like churn prediction, where naive splitting could result in skewed class representations and biased models.

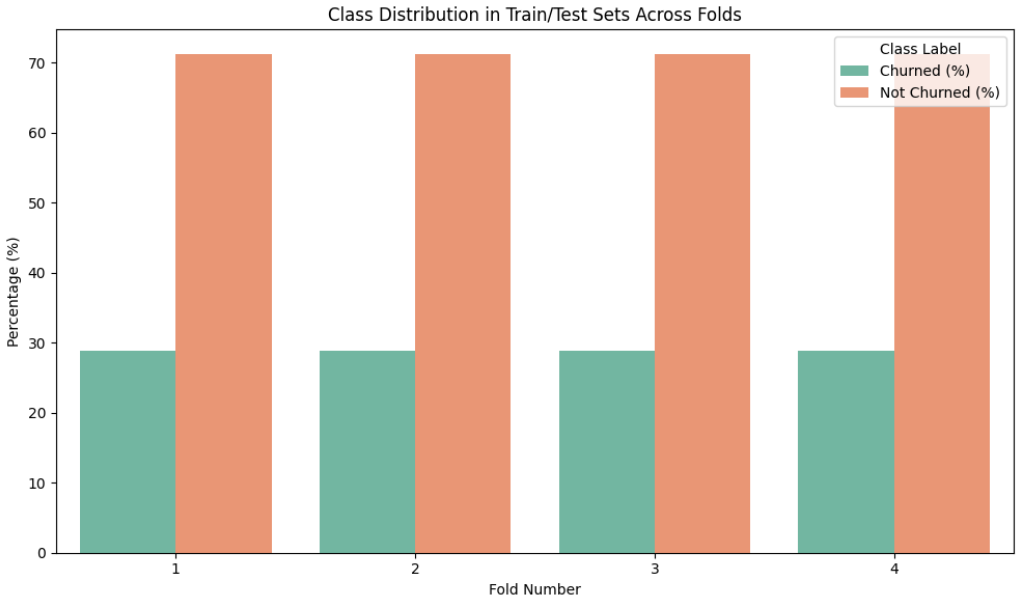

Figure 13: Class Distribution after 4-fold cross validation

Figure 10 confirms that the stratified K-fold cross-validation maintained a consistent distribution of the target variable across all four folds. In each case, the training and testing subsets reflect the original class imbalance, with approximately 71% of customers labeled as non-churned and 29% as churned. This balanced preservation ensures that both classes are adequately represented during each iteration, allowing the models to be fairly evaluated and reducing the risk of biased performance metrics due to class imbalance. Such consistency is essential for reliable model validation in churn prediction tasks.

4.4.2 Classifiers Hyperparameter optimization

To enhance the performance and generalizability of each classifier, this study employed RandomizedSearchCV, a hyperparameter optimization technique from the scikit-learn library. Unlike grid search, which exhaustively tries all possible parameter combinations,

randomized search evaluates a fixed number of random combinations drawn from specified distributions, making it more efficient for large search spaces. In this study, the number of iterations (n_iter) was set to 10, meaning 10 different sets of hyperparameters were sampled for each classifier.

The performance of each sampled model was evaluated using 3-fold cross-validation (cv=3), which means each training set (from the outer 4-fold stratified split) was further divided into three parts to validate the hyperparameter combinations. This setting was chosen as a balance between computational efficiency and performance stability, allowing robust tuning without excessive processing time.

The scoring metric used for model selection was the F1-score, which is particularly suitable for imbalanced datasets like this one, as it balances precision and recall. Additionally, n_jobs=-1 was specified to parallelize computations across all available CPU cores, significantly reducing training time. The random_state=42 ensures reproducibility of results.

Each classifier had its own custom-defined parameter search space, consisting of appropriate ranges or distributions for critical hyperparameters like max_depth, learning_rate, or n_estimators, depending on the algorithm. These were based on common practices and prior research in churn prediction tasks. Once the best parameters were found, the resulting optimized model was retained and evaluated on the corresponding test set from the outer cross-validation loop.

Table 4: Optimized Hyperparameters for each Classifier

Classifier Name	Optimized Hyperparameters
k-Nearest Neighbors (kNN)	{'n_neighbors': 2, 'p': 2, 'weights': 'distance'}
Linear Discriminant Analysis	{'shrinkage': 'auto', 'solver': 'eigen', 'tol': 0.0093}
Logistic Regression	{'C': 3.75, 'penalty': 'l1', 'solver': 'liblinear'}
Decision Trees	{'criterion': 'entropy', 'max_depth': 17, 'min_samples_leaf': 3, 'min_samples_split': 5}
Random Forest	{'criterion': 'gini', 'max_depth': 14, 'min_samples_leaf': 2, 'min_samples_split': 3, 'n_estimators': 58}
NaiveBayes	{'var_smoothing': 3.74}
AdaBoost	{'learning_rate': 0.364, 'n_estimators': 199}
XGBoost	{'colsample_bytree': 0.749, 'learning_rate': 0.295, 'max_depth': 5, 'n_estimators': 171, 'reg_alpha': 0.698, 'reg_lambda': 1.468, 'subsample':0.662}
GradientBoosting	Best params: {'learning_rate': 0.294, 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 3, 'n_estimators': 108, 'subsample':0.606}

4.4.3 Neural Network Hyperparameter Optimization

To enhance the performance and generalization of the neural network, Keras Tuner was employed for automatic hyperparameter optimization. Unlike manual selection, Keras

Tuner systematically explores combinations of architectural and training parameters using a randomized search strategy. Key parameters tuned include the number of hidden layers, number of units per layer, dropout rates, and the learning rate. The tuning process evaluated multiple models based on validation accuracy, ultimately selecting the configuration that achieved the best performance on the validation set. This allowed for a data-driven approach to architecture design, improving model robustness without exhaustive manual experimentation.

The results of Keras Tuner hyperparameter optimization are presented in the table below.

Table 5: Optimized Hyperparameters using Keras Tuner

Hyperparameter	**Value**
Number of layers	5
Units per layer	[64, 224, 32, 64, 32]
Dropout per layer	[0.1, 0.2, 0, 0.4, 0]
Input units	64
Input dropout	0.2
Learning rate	0.0001
Batch size	64
Epochs	50

While the optimized configuration achieved promising results during tuning, it demonstrated significant overfitting when evaluated across multiple validation folds. The training accuracy improved rapidly, but the validation performance did not follow the same trajectory, and recall and F1 scores dropped notably in the test sets. This outcome suggests that the more complex, deeper architecture lacked sufficient regularization and failed to generalize effectively. As a result, the tuned model was not adopted for the remaining experiments. Instead, a simpler, more stable architecture was selected based on empirical performance and common deep learning best practices.

4.4.4 Neural Network Architecture

To complement the performance of traditional machine learning models, a feedforward neural network was developed and trained for the churn classification task. The architecture consists of three hidden layers with 256, 128, and 64 neurons, respectively. These layers use the ReLU (Rectified Linear Unit) activation function, which is widely adopted due to its ability to introduce non-linearity while maintaining computational efficiency and avoiding vanishing gradient issues during training.

The output layer consists of a single neuron with a sigmoid activation function, appropriate for binary classification tasks as it outputs a probability score between 0 and 1 indicating the likelihood of churn.

To reduce the risk of overfitting, dropout layers were incorporated after each hidden layer. The dropout rate was set to 0.3 applied after the first hidden layer and 0.2 after the

second, meaning that 30% of the neurons were randomly deactivated during training at each iteration. This technique randomly deactivates a fraction of neurons during each training iteration, encouraging the model to develop more robust internal representations.

The model was compiled using the Adam optimizer, a gradient-based optimization algorithm known for its fast convergence and adaptive learning rates. The learning rate was set to 0.0005, a commonly used starting point that provides a balance between learning speed and model stability. The loss function selected was binary crossentropy, which is the standard choice for binary classification problems as it quantifies the difference between the predicted probabilities and the actual class labels.

Training was performed over a maximum of 50 epochs with a batch size of 64, allowing the model to update its weights after every 64 samples. To prevent unnecessary training beyond the optimal point and reduce overfitting, early stopping was implemented. This mechanism monitored the validation loss and stopped training if no improvement was observed over 8 consecutive epochs (patience=8). This approach helps preserve generalization performance by halting training at the point of convergence.

Overall, the neural network architecture was designed to be moderately deep and regularized, with hyperparameters chosen based on common practices in classification tasks and empirical observations from initial tuning.

4.5 Model Testing

To evaluate the predictive performance of all models used in this study—including traditional classifiers and the neural network—several standard classification metrics were employed. Given the binary nature of the churn prediction task (churned vs. not churned) and the presence of a class imbalance, selecting metrics that offer a balanced view of both classes is essential.

The following performance indicators were used to assess model behavior:

Accuracy

Accuracy represents the proportion of correctly predicted observations (both churned and non-churned) to the total number of predictions. It is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where:

- **TP (True Positives):** correctly predicted churners
- **TN (True Negatives):** correctly predicted non-churners

- **FP (False Positives):** non-churners incorrectly predicted as churners
- **FN (False Negatives):** churners incorrectly predicted as non-churners

While accuracy is a widely used metric, it can be misleading in imbalanced datasets like the one used in this study. For example, if 70% of customers are non-churners, a model that always predicts "no churn" would still achieve 70% accuracy without offering real predictive value.

Precision

Precision measures the correctness of positive (churn) predictions. It is the ratio of true positives to all instances predicted as positive:

$$Precision = \frac{TP}{TP + FP}$$

Recall (Sensitivity or True Positive Rate)

Recall evaluates the model's ability to identify actual churners. It is calculated as:

$$Recall = \frac{TP}{TP + FN}$$

A high recall indicates that most churners were correctly identified, even if it means some false positives occurred. In churn prediction, recall is often considered more critical than precision, as missing actual churners could lead to higher revenue loss.

F1 Score

The F1 score is the harmonic mean of precision and recall. It balances the two metrics and is particularly useful when there is an uneven class distribution:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

A high F1 score indicates that the model achieves both good precision and recall performance, making it a strong indicator of overall classification quality.

ROC-AUC (Receiver Operating Characteristic – Area Under Curve)

The ROC curve plots the True Positive Rate (Recall) against the False Positive Rate across different threshold values. The Area Under the ROC Curve (AUC) provides an aggregate measure of performance across all classification thresholds:

$$AUC = \int_0^1 TPR(FPR), dFPR$$

An AUC score of 0.5 suggests no discriminative power (random guessing), while a score of 1.0 indicates perfect classification. AUC is especially helpful in comparing classifiers' ability to rank instances correctly, independent of a specific decision threshold.

Accuracy and Loss Curve

In addition to standard classification metrics, training and validation curves for loss and accuracy were plotted across epochs to visually assess the learning dynamics and generalization behavior of the neural network. The **accuracy curve** illustrates the proportion of correct predictions on both the training and validation sets over time, providing insight into the model's learning progression. Similarly, the **loss curve** reflects the model's optimization performance, indicating how well it minimizes the binary cross-entropy loss function during training.

These visualizations are essential for detecting signs of overfitting or underfitting. A consistent divergence between training and validation accuracy, or a plateau in validation loss while training loss continues to decrease, often suggests overfitting. In the present experiments, early stopping was implemented to halt training once the validation loss failed to improve for a predefined number of epochs, thereby preserving generalization and preventing excessive model complexity. By monitoring these curves, the training process was not only optimized for performance but also regulated for stability and robustness across all folds.

4.6 Results Analysis

This chapter presents the performance evaluation of all models developed for the customer churn prediction task. After completing the model development phase, each algorithm was assessed using stratified 4-fold cross-validation, ensuring a balanced distribution of churned and non-churned customers across all splits. This allowed for a robust and equitable testing framework in which every sample was used for both training and validation.

In total, five experiments were conducted, each designed to explore the impact of specific methodological choices:

1. Experiment 1 (Baseline): Default training using the original imbalanced dataset without any adjustment.
2. Experiment 2: Class weighting applied to cost-sensitive algorithms to account for the class imbalance.
3. Experiment 3: Increased hyperparameter tuning depth by extending the number of iterations in the randomized search and switching the evaluation metric from F1 score to recall.
4. Experiment 4: SMOTE (Synthetic Minority Over-sampling Technique) was used to synthetically balance the dataset during training.
5. Experiment 5: Default training using the top 20 important features.

For all experiments, the best model parameters were selected through RandomizedSearchCV, as explained in the previous chapter. These tuned hyperparameters were used consistently across folds to evaluate model stability, generalization, and performance consistency.

The evaluation is based on key classification metrics, including Accuracy, Precision, Recall, and F1 Score, with special emphasis placed on Recall and F1 Score due to the imbalanced nature of the dataset. In the following sections, performance visualizations such as bar charts, confusion matrices, and ROC curves are presented to summarize classifier behavior and facilitate comparison across models and experiments.

Experiment 1- Baseline

Below is the first experiment serves as the baseline evaluation for all classifiers, trained and tested on the original imbalanced dataset without applying any sampling or weighting adjustments.

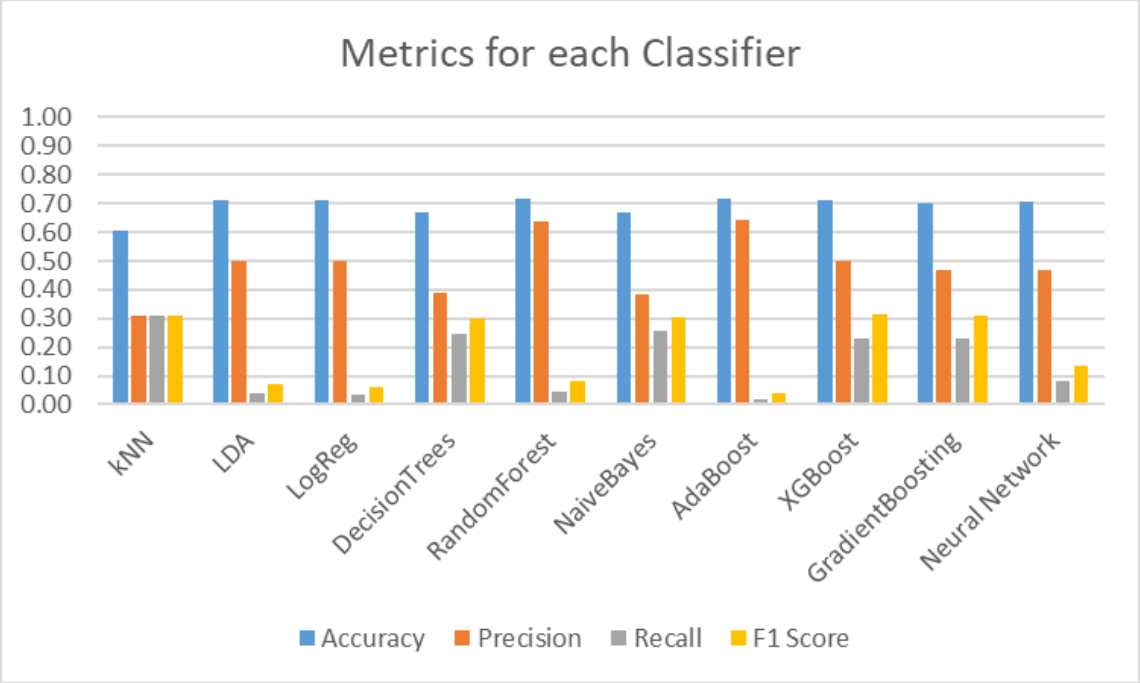

Figure 14: Overall accuracy, precision, recall, f1-score for the first experiment

In this initial experiment, models were trained and tested on the original imbalanced dataset without applying any class balancing strategies. While accuracy values across all classifiers remained relatively consistent—ranging from 60% (kNN) to 72% (Random Forest)—a deeper look at recall and F1 score reveals critical limitations in performance. This pattern suggests that models may be biased toward predicting the majority class (non-churners), a typical issue in imbalanced classification problems.

XGBoost emerged as the top performer in terms of F1 score (0.32) and shared the highest accuracy (71%) alongside Logistic Regression and AdaBoost. Its ability to handle non-linear patterns and regularization capabilities likely contributed to its superior generalization despite the class imbalance. **Gradient Boosting** followed closely with an F1 score of 0.31, reinforcing the strength of ensemble boosting methods in this task.

Interestingly, **k-Nearest Neighbors (kNN)** achieved the highest recall (0.31), indicating it was more successful than others at identifying churners. However, its lower accuracy (0.60) and moderate precision reflect a higher rate of false positives, which may not be acceptable depending on the business objective.

AdaBoost, despite scoring well in terms of accuracy (0.71) and precision (0.64), delivered the lowest F1 score (0.04) and recall (0.02)**—**highlighting its extreme imbalance in predictions. This likely stems from its sequential learning design, which is more sensitive to class imbalance, leading to severe underprediction of the minority class.

A noticeable trend is that accuracy values are misleadingly high across models. Since 71% of the dataset belongs to the non-churn class, models can achieve high accuracy simply by predicting the majority class. However, this comes at the cost of identifying actual churners, as evident in the universally low recall values (except for kNN and the ensemble models).

Naive Bayes also showed poor F1 performance (0.08) due to low recall (0.04) despite a moderate accuracy of 67%. The probabilistic assumptions of this model (e.g., feature independence) may not align well with the complex relationships present in telecom churn data.

These findings clearly illustrate that F1 score and recall provide a more realistic reflection of model effectiveness for churn prediction than accuracy alone. The dominance of ensemble methods like XGBoost and Gradient Boosting further suggests that leveraging multiple weak learners improves sensitivity to minority class patterns. Conversely, linear models and algorithms based on distance or probability assumptions (like Logistic Regression and Naive Bayes) struggle without additional balancing or feature transformation.

To further support the evaluation of classifier performance in the baseline scenario, confusion matrices and ROC-AUC plots were generated for three representative models: XGBoost, k-Nearest Neighbors (kNN), and AdaBoost. These visual tools provide deeper insights into the strengths and weaknesses of each algorithm beyond summary metrics.

XGBoost

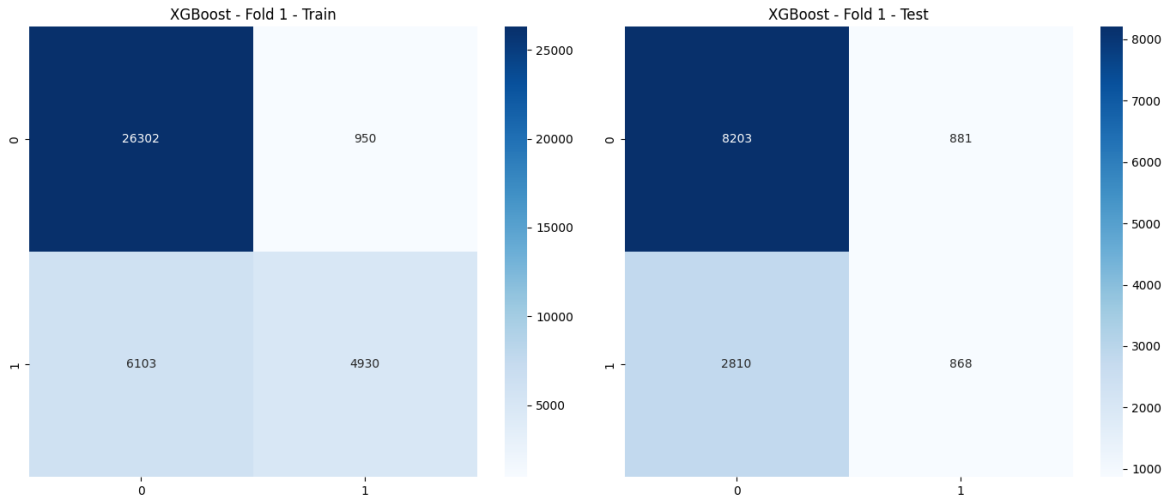

Figure 15: XGBoost Confusion Matrix (Experiment 1)

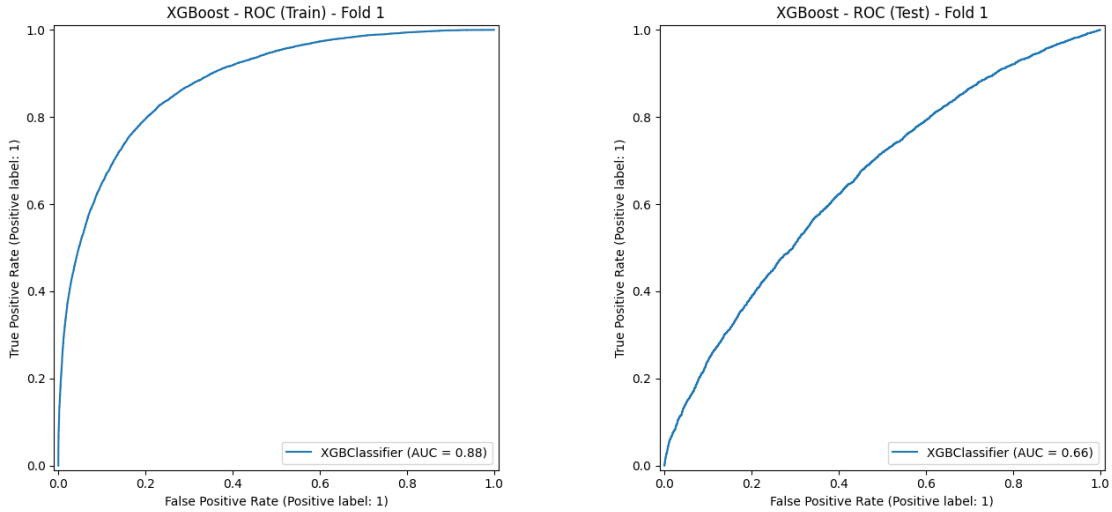

Figure 16: XGBoost ROC Curve (Experiment 1)

XGBoost was selected due to its overall best performance in terms of F1 score, making it a suitable benchmark for both sensitivity and balance in predictions. Its confusion matrix on the test fold shows a reasonably good balance between detecting churners and avoiding false positives, although a significant number of churners were still missed—illustrating the impact of class imbalance. The corresponding ROC curves highlight a strong distinction capability on the training set (AUC = 0.88), but with a noticeable decline

on the test set (AUC = 0.66), suggesting some degree of overfitting. This contrast reinforces the importance of evaluating models not only on in-sample performance but also on their generalization ability.

k-NN

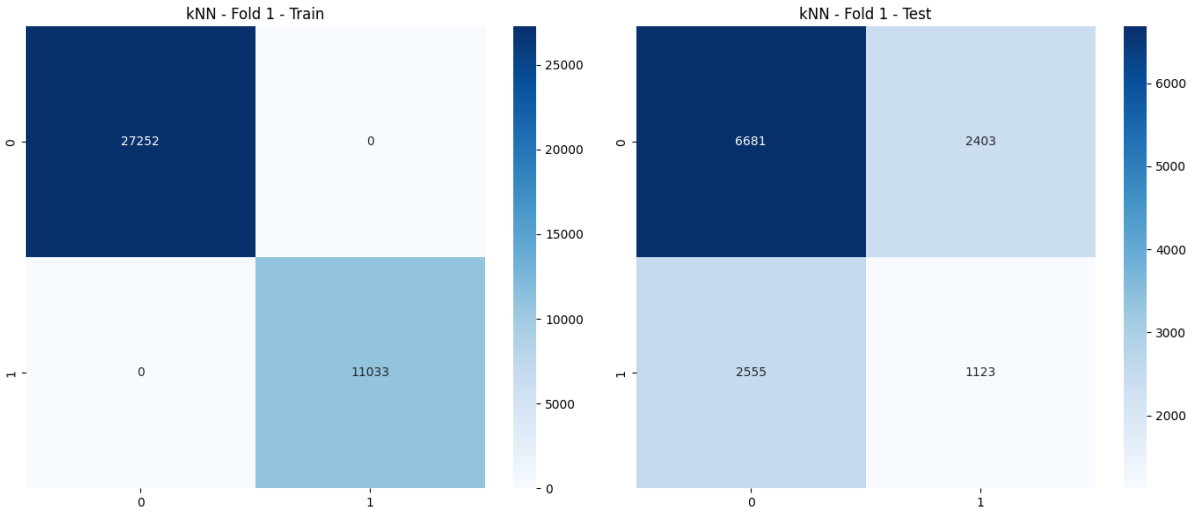

Figure 17:k-NN Confusion Matrix (Experiment 1)

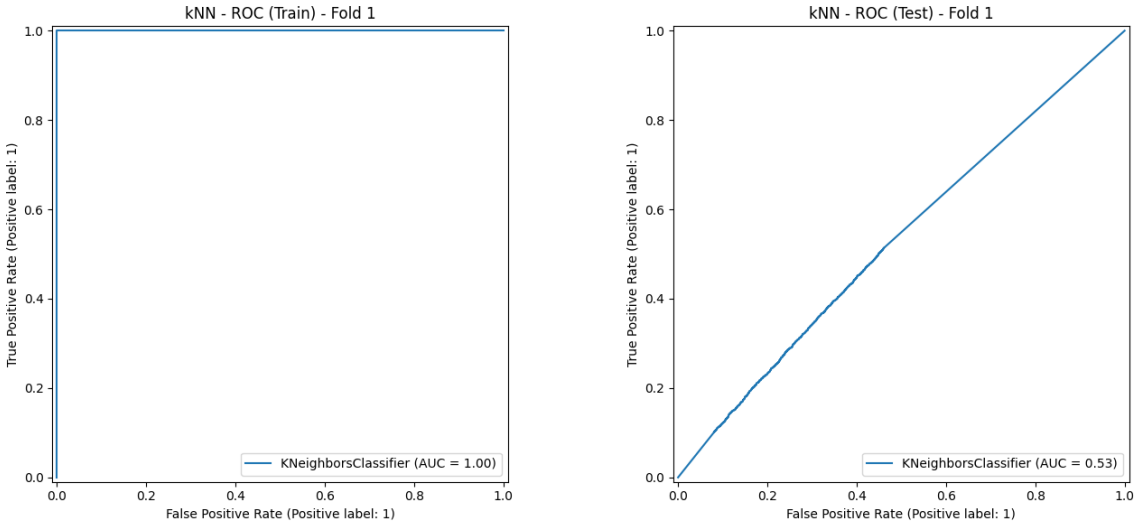

Figure 18: k-NN ROC Curve (Experiment 1)

k-Nearest Neighbors (kNN) was selected for visualization due to its notably high recall, which reflects its relative success in identifying churners. However, this performance came at the cost of precision, as shown by a substantial number of false positives in the test set. The confusion matrix clearly demonstrates this trade-off: while kNN correctly

identified many actual churners, it also misclassified a significant portion of non-churners as churners. The ROC curve further emphasizes the model's instability, with a perfect AUC of 1.00 on the training data—indicative of overfitting—and a sharp drop to 0.53 on the test set, revealing poor generalization. These results highlight kNN's sensitivity to the training data distribution and reinforce the need for caution when deploying recall-heavy models in real-world settings.

AdaBoost

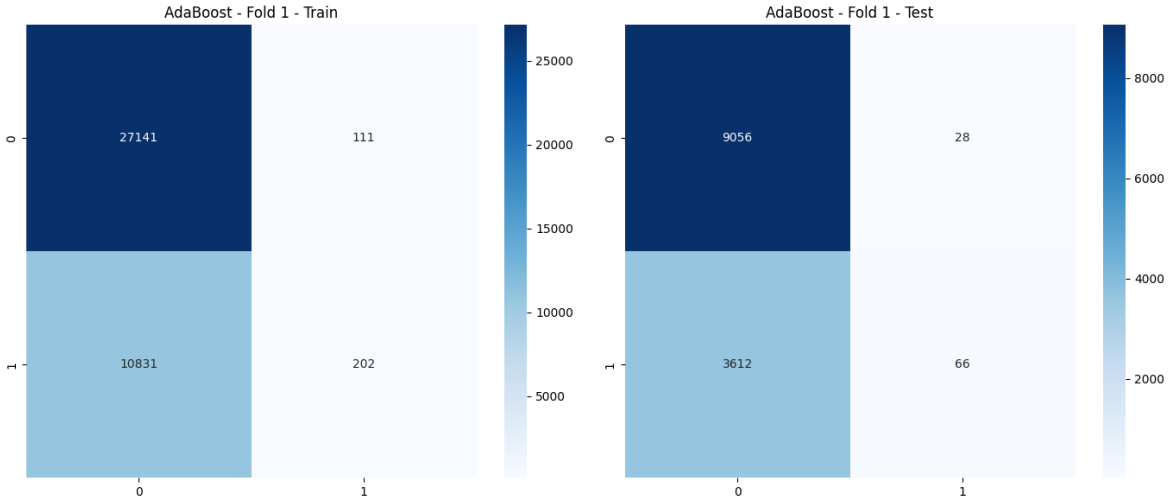

Figure 19: AdaBoost Confusion Matrix (Experiment 1)

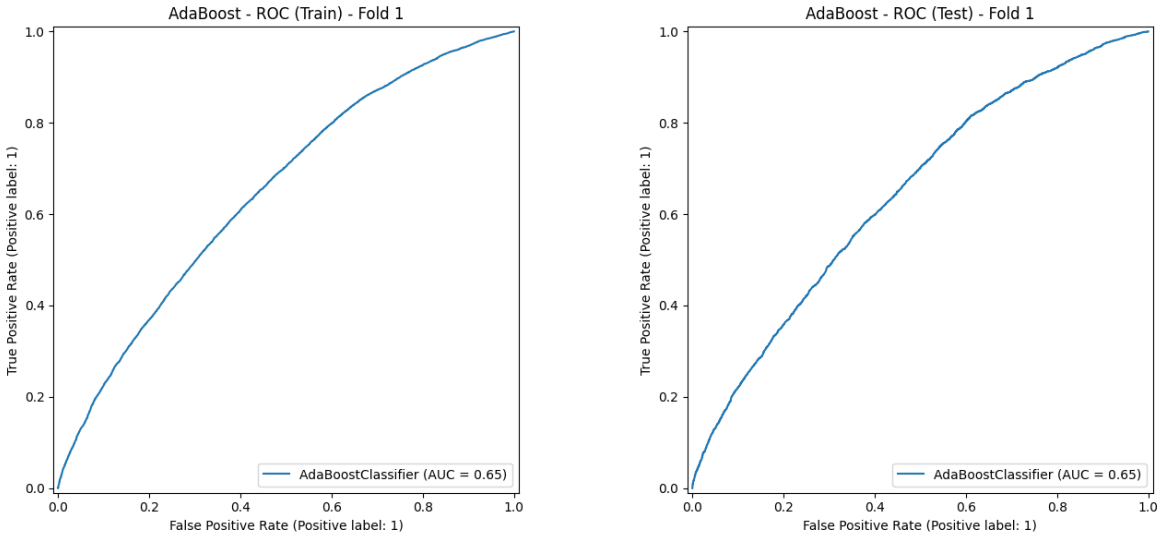

Figure 20: AdaBoost ROC Curve (Experiment 1)

AdaBoost was included to illustrate the limitations of relying solely on accuracy as a performance measure. While the model achieved high accuracy due to correctly classifying the majority of non-churners, its confusion matrix reveals a severe imbalance in predictive behavior. Specifically, it failed to identify the vast majority of actual churners, capturing only a small fraction and misclassifying thousands. This indicates a strong bias

toward the majority class. The accompanying ROC curves support this observation: although both training and testing AUC scores were moderately consistent at 0.65, the model's limited ability to distinguish between churners and non-churners remains evident. These results reinforce that high accuracy in imbalanced settings can be misleading and must be interpreted alongside class-sensitive metrics such as recall and F1 score.

Neural Network

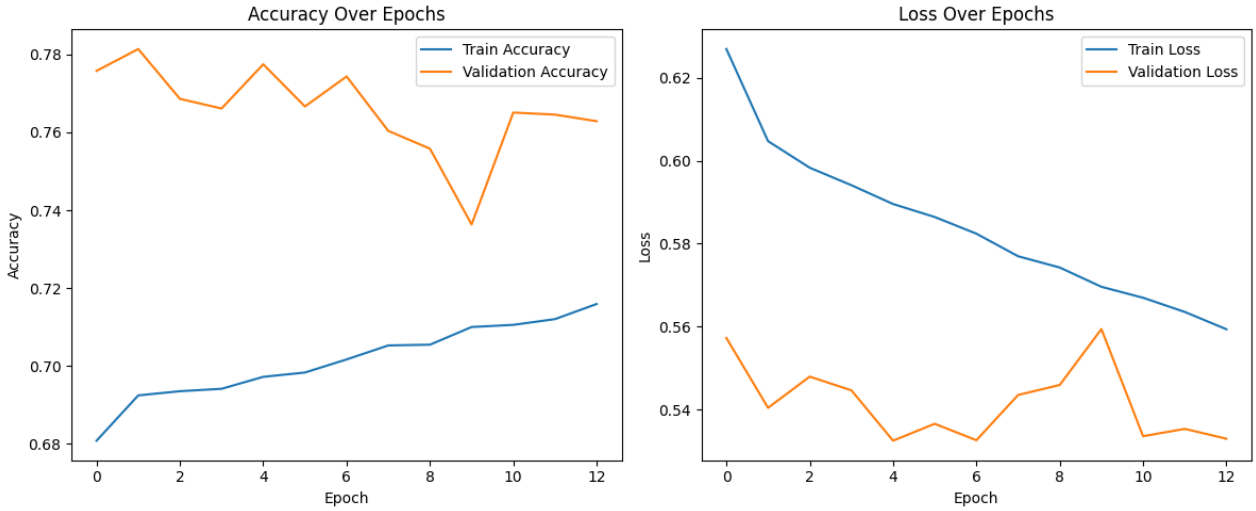

Figure 21:Train and Validation Accuracy/ Loss Curves (Experiment 1)

Lastly, the neural network was included as a deep learning benchmark to compare against traditional classifiers. As reflected in the performance metrics, it achieved moderate accuracy (0.70) but delivered lower recall and F1 scores relative to the best-performing ensemble models. Despite being capable of modeling non-linear relationships, the network struggled to generalize effectively in this imbalanced classification context. This is further supported by the accuracy and loss curves. The training accuracy steadily improved over the epochs, while validation accuracy remained relatively flat, indicating limited generalization gain. Similarly, the training loss decreased consistently, but the validation loss showed oscillations without a clear downward trend—suggesting that the model began to overfit despite the use of dropout layers and batch normalization. Early stopping was employed to prevent excessive overfitting, and training halted after 12 epochs when no consistent improvement in validation loss was observed. These patterns highlight the sensitivity of neural networks to data imbalance and emphasize the importance of architecture tuning and regularization when applying deep models to customer churn prediction.

Experiment 2- Cost-Sensitive Learning

The second experiment applies a cost-sensitive learning approach by integrating class weighting into the model training process. Given the notable imbalance in the dataset—where churners account for approximately 29% of total observations—standard learning algorithms tend to exhibit a bias toward the majority class. To counteract this effect, models that support internal class weighting were configured accordingly. Specifically, classifiers such as Logistic Regression, Decision Trees, Random Forest and the Neural Network were trained using *class_weight='balanced'*. For boosting models like XGBoost the *scale_pos_weight* parameter was employed to simulate cost sensitivity by assigning greater weight to minority class instances during optimization. However, other classifiers—including k-Nearest Neighbors (kNN), Linear Discriminant Analysis (LDA), Naive Bayes, AdaBoost, and Gradient Boosting—were not trained with class weighting, either due to the lack of native support for this feature or the absence of a straightforward mechanism for incorporating it.

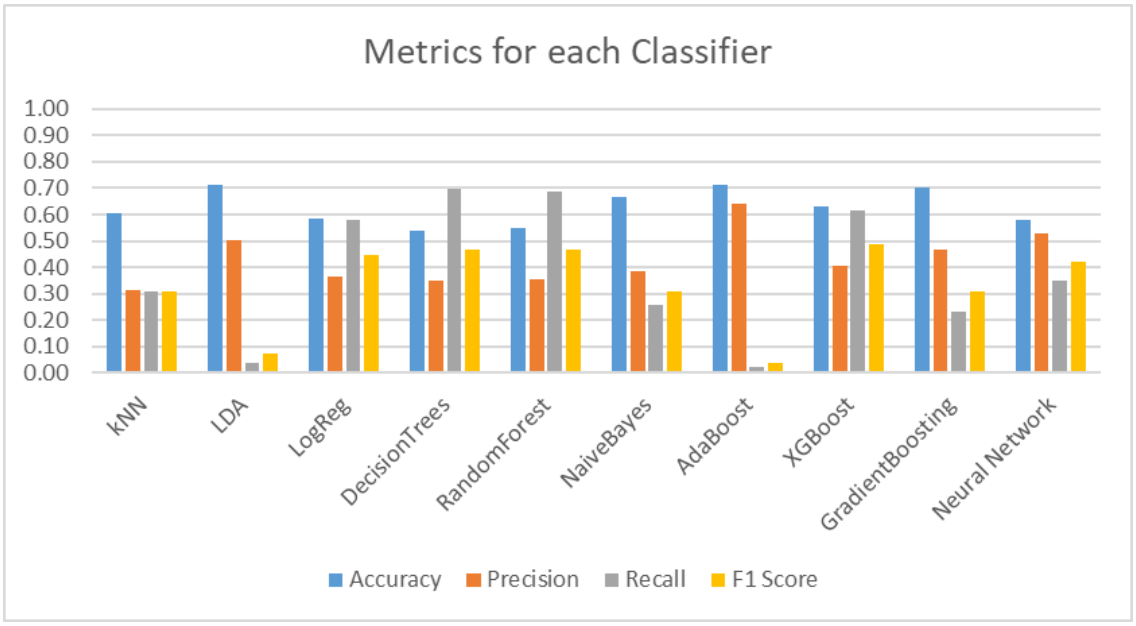

Figure 22: Overall accuracy, precision, recall, f1-score for the second experiment

The results demonstrate a significant improvement in performance metrics sensitive to minority class detection. XGBoost achieved the highest F1 score of 0.49, alongside a strong recall of 0.61, indicating its effectiveness in identifying churners without sacrificing overall balance. Decision Trees and Random Forest also performed well under cost-sensitive conditions, reaching recall values of 0.70 and 0.69 respectively, albeit with moderate precision. Logistic Regression, which previously struggled in recall, showed a marked improvement with an F1 score of 0.45. The Neural Network also benefited from cost-sensitive training, showing improved recall and F1 performance compared to its baseline results. These findings affirm that cost-sensitive learning strategies improve the

model's ability to recognize churners, a critical requirement in churn prediction tasks, especially when class imbalance is substantial.

XGBoost

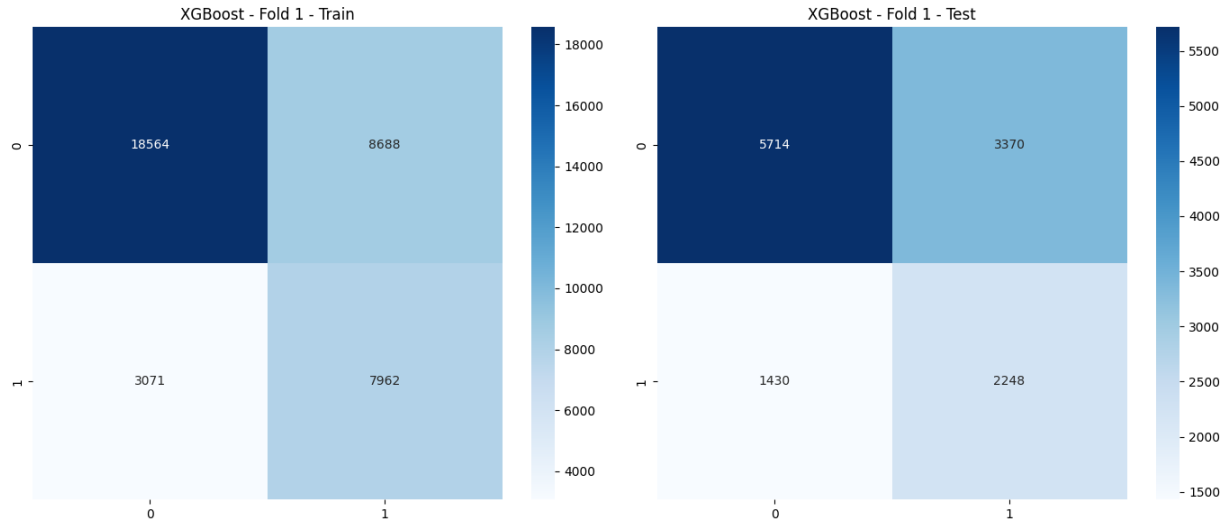

Figure 23: XGBoost Confusion Matrix (Experiment 2)

The confusion matrix from Experiment 2 demonstrates a significant improvement in churn detection (true positives), with recall more than doubling compared to the default XGBoost. While this came at the cost of an increased false positive rate (and thus lower precision), the overall F1 score improved considerably. This trade-off is typical and acceptable in churn prediction tasks, where catching more churners is often more valuable than maintaining perfect precision. The cost-sensitive version of XGBoost is clearly better suited to the imbalanced nature of the dataset.

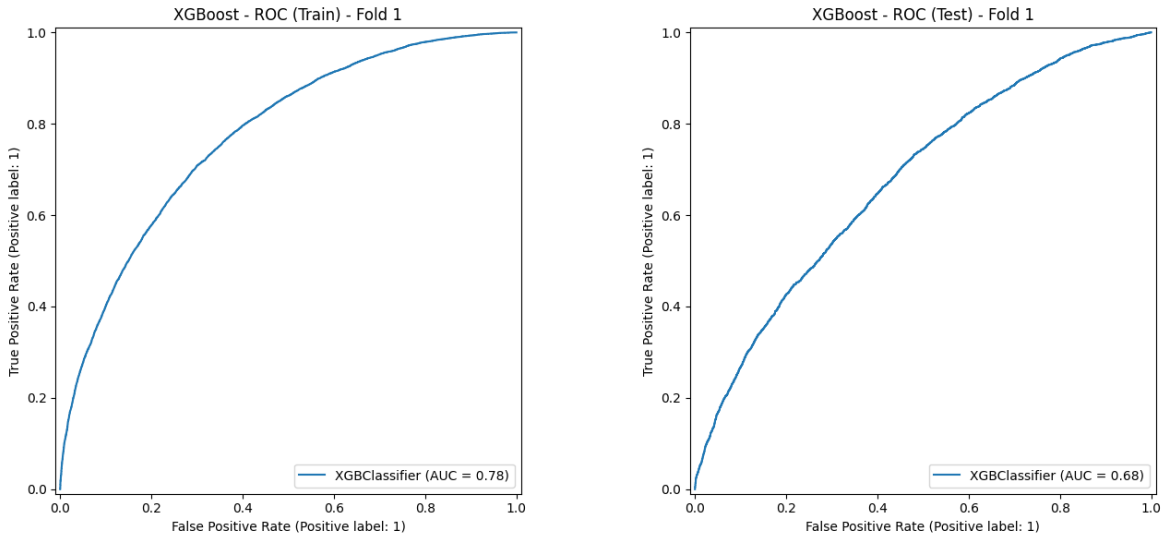

Figure 24: XGBoost ROC Curve (Experiment 2)

In Experiment 2, the XGBoost ROC curves indicate better generalization and less overfitting compared to the default model. While the AUC on the training set decreased (from 0.88 to 0.78), this is expected and actually desirable — it suggests the model is no longer overfitting to the majority class. The test AUC increased slightly from 0.66 to 0.68, which, combined with the significant boost in recall, confirms that class weighting helped the model become more sensitive to the minority class while maintaining its ability to distinguish between classes across thresholds.

Neural Network

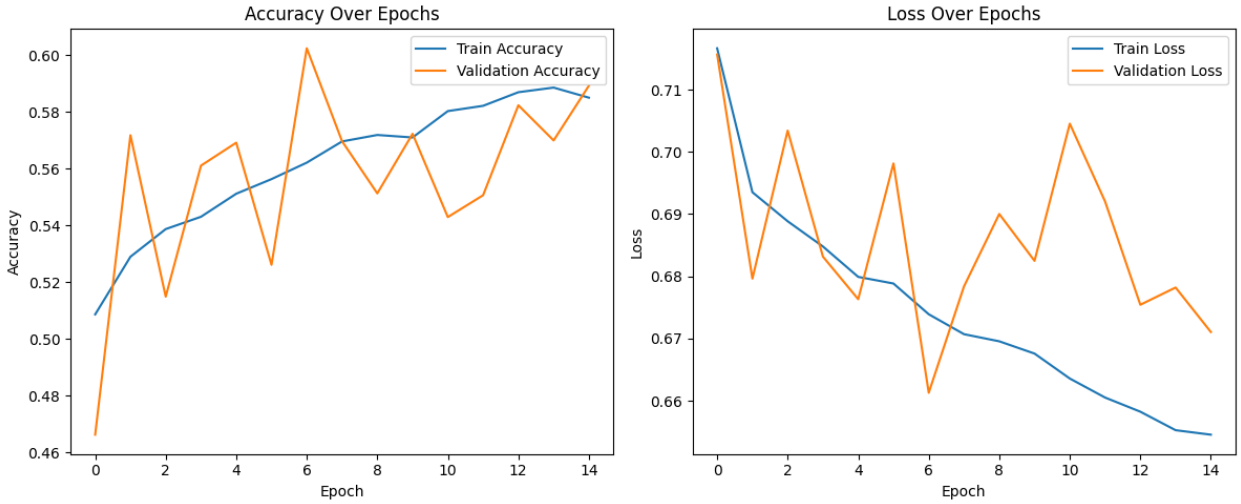

Figure 25: Train and Validation Accuracy/ Loss Curves (Experiment2)

The learning curves for the baseline neural network (Experiment 1) and its class-weighted counterpart (Experiment 2) reveal important differences in performance behavior under different training objectives. In Experiment 1, the model achieved higher accuracy on both the training and validation sets, with the validation accuracy peaking near 78%. However, this performance came at the cost of low recall, meaning the model prioritized overall accuracy—driven largely by correctly predicting the dominant (non-churn) class—while failing to adequately detect churners.

In contrast, the model in Experiment 2 was trained with class weighting, shifting the learning emphasis toward the minority class. This modification led to lower overall accuracy (train and validation accuracy capped below 61%) and higher loss values, particularly on the training set. These outcomes are expected and appropriate: in optimizing for a better balance between false positives and false negatives, the model sacrificed raw accuracy to increase true positive rates for churners.

The validation loss curve in Experiment 2 shows greater fluctuation, a result of the increased challenge in optimizing a cost-sensitive objective under class imbalance. Nevertheless, the training trajectory remains stable, and early stopping occurred without signs of overfitting.

Experiment 3- Recall-Based Scoring

The third experiment explores the impact of optimizing models specifically for recall, rather than relying on default performance metrics such as accuracy or F1 score. This shift was motivated by the nature of the churn prediction problem, where correctly identifying churners—the minority class—is of primary importance. In this context, recall serves as a more meaningful evaluation criterion, as it directly measures the proportion of actual churners successfully detected by the model.

To implement this, all traditional machine learning classifiers were re-tuned using RandomizedSearchCV with scoring='recall', ensuring that hyperparameter selection explicitly prioritized sensitivity to the positive class. As this approach relies on scikit-learn's model selection interface, the neural network—implemented separately without RandomizedSearchCV—was excluded from this experiment. The goal of this experiment is to assess whether recall-centric optimization leads to a tangible improvement in minority class detection, and how it compares to previous strategies such as cost-sensitive learning.

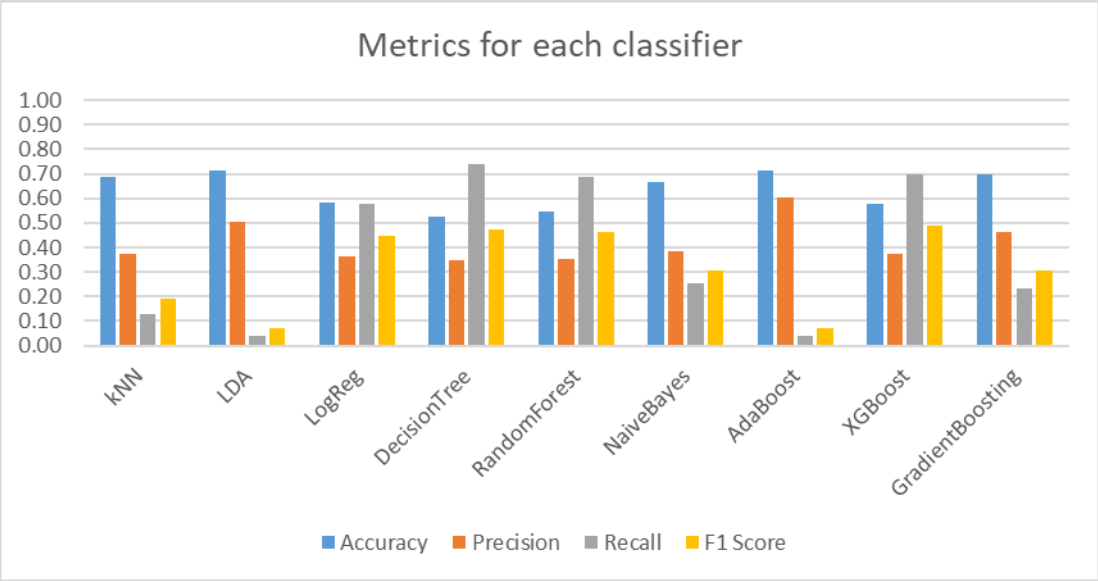

Figure 26: Overall accuracy, precision, recall, f1-score for the third experiment

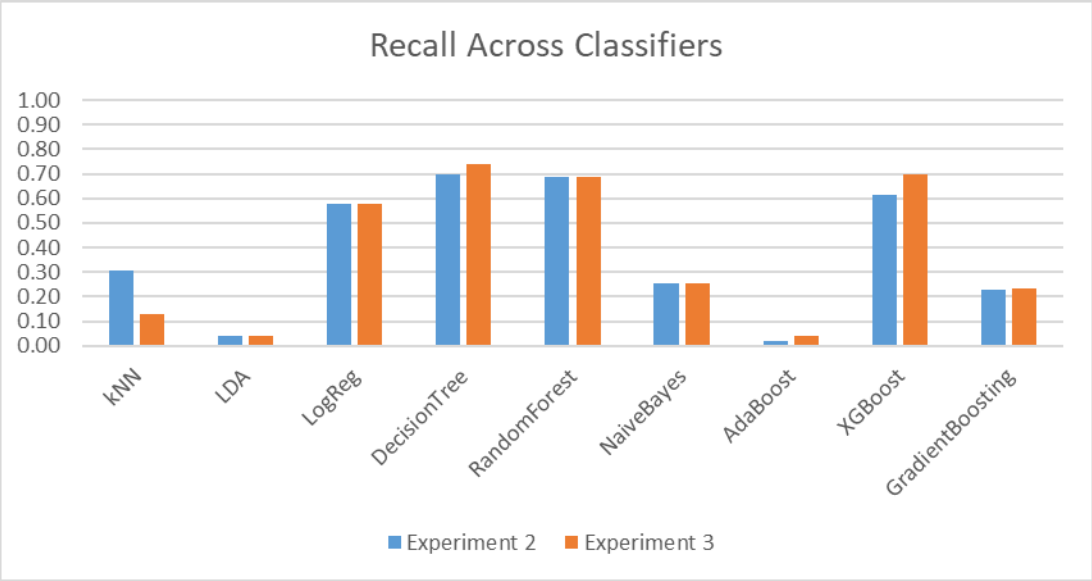

Figure 27: Recall comparison between Experiment 2 and Experiment 3

Figures 26 and 27 present the results of the third experiment, where all classifiers (excluding the neural network) were trained using cost-sensitive learning and tuned specifically with scoring='recall' as the optimization objective. The intention was to further increase sensitivity to churners by explicitly favoring recall during hyperparameter selection.

Surprisingly, as shown in the first plot, recall did not universally improve across all classifiers compared to Experiment 2. While models such as Decision Tree and XGBoost demonstrated slight improvements or maintained already strong recall levels (above 0.7), others—like kNN, LDA, and even Logistic Regression—saw either no meaningful gain or slight degradation in recall performance. This outcome, although counterintuitive at first, is possible due to the stochastic nature of hyperparameter optimization, interactions with class weighting, and how recall behaves in imbalanced settings. Optimizing for recall during cross-validation may occasionally result in hyperparameters that generalize poorly, or overfit to recall on the validation split but not on the test fold. In some models, especially those with fewer degrees of freedom (like LDA or Naive Bayes), the influence of hyperparameter tuning is inherently limited.

The first plot, which shows the full set of performance metrics for each classifier in Experiment 3, supports this interpretation. It is evident that in some cases—such as AdaBoost—recall increased only marginally, but at the cost of significantly reduced precision, leading to low F1 scores. These shifts highlight the precision-recall trade-off that emerges when recall is optimized in isolation. Ultimately, this experiment reinforces that while recall-oriented optimization aligns well with churn prediction goals, its benefits vary depending on model complexity, sensitivity to hyperparameters, and interaction with cost-sensitive training mechanisms.

Experiment 4- SMOTE

In this fourth experiment, the Synthetic Minority Over-sampling Technique (SMOTE) was applied across all classifiers to address class imbalance at the data level. Rather than adjusting model behavior through class weighting or scoring functions, SMOTE balances the dataset by synthetically generating new instances of the minority class (churners) based on feature similarities. This method ensures that all classifiers—traditional and neural network alike—train on a more balanced distribution, potentially improving their ability to detect churners. SMOTE was applied exclusively to the training folds during cross-validation to avoid information leakage, while test sets maintained their original imbalanced distribution. The goal of this experiment is to assess whether oversampling enhances recall and F1 score, particularly for models that previously underperformed in minority class detection.

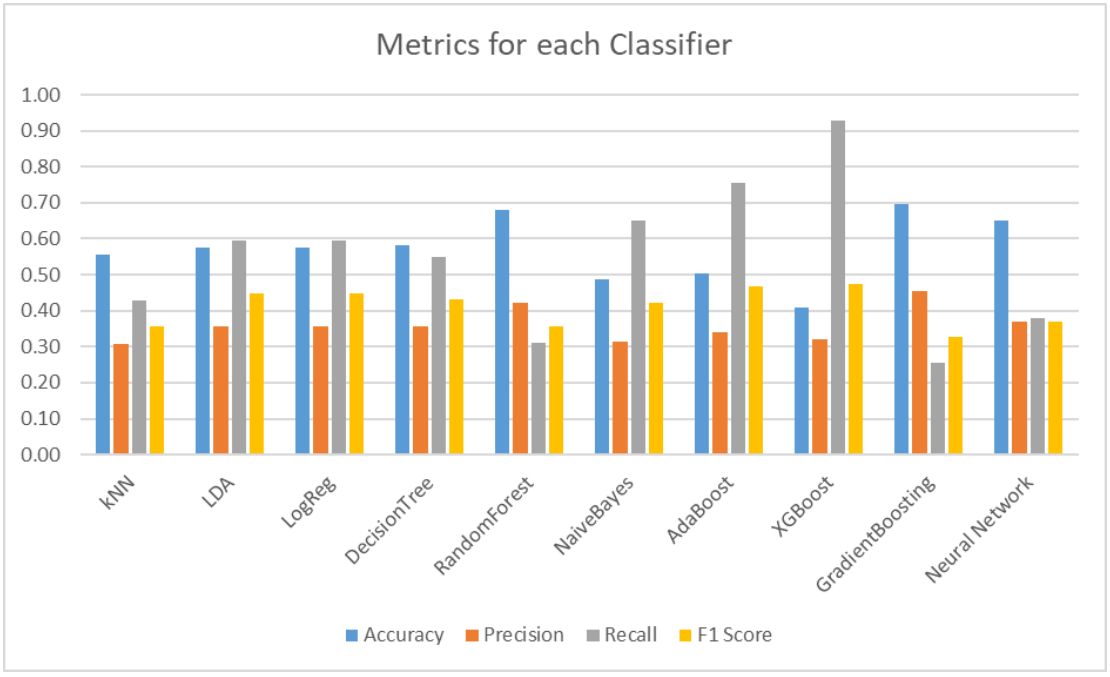

Figure 28: Overall accuracy, precision, recall, f1-score for the fourth experiment

The performance metrics in Figure X show that SMOTE produced notable improvements in recall across several classifiers, with XGBoost and AdaBoost demonstrating especially strong gains—both exceeding 0.70 in recall. These values mark a substantial improvement over Experiments 1–3, where recall values for these models were generally lower, even under cost-sensitive learning. Gradient Boosting also saw a meaningful recall increase, though it lagged slightly behind its ensemble counterparts.

Interestingly, classifiers such as Logistic Regression, LDA, and k-NN achieved a more balanced trade-off between precision and recall after SMOTE, resulting in stronger F1 scores. This indicates that SMOTE helped these models better generalize across both classes, not just improving sensitivity. The neural network, which previously showed

instability under class weighting, demonstrated more stable and moderate performance in this experiment, suggesting that oversampling may provide a more effective strategy than class weighting in this context.

However, it's worth noting that accuracy generally declined across most models—this is expected with SMOTE, as the synthetic minority class samples lead to more true positives but also increase false positives, especially on the imbalanced test set. This shift emphasizes the importance of relying on recall and F1 score over accuracy when evaluating performance on imbalanced datasets.

Overall, this experiment reinforces that SMOTE is a powerful and broadly applicable strategy for improving minority class detection across a wide range of classifiers, particularly when the goal is to maximize recall and ensure fair representation of churners during training.

Neural Network

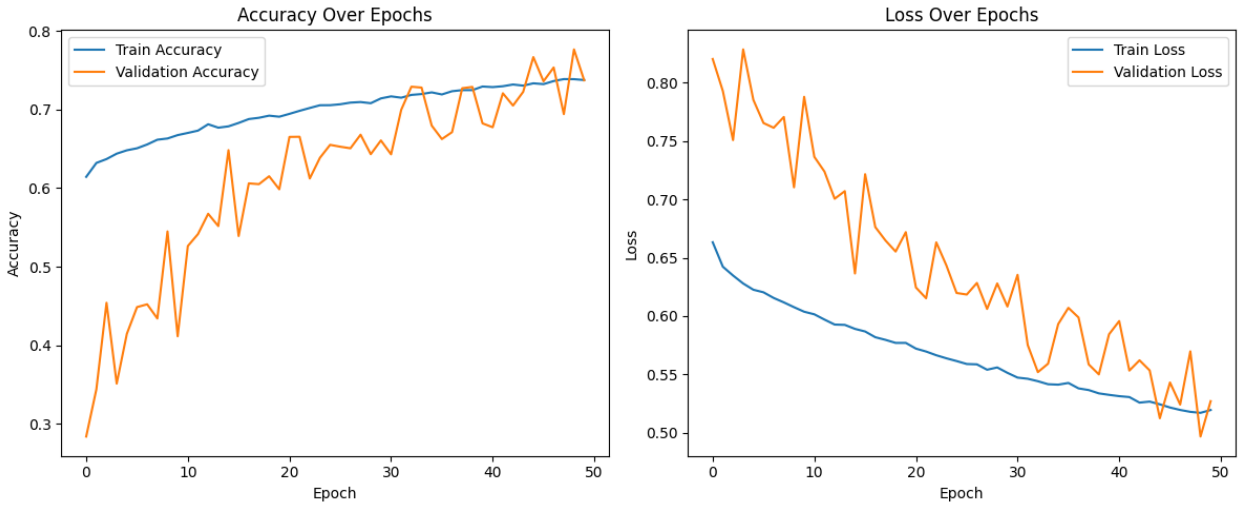

Figure 29: Train and Validation Accuracy/Loss Curves (Experiment 4)

Figure 29 presents the training and validation curves of the neural network under SMOTE augmentation. The model demonstrates consistent learning progression, with both training and validation accuracy steadily increasing throughout the epochs. Although validation loss exhibits noticeable fluctuations—an expected consequence of synthetic sample variability—the overall trajectory does not suggest overfitting, as the gap between training and validation metrics remains controlled. The impact of SMOTE is further evidenced in the performance metrics: recall improved significantly, indicating enhanced sensitivity to the minority (churn) class. However, this improvement came at a modest cost to the F1 score, which experienced a slight decline due to reduced precision. This trade-off underscores a typical effect of oversampling, where the model becomes more capable of identifying positive cases but may also generate a higher number of false

positives. Overall, the SMOTE-based training approach achieved its intended goal of boosting recall while maintaining acceptable balance across performance metrics.

Experiment 5- Feature importance

In the fifth and final experiment, the focus shifted from rebalancing techniques and hyperparameter tuning to dimensionality reduction through feature selection. Specifically, the top 20 most significant features identified using a Random Forest classifier were used to retrain all models under the same default conditions as the baseline experiment. This approach aimed to evaluate whether removing less relevant or redundant features could enhance model performance by simplifying the input space, reducing noise, and potentially increasing generalization capability without altering the class imbalance or other training parameters.

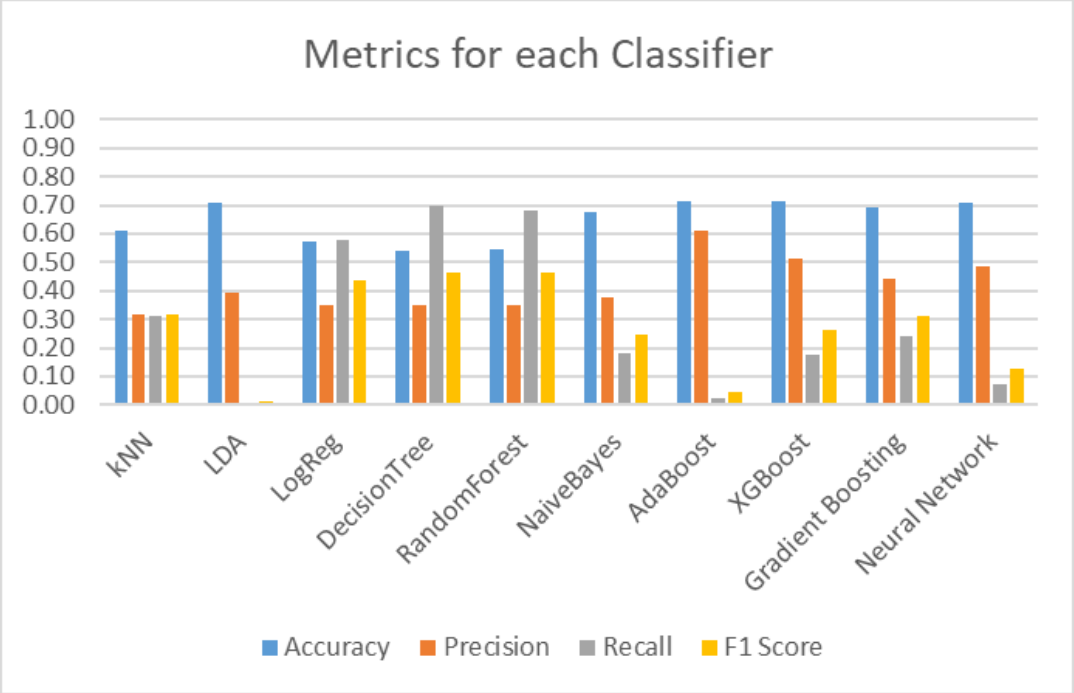

Figure 30: Overall accuracy, precision, recall, f1-score for the fifth experiment

The results of this experiment show that using only the top 20 most informative features had mixed effects on classifier performance. While overall accuracy remained largely similar compared to the baseline, some classifiers demonstrated slight improvements in precision and F1 score, indicating that reducing input dimensionality helped focus the learning process on the most predictive signals. When comparing with the Experiment 1 simpler models like Decision Trees and Random Forests showed notable improvements in F1 scores, highlighting that these classifiers benefit from more focused input features by reducing noise. Logistic Regression also recorded an increase in both recall and F1 score, suggesting that removing less relevant variables strengthened its predictive power for churn detection. In contrast, LDA experienced a decline in both recall and F1 score,

indicating that its performance may rely on the complete feature space to capture discriminative variance. XGBoost, which consistently performed strongly in other experiments, showed a slight drop in F1 score with feature reduction, implying that its boosting mechanism effectively handles large feature sets without needing aggressive dimensionality cuts. Overall, the fifth experiment demonstrates that careful feature selection can improve the efficiency and effectiveness of certain models, but its benefit is not universal and must be balanced with each algorithm's learning strategy.

Accuracy Across All Experiments

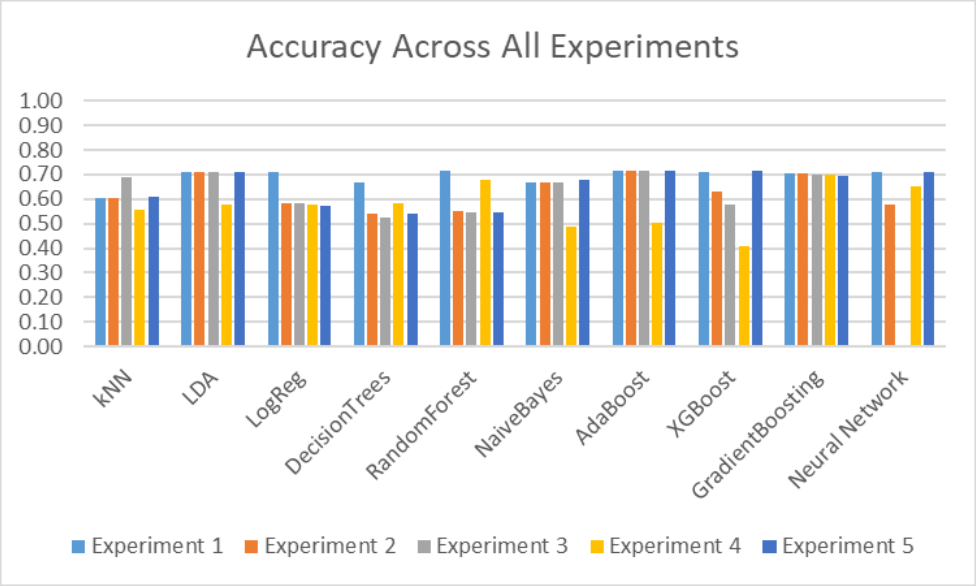

Figure 29: Classifier Accuracy Trends Across All Experimental Setups

The bar chart above presents the evolution of accuracy across all four experimental setups for each classifier. As expected, classifiers generally achieved the highest accuracy in the baseline experiments (Experiment 1,5), where no special measures were taken to address class imbalance. However, as cost-sensitive learning and SMOTE were introduced in subsequent experiments, slight declines in accuracy were observed for several models, including Logistic Regression, Decision Trees, Random Forest, XGBoost, Neural Network. This reduction reflects a shift in model behavior toward better recall at the expense of precision and overall accuracy. Notably, ensemble models maintained relatively stable accuracy, suggesting they adapt well to varying training strategies while preserving generalization performance.

Precision Across All Experiments

While recall is crucial for identifying churners, precision plays an equally important role in ensuring that the positive predictions made by the model are correct. Precision is defined as the proportion of true positives out of all predicted positives, meaning it quantifies the model's ability to avoid false alarms. A lower precision, as observed in later experiments, indicates that many customers flagged as churners were in fact loyal—a costly outcome in real-world applications. This can lead to inefficient resource allocation, where retention efforts (e.g., discounts, outreach) are spent on users who were unlikely to churn. Therefore, while boosting recall helps capture more actual churners, it must be balanced with acceptable precision levels to maintain operational and financial efficiency. This highlights the importance of considering F1 score as a harmonized metric, especially in domains where both false positives and false negatives carry significant consequences.

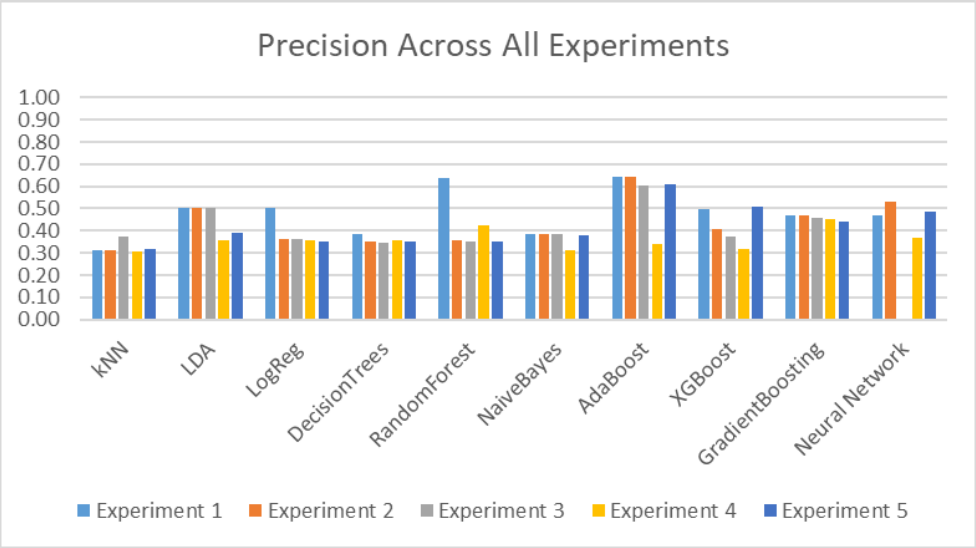

Figure 30: Classifier Precision Trends Across All Experimental Setups

This plot illustrates how precision values evolved for each classifier throughout the four experimental settings. A consistent trend is the gradual reduction in precision as recall-oriented strategies were applied, particularly evident in Random Forest, AdaBoost, and XGBoost. This trade-off reflects the models' shift toward capturing more true positives at the expense of increasing false positives. Notably, classifiers such as k-Nearest Neighbors (kNN), Decision Trees, Logistic Regression, and Naive Bayes consistently exhibited low precision across all experiments, suggesting a tendency to overpredict the minority class. This behavior results in a high number of false positives, which may compromise the reliability of churn identification in real-world applications. Meanwhile, AdaBoost maintained high precision in the baseline scenario but experienced a sharp

decline under cost-sensitive and resampling conditions, highlighting its sensitivity to shifts in class distribution and optimization focus.

Recall Across All Experiments

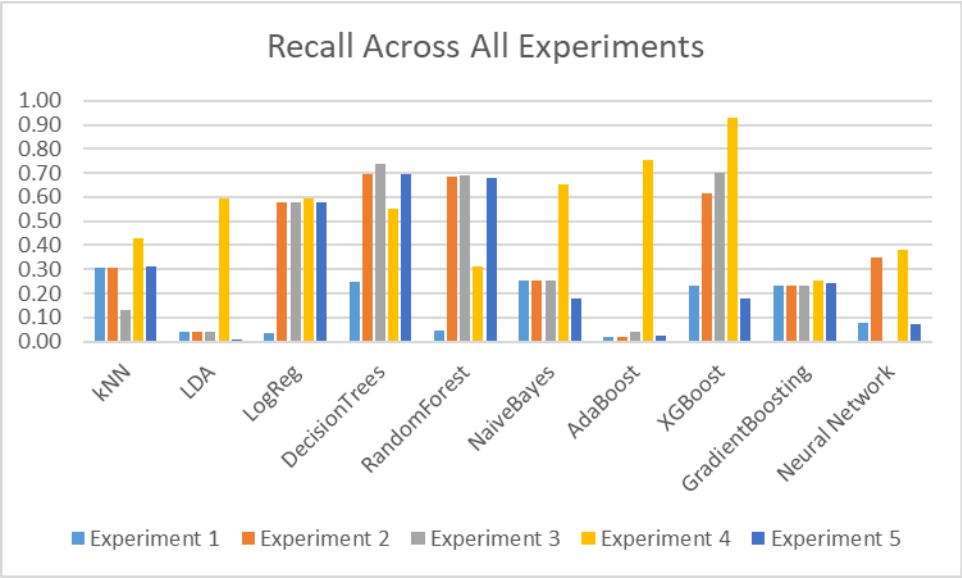

Figure 31: Classifier Recall Trends Across All Experimental Setups

This plot highlights the evolution of recall scores across all four experiments, underscoring the progressive impact of class imbalance mitigation techniques. Recall, a critical metric for churn prediction, reflects a model's ability to correctly identify churners (true positives). Most classifiers exhibit clear gains from the second experiment onward, where class weighting was introduced. Decision Trees and Random Forest, for instance, show stable and strong recall improvements, particularly in Experiment 3, where recall scoring further optimized their sensitivity to the minority class. XGBoost demonstrates a remarkable trajectory, culminating in the highest recall (above 90%) during the SMOTE-enhanced Experiment 4. This indicates its strong adaptability to both cost-sensitive learning and synthetic resampling. Notably, models like LDA, AdaBoost and the Neural Network also benefits from SMOTE in the final experiment, with a substantial recall increase compared to prior experiments. Overall, the recall progression across experiments confirms the effectiveness of targeted optimization strategies in improving the model's ability to detect churn, a vital objective in imbalanced classification tasks.

F1 Score Across All Experiments

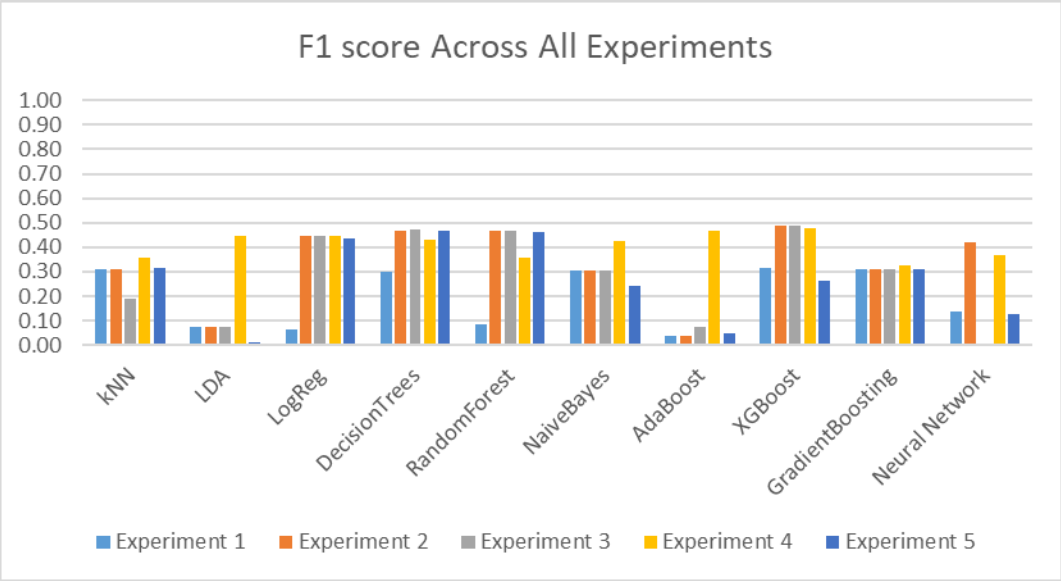

Figure 32: Classifier F1 Score Trends Across All Experimental Setups

The F1 score, which balances precision and recall, is a critical metric for evaluating classifier performance in imbalanced classification tasks such as churn prediction. Among all classifiers, XGBoost consistently achieved the highest F1 scores, reaching 0.49 in both Experiments 2 and 3, and maintaining a competitive 0.48 in Experiment 4, confirming its robustness under cost-sensitive learning and SMOTE. Decision Trees also performed strongly across all experiments, particularly under recall-focused and resampling strategies. While Gradient Boosting showed steady performance, it did not surpass the more adaptable tree-based models in this metric. AdaBoost, although underperforming in the earlier experiments, exhibited a surprising improvement in Experiment 4, where it nearly reached the top with an F1 score of 0.47, indicating that it can benefit substantially from synthetic oversampling. Logistic Regression, Random Forest, and Naive Bayes demonstrated consistent improvements in F1 score after the baseline, validating the positive impact of class weighting and resampling. Overall, this trend highlights how different balancing techniques influence the interplay between precision and recall, with ensemble methods like XGBoost and Decision Trees emerging as particularly effective for balanced churn classification.

5. Conclusions

This thesis explored the application and comparative evaluation of various machine learning models for predicting customer churn in the telecommunications sector. Recognizing the challenges posed by class imbalance in churn prediction—where churners typically represent a minority of the customer base—the study systematically investigated how different modeling techniques, evaluation strategies, and data preprocessing methods affect classification performance.

Initially, baseline models trained on imbalanced data highlighted the limitations of relying solely on accuracy as a performance metric. While many classifiers achieved high accuracy due to the dominance of the non-churn class, their recall and F1 scores were substantially lower, reflecting poor performance in correctly identifying churners. Ensemble methods, particularly XGBoost, demonstrated comparatively stronger generalization, achieving the highest F1 scores in the baseline setting.

The second experimental phase introduced cost-sensitive learning by applying class weighting techniques to penalize the misclassification of minority class instances. This adjustment resulted in a notable improvement in minority class recall and overall F1 scores for models supporting class weighting. XGBoost, Decision Trees, and Random Forests showed strong gains, confirming the effectiveness of this approach. The neural network also benefited from this strategy, exhibiting improved sensitivity to churners without excessive overfitting.

In the third experiment, hyperparameter optimization was guided by recall scoring**,** rather than F1, to further prioritize the correct classification of churners. Although this method was expected to boost recall, the results varied across classifiers, with some failing to improve due to model-specific characteristics or optimization constraints. Nonetheless, several models maintained or slightly improved their recall while preserving acceptable F1 performance, underlining the trade-offs between sensitivity and precision in imbalanced settings.

In the fourth experiment employed SMOTE (Synthetic Minority Oversampling Technique) to synthetically rebalance the training dataset. This resampling strategy further enhanced the minority class representation, leading to recall improvements in nearly all models. However, this came at the cost of reduced precision and, in some cases, a slight drop in F1 scores—an expected trade-off in synthetic oversampling approaches. Notably, AdaBoost, which had consistently underperformed in previous experiments, showed a marked improvement under SMOTE, nearly matching top-performing classifiers in F1 score.Throughout all experiments, XGBoost consistently emerged as the top-performing model, demonstrating robustness and adaptability to various balancing and optimization techniques. Decision Trees and Random Forests also proved effective, particularly under cost-sensitive and SMOTE conditions. In contrast, simpler models such as k-Nearest Neighbors and Naive Bayes struggled to balance recall and precision, even with support from balancing techniques.

In the fifth and final experiment, the focus shifted to evaluating the impact of dimensionality reduction by retraining the baseline models using only the top 20 most important features, as determined by a Random Forest feature importance analysis. This approach demonstrated that selective feature input can benefit certain models, with Decision Trees, Random Forest, and Logistic Regression all showing higher F1 scores compared to their baseline counterparts. This indicates that reducing less relevant variables can help some algorithms focus on stronger predictive patterns. However, models like LDA and XGBoost did not experience similar gains, highlighting that robust ensemble methods or linear discriminants may rely on the full feature space to maintain optimal performance. Overall, the fifth experiment reinforced the importance of thoughtful feature selection, confirming that while it can enhance interpretability and efficiency, its impact depends on the specific learning characteristics of each classifier.

Final Reflections

This research underscores the importance of selecting appropriate evaluation metrics—especially recall and F1 score—when dealing with imbalanced classification problems such as churn prediction. It also highlights the critical role of preprocessing techniques, model choice, and metric-aligned optimization in enhancing predictive performance. No single method universally outperforms in all contexts; rather, the most effective approach often lies in tailoring model configurations to the problem's unique challenges. Future work could explore hybrid models, ensemble stacking, or cost-sensitive deep learning to further advance performance in real-world churn scenarios.

6. References

1. Bhatnagar, A., Srivastava, S. Customer Churn Prediction: A Machine Learning Approach with Data Balancing for Telecom Industry (2025) *International Journal of Computing*
2. Kirgiz, O.B., Kiygi-Calli, M., Cagliyor, S., El Oraiby, M. "Assessing the effectiveness of OTT services, branded apps, and gamified loyalty giveaways on mobile customer churn in the telecom industry: A machine-learning approach" *Telecommunications Policy* 2024
3. Ong, J.-X., Tong, G.-K., Khor, K.-C., Haw, S.-C. "Enhancing Customer Churn Prediction With Resampling: A Comparative Study" *TEM Journal* 2024
4. Abbasimehr, H., Bahrini, A. "Improving churn prediction using imperialist competitive algorithm for feature selection in telecom" *International Journal of Business Information Systems* 2024

5. Lukita, C., Bakti, L. D., Rusilowati, U., Sutarman, A., & Rahardja, U. (2023). Predictive and analytics using data mining and machine learning for customer churn prediction. *Journal of Applied Data Sciences*, *4*(4), 454-465.
6. Fareniuk, Y., Zatonatska, T., Dluhopolskyi, O., Kovalenko, O. "CUSTOMER CHURN PREDICTION MODEL: A CASE OF THE TELECOMMUNICATION MARKET" *ECONOMICS - Innovative and Economics Research Journal* 2022
7. Tariq, M.U., Babar, M., Poulin, M., Khattak, A.S. "Distributed model for customer churn prediction using convolutional neural network" *Journal of Modelling in Management* 2022
8. Shabankareh, M.J., Shabankareh, M.A., Nazarian, A., Ranjbaran, A., Seyyedamiri, N. "A Stacking-Based Data Mining Solution to Customer Churn Prediction" *Journal of Relationship Marketing* 2022
9. Vo, N.N.Y., Liu, S., Li, X., Xu, G. "Leveraging unstructured call log data for customer churn prediction" *Knowledge-Based Systems* 2021
10. Rani, B., Kant, S. "Semi-Supervised Learning Approach to Improve Machine Learning Algorithms for Churn Analysis in Telecommunication" *International Journal of Computer Information Systems and Industrial Management Applications* 2020
11. Khan, Y., Shafiq, S., Naeem, A., Hussain, S., Ahmed, S., Safwan, N. Customers churn prediction using Artificial Neural Networks (ANN) in telecom industry (2019) *International Journal of Advanced Computer Science and Applications*
12. Alzubaidi, A.M.N., Al-Shamery, E.S. "Churners prediction based on mining the content of social network taxonomy" *International Journal of Recent Technology and Engineering* 2019
13. Pamina, J., Dhiliphan Rajkumar, T., Kiruthika, S., Suganya, T., Femila, F. "Exploring hybrid and ensemble models for customer churn prediction in telecom sector" *International Journal of Recent Technology and Engineering* 2019
14. Coussement, K., Lessmann, S., Verstraeten, G. "A comparative analysis of data preparation algorithms for customer churn prediction: A case study in the telecommunication industry" *Decision Support Systems* 2017
15. Mamčenko, J., Gasimov, J. "Customer churn prediction in mobile operator using combined model" *ICEIS 2014 - Proceedings of the 16th International Conference on Enterprise Information Systems* 2014
16. Abbasimehr, H., Setak, M., Soroor, J. "A framework for identification of high-value customers by including social network based variables for churn prediction using neuro-fuzzy techniques" *International Journal of Production Research* 2013
17. Keramati, A., Jafari-Marandi, R., Aliannejadi, M., Ahmadian, I., Mozaffari, M., & Abbasi, U. (2014). Improved churn prediction in telecommunication industry using data mining techniques. *Applied Soft Computing*, *24*, 994-1012.

18. Zhang, X., Zhu, J., Xu, S., Wan, Y. "Predicting customer churn through interpersonal influence" *Knowledge-Based Systems* 2012
19. Huang, B., Kechadi, M. T., & Buckley, B. (2012). Customer churn prediction in telecommunications. *Expert Systems with Applications*, *39*(1), 1414-1425.
20. Burez, J., & Van den Poel, D. (2009). Handling class imbalance in customer churn prediction. *Expert Systems with Applications*, *36*(3), 4626-4636.