LUNDS UNIVERSITET
Lunds Tekniska Högskola

Predicting customer churn rate with machine learning, for use in customer lifetime value calculations in the telecommunications industry

29-05-2024

Joel Karnehed
&
Albert Regnell

Supervisor: Andreas Johansson
Examiner: Anders Vilhelmsson
Lund University, Faculty of Engineering
EXTM10, Degree Project in Financial Economics, 30 Credits
Master Thesis, Master of Science in Industrial Engineering and Management

Abstract

In the rapidly evolving telecommunications industry, customer retention is crucial for maintaining profitability. This report investigates the performance of various machine learning models in predicting customer churn rates, which are integral to calculating Customer Lifetime Value (CLV). The focus is on a comparative analysis of several models including logistic regression, ridge regression, lasso regression, random forest, AdaBoost, XGBoost, and support vector machines, using two different datasets from the telecom sector. Our research explores the generalizability of churn prediction models, aiming to identify whether a particular model consistently excels across various datasets. Additionally, the study examines the appropriate evaluation metrics for assessing model performance, highlighting the significance of the Brier score in calculating churn probabilities. The findings indicate that XGBoost is the top-performing model on both datasets, leading to the recommendation for telecom operators to adopt XGBoost for their churn rate calculations. However, since model effectiveness varies by dataset, model selection should be done with caution. Selecting a model should consider more than just performance, including factors like interpretability and ease of implementation.

Key words

Churn Rate Analysis, Customer Lifetime Value (CLV), Logistic regression, Machine Learning, CART, Data Imbalance

Acknowledgements

We extend our sincere thanks to Andreas Johansson for his valuable guidance and consistent support throughout our research. His expertise and constructive feedback have greatly contributed to the success of this project. We are also grateful to our company supervisors, who must remain anonymous, for their insights and support.

Table of contents

1 Introduction

1.1 Background

With the rapidly advancing telecommunications industry and evolving customer expectations, telecom operators face challenges acquiring new and retaining existing customers, by ensuring loyalty and satisfaction. Customer churn rates can significantly impact a telecom company's bottom line, which makes customer retention a top priority.

Given this context, customer churn has become increasingly common for companies to assess, and research has been conducted to develop advanced modeling and data-mining techniques over the past years. Churn rate can be used in many business cases to focus retention and marketing activities, and one of the most commonly used cases is as input in customer lifetime value calculations (CLV). (Kotler et al. 2008, pp. 385-387) In the case of this report, a customer specific churn rate model will be generated for telecom subscribers.

Recent research has evaluated various machine learning algorithms for churn modeling. However, many studies have focused on single datasets, raising concerns about result generalizability. To address this, this study tests all relevant models on two distinct customer datasets from the telecom industry. One publicly available dataset from Kaggle, and one proprietary from a major telecom operator. Further, existing literature lacks consensus on the purpose of churn modeling, a gap this study aims to fill. By introducing different performance metrics, we aim to identify the optimal churn modeling technique for use in CLV calculations.

1.2 Purpose

This thesis aims to fill the gap in existing literature by defining a clear objective for churn modeling, namely CLV calculations in contractual settings. We will investigate churn rate modeling using machine learning algorithms, specifically

focusing on telecom operator subscribers. Additionally, we aim to address the limitations of prior research, which often lacked generalizability due to single dataset analyses. By testing various models on two different customer datasets from the telecom industry, we aim to identify the most effective churn modeling technique.

1.3.1 Research questions

Which model provides the most precise predictions of churn probabilities for telecom customers, making it suitable for integration into Customer Lifetime Value calculations?

2 Theoretical background

This chapter gives a theoretical background of the subject, beginning with CLV calculations to highlight the importance of churn rate analysis. Churn models from prior works are then presented and further explained. Additionally, an analysis of evaluation parameters is presented and each parameter is explained in detail.

2.1 Customer lifetime value

The idea of CLV was first introduced by Kotler (1974), when introducing the "*present value of the future profit stream expected over a given time horizon of transacting with the customer".* Berger and Nasr (1998) further points out that loyal customers should be sought after by companies, as they will be the most valuable in the long term. They support this by introducing formula (1) for calculating CLV as follows:

$$CLV = \sum_{t=1}^{n} \pi(t) \frac{\rho^{t}}{(1+d)^{t}} \quad (1)$$

With π(t) profit or value generated each time period, ρ retention rate and d is the discount rate over a time period n. By this definition there are two parameters that can be adjusted by marketing activities to increase CLV, namely revenue and retention rate. For a contractual service like telecommunication subscriptions, the revenue for each customer is rather stable over time, making retention rate the single most important parameter in CLV calculations. This idea is supported by Ryals and Knox (2005) who suggest that retention campaigns are the most important factor to increase customer value. As a result, the upcoming section will present prior literature on churn modeling.

2.2 Modeling churn rates

It is clear that the retention rate of a customer is an important parameter when calculating CLV. Reichheld and Sasser (1990) suggests that an increase in customer retention from 85% to 90% could result in a net present value profit increase of up to 95%. Ryals and Knox (2005) further develops this idea by finding that the greatest increase in CLV comes from increasing retention on the most valuable customers, and on customers where an increase is actually possible. As a result, proper and accurate modeling of customer-specific retention rates is essential for understanding and later maximizing CLV.

2.2.1 Models from previous works

According to Kumar and Reinartz (2016), many studies between the years 2000 and 2007 used logistic regression to model churn rate. Many times it was also modeled as aggregated churn for customer segments or cohorts. However, recently different machine learning algorithms have become increasingly popular given the possibility of modeling churn rates for individual customers, among other reasons. (Kumar & Reinartz 2016)

Glady, Baesens, and Croux (2009) compares three machine learning algorithms to a baseline logistic regression. Their results show that a neural network has the overall highest accuracy compared to a decision tree and AdaBoost classifier, as well as in comparison to the logistic regression. However, when considering the profit loss of a misclassification, the cost-sensitive decision tree and AdaBoost were the best models. In a similar way, Buckinx and Van den Poel (2005) compares logistic regression with random forest and neural networks. They conclude that the random forest and neural network models perform equally well, but emphasize that the simplicity of computation and interpretation of the random forest model makes it superior.

Looking at newer studies, the basic logistic regression model is further developed with Lasso and Ridge penalty functions. Pebrianti et al. (2022) suggests that Lasso performs the best out of all logistic regression models. However, these authors then compared the XGBoost machine learning model to all previous

models, and it was found to outperform them all. Similarly, Galal et al. (2022) tested the Gradient Boosting model - a simpler version of XGBoost - finding a similar result to Pebrianti et al. (2022). Further, Zatonatska et al. (2023) revealed that XGBoost delivers high overall performance, but concluded that a Support Vector Machine model produced the greatest sensitivity in churn classification. However, one should not dismiss the simplicity of implementation and interpretation as a reason to use a logistic regression when modeling churn rate in a company. (Zatonatska et al. 2023; Pebrianti et al. 2022)

Considering the insights from prior research, this study seeks to compare the recommended models to conclude whether one model can outperform the others. We also aim to address generalizability by testing them on two datasets, which no other author has done. Furthermore, it is worth noting that these studies have focused on classification performance, and not considered probability validity. Hence, the models under evaluation include logistic regression, logistic ridge regression, logistic lasso regression, random forest, AdaBoost, XGBoost and support vector machine. Artificial neural networks will be excluded due to their complexity in implementation, given the numerous different architectures and methods of implementation. This is because ease of implementation is a prerequisite for a good model in business cases (Zatonatska et al. 2023; Pebrianti et al. 2022). In the following sections, the theory and background of each chosen model will be explained.

2.2.1.1 Logistic regression

Logistic regression is useful when the response variable is binary - e.g. when using customer data to predict churn risk. Let Y=1 be the response corresponding to a customer churning, and Y=0 be the response to a customer staying. Instead of modeling the response Y, logistic regression models the probability that Y = 1. In such applications one uses the model in equation 2.

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\ldots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\ldots+\beta_p X_p}} \quad (2)$$

The output will have a value between 0 and 1, where X is the vector of explanatory variables, and β are the unknown parameters to be estimated from the data. The coefficients are estimated using maximum likelihood, which results in minimizing the loss function, which for logistic regression is the binary cross-entropy loss. Logistic regression is advantageous because, under the transformation (3), which is the log of the odds, one obtains a linear model.

$$log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p \quad (3)$$

Thus the model can be interpreted similarly to normal linear multiple regressions. Note that increasing X_i with one unit changes the log odds with β_i, not the probability. Logistic regression can be modified to handle categorical explanatory variables through definition of dummy variables, which although can become impractical if there are many categories. (James et al. 2023)

Logistic regression offers several advantages. Firstly, the model coefficients can be translated to a determined increase in the dependent variable, which is not possible with CART-based methods. Secondly, it doesn't necessitate that variables follow a normal distribution, making it versatile across different data distributions. Additionally, it handles heteroskedasticity between variables with minimal distortion to results, reducing the need for complex mathematical transformations. However, a key consideration is the need to reduce the number of features used, often achieved through backward elimination. This iterative process removes insignificant variables from the model one at a time, ensuring that only sufficiently significant variables remain. (Szanto 2023)

On the downside, logistic regression is sensitive to outliers, emphasizing the importance of outlier management prior to analysis. Furthermore, when employing logistic regression, issues such as correlation and multicollinearity between variables can arise, necessitating thorough investigation before model setup. The final model is constructed by considering collinearity, significance, and classification accuracy together. (Szanto 2023)

2.2.1.2 Ridge regression

Ridge regression is a regularization technique aimed at tempering the variance in a model's predictions, although at the expense of introducing a degree of bias. While closely resembling standard regression, ridge regression diverges by adding a penalty term into the loss function:

$$Penalty = \lambda \sum_{j=1}^{p} \beta_j^2, \quad (4)$$

This added term, known as the shrinkage penalty, is small when the coefficients are close to zero. Consequently, it shrinks the coefficients towards zero, mitigating the impact of multicollinearity. The lambda parameter serves as a tuning parameter, regulating the interplay between the shrinkage penalty and the original loss function. At $\lambda = 0$, the penalty term equals zero, reverting ridge regression to normal regression. However, as λ approaches infinity, the impact of the shrinkage penalty increases, causing the coefficients to approach zero. Consequently, the flexibility of the model decreases, resulting in reduced variance but heightened bias. The features should be standardized to have expected value zero and variance one before implementing the model to make sure that the regularization penalizes the coefficients equally. (James et al. 2023)

This tradeoff makes the model less sensitive to fluctuations in the training data, decreasing the risk of overfitting. Identifying an appropriate value for the lambda parameter is essential for striking a balance between bias and variance. Ridge regression performs particularly well in scenarios where standard regression exhibits high variance, a common occurrence in datasets with limited observations. (James et al. 2023)

2.2.1.3 Lasso regression

Ridge regression comes with an obvious drawback. While it shrinks all coefficients towards zero, it never sets any of them precisely to zero. Hence, although increasing the value of λ in ridge regression will reduce the magnitudes of coefficients, it won't exclude any variables. This may not affect the accuracy of

the model, but can pose a challenge in situations where we have a large number of input features and we want to build a model that includes only the most important ones. To address this limitation, lasso regression offers a solution. It introduces the absolute value of coefficients as a penalty in the loss function:

$$Penalty = \lambda \sum_{j=1}^{p} \left| \beta_j \right| \quad (5)$$

Comparing equation (4) to (5), we see that the only difference between lasso and ridge regression is that the β_j^2 term in the ridge regression penalty (4) has been replaced by $|\beta j|$ in the lasso penalty (5). This penalty ensures that some coefficient estimates can be exactly zero when the tuning parameter λ is large enough. As a result, lasso regression performs feature selection, making it easier to interpret models compared to ridge regression. As with ridge regression, selecting an appropriate value for λ in lasso regression is crucial and can be done through techniques like cross-validation. Lasso is expected to perform better than ridge in situations with a large amount of input features, but with only a few significant ones. (James et al. 2023)

2.2.1.4 Random forest

A CART (Classification and Regression Tree) works by recursively splitting the data into subsets based on feature values that maximize the separation of the target variable. At each node, the algorithm selects the feature and threshold that best reduce a chosen impurity measure (like Gini impurity for classification or mean squared error for regression). The process continues until a stopping criterion is met, resulting in a tree structure where each leaf node represents a predicted outcome. Classification and decision trees (CARTs) generally exhibit a notable issue known as high variance. This implies that if the training data were randomly divided into two parts and a decision tree constructed for each segment, the outcomes obtained could vary significantly. In contrast, methods with low variance produce consistent results when applied to different datasets. For instance, linear regression tends to demonstrate low variance, especially when the ratio of observations to features is relatively large. (James et al. 2023)

As averaging a set of observations reduces the variance, a natural approach to diminish variance and enhance the accuracy of a statistical learning method on the test set involves extracting multiple training sets from the population, constructing individual prediction models using each training set, referred to as an ensemble. The predictions from the models are subsequently averaged to generate one final prediction. When the training sets are extracted by Bootstrapping on the original dataset, this method is called Bootstrap aggregation, commonly referred to as bagging. This serves as a versatile technique for mitigating the variance inherent in statistical learning methods. A technique commonly used in conjunction with decision trees. (James et al. 2023)

Random Forest Classification is an ensemble learning technique based on the bagging method, aimed at improving classification accuracy and robustness. It constructs multiple decision trees on bootstrapped training samples. Each time a split is considered in a tree, a random subset of features is considered as split candidates, and the feature that optimizes information gain or Gini impurity is chosen. This creates a diversity among all trees which mitigates overfitting and enhances model resilience. Once trained, the random forest aggregates predictions from individual trees to make the final classification decision. (James et al. 2023)

Feature selection is intrinsic to Random Forests, which promotes a diverse set of features in tree construction. This approach helps prevent feature dominance and overfitting, ensuring the ensemble benefits from a wide range of features to improve generalization. How often each feature is used as a split into the forest can then be interpreted as feature importance. (GeeksForGeeks 2024).

A number of hyperparameters can be tuned to optimize the performance of the random forest. The parameters decide characteristics of the forest, such as how many trees it should consist of, how many features that should be considered in every split, how deep and wide the trees should be, and how many leaves they should have. More leaves allows for more specific predictions, but also increases the risk of overfitting. To reduce overfitting, the hyperparameters are tuned according to the performance on the validation set. (Scikit-learn 2024)

2.2.1.5 AdaBoost

Boosting, another ensemble method for enhancing decision tree predictions, is a general approach applicable to various statistical learning methods for regression or classification. Boosting operates similarly to bagging, but with a key distinction: the trees are developed sequentially using insights from the previously constructed trees. Unlike bagging, boosting does not employ bootstrap sampling; instead, each tree is trained on a modified version of the original dataset. (James et al. 2023)

AdaBoost is a boosting method that trains and deploys decision trees in series. By implementing boosting, AdaBoost puts together a lineup of weak classifiers in succession. Each weak classifier attempts to rectify the misclassifications made by its predecessor, gradually transforming a collection of weak learners into a robust classifier. The decision trees employed in AdaBoost, often referred to as "stumps," are deliberately kept shallow to avoid overfitting while maintaining a level of bias. Each tree is tailored to rectify the shortcomings of its forerunner by amplifying the weight of previously misclassified samples to sharpen the focus of subsequent classifiers. As more weak classifiers join the series, the overall classification accuracy improves. The sequential learning can give the method an advantage over methods like Random Forest, but as the algorithm continuously increases the weights on the misclassifications, this can potentially lead to overfitting and a decline in generalization performance. Furthermore, AdaBoost can be particularly effective for imbalanced datasets, but may falter in the presence of noise. (Misra & Li 2020, pp. 243-287)

2.2.1.6 XGBoost

XGBoost represents an advancement in boosting algorithms, offering an efficient implementation of gradient boosting decision trees that can set multiple weak classifiers into one strong classifier. Through iterative refinement, the algorithm constructs new trees to address the residual errors left by preceding trees, progressively improving accuracy with each iteration. To prevent overfitting, XGBoost incorporates regularization, introducing a regularization term into the loss function. This term, in the context of decision regression trees, is based on the number of leaf nodes and their associated values. If the regularization

parameter is set to zero, XGBoost reverts to a conventional boosting model. (Tao et al. 2023)

During the iterative training process, XGBoost aims to minimize the objective function by iteratively refining the model, finding a balance between reducing the loss function and controlling the model's complexity. (Tao et al. 2023)

2.2.1.7 Support Vector Machine

The support vector machine (SVM) builds upon the foundational concept of the maximal margin classifier, which aims to separate classes with hyperplanes. While the maximal margin classifier is elegant and straightforward, its applicability is limited as it requires linear separability between classes, making it unsuitable for many datasets. To address this limitation, the support vector classifier was developed, which extends the maximal margin classifier to handle a wider range of scenarios. Then further advancement led to the SVM, which is designed to accommodate non-linear class boundaries. It manages this by enlarging the feature space using kernels. The improved capabilities of the non-linear kernels compared to the linear kernel is illustrated in figure 1 below. The linear kernel does not manage to separate the classes in figure 1, while the polynomial kernel and radial kernel manage almost perfectly. (James et al. 2023)

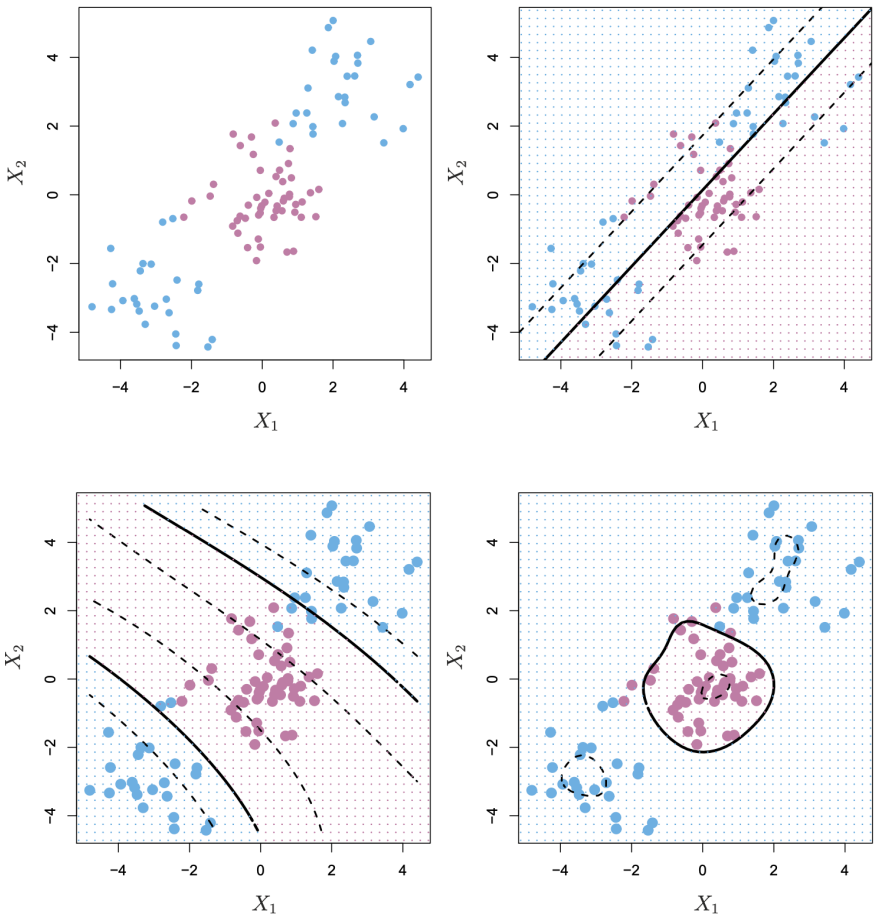

Figure 1: The upper left figure shows observations in a space with features X_1 and X_2. The color of the dot shows which class the observation belongs to. The upper right shows SVM done with a linear kernel, bottom left with a polynomial kernel of degree 3, and the bottom right with a radial kernel. (James et al. 2023)

A kernel function quantifies the similarity between two observations, which is used when the vector classifier divides the observations into different classes. For instance, a simple choice for a kernel could be a linear kernel, which essentially mirrors the support vector classifier. The linear kernel measures the similarity of observation pairs using Pearson correlation, as it assumes linearity in the features. Suppose that we have a n x p data matrix **X** that consists of n training observations in a p-dimensional space:

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

where all observations fall into one of two binary classes. Then the linear kernel can be written as in equation (6)

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{6}$$

Alternatively, equation (7) presents another kernel option, known as a polynomial kernel of degree d, where d is a positive integer. By employing such a kernel with d>1 in the support vector classifier algorithm, a more flexible decision boundary is achieved. This kernel effectively transforms the classification problem into a higher-dimensional space, involving polynomial features of degree d, enhancing the classifier's ability to capture non-linear relationships. The polynomial kernel is shown in equation (7).

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d \tag{7}$$

Combining the support vector classifier with a non-linear kernel, such as the polynomial kernel in equation (7), results in a support vector machine (SVM). The function for the classifying vector in the support vector machine is shown in equation (8), where the choice of kernel clearly decides the characteristics of the classifier.

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i) \tag{8}$$

Besides the polynomial kernel, two other available non-linear kernels are the sigmoid and radial kernel. The radial kernel is commonly used and uses the euclidean distance for predictions, where training observations with big distance from each other has little predictive impact on each other. The formula for the radial kernel is shown in equation (9).

$$K(x_i, x_{i'}) = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \quad (9)$$

SVM has strong underlying connections to logistic regression, and due to the similarities between their loss functions, logistic regression and the support vector classifier often give very similar results. When the classes are well separated, SVMs tend to behave better than logistic regression; in more overlapping regimes, logistic regression is often preferred. (James et al. 2023) A disadvantage with SVM is that it doesn't directly provide a predicted probability, but when implemented using scikit-learn, instead calculates the probability by using an expensive five-fold cross-validation. There is a tuning parameter C in the loss function that determines the extent to which the model underfits or overfits the data. (scikit-learn 2024)

2.2.2 Evaluation parameters from previous works

After generating the models, the models' predictive performances need to be evaluated and compared using different metrics. Galal et al. (2022) uses accuracy and AUC-ROC as performance measures, while Pebrianti et al. (2022) are using the AUC-ROC as well as accuracy, but are also including other outputs from the confusion matrix; precision and recall. The precision and recall adds an extra nuance of insight to the accuracy, which is valuable in cases of e.g. an imbalanced data set.

2.2.2.1 Confusion matrix

A confusion matrix offers popular performance measures in classification problems, as it compares the actual outcome with the predicted values. The diagonal represents the correctly classified data points, with the top left corner being the true negative, and the bottom right being the true positive. The subdiagonal is the false classified data points, the top right corner being the falsely positive, and the bottom left being the falsely negative. The general confusion matrix is shown in figure 2. (Kalkarni et al. (2020))

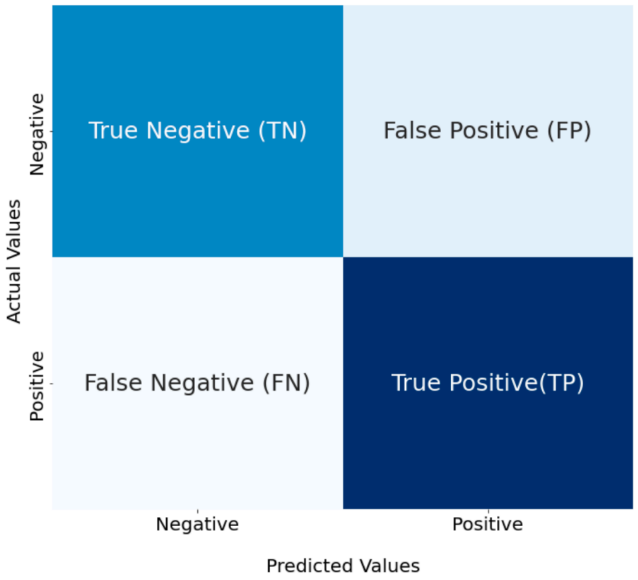

Figure 2: The general definition of a confusion matrix.

The confusion matrix counts are utilized to derive performance metrics, especially accuracy, precision, and recall. Accuracy represents the proportion of accurately classified data points. Precision indicates the percentage of accurately classified positive data points, while recall indicates the percentage of correctly predicted positive instances out of all predicted positive instances. Hence, the perfect model has an accuracy, precision and recall of 100%. It's essential to consider the class distribution when interpreting accuracy, particularly if one class dominates the dataset, as consistently predicting the dominant class can inflate accuracy. Precision and recall offer valuable insights into areas where the model struggles to make accurate predictions. (Kulkarni et al. 2020)

Furthermore, the F1-score, which is the harmonic mean of precision and recall, offers a more balanced measure of model performance, especially for imbalanced data. It ensures that both the minority and majority classes are considered, balancing the trade-off between precision and recall. This makes the F1-score a more robust and informative metric in cases where class distribution is skewed. (Brownlee 2019) Accuracy, precision, recall and F1-score will all be used when evaluating the models and their formulas are shown in the equations below.

$$Accuracy = \frac{TP + TN}{TP+TN+FP+FN} \quad (10)$$

$$Precision = \frac{TP}{TP + FP} \quad (11)$$

$$Recall = \frac{TP}{TP + FN} \quad (12)$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad (13)$$

2.2.2.2 AUC-ROC

The Receiver Operating Characteristic curve (ROC) is a graphical tool for assessing the effectiveness of binary classification methods. It's frequently employed to pinpoint the optimal threshold for maximizing model performance. On the ROC curve, the x-axis represents "1 - specificity" and the y-axis represents recall, with the curve being plotted for threshold values between 0 and 1. Specificity indicates the proportion of true negative predictions among all actual negatives, with 100% being the ideal value. (Nahm 2022)

Ideally, the curve should hug the upper left corner, representing a perfect (0%,100%) ratio. To evaluate this, the AUC (Area Under Curve) is utilized. A perfect model achieves an AUC of 100%, while random guessing yields an AUC of 50%. Hence, a model's AUC should exceed 50% to be meaningful and typically surpass 80% to be deemed good, although this standard varies based on the complexity of the modeling task. (Nahm 2022) Figure 3 shows interpretation of different AUC-values.

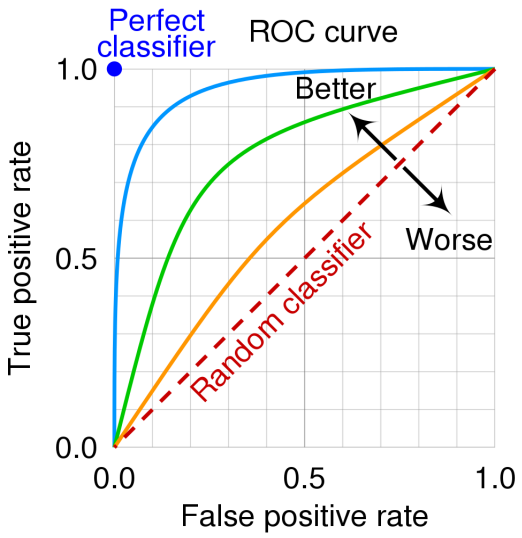

Figure 3: An example of 4 different ROC-curves, representing different levels of performance in the underlying models. (Lee 2021)

2.2.2.3 Brier Score

A notable oversight in prior churn rate literature is the absence of discussion on the validity of output probabilities. While churn calculations commonly focus on the accuracy of binary classification of churners and non-churners, the evaluation of churn probabilities becomes crucial when used in CLV modeling. As a result, the predicted probabilities must be evaluated and compared to real outcomes. In other areas of research, a common way to do this is using Brier score (Dankowski & Ziegler 2016; Lessman et al. 2015).

The Brier score is the mean-squared difference between actual binary outcome and predicted probability and is estimated by:

$$BS = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{p}(y_i = 1))^2 \quad (14)$$

with observed outcome y_i for observations i = 1,...,n. It has been confirmed that Brier score is a proper score, meaning it cannot be enhanced by consistently predicting probability values other than the optimal estimate. Hence, a low score implies that predicted probabilities match true probabilities. (Malley et al. 2012)

In this study, Brier score will be used to assess the predicted churn probabilities for each model, as we suggest churn rate to be used as probabilities in CLV calculations.

2.2.3 Handling imbalanced data

There are many ways of handling imbalanced data. According to Dr. Brownlee (2020) the first step is to make sure that appropriate performance metrics are used, which was discussed in section 2.1.2.1. After that, a second step could be to balance the dataset using either over or under-sampling. One of the best techniques being the oversampling technique called Synthethic Minority Oversampling Technique (SMOTE). (Brownlee 2020) Pradipta et al. (2021) agrees with Brownlee regarding the usefulness of SMOTE. Rather than duplicating minority class instances, SMOTE creates new synthetic data points by interpolating between existing ones. These interpolating lines extend to the k nearest neighbors in the minority class, determined using the K nearest neighbors (KNN) algorithm. The neighboring points are scaled by a random number between 0 and 1 and added to the previously selected feature vector. Distance measurement between sample data points employs the Euclidean distance method. This approach significantly enhances the minority class without mere replication of existing instances. A potential drawback of SMOTE is introduction of bias by generating unrealistic synthetic samples, especially if the minority class instances are noisy or if the feature space is complex. (Pradipta et al. 2021)

There is no established answer to the ideal ratio between the two classes in a dataset; it must be tailored to each specific case. Sometimes, balancing the data may not improve performance at all. (Pradipta et al. 2021)

3 Methodology

This report will take inspiration from Galal et al. (2022) and Pebrianti et al. (2022) in the modeling framework and preprocessing of data, as well as using recommendations from chapter 2 for more detailed decisions. Chapter 3 consists of an explanation of the steps that will be used in data pre-processing as well as model implementation. The steps will be executed and results presented in chapter 4 and chapter 5 for dataset 1 and 2 respectively.

3.1 Data and missing values

First, the structure of the dataset will be evaluated by presenting the number of features and data points. Missing values will either be imputed or encoded as a dummy variable and handled differently for each dataset.

3.2 Exploratory data analysis

Variable and churn analysis will then be conducted. The reason is to find patterns and trends in order to create a naive predictor for comparison to other models. It is also important to detect imbalance between churners and non-churners. As explained in 2.2.3, how to optimally handle imbalance varies for each dataset and model, which is why it will be done in connection to each model if the exploratory analysis indicates imbalance.

3.3 Correlation and multicollinearity

Before modeling any data, we must assess correlation and multicollinearity between predictor variables. High correlation and multicollinearity can lead to difficulty in interpretation of data and redundant information, resulting in model complexity and bias. Logistic regression is impacted by increased variance in coefficient estimates and model bias. Furthermore, it makes analysis of feature importance for the machine learning models more difficult, as highly collinear features will seem less important. (Harrell 2015, p.78-81) Hence, the correlation matrix for each dataset will be calculated and if two features have a correlation of more than 0.8, one will be removed.

While correlation matrices provide a broad overview, they may not detect all collinearity issues, including multicollinearity involving three or more features. To address multicollinearity, the Variance Inflation Factor (VIF) can be utilized. The VIF estimates the extent to which regression coefficients' variance increases due to multicollinearity, with a common cutoff value of 5 used to identify problematic variables. (James et al. 2023)

3.4 Splitting the dataset

When building the models, one dataset is needed to train the parameters in the model. Then the hyperparameters need to be tuned, which requires another dataset. Additionally, an out-of-sample dataset is needed, which represents unseen data that is used to evaluate the generalization capabilities of the model. Therefore, the original dataset is divided into three parts: train, validation, and test. The proportions for splitting the dataset is subjective and should be adapted depending on the dataset. Goodfellow, Bengio and Courville recommend a split of 60% training, 20% validation, and 20% test. However, they claim that for a large dataset the proportion of validation and test data can be increased, since there are enough data points to still achieve comprehensive training. This study uses two relatively large datasets, and since the latter model aims to be implemented by the company, having a reliable estimation of the generalization capability is important. Therefore a large validation and test set is chosen, with a split of 50%, 25%, and 25%, respectively. (Goodfellow, Bengio & Courville 2016, pp 96-161)

3.5 Detecting and handling outliers

Szanto (2023) and Nyitrai and Virag (2019) all advocate for the use of dynamic winsorization in logistic regression models for bankruptcy prediction, where outliers are replaced with the nearest non-outlier values. This method considers observations as outliers if they deviate more than 2 standard deviations from the mean. The process involves an iterative recalibration of the mean and standard deviation until no outliers remain. (Nyitrai & Virag 2019) Although decision trees handle outliers well, logistic regression models, including those using Ridge and Lasso, can be negatively impacted by outliers. Therefore, dynamic winsorization is employed on the logistic regression models. However, given the significant dataset dependency in outlier management, model performance on

outlier-mitigated data will be compared against performance on original data to determine the preferred dataset for modeling.

3.6 Model implementation

Given the literature study on previous works, the following models will be implemented in this thesis, using available established libraries in python:

I. Naive predictor
II. Logistic regression
    A. Logistic Lasso regression
    B. Logistic Ridge regression
III. Random forest
IV. XGBoost
V. AdaBoost
VI. Support vector machine

3.7 Optimizing models

We use grid search with cross validation to optimize the baseline model by tuning the hyperparameters, as suggested by George and Sumathi, (2020). When using grid search, the model is trained and validated for each combination of parameters in a given grid of hyperparameters to try. Cross validation is used to further split the validation data into a given number of sets, 3 in this study, and each set is used as validation data once, and the other two are used to train the model with the parameters in the grid, with each combination of parameters being used once. The output is the combination of parameters that give the best result on all 3 cross validated data sets. Note that splitting the validation data into smaller parts mitigates the risk of the optimization being biased to the validation set. (George & Sumathi 2020) In this study, accuracy and F1 score will be used to assess the results in dataset 1 and 2 respectively, as they are differently balanced.

3.8 Model evaluation

Once the models are optimized and churn predictions are generated, we will calculate and compare the performance metrics outlined in section 2.2.2 for each

model. While the Brier score and accuracy will be prioritized, a comprehensive evaluation including all metrics is essential for a complete performance assessment.

4 Results from dataset 1

In this section the data preparation and model implementation for dataset 1 will be presented, along with model performance results and a short discussion on the dataset specific results.

4.1 Data preparation

4.1.1 Structure of data

Dataset 1 is a publicly available dataset, consisting of anonymized customer data from 72,274 customers of a telecom operator between the years 2007 and 2019 (Sabri Kunt 2019). The data points have been binary classified as churners and non-churners, depending on if the customer has left the company or not. There are 10 predictor variables containing data about the customer contract and usage, specified in table 1.

Table 1: Description of features in the dataset.

Feature	**Description of feature**
Customer Id	Unique Id for each customer
Is TV subscriber	If the customer had a TV subscription with the company
Is movie package subscriber	If the customer has a movie package subscription with the company
Subscription age	Number of years the customer has used the company's services
Bill average	Average billing over the past 3 months
Contract duration	Number of years remaining for the customer contract
Customer service calls	Number of times the customer has called the call center for a service failure
Download average	Average downloaded data over the past 3 months (GB)
Upload average	Average uploaded data over the past 3 months (GB)
Download over limit	How many months the user has used data over their contract limit over the past 9 months
Churn	Binary target variable. 1 if customer has ended their service and 0 if they still have a contract

4.1.2 Missing values

The dataset comprises 72,274 customers with 10 input parameters and a churn rate. One column, *id*, lacks useful information and is therefore removed. Additionally, the columns *remaining contract*, *download average*, and *upload average* contain missing values. The missing values in the *remaining contract* are redundant, which suggests a systematic error of leaving the column empty instead of explicitly writing that the customer doesn't have a contract. Therefore the missing values are replaced with zeros. Conversely, missing values in the download average and upload average columns are more scarce, which would indicate data anomalies, rather than an explainable error. Given the few rows with missing values, it is deemed more appropriate to drop these customers than trying to impute values in the missing fields. This might be seen as biased, but because of the small fraction of observations missing, the random nature of the missing values, and the large data set, deletion can be justified. (Harrell 2015, p.45-48) This results in 71,983 observations with nine available features.

4.1.3 Exploratory data analysis

While imbalanced data is common in churn rate analysis with a small proportion of churned customers, figure 4 reveals a relatively balanced dataset regarding churned customers. This indicates that the company has saved data from churned customers, meaning that the data contains all customers that have ever been with the company.

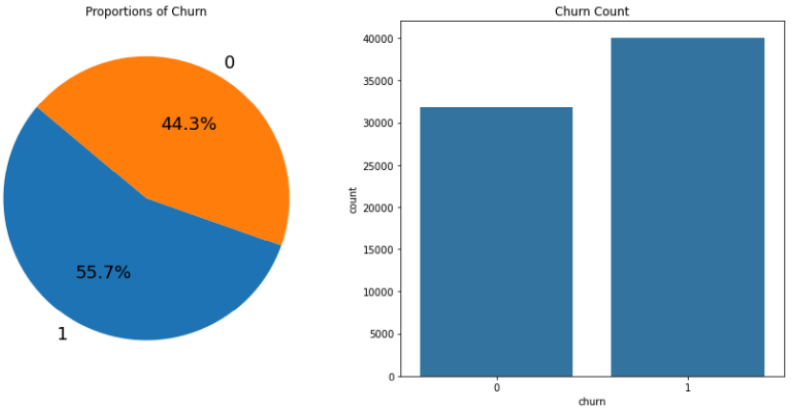

Figure 4: Proportion and number of churned customers in training data.

When evaluating the different models performances, it is useful to compare the performances to the simplest possible baseline model, also called a naive predictor. To create this naive predictor, some basic understanding of how the churn rate depends on the different features is needed. Therefore individual plots depicting each feature's dependency on customer churn are generated, as shown in Figure 5.

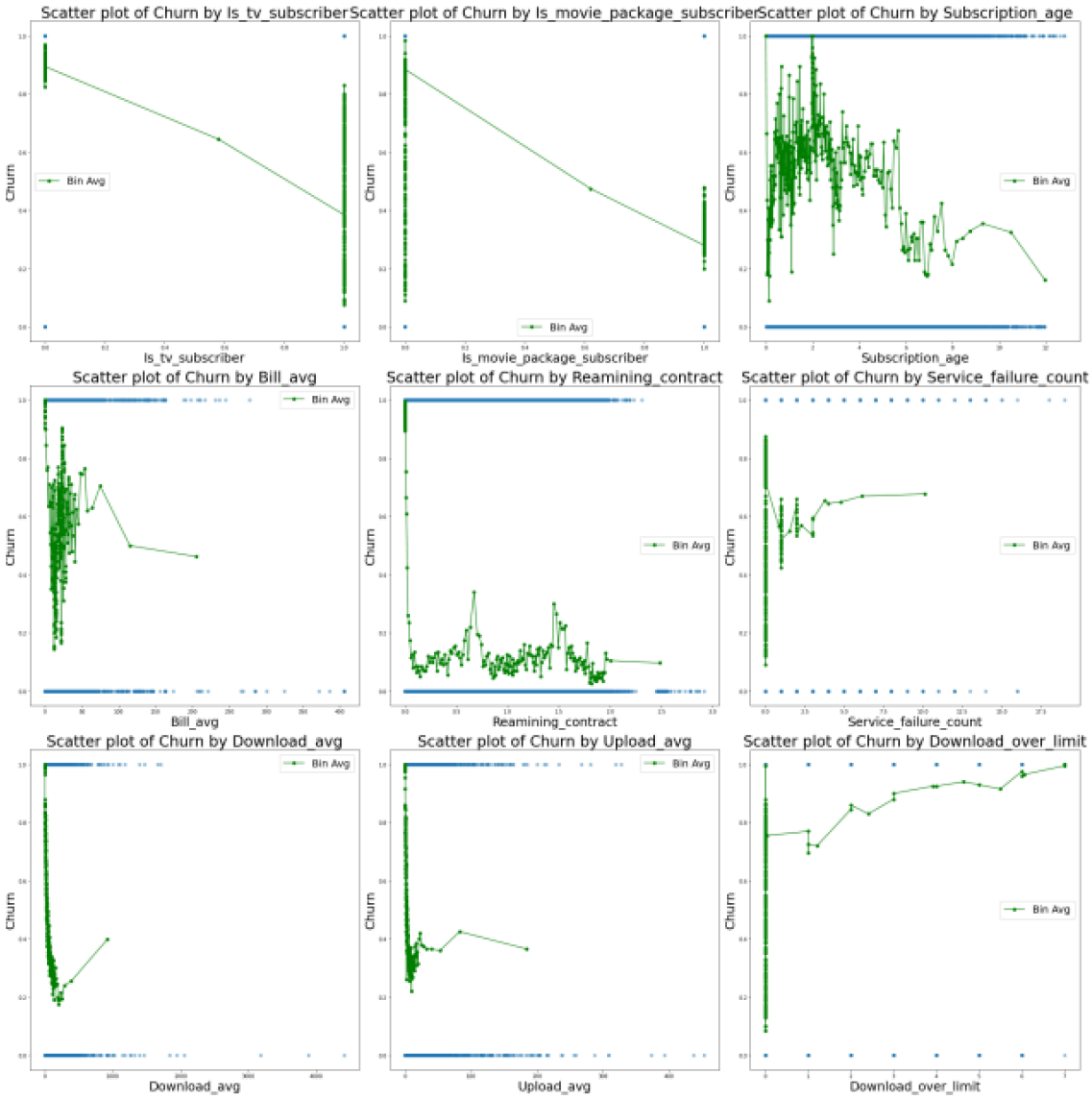

Figure 5: The blue dots representing every observation's feature value and their churn value. The green dots represent the average churn rate for bins of 200 observations in ascending order.

The bins (green dots) show that customers with an active contract display a notably lower churn rate compared to those without. This observation aligns with expectations, as customers are less inclined to churn when bound by contractual obligations. Hence, this feature constitutes a strong candidate for use in the naive predictor. Furthermore, the plot shows that when customers have a non-zero download average and upload average their expected churn rate decreases drastically. Downloading over the limit has the opposite effect, instantly leading to a high expected churn rate.

The blue dots in figure 5 reveal that *service failure*, *download average* and *upload average* have some extreme outliers. For instance, while the median for *download average* stands at 27.8 GB, certain observations exhibit *download average* surpassing 2000 GB. Notably, an outlier having a *download average* 69 standard deviations above the median may prove unsuitable for regression purposes.

4.1.4 Correlation and multicollinearity

The correlation matrix, illustrating all multiple correlations between predictor features and output variable is shown in figure 6. From the correlation matrix, we can conclude that the largest absolute correlation is at 0.55 between features *upload average* and *download average*. This is not considered too high and will therefore not have a big impact on model performance or inference. The correlation matrix also reinforces what we saw in figure 5, that the *remaining contract* has a strong correlation with the churn rate.

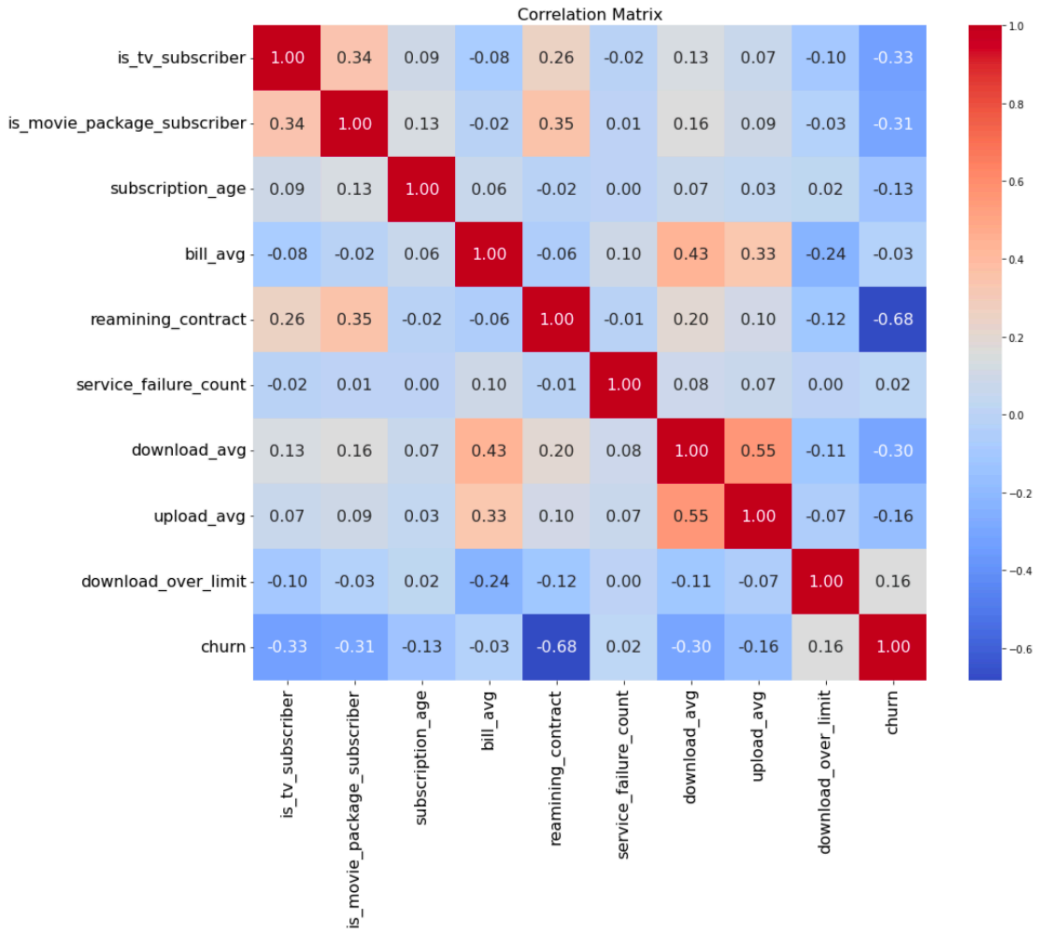

Figure 6: Correlation matrix illustrating correlation between each feature in the dataset.

To further analyze the collinearity, the VIF value for every feature is calculated and is presented in table A1 in the appendix. The VIF shows that the data exhibit some multicollinearity, but since all values are under 5 it is not deemed to be a problem.

4.2 Model implementation

In this section, each model's implementation and results will be presented to later be summarized and discussed in section 4.3.

4.2.1 Naive predictor

To get a fair picture of how well a model performs it is helpful to compare it to a naive model; a model that is the simplest possible but still has some logic behind it. By looking at figure 6 in section 4.1.3 one can see that the churn rate is strongly dependent on if the customer has a remaining contract or not - which is reasonable. Actually, it turns out that only 11.89% of the customers with remaining contracts churned, and 95.08% of the customers that did not have a contract churned. Hence, a reasonable model is to predict that all the customers with contract stays, and all the customers without contract churns. This prediction gave surprisingly good results. The proportions 11.89% and 95.08% are used as the predicted probabilities for the two different classes. The validation confusion matrix is shown in figure 7 and the accuracy and brier score is presented in table 2.

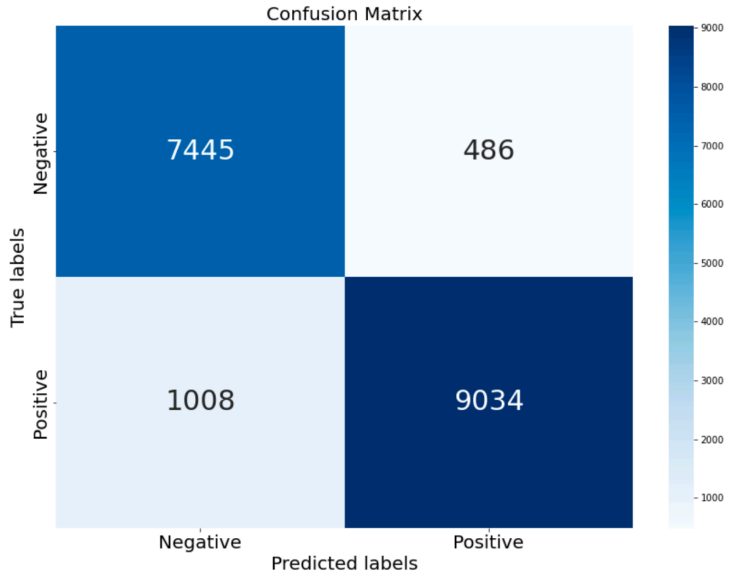

Figure 7: Confusion matrix on the validation data for the naive predictor.

Table 2: Performance metrics for the naive predictor.

Performance metric	Validation results	Test results
Accuracy	0.9169	0.9190
Precision	0.9519	0.9519
Recall	0.9519	0.9010
F1 Score	0.9528	0.9285
ROC AUC Score	0.9192	0.9215
Brier Score	0.0732	0.0732

4.2.2 Logistic regression

The logistic regression model was initially constructed without handling outliers and using all available features. Upon analysis, both *bill average* and *upload average* were found to be insignificant predictors of churn. Consequently, according to the method of backward elimination, these features were removed from the model. Then, this model was compared to a model generated with the same method but after handling outliers using winsorization. The first model performed better regarding validation accuracy, so it was decided to let the outliers be unhandled.

To validate the performance of the model with the features *bill average* and *upload average* removed, a partial likelihood ratio test was conducted. This test compared the performance of the reduced model (without *bill average* and *upload average*) to the original model, showing that the reduced model exhibited no significant decrease in performance. This was supported by lower Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) values, indicating a better model fit. Subsequently, a global likelihood ratio test confirmed that the reduced model significantly outperformed the null model.

Hence, the model is chosen as the final model, with 0.8698 in training accuracy, and the confusion matrix and performance metrics shown in figure 8 and table 3 respectively.

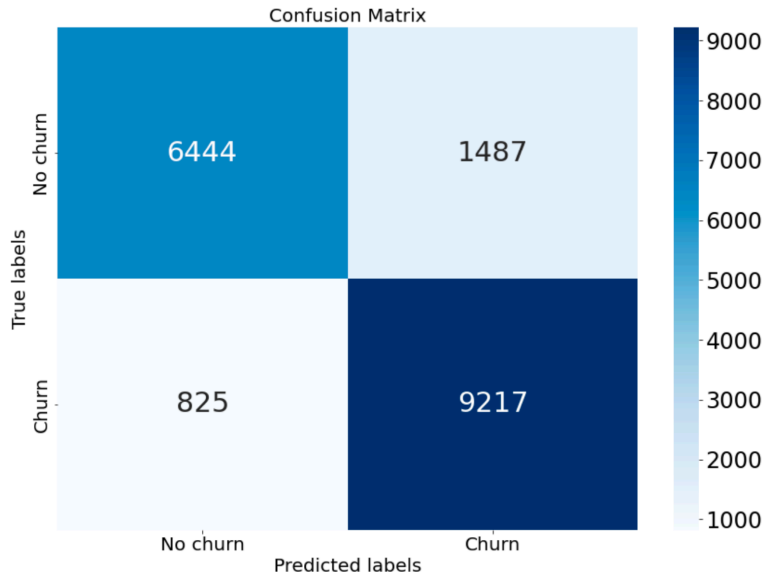

Figure 8: Confusion matrix on the validation data for Logistic Regression.

Table 3: Logistic Regression performance metrics.

Performance metric	Validation results	Test results
Accuracy	0.8714	0.8745
Precision	0.8611	0.8631
Recall	0.9178	0.9221
F1 Score	0.8886	0.8917
ROC AUC Score	0.9346	0.9349
Brier Score	0.0980	0.0973

4.2.3 Logistic Ridge Regression

Logistic Ridge Regression requires that features are standardized to ensure consistent treatment by regularization. Thus, the data is standardized so that the features have a mean of zero and variance of one. Ridge regression inherently applies feature selection, removing the need for manual feature removal.

A grid search with cross-validation regarding accuracy on the validation data was used to identify the optimal regularization parameter. C, the inverse of the lambda parameter, was explored over eight values from 0.001 to 10,000. The grid search revealed a peak accuracy at a regularization value of 10.

Training a model with a regularization parameter of 10 yielded the following confusion matrix and performance metrics:

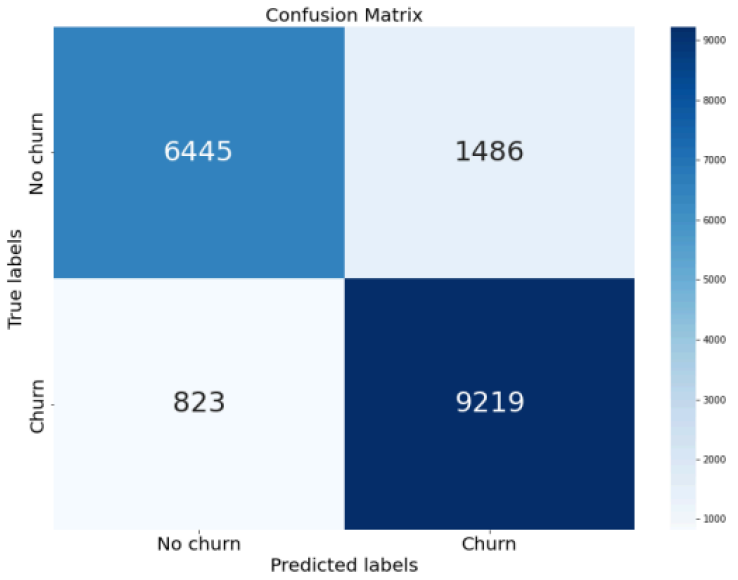

Figure 9: Confusion matrix on the validation data for Logistic Ridge Regression

Table 4: Logistic Regression performance metrics.

Performance metric	Validation results	Test results
Accuracy	0.8715	0.8746
Precision	0.8612	0.8632
Recall	0.9180	0.9222
F1 Score	0.8887	0.8918
ROC AUC Score	0.9345	0.9350
Brier Score	0.0980	0.0973

Similarly, we applied the same method to the dataset with outliers handled as in section 3.5. However, the initial model outperformed the one based on the winsorized dataset. Thus, we retained the outliers in the dataset.

4.2.4 Logistic Lasso Regression

Similar to Ridge regression, the dataset undergoes standardization to achieve a mean of zero and a variance of one. Subsequently, a grid search is executed to pinpoint the optimal regularization parameter for maximizing accuracy. The grid search identifies the optimal value 0.1.

Following this, the identical procedure is repeated for the winsorized data. This model exhibits inferior performance, leading to the selection of the model based on the original data. The validation confusion matrix of this model is shown in figure 10.

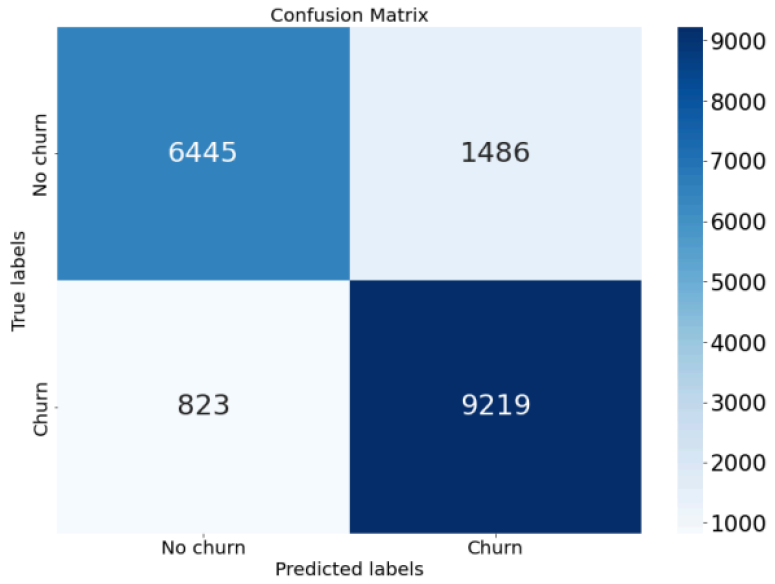

Figure 10: Confusion matrix on the validation data for Logistic Ridge Regression

The training accuracy was 0.8699. The performance metrics on the validation and test data are shown in table 5.

Table 5: Logistic Lasso Regression performance metrics.

Performance metric	Validation results	Test results
Accuracy	0.8715	0.8746
Precision	0.8612	0.8632
Recall	0.9180	0.9222
F1 Score	0.8887	0.8918
ROC AUC Score	0.9345	0.9350
Brier Score	0.0980	0.0973

The performance of the Lasso and Ridge are identical, and they closely mirror that of logistic regression. This is consistent as Ridge and Lasso regression performs well with numerous features, but with only nine features in our

dataset, logistic regression demonstrated comparable performance. Hence, the regularization does not have a notable impact on the generalization performance.

4.2.5 Random forest

The random forest model is first trained with the standard parameters from Scikit learn, on the original training data without any modifications of outliers or standardization.

Through trial and error we found some hyperparameters that gave a higher accuracy to the model, which were later optimized with grid search and cross validation, as explained in 3.7. The input for grid search were 4 different values of each of the parameters: number of estimators (trees in forest), minimum samples to be considered a leaf node, maximum depth per tree and minimum samples to do a split. This resulted in the optimal random forest model with the following hyperparameters:

- `n_estimators = 300`
- `min_samples_leaf=1`
- `max_depth=None`
- `min_samples_split=5`

The model resulted in the confusion matrix shown in figure 11 and the performance metrics specified in table 6. The training accuracy was 0.9834.

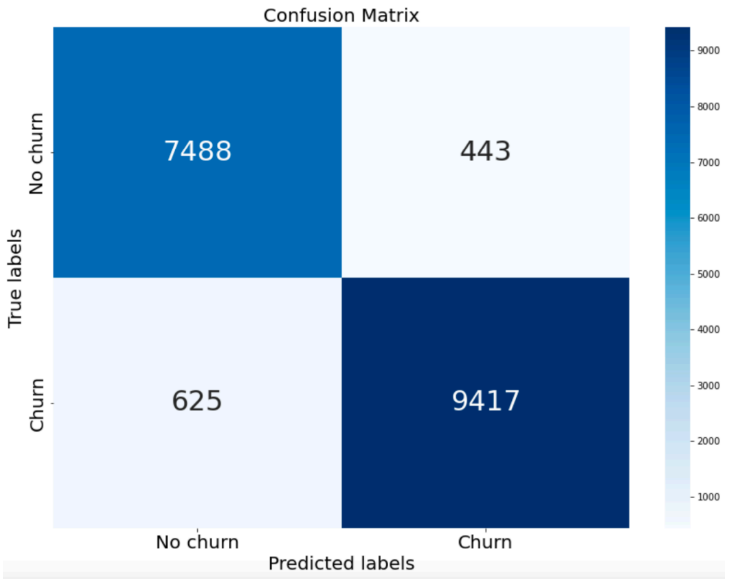

Figure 11: Confusion matrix on the validation data for Random Forest.

Table 6: Random forest performance metrics

Performance metric	Validation results	Test results
Accuracy	0.9409	0.9409
Precision	0.9564	0.9566
Recall	0.9378	0.9369
F1 Score	0.9466	0.9467
ROC AUC Score	0.9788	0.9786
Brier Score	0.0479	0.0477

4.2.6 XGBoost

The XGBoost model was implemented in the same way as random forest, with the standard parameters from Scikit learn. After initial testing of some parameters, a grid search with cross validation on validation data was conducted on the following hyperparameters: number of estimators, maximum depth per tree, and learning rate, with 4 different values on each parameter. Grid search resulted in the following optimal hyperparameters of the model:

- `n_estimators=1000`
- `max_depth=8`
- `learning_rate=0.01`

The model resulted in the confusion matrix shown in figure 12 and the performance metrics specified in table 7. Training accuracy: 0.9529.

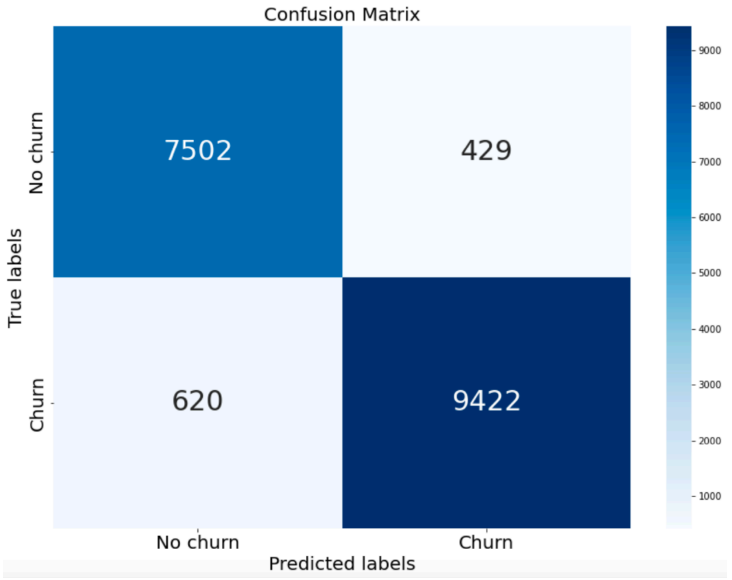

Figure 12: Confusion matrix on the validation data for XGBoost.

Table 7: XGBoost performance metrics

Performance metric	Validation results	Test results
Accuracy	0.9416	0.9416
Precision	0.9565	0.9580
Recall	0.9383	0.9368
F1 Score	0.9473	0.9473
ROC AUC Score	0.9796	0.9786
Brier Score	0.0471	0.0471

4.2.7 AdaBoost

The AdaBoost model was implemented in the same way as the previous CART ensemble models. The input for grid search with cross validation was 4 different values on the parameters: number of estimators, and learning rate. This resulted in the optimal AdaBoost model with the following hyperparameters:

- `n_estimators=1500`
- `learning_rate=0.6`

The model resulted in the confusion matrix shown in figure 13 and the performance metrics specified in table 8. Training accuracy: 0.9347.

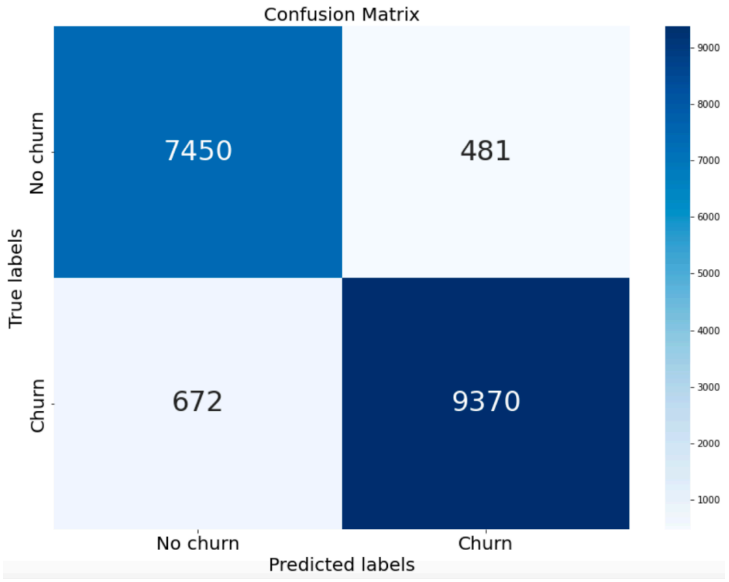

Figure 13: Confusion matrix on the validation data for AdaBoost.

Table 8: AdaBoost performance metrics

Performance metric	Validation results	Test results
Accuracy	0.9341	0.9363
Precision	0.9486	0.9516
Recall	0.9325	0.9338
F1 Score	0.9405	0.9426
ROC AUC Score	0.9658	0.9641
Brier Score	0.1651	0.1644

4.2.8 Support Vector Machine

First, the type of kernel had to be determined. To achieve this, models with linear, sigmoid, radial, and polynomial kernels with degrees ranging from 1 to 10 were created. The data was standardized to mean zero and variance one to ease the fitting of the models. Among the different kernels, the radial kernel model exhibited superior performance metrics.

Next, the regularization parameter C for the radial kernel model was fine-tuned. Utilizing grid search with cross-validation, it was determined that C=10 yielded the best performance.

Subsequently, the model was fitted, and predicted probabilities along with performance measures were calculated. The accuracy on the training data was 0.9181. The resulting validation confusion matrix and performance metrics are presented in figure 14 and table 9 respectively.

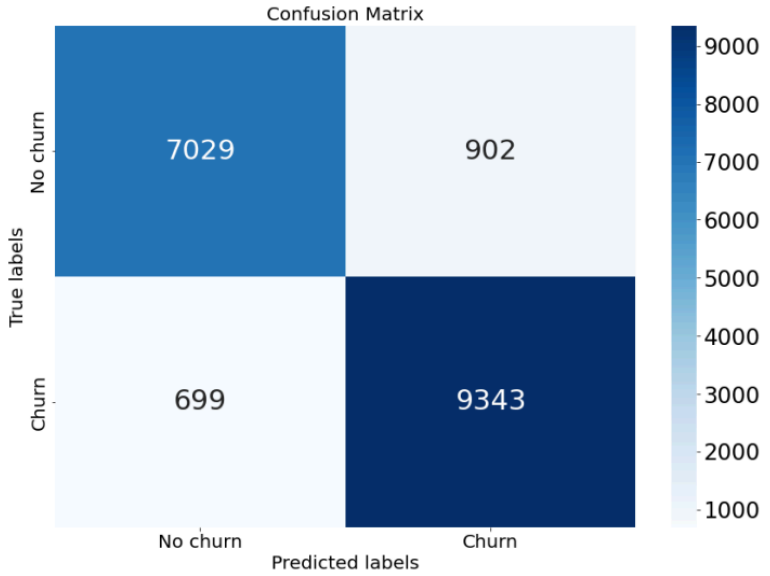

Figure 14: Confusion matrix on the validation data for SVM.

Table 9: SVM performance metrics.

Performance metric	Validation	Test
Accuracy	0.9130	0.9118
Precision	0.9165	0.9165
Recall	0.9289	0.9269
F1-score	0.9227	0.9217
ROC AUC	0.9517	0.9506
Brier score	0.0731	0.0735

4.3 Evaluating the models

The test results for all the models are summarized in table 10 below.

Table 10: Summary of the models performance metrics on the test data.

Model	Accuracy	Brier Score
Naive predictor	0.9190	0.0732
Logistic Regression	0.8745	0.0973
Logistic Ridge Regression	0.8746	0.0973
Logistic Lasso Regression	0.8746	0.0973
Random forest	0.9409	0.0477
XGBoost	0.9416	0.0471
AdaBoost	0.9363	0.1644
SVM	0.9118	0.0735

As table 11 shows, the XGBoost model has the highest accuracy, followed closely by the random forest and AdaBoost respectively. All three tree-based methods outperform the naive predictor, achieving an accuracy of about 94% on the test data. Therefore, the choice between them depends on the specific goals of the model. Since our goal is to accurately predict churn for CLV, it is crucial that the models' predicted probabilities are reliable, which can be assessed using the Brier score.

The tree-based methods clearly outperform the logistic regression models in terms of both accuracy and Brier score. This suggests that the data has complex relationships that decision trees can capture well, but logistic regression struggles with. This conclusion is further strengthened by the fact that the naive predictor outperforms the logistic regression models as well. The naive predictor bases its predictions on two valuable segments: customers with and without contracts. The decision trees methods are perfectly fit to capture this as they use it as a cutoff value in the first couple of splits. However, logistic regression cannot utilize this as the remaining contract is not a binary variable. Therefore, the

logistic regression based models could probably be improved if a dummy variable was added. This would be achievable with this dataset, but as churn datasets can have a lot more features, it wouldn't be realistic to do the analysis necessary to find those kinds of valuable splits. More importantly, one of the big advantages of the decision tree based methods is that they are capable of finding such complex relationships without manual help.

Another notable result is that all the logistic regression based models have almost identical performance. This is probably because the number of features is relatively few, and the dataset is large, which mitigates the risk of overfitting and makes regularization unnecessary.

Interestingly, although XGBoost and AdaBoost have similar accuracies, their Brier scores differ significantly. The histograms in figure 15 reveal that AdaBoost's predictions are uncertain and distributed among three groups, whereas XGBoost's predicted probabilities are concentrated near 0 and 1. XGBoost's histogram indicates a high level of confidence in its predictions, which might suggest overfitting or very strong predictive power depending on the context. AdaBoost's more spread-out probabilities reflect its tendency to combine weaker learners into a more moderated ensemble, leading to a wider range of confidence levels in its predictions. While it might seem unrealistic for XGBoost to be so confident, the high accuracy achieved by the naive model suggests that there are clear trends in the features that a machine learning model could leverage to make confident predictions. This is supported by XGBoost's low Brier score, indicating minimal misclassification while still being confident in its predictions.

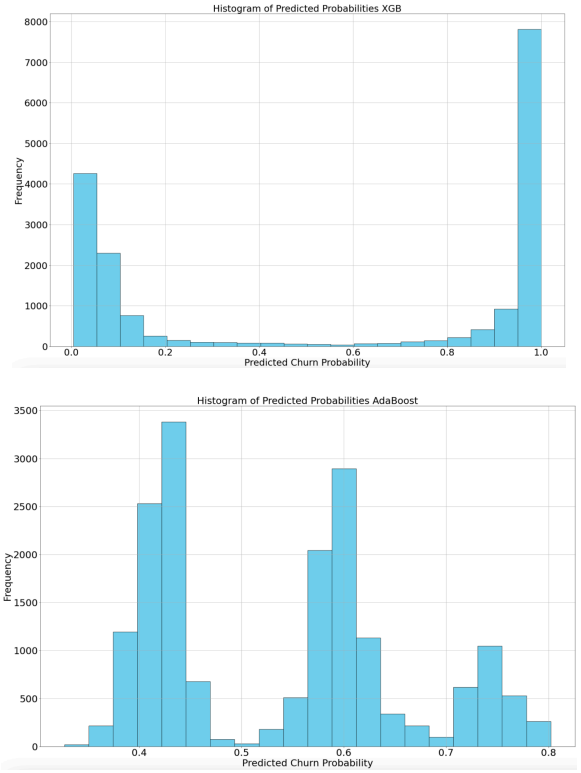

Figure 15: Histogram of the XGBoost (top) and AdaBoost (bottom) predicted probabilities on the test data.

XGBoost also has a slightly lower Brier score than random forest, making it the best model in both accuracy and Brier score—two key metrics for creating a CLV model.

5 Results from dataset 2

In this section the data preparation and model implementation for dataset 2 will be presented, along with model performance results and a short discussion on the dataset specific results.

Dataset 2 focuses on the churn rate among customers of a telecommunications company. Each data point represents an individual subscription, featuring details about the subscriber as well as the specifics of their subscription plan in March 2023. The churn status is determined by whether the subscription remains active after a 12-month period. The data is confidential and will be handled accordingly.

5.1 Data preparation

The same data preparation steps applied to dataset 1 were also used for dataset 2. However, due to the larger size of dataset 2, some visual analyses could not be conducted. Also, since the data is confidential, the specific feature names will not be shared in this report. The focus will be on how the models manage to perform on the data.

5.1.1 Structure of data

The original dataset consists of 127 features and 188431 datapoints. The features are a combination of continuous, binary, ordinal categorical and nominal categorical. The features consist of similar information as dataset 1 but include more details regarding customer specific information, subscription specifications, previous subscriptions as well as information on what other product groups the customer has.

5.1.2 Missing values

Data points in several columns were missing. These were handled in close contact with the company, to make sure that no information is distorted. The missing values in the binary columns were assumed to correspond to zeros. Others were harder to interpret, and in some cases had to be imputed with its own category.

5.1.3 Categorizing variables

The ordinal features were encoded with integers in appropriate order. The categories in nominal features were turned into dummy variables, as no mutual order could be assumed, for instance type of company or geographical region. This increased the number of columns drastically and some of the dummy variables were removed as there were too few observations belonging to that category.

5.1.4 Exploratory analysis

Figure 16 shows the percentage and number of churned customers in the training data. 88.3% doesn't churn in a year's time, which creates a heavily imbalanced dataset.

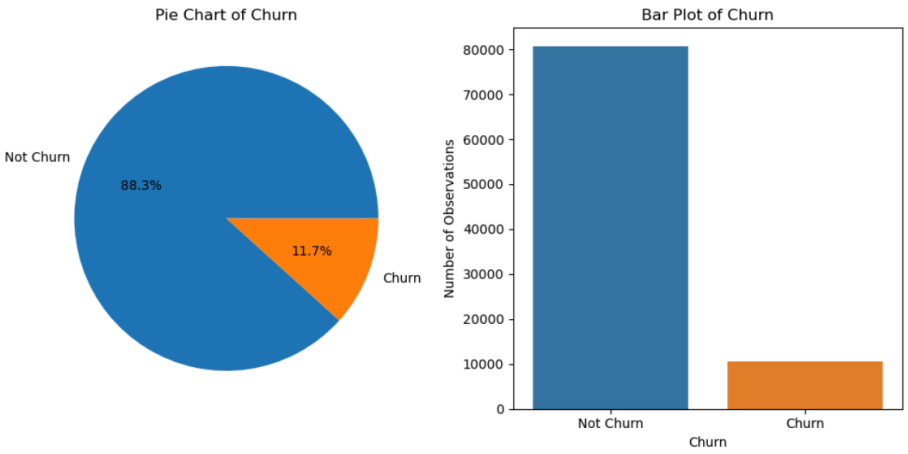

Figure 16: Proportion and number of churned customers in training data.

The graphical visualization of the relationship between the features and churn that was made for dataset 1 will not be conducted here since we suggest it is a too comprehensive analysis only to find a naive predictor, and would therefore not be considered naive. Instead, we chose a feature that intuitively makes sense to explain churn rate, in section 5.2.1.

5.1.5 Correlation and multicollinearity

The correlation matrix is calculated and if feature pairs exhibit a correlation exceeding 0.8, one of them is removed. A visualization of the correlation matrix will not be included in this report as there are too many features for it to give any clear information.

The VIF analysis revealed that nearly all dummy variables exhibited an infinite VIF value. This occurs because dummy variables originating from the same feature become linear combinations of each other. Consequently, the VIF value is not applicable to these features. This may leave some unhandled multicollinearity, however, the feature importance analysis that the dummies enable are regarded as more important. Regarding the other features, those with VIF values exceeding 5 were progressively removed.

5.1.6 Handling imbalance

As the dataset exhibits a churn rate of 12%, it is imbalanced. To address this, as detailed in section 2.2.2.1, F1 score will be used as a metric in the grid search instead of accuracy. Additionally, SMOTE will be employed to oversample the training data for better balance. Since there's no fixed desired class ratio, five different ratios, including no oversampling, will be evaluated and selected individually for each method.

5.2 Model implementation

The training of the models followed the same procedure as for dataset 1, with the difference that the training data was oversampled for some of the models to make the dataset more balanced. For every model, different ratios of oversampling were tested, and the ratio that gave the best combination of validation accuracy, F1 score and brier score was chosen.

5.2.1 Naive predictor

The naive predictor should be straightforward, yet provide a compelling argumentative basis, serving as a benchmark for comparing other models. One simple baseline could be to consistently predict no churn, which would yield a validation accuracy of 88%. To enhance this approach slightly, the naive predictor will be a logistic regression model with a single parameter. Unlike dataset 1, the large number of features in this dataset obstructs the possibility of using any visual analysis to identify the most effective solo predictor. Therefore, customer length is selected as it intuitively seems to have a significant interdependence with churn rate.

The naive predictor had best performance when trained without oversampling. The training accuracy was 0.8832. The confusion matrix corresponding to the validation data, and the performance metrics, are shown in figure 17 and table 11.

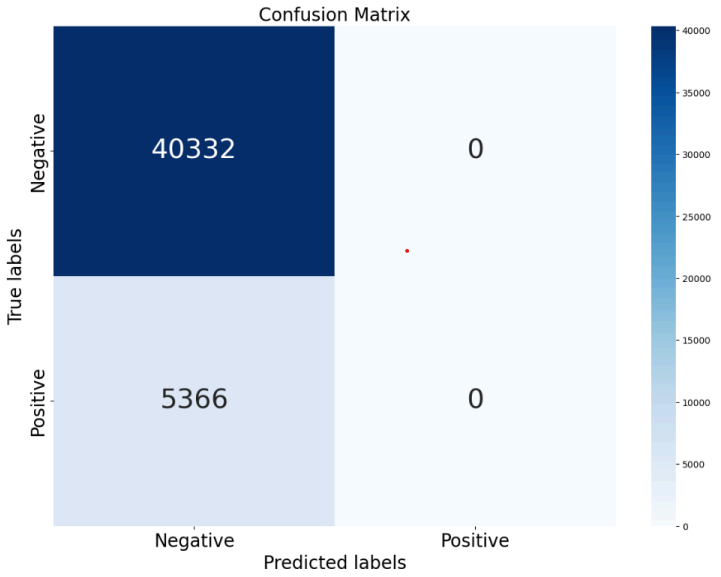

Figure 17: Confusion matrix on the validation data for the naive predictor.

Table 11: Performance measures naive predictor.

Performance metric	Validation results	Test results
Accuracy	0.8826	0.8858
Precision	0.0000	0.0000
Recall	0.0000	0.0000
F1 Score	0.0000	0.0000
ROC AUC Score	0.5311	0.5333
Brier Score	0.1035	0.1010

It turns out that the naive predictor predicts no churn for all observations, as seen in figure 17, which gives it an accuracy of 88%. However, the brier score is 0.1010 on the test data, which is slightly better than a brier score of 0.1036 that is achieved when the probability 12% is predicted for all observations.

5.2.2 Logistic regression

The logistic regression model was trained using the 15 most important features from the random forest model described in 5.2.5. A baseline model was optimized using backward elimination of variables, leaving only the statistically significant ones. A partial likelihood test and the AIC and BIC supported that this model was better than the initial one. Outliers were then handled using dynamic winsorization, and the outcome was used to fit another model. The results of the two models were compared, and once again the model performed better when not handling outliers.

Four different oversampling strategies with SMOTE were also tested and evaluated. However, the model performance decreased, as both accuracy, Brier score and ROC AUC worsened, resulting in no oversampling being made in the final model. As a result, the first model was the best performing, with metrics presented in table 12 and the training accuracy being 0.8832. The confusion matrix in figure 18 shows that the model did not classify any customer as churned.

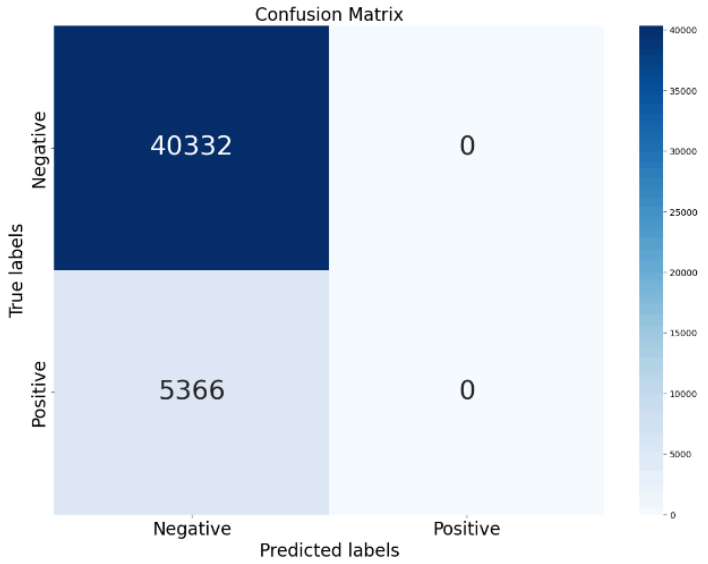

Figure 18: Confusion matrix on the validation data for logistic regression.

Table 12: Logistic regression performance metrics.

Performance metric	Validation results	Test results
Accuracy	0.8826	0.8858
Precision	0.0000	0.0000
Recall	0.0000	0.0000
F1 Score	0.0000	0.0000
ROC AUC Score	0.6295	0.6306
Brier Score	0.1014	0.0991

5.2.3 Logistic Ridge regression

In contrast to the logistic regression, the ridge regression was trained on all the features. The training data was standardized, and the performance analysis showed that the model performed best without oversampling. The grid search gave that the optimal value of the regularization parameter was found to be 0.00001, giving a F1-score of 0, suggesting that it did not predict any churns. The training accuracy was 0.8832. The confusion matrix and the performance metrics are presented in figure 19 and table 13.

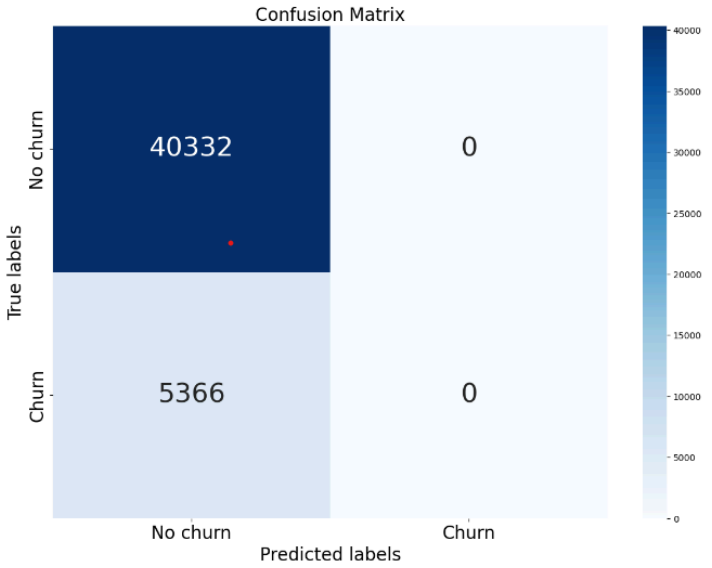

Figure 19: Confusion matrix on the validation data for logistic Ridge regression.

Table 13: Logistic Ridge regression performance metrics.

Performance metric	Validation results	Test results
Accuracy	0.8826	0.8858
Precision	0.0000	0.0000
Recall	0.0000	0.0000
F1 Score	0.0000	0.0000
ROC AUC Score	0.6329	0.6300
Brier Score	0.1028	0.1004

Notably, the Ridge regression got the same accuracy as the logistic regression, with all predictions being "no churn". However, the brier score on the test data is actually worse than the logistic regression, but slightly better than the naive predictor.

5.2.4 Logistic Lasso regression

Similarly to Ridge regression, the Lasso regression was trained on the standardized set of all the features. The model performed best without oversampling with a training accuracy of 0.8831. The grid search showed that the optimal value for regularization parameter was found to be 0.1. The confusion matrix and the performance metrics are presented in figure 20 and table 14.

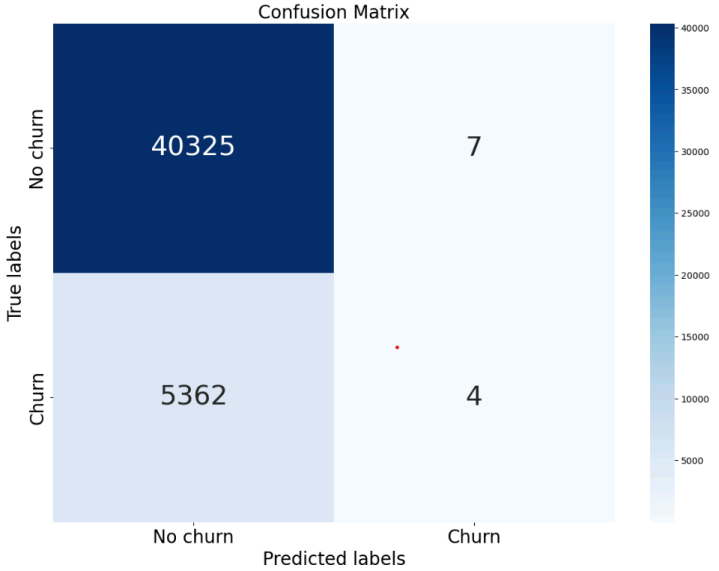

Figure 20: Confusion matrix for Lasso regression om validation data.

Table 14: Lasso regression performance metrics.

Performance metric	Validation results	Test results
Accuracy	0.8825	0.8857
Precision	0.3636	0.1667
Recall	0.0007	0.0002
F1 Score	0.0015	0.0004
ROC AUC Score	0.6584	0.6532
Brier Score	0.1002	0.0982

The Lasso regression outperforms both the naive predictor and logistic regression. This indicates that Lasso regression's capability to shrink coefficients to zero provides it with a useful advantage for this dataset.

5.2.5 Random forest

The random forest model was implemented with grid search on the validation data, which gave the following optimal hyperparameters:

- `n_estimators = 600`
- `min_samples_leaf=1`
- `max_depth=50`
- `min_samples_split=2`

The model was further enhanced with oversampling of the minority class using SMOTE. Three different sampling ratios were evaluated resulting in the optimal ratio of 0.4:1 between the minority and majority class. Training accuracy was 0.9995, indicating overfitting to training data. Despite this, the model managed to generalize better than all logistic regression models, considering the results on the validation and test set, seen in table 15.

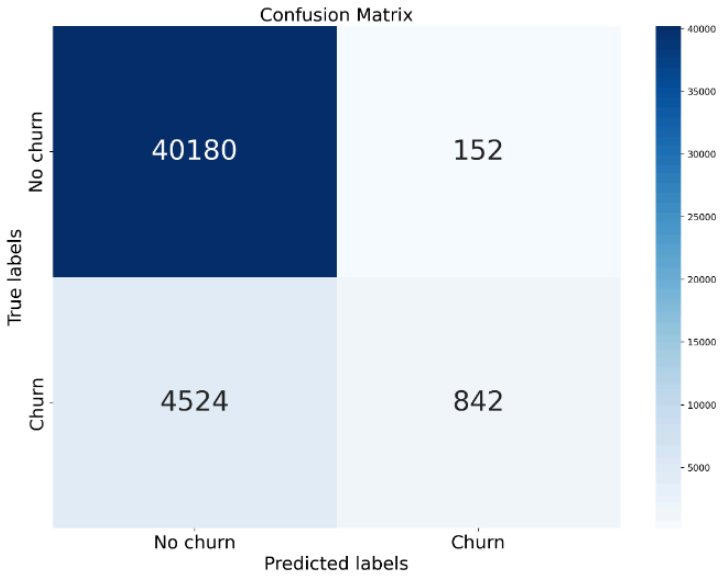

Figure 21: Confusion matrix on the validation data for Random forest.

Table 15: Random forest performance metrics

Performance metric	Validation results	Test results
Accuracy	0.8977	0.9007
Precision	0.8471	0.8355
Recall	0.1569	0.1625
F1 Score	0.2648	0.2721
ROC AUC Score	0.7731	0.7730
Brier Score	0.0862	0.0845

5.2.6 XGBoost

XGBoost was first implemented using grid search with 4 different values on each hyperparameter resulting in the following optimal values:

- `n_estimators=500`
- `max_depth=12`
- `learning_rate=0.1`

Furthermore, SMOTE was used to experiment with oversampling, indicating an optimal ratio of the minority class of 0.4:1. The optimal model achieved a training accuracy of 0.9977 and other performance metrics shown in table 16, as well as

the confusion matrix in figure 22. Similar to random forest, XGBoost demonstrated good generalization to new data, although the high training accuracy suggested potential overfitting.

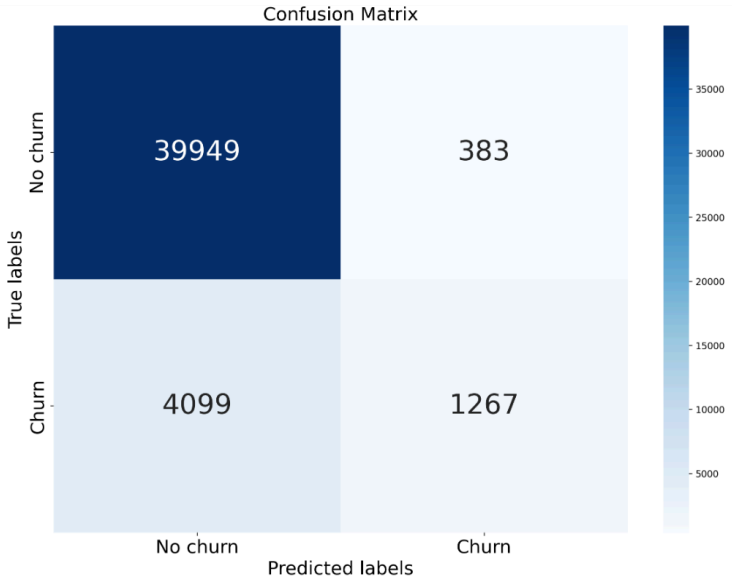

Figure 22: Confusion matrix on the validation data for XGBoost.

Table 16: Performance metrics for XGBoost.

Performance metric	Validation results	Test results
Accuracy	0.9019	0.9043
Precision	0.7679	0.7548
Recall	0.2361	0.2390
F1 Score	0.3612	0.3631
ROC AUC Score	0.7838	0.7839
Brier Score	0.0828	0.0805

5.2.7 AdaBoost

Optimizing the hyperparameters with grid search revealed that the optimal number of trees and learning rate were 1 and 1000, respectively. Additionally, it was observed that the model performed best when the dataset was not oversampled. The performance metrics can be found in table 17, and the

confusion matrix is illustrated in figure 23. The training accuracy of 0.8832 indicates very few instances of classified churned customers.

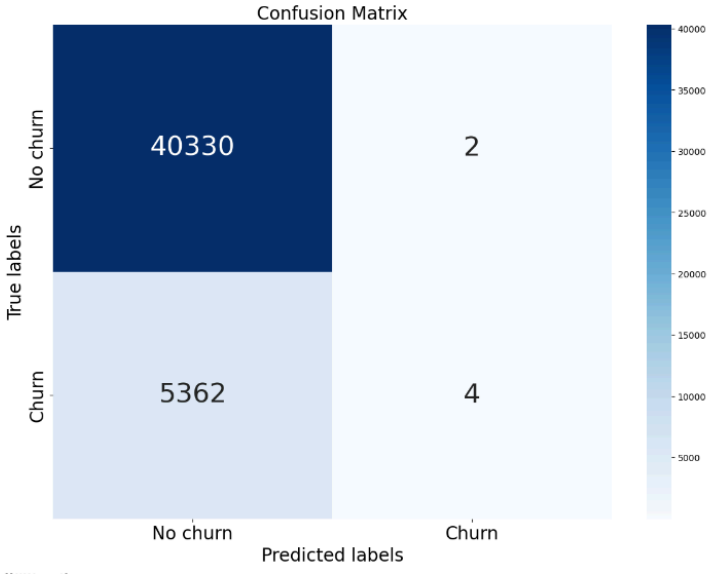

Figure 23: Confusion matrix on the validation data for AdaBoost.

Table 17: Performance metrics for AdaBoost.

Performance metric	Validation results	Test results
Accuracy	0.8826	0.8859
Precision	0.6667	0.6667
Recall	0.0007	0.0012
F1 Score	0.0015	0.0023
ROC AUC Score	0.6846	0.6875
Brier Score	0.2095	0.2092

5.2.8 Support Vector Machine

The data underwent standardization before selecting the radial kernel, which demonstrated the best performance on the validation data. Subsequently, the regularization parameter was fine-tuned through cross-validation on the validation data, with the optimal value determined to be 10. The data was then oversampled, achieving the most effective ratio of 0.4:1. The model achieved a

training accuracy of 0.7053, with the confusion matrix detailed in figure 24 and performance metrics outlined in table 18.

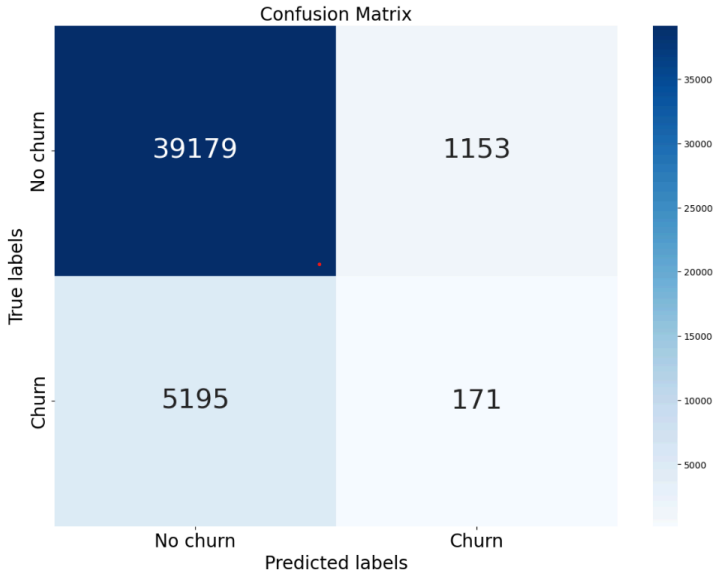

Figure 24: Confusion matrix on the validation data for SVM.

Table 18: Performance metrics for SVM.

Performance metric	Validation results	Test results
Accuracy	0.8611	0.8654
Precision	0.1292	0.1179
Recall	0.0319	0.0276
F1 Score	0.0511	0.0447
ROC AUC Score	0.5208	0.5096
Brier Score	0.1296	0.1282

The model achieved an accuracy of 86.54% and a Brier score of 0.1282. These were the worst results among all models, including the naive predictor. This indicates that a SVM is not suitable for such a large and complex dataset.

5.3 Evaluation of models

The out-of-sample results from dataset 2 are summarized in table 19.

Table 19: Summary of the models performance metrics on the test data.

Model	Accuracy	Brier Score	F1-score
Naive predictor	0.8858	0.1010	0.0000
Logistic Regression	0.8858	0.0991	0.0000
Logistic Ridge Regression	0.8858	0.1004	0.0000
Logistic Lasso Regression	0.8857	0.0982	0.0004
Random forest	0.9007	0.0845	0.2721
XGBoost	0.9043	0.0805	0.3631
AdaBoost	0.8859	0.2092	0.0023
SVM	0.8654	0.1282	0.0447

When modeling the second dataset, the primary challenge was the imbalanced nature of it, which significantly hindered the predictive performance of the models, particularly in identifying the churned customers accurately.

The imbalanced dataset primarily affected the logistic and ridge regression models, which failed to classify any customers as churned. This outcome suggests that the logistic regression models, with its default threshold of 0.5, was biased towards the majority class, leading to a significant underestimation of the minority class (churned customers). Adjusting the classification threshold did show some improvement in identifying churned customers; however, this came at the expense of overall accuracy and would also not improve the Brier score. This trade-off highlights the sensitivity of logistic regression to class imbalance and threshold settings.

In contrast, decision tree-based methods like XGBoost and Random Forest

demonstrated superior performance across most metrics. Both models performed robustly due to their mechanism of building multiple decision trees and aggregating their outcomes, which reduces the variance and bias. However, AdaBoost continued to perform poorly, potentially indicating the presence of noise in the data, to which AdaBoost is sensitive. XGBoost, in particular, excelled slightly in all metrics except precision, when compared to Random Forest. The superior performance of XGBoost can be attributed to XGBoost's gradient boosting mechanism that focuses on correcting the errors of previously built trees, thereby adapting better to the imbalanced data. However, XGBoost only achieves 2.8 percentage points higher accuracy than the naive predictor, indicating the complexity in predicting churn for this dataset.

However, the Brier scores for all models were higher on the second dataset, suggesting that the calculated probabilities might not be accurate. Additionally, the Brier score for XGBoost did not considerably outperform other models in the second dataset. To verify the correctness of these probabilities, we will illustrate the calibration curve in figure 25 as recommended by Dankowski and Ziegler (2016) and Chamberlain et al. (2017). This approach will help confirm that the predicted probabilities align with the actual likelihood of an event, with a perfectly calibrated model lying along the diagonal line. As seen in figure 25, the XGBoost model is highly calibrated, which together with the lowest Brier score of all models proves to be highly effective for predicting churn probabilities.

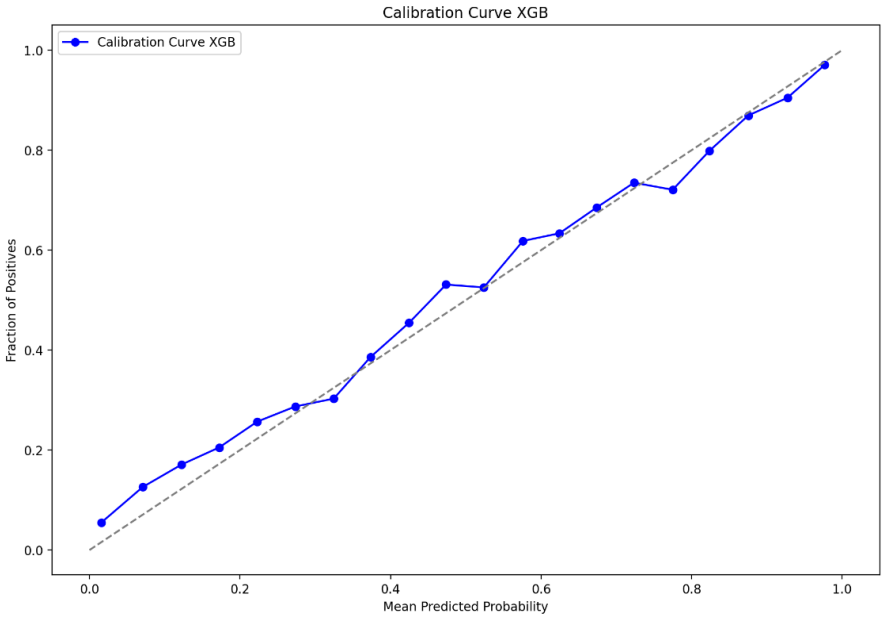

Figure 25: Calibration curve for the XGBoost model, with 20 probability bins. Predicted probabilities on the x-axis and fraction of positives on the y-axis.

Another interesting observation from our experiments was that most methods performed best without the application of oversampling techniques to balance the class distribution. This could indicate introduction of noise, especially if the synthetic samples do not represent the true underlying distribution of the minority class. A potential improvement could be to assess another oversampling method or undersampling of the majority class instead.

6 Discussion and conclusion

The report sets out to identify the most effective model for predicting churn rates, a crucial component in Customer Lifetime Value (CLV) analysis, which plays a significant role in managing customer retention. It compares traditional methods, such as enhanced logistic regression, with modern machine learning techniques that have been prevalent in recent churn rate studies. To align the study with practical applications in CLV calculations, the Brier score and accuracy were chosen as the primary performance metrics, as well as F1 score for the imbalanced dataset.

The analysis involved two datasets that differed significantly in complexity. The second dataset was more challenging due to its imbalance and the greater number of data points and features it contained. Despite these challenges, XGBoost consistently delivered superior performance across both datasets in almost all metrics, with Random Forest ranking a close second. This underscores the effectiveness and flexibility of ensemble CART methods in managing data imbalances and complex patterns. These findings align with recent churn rate research, which highlights decision trees enhanced with boosting and bagging techniques as the best performing models (Zatonatska et al. 2023; Pebrianti et al. 2022; Galal et al. 2022). This thesis further strengthens these findings by testing the models on two distinct datasets, providing a more robust validation than previous studies that typically used only one. However, it is important not to overlook the application of churn rate analysis in telecom companies, where ease of implementation and interpretability are critical metrics that this study does not address.

Despite the small accuracy gap between the naive predictor and XGBoost in the second dataset, which might make the machine learning model seem redundant, the superior Brier score of XGBoost is critical as it reflects the probability of churn rather than mere classification. This distinction is vital in CLV calculations and general business cases where understanding the likelihood of churn is more beneficial than simply classifying customers. It has also been proved that the predicted probabilities in the XGBoost model are well calibrated, which further argues for CLV calculations as an appropriate use case.

In conclusion, and in response to the main question of this report, the analysis demonstrates that XGBoost outperforms both regression models and modern machine learning methods in modeling churn rates for Customer Lifetime Value (CLV) calculations, consistently achieving superior results across various datasets. Telecom operators, and similar subscription based businesses, can leverage the insights from this study to refine their predictive analytics strategies. By implementing an XGBoost model, which adapts well to various data conditions and shows consistent performance, operators can better predict churn probabilities and tailor their customer retention strategies accordingly. However, operators should also consider the complexity of model deployment and the interpretability of model outputs, especially in a business context where explaining predictions can be as important as making them. One should also acknowledge that in the rapidly evolving field of machine learning, it is crucial to stay open to new and potentially improved models. As technology advances, embracing these changes can significantly enhance predictive analytics and keep strategies cutting-edge.

6.1 Drawbacks and Further Research

The study, while insightful, was limited to only assessing predictive performance without a discussion on interpretability and usefulness of models. High-performing models like XGBoost, although effective, are complex and lack the interpretability of simpler models such as logistic regression. This could be a significant drawback in business settings where understanding the rationale behind model decisions is crucial. Logistic regression, for instance, can provide insights into how changes in specific parameters might affect churn predictions. Unfortunately logistic regression performed worse than almost all the machine learning models on both datasets, even with Ridge and Lasso enhancements. Further research should therefore focus on enhancing the interpretability of complex models like XGBoost and Random Forest, making them more suitable for business applications where explaining predictions is essential.

Another major limitation is that the current models are based on a snapshot of churn, which could be influenced by specific market conditions prevalent during the dataset's timeframe. Longitudinal studies that track churn prediction performance over time would offer valuable insights into the models' stability and adaptability to changing market conditions, particularly for the second dataset which covers only one year. Such studies would reduce sensitivity to temporary market fluctuations and provide a more robust framework for understanding and predicting customer behavior over time.

While predicting churn is a critical component of Customer Lifetime Value (CLV) analysis, it does not provide a complete picture. To accurately calculate CLV, predicted income and associated costs must also be considered, in which there was no available data for this study. These elements could be analyzed separately; however, integrating them provides a more comprehensive evaluation. Comparing the implications of churn model predictions on CLV with actual real-world CLVs can offer valuable insights, ensuring that the models not only predict churn effectively but also align closely with true customer value outcomes. This holistic approach enhances the relevance and applicability of the predictive models in strategic decision-making.

References

Berger, P.D. and Nasr, N.I., 1998. Customer lifetime value: Marketing models and applications. *Journal of Interactive Marketing*, 12(1), pp.17-30. doi: 10.1002/(SICI)1520-6653(199824)12:1<17::AID-DIR3>3.0.CO;2-K.

Brownlee, J., 2019. Classification accuracy is not enough: More performance measures you can use. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/ [Accessed 14 May 2024].

Buckinx, W. and Van den Poel, D., 2005. Customer base analysis: partial defection of behaviourally loyal clients in a non-contractual FMCG retail setting. *European Journal of Operational Research*, 164(1), pp.252-268. https://doi.org/10.1016/j.ejor.2003.12.010

Chamberlain, B.P., Cardoso, A., Liu, C.B., Pagliari, R. and Deisenroth, M.P., 2017. Customer lifetime value prediction using embeddings. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.1753-1762. https://doi.org/10.1145/3097983.3098123.

Dankowski, T. and Ziegler, A., 2016. Calibrating random forests for probability estimation. *Statistics in medicine*, *35*(22), pp.3949-3960.

Galal, M., Rady, S. and Aref, M., 2022. Enhancing Customer Churn Prediction in Digital Banking using Ensemble Modeling. In *2022 4th Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, Giza, Egypt, pp.21-25. doi: 10.1109/NILES56402.2022.9942408.

GeeksforGeeks. 2024. *Random Forest Classifier using Scikit-learn.* Available at: https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/. [Accessed 16 may 2024]

George, S. and Sumathi, B., 2020. Grid search tuning of hyperparameters in random forest classifier for customer feedback sentiment prediction. *International Journal of Advanced Computer Science and Applications*, 11. Available at: https://thesai.org/Downloads/Volume11No9/Paper_20-Grid_Search_Tuning_of_Hyperparameters.pdf

Glady, N., Baesens, B. and Croux, C., 2009. Modeling churn using customer lifetime value. *European Journal of Operational Research*, 197(1), pp.402-411.

Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep Learning*. MIT Press, pp 96-161. Available at: http://www.deeplearningbook.org.

Harrell, F.E., 2015. *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*. 2nd ed. Springer.

James, G., Witten, D., Hastie, T., Tibshirani, R., Taylor, J. 2023. *An Introduction to Statistical Learning*. Springer Texts in Statistics. Springer, Cham. https://doi.org/10.1007/978-3-031-38747-0_1

Kotler, P., 1974. Marketing During Periods of Shortage. *Journal of Marketing*, 38(3), pp.20-29. doi: 10.1177/002224297403800305.

Kotler, P., Armstrong, G., Wong, V. and Saunders, J., 2008. *Principles of Marketing*. 5th European ed. Pearson Education.

Kulkarni, A., Chong, D. and Batarseh, F.A., 2020. Foundations of data imbalance and solutions for a data democracy. In *Data democracy*, pp. 83-106. Academic Press. https://doi.org/10.1016/B978-0-12-818366-3.00005-8.

Kumar, V. and Reinartz, W., 2016. Creating Enduring Customer Value. *Journal of Marketing*, 80(6), pp.36-68. doi: 10.1509/jm.15.0414.

Lee, C.M.G., 2021. Roc curve [svg]. Available at: https://commons.wikimedia.org/wiki/File:Roc_curve.svg#filelinks [Accessed 14 May 2024].

Lessmann, S., Baesens, B., Seow, H.V. and Thomas, L.C., 2015. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, 247(1), pp.124-136.

Malley, J.D., Kruppa, J., Dasgupta, A., Malley, K.G. and Ziegler, A., 2012. Probability machines. *Methods of information in medicine*, 51(01), pp.74-81.

Misra, S., Li, H. and He, J., 2020. Noninvasive fracture characterization based on the classification of sonic wave travel times. *Machine learning for subsurface characterization*, *4*, pp.243-287.

Nahm, F.S., 2022. Receiver operating characteristic curve: overview and practical use for clinicians. *Korean journal of anesthesiology*, *75*(1), p.25.

Pebrianti, D., Istinabiyah, D.D., Bayuaji, L. and Rusdah, 2022. Hybrid method for churn prediction model in the case of telecommunication companies. In: *2022 9th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, Jakarta, Indonesia, pp.161-166. doi: 10.23919/EECSI56542.2022.9946535.

Pradipta, G.A., Wardoyo, R., Musdholifah, A., Sanjaya, I.N.H. and Ismail, M., 2021. SMOTE for Handling Imbalanced Data Problem: A Review. In *2021 Sixth International Conference on Informatics and Computing (ICIC)*, Jakarta, Indonesia, pp.1-8. doi: 10.1109/ICIC54025.2021.9632912.

Reichheld, F. and Sasser, W.E. Jr, 1990. Zero defections: quality comes to services. *Harvard Business Review*, 68(5), pp.105-111.

Ryals, L.J. and Knox, S., 2005. Measuring risk-adjusted customer lifetime value and its impact on relationship marketing strategies and shareholder value. *European Journal of Marketing*, 39(5/6), pp.456-472. doi: 10.1108/03090560510590665.

Sabri Kunt, M., 2019. *Internet Service Provider Customer Churn* [Dataset]. Kaggle. Available at: https://www.kaggle.com/datasets/mehmetsabrikunt/internet-service-churn [Accessed 13 March 2024]

Scikit learn. 2024. *sklearn.ensemble.RandomForestClassifier*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed 16 May 2024]

Zatonatska, T., Fareniuk, Y. and Shpyrko, V., 2023. Churn Rate Modeling for Telecommunication Operators Using Data Science Methods. *Marketing and Management of Innovations*, 2, pp.163–173. https://doi.org/10.21272/mmi.2023.2-15

Appendix

Table A1: VIF for all the features.

Feature	**VIF**
is_tv_subscriber	3.679251
is_movie_package_subscriber	1.879948
subscription_age	2.333280
bill_avg	3.010977
reamining_contract	1.869700
service_failure_count	1.123313
download_avg	2.497075
upload_avg	1.734142
download_over_limit	1.058071