VRIJE UNIVERSITEIT AMSTERDAM

meewind

Master Thesis

Predicting Investor Churn in Sustainable Investment Funds: A Machine Learning Approach

Author: Arthur Trautwein (2677633)

First supervisor: Anil Yaman
Daily supervisor: Emilie Hardick (Meewind)
Second reader: Ronald Meester

A thesis submitted in fulfillment of the requirements for the VU Master of Science degree in Business Analytics

February 18, 2026

Predicting Investor Churn in Sustainable Investment Funds: A Machine Learning Approach

Arthur Trautwein
Internship Report

Vrije Universiteit Amsterdam
Faculty of Science
Business Analytics
De Boelelaan 1081a
1081 HV Amsterdam

Host organization:
Meewind
Nieuwe Gracht 13
2011 NB Haarlem

February 18, 2026

Acknowledgements

The purpose of this thesis is to fulfill the requirements to obtain the Master of Science (MSc) degree in Business Analytics at the Vrije Universiteit in Amsterdam. The Master's degree in Business Analytics is a two-year program that ends with an internship, in this case at Meewind, where a master's thesis must be written.

This project was done under the supervision of Anil Yaman on behalf of the Vrije Universiteit and Emilie Hardick on behalf of Meewind. I would like to thank both for being my supervisors and for their guidance and support throughout the project. Moreover, I would like to thank all the other employees of Meewind for their involvement during my internship, and Ronald Meester, for being my second reader. Finally, my special thanks go to my family and friends for their support, encouragement, and love.

Abstract

This thesis investigates the use of machine learning techniques to predict investor churn at Meewind, a Dutch investment fund specializing in sustainable energy projects. In the asset management industry, client retention is a critical determinant of financial stability and long-term performance, making the early identification of potential churners an important strategic objective. However, this task is complicated by a large class imbalance, as only a small proportion of investors terminate their participation. To address this challenge, two imbalance correction strategies, cost-sensitive learning and the Synthetic Minority Oversampling Technique (SMOTE), were applied across a range of classification models.

Model performance was evaluated using both random and temporally separated train–test splits to simulate academic and real deployment conditions. Recall was emphasized as the most business-critical metric, since missing churners carries a substantially higher cost than incorrectly flagging non-churners. The results show that imbalance correction notably improves minority-class detection, while hyperparameter optimization further enhances predictive power and model robustness. Among all models, XGBoost performed strongest under both evaluation strategies, achieving a recall of 77% and a ROC-AUC of 0.85 in the temporal split, a level of performance that falls within the operational tolerance of the organization for proactive retention efforts. Feature importance analysis highlights the key role of transactional behavior, particularly recent sales activity and dividend choices, as well as engagement-related variables such as login frequency.

These findings demonstrate that advanced ensemble models, when combined with appropriate imbalance handling, can provide reliable and actionable early-warning signals for investor churn.

Contents

1

Introduction

Business Context In recent years, the financial industry has undergone substantial transformation driven by technological innovation, stricter regulatory standards, and an increasing emphasis on sustainability. In parallel, investors have shown growing interest in renewable energy and sustainable infrastructure, supported by EU regulations and increasing awareness of climate change. As a result, investment funds focused on financing the energy transition have expanded rapidly and now play a key role in mobilizing private capital to support climate objectives.

In this context, Meewind occupies a distinctive position as a Dutch investment fund dedicated to financing sustainable energy projects. Founded with the mission of making renewable energy accessible to a broad investor base, Meewind directs private capital into projects such as offshore wind farms, solar parks, and other sustainable technology ventures. Through its various sub-funds, including ZeeWind, Regionaal Duurzaam and Energie Transitie Fonds, the organization enables participants to contribute to the energy transition while pursuing long-term financial returns.

Like many investment organizations, Meewind faces the challenge of retaining its existing participants. Churn, investors discontinuing their participation by liquidating their holdings, directly reduces assets under management, undermines long-term planning, and can weaken investor confidence. Research in marketing and customer relationship management consistently shows that retaining existing clients is significantly more cost-effective than acquiring new ones; acquiring a new customer may cost up to five times more than retaining an existing one (Reichheld and Sasser, 1990). For an investment fund, where relationships are trust-based and long-term, participant retention is therefore not only financially essential but strategically influential. High churn can offset even substantial marketing efforts, whereas a stable and loyal investor base forms a foundation for sustainable growth.

From a growth perspective, preventing churn is, therefore, not only defensive but also strategic. Even significant marketing expenditures to attract new participants can result in limited net growth if high churn persists. In contrast, a loyal and stable participant base forms a foundation for sustainable expansion, as satisfied investors are more likely to reinvest, increase their holdings, and recommend the fund to others. This compounding effect makes churn prevention an essential component of Meewind's growth strategy.

To address this challenge, Meewind can take advantage of modern data-driven methods such as machine learning to predict and mitigate churn. By identifying behavioral patterns that signal potential attrition, the fund can design proactive interventions, ranging from personalized communication to customized reinvestment opportunities. Incorporating churn prediction into strategic decision-making thus offers Meewind the opportunity to improve participant satisfaction, reduce acquisition costs, and reinforce its role as a frontrunner in sustainable investment.

Research Question This thesis investigates whether machine learning techniques can be used to effectively predict investor churn within Meewind's diverse and sustainability-focused investment portfolio. The central research question is:

To what extent can customer churn be predicted for an investment fund like Meewind using machine learning models? Furthermore, which models are most effective in predicting churn within this specific context?

The central objective is to develop a predictive framework capable of identifying at-risk investors before they withdraw their investments. In doing so, the study addresses three methodological challenges common to churn prediction: data creation, from all the information that is collected creating meaningful features, class imbalance, where the number of churners is disproportionately small relative to the total customer base, and model interpretability, which is critical for the practical adoption of machine learning in financial institutions.

To address these challenges, several classification algorithms were selected, including Logistic Regression, Decision Trees, Random Forests, Gradient Boosting, Support Vector Machines (SVM), Extremely Randomized Trees (ExtraTrees), and Extreme Gradient Boosting (XGBoost). The choice of these models is motivated by two considerations. First, Meewind explicitly requires interpretable or semi-interpretable models to support transparency and trust in decision-making. This favors so-called "glass box" models and ensemble methods for which feature importance and decision logic can be meaningfully inspected, while fully opaque black-box approaches are less suitable for operational use.

Second, the selected models are strongly grounded in existing churn prediction literature. Recent survey and review studies show that there is no single universally dominant algorithm for churn prediction; model performance is highly context-dependent and varies across industries and data structures. Nevertheless, ensemble and kernel-based methods such as Random Forests, XGBoost, and SVMs consistently emerge as top-performing approaches across a wide range of churn applications (Geiler and Klein, 2022; Manzoor et al., 2021; Sam and Kannan, 2023). Comprehensive reviews by Manzoor et al. (2021) and Geiler and Klein (2022) demonstrate that while black-box models often achieve the highest predictive accuracy, interpretable ensemble methods frequently offer a strong compromise between performance and explainability, making them attractive for business practitioners. Empirical studies in related domains, such as telecommunications, further confirm the strong performance of tree-based ensembles, particularly XGBoost and Random Forests, even in the presence of class imbalance (Lalwani et al., 2019; Sam and Kannan, 2023).

Based on these insights, this study adopts a broad but focused model set that reflects both state-of-the-art practices in churn prediction research and the practical constraints imposed by Meewind's operational and interpretability requirements. Both random and temporal validation strategies were implemented to assess model generalization and real-world applicability, while imbalance correction was performed using SMOTE and cost-sensitive learning. Hyperparameter optimization via Grid Search, Random Search, and Bayesian Optimization was used to further enhance model performance.

Practical and Methodological Challenges Despite the clear strategic relevance of churn prediction, applying machine learning in this context presents several non-trivial challenges. First, the available data was not originally designed for analytical purposes. Meewind's database primarily supports operational processes and regulatory compliance, resulting in a highly normalized relational structure with many variables that are intended for front-end functionality rather than behavioral analysis. Transforming this operational database into an analytics-ready dataset required substantial preprocessing, including table aggregation, feature construction, and the careful selection of variables that meaningfully reflect investor behavior.

A particular difficulty arises from the temporal nature of investor activity. Although churn is inherently a time-dependent phenomenon, much of the raw temporal information is not retained in its original form. Intermediate states and historical changes are often overwritten or removed as part of routine database maintenance. As a result, key behavioral indicators, such as changes in investment patterns or engagement intensity over time,

had to be reconstructed indirectly through feature engineering. This increases both the complexity of the modeling process and the risk of information loss, which makes robust validation strategies essential.

Second, the dataset is characterized by severe class imbalance. Only a small fraction of investors ultimately churn, while the vast majority remain active. Although this reflects the desirable business reality of high retention, it poses a significant challenge for machine learning models. Standard classification algorithms tend to favor the majority class, leading to a misleadingly high accuracy while failing to identify churners. Without appropriate correction, models may appear performant but offer little practical value for retention strategies. Addressing this imbalance is therefore a central methodological concern throughout the study.

Finally, model interpretability imposes an additional constraint. While highly complex black-box models often achieve superior predictive performance, Meewind explicitly requires transparency and explainability to support trust, regulatory accountability, and actionable decision-making. This requirement limits the unrestricted use of opaque deep learning approaches and necessitates a careful balance between predictive accuracy and interpretability. Consequently, the selected modeling approach prioritizes algorithms that allow insight into feature importance and decision logic, even when this comes at the cost of additional modeling complexity.

Taken together, these challenges underline that churn prediction at Meewind is not merely a modeling exercise, but a comprehensive data and methodological problem. Successfully addressing it requires combining domain understanding, careful data preparation, tailored validation strategies, and the deliberate selection of interpretable yet powerful machine learning models.

Contribution The contribution of this thesis is threefold, encompassing methodological, empirical, and practical dimensions.

First, from a methodological perspective, this study develops and evaluates a comprehensive machine learning framework for churn prediction that is specifically designed for real-world deployment in financial institutions. Rather than focusing on a single algorithm, the research systematically compares a broad range of interpretable and ensemble-based classification models under consistent evaluation conditions. The framework addresses several challenges that are frequently treated in isolation in existing literature, including feature engineering from operational data, severe class imbalance, hyperparameter optimization under computational constraints, and validation under both random and temporal data

splits. By integrating these elements into a single, coherent pipeline, the thesis provides a structured and reproducible approach that can be adapted to similar churn prediction problems in other investment and financial service contexts.

Second, the thesis contributes empirically by applying churn prediction techniques to the domain of sustainable investment funds, a setting that remains underrepresented in the churn prediction literature. Unlike commonly studied sectors such as telecommunications or subscription-based services, investor churn in sustainable finance is characterized by long-term participation, trust-based relationships, and irregular behavioral signals. The transformation of Meewind's operational database into an analytics-ready dataset, including the reconstruction of temporal features that were not explicitly stored, illustrates how meaningful predictive signals can be extracted even when historical behavioral data is incomplete or fragmented. While the specific feature engineering is tailored to Meewind's data environment, the identified types of indicators, such as engagement dynamics, tenure effects, and changes in interaction behavior, are likely to be relevant across comparable investment settings.

Third, the research delivers direct practical value for Meewind by providing an interpretable decision-support tool for proactive churn management. The resulting models are designed to be used by business stakeholders rather than as fully automated systems, generating prioritized lists of account identifiers that can be targeted through focused marketing, communication, or retention strategies. By emphasizing interpretability alongside predictive performance, the framework aligns with regulatory expectations and organizational requirements within the financial sector. In addition, the structured evaluation strategy ensures that model outputs can be trusted when applied to future data, supporting informed and timely intervention. As such, the thesis bridges the gap between academic machine learning research and actionable business intelligence in a sustainable finance context.

Overview of Results The empirical results indicate that machine learning models are capable of identifying meaningful patterns associated with investor churn at Meewind, despite the severe class imbalance and the limitations of the available operational data. Baseline models performed poorly in identifying churners, confirming that accuracy alone is insufficient in this context and that imbalance-aware modelling is essential. After applying class imbalance correction and systematic hyperparameter optimisation, ensemble-based methods substantially improved minority-class detection. In particular, tree-based ensembles outperformed simpler models, with XGBoost emerging as the strongest overall

performer. The final tuned XGBoost model achieved a recall of approximately 77% for churners while maintaining a strong ROC-AUC under both random and temporal validation schemes. Importantly, performance remained stable when evaluated on future, unseen data, suggesting that the learned relationships generalize beyond the training sample. Feature importance and SHAP analyses further demonstrate that the resulting model is not only predictive but also interpretable, revealing economically intuitive drivers of churn related to transaction behaviour, engagement dynamics, and investor tenure. Together, these findings suggest that machine learning can support a structured and data-driven approach to proactive churn management at Meewind.

2

Background

Before presenting the related literature on churn, it is essential to outline the methodological background that underpins the models used in this study. Customer churn prediction is based on a wide range of machine learning techniques, each with its own assumptions, strengths, and limitations. Although the following sections do not aim to provide an exhaustive treatment of these methods, they summarize the concepts that a reader should be familiar with in order to understand the modelling choices made in this research. The models range from interpretable statistical models to advanced ensemble algorithms, this background establishes the foundation on which the subsequent analysis is built.

2.1 Logistic Regression

Logistic regression is one of the most widely used statistical methods for binary classification problems and has become a common baseline in churn prediction tasks. Its popularity comes from its balance between simplicity, interpretability, and predictive power. Although its origins are in statistics, logistic regression is frequently applied in machine learning contexts as a supervised classification technique (Sperandei, 2014).

The method is closely related to linear regression, as both estimate relationships between a dependent variable and one or more independent variables. However, unlike linear regression, which assumes a continuous outcome and models it as a linear function of predictors, logistic regression is designed for categorical outcomes, most commonly binary variables, such as churn versus non-churn. Directly applying linear regression to binary results can lead to predictions outside the $[0, 1]$ interval and violate statistical assumptions such as homoscedasticity and normality of errors. Logistic regression resolves this by modelling

the probability of the outcome using the logistic (sigmoid) function, which ensures that the predictions fall within valid probability bounds.

Formally, the model can be expressed as:

$$P(Y = 1 \mid X) = \pi(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k)}},$$

where Y is the binary dependent variable, $X_1, X_2, \ldots, X_k$ are the independent variables, and $\beta_0, \beta_1, \ldots, \beta_k$ are the estimated parameters. Instead of modelling the probability directly, logistic regression models the log-odds (logit) of the outcome:

$$\text{logit}(\pi(X)) = \ln\left(\frac{\pi(X)}{1 - \pi(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k.$$

This transformation links the linear predictor to the probability space, allowing coefficients to be interpreted as the change in the log-odds of the outcome associated with a one-unit increase in the predictor, holding other variables constant. Exponentiating coefficients yield odds ratios, which are often more intuitive to interpret in applied contexts such as customer churn.

Logistic regression can accommodate both continuous and categorical independent variables, making it highly versatile. The flexibility to include interaction terms and polynomial effects also enables the model to capture more complex relationships, although care must be taken to avoid overfitting.

Another important consideration is the inclusion and selection of variables. Since logistic regression assumes a linear relationship between predictors and the logit of the outcome, irrelevant or highly correlated predictors can degrade model performance. Various strategies exist to guide variable selection, including stepwise procedures, information criteria such as AIC and BIC, and regularization techniques (e.g., Lasso or Ridge regression). Sperandei (2014) emphasizes that careful selection of predictors, informed by both statistical fit and theoretical understanding of the problem domain, is essential to ensure that the model remains both interpretable and robust.

2.2 Decision trees

Having examined the previous method, the natural next step is to turn to decision trees, one of the most widely used techniques in machine learning and data mining. Decision trees are intuitive models that represent a series of hierarchical decisions that resemble the structure of a flow chart. Each internal node corresponds to a test on an attribute, each

branch represents the outcome of that test, and each leaf node denotes a final prediction or class assignment (Rokach and Maimon, 2005). Their simplicity and transparency have made them popular in business applications, where interpretability is often as important as predictive accuracy.

The construction of a decision tree involves recursively partitioning the data into subsets that are increasingly homogeneous with respect to the target variable. At each node, the algorithm selects the attribute that maximizes the separation of classes, a process guided by the splitting criteria. Among the most common are the information gain, which is derived from Shannon's entropy, and the Gini index. For a dataset S with class proportions p_i, the entropy is defined as:

$$H(S) = -\sum_{i=1}^{k} p_i \log_2 p_i,$$

and information gain measures the reduction in entropy achieved by splitting on a given attribute. Similarly, the Gini index,

$$G(S) = 1 - \sum_{i=1}^{k} p_i^2,$$

quantifies impurity, with lower values indicating purer subsets. These measures ensure that the chosen splits maximize class separation, leading to clearer decision boundaries.

Recursive partitioning continues until a stopping criterion is reached, which may be based on maximum tree depth, minimum sample size at a node, or the absence of significant improvement in predictive performance. Without such a criterion, a tree can grow excessively large, capturing noise in the training data.

A central challenge with decision trees is their tendency to overfit. Fully grown trees often exhibit high variance, perfectly classifying the training set, but performing poorly on unseen data. To mitigate this, pruning strategies are applied. Pre-pruning halts tree growth early, based on stopping rules such as maximum depth or minimum leaf size, while post-pruning involves first growing a large tree and then removing branches that contribute little to predictive accuracy. Techniques like cost-complexity pruning balance the trade-off between model complexity and generalization by penalizing overly complex trees. These approaches reduce variance and improve the robustness of the model.

Decision trees offer several important advantages. They are highly interpretable and can be visualized in a way that makes the decision process transparent to managers and stakeholders. They require little preprocessing of data, as they can naturally handle both

categorical and continuous variables, and they are capable of modeling non-linear relationships and variable interactions.

However, these strengths are tempered by significant weaknesses. Trees are notoriously unstable: small changes in the training data can lead to very different structures, making them sensitive to noise. Furthermore, a single tree often has limited predictive power compared to more advanced models, particularly when patterns are complex and involve subtle interactions among variables. This has motivated the development of ensemble methods, such as Random Forests and Gradient Boosted Trees, which aggregate the predictions of many trees to achieve greater accuracy and stability.

2.3 Random Forest

Closely related to decision trees is the Random Forest method, first introduced by Breiman (2001). Although decision trees are simple and interpretable, they are also prone to instability and overfitting. Random Forests address these weaknesses by constructing an ensemble of decision trees and combining their predictions, thereby improving both accuracy and robustness.

The key idea behind Random Forests is the combination of bagging and random feature selection. In bagging, multiple bootstrap samples are drawn from the training dataset and a separate decision tree is fitted to each sample. The predictions are then aggregated, by majority vote in the classification tasks or by averaging in the regression tasks, to produce the final result. This reduces variance by averaging over many different models, counteracting the instability of single trees.

To further enhance diversity among trees, Random Forests introduce randomness at the feature level. At each split in a tree, only a random subset of predictors is considered when determining the best split. This mechanism decorrelates the trees, preventing a few strong predictors from dominating all splits and increasing the ensemble's ability to capture varied aspects of the data.

The prediction function of a Random Forest classifier can be expressed as:

$$\hat{f}(x) = \text{majority\_vote}\left(h_1(x), h_2(x), \ldots, h_B(x)\right),$$

where $h_b(x)$ denotes the prediction of the b-th tree and B is the total number of trees in the forest. As $B \to \infty$, the law of large numbers ensures that the ensemble converges to a stable predictor with low variance. Breiman (2001) formalized this improvement in terms of the accuracy of individual trees and the degree of similarity among trees, showing that

a Random Forest achieves strong performance when individual trees are both reasonably accurate and weakly correlated.

Random Forests offer several advantages. They retain much of the interpretability of decision trees, while greatly improving predictive accuracy and stability. They can naturally handle high-dimensional data, mixed variable types, and complex, non-linear interactions. Furthermore, Random Forests provide internal estimates of the generalization error through out-of-bag (OOB) samples, those observations not included in a tree's bootstrap sample, which can be used for unbiased performance evaluation. Variable importance measures are another valuable by-product, allowing researchers to identify which features contribute most to prediction.

However, Random Forests are not without drawbacks. They sacrifice some interpretability compared to a single decision tree, as the ensemble structure makes it harder to trace individual predictions. Computational costs can also be significant, especially with large datasets and many trees. In addition, while Random Forests reduce overfitting compared to individual trees, they may still underperform more sophisticated ensemble methods like gradient boosting in scenarios with highly imbalanced data or subtle class boundaries.

2.4 Gradient Boosting

Building upon the ideas of ensemble learning, another influential method is Gradient Boosting, originally formalized by Friedman (2001) and later elaborated in more practical detail by Natekin and Knoll (2013). While Random Forests reduce variance by averaging many decorrelated decision trees, Gradient Boosting takes a different approach: it builds trees sequentially, each new tree focusing on correcting the residual errors of the previous ensemble. This iterative refinement process often leads to highly accurate models, making Gradient Boosting one of the most powerful machine learning techniques available for structured data.

At its core, Gradient Boosting is an additive model:

$$F_M(x) = \sum_{m=1}^{M} \gamma_m h_m(x),$$

where $h_m(x)$ are weak learners, typically shallow decision trees, and γ_m are weights determined during training. The algorithm progresses stage by stage: at each step m, a new learner h_m is fitted to the negative gradient of the loss function with respect to the

prediction of the current model. This explains the name "gradient boosting," as the method essentially performs gradient descent in function space rather than parameter space.

Friedman (2001) showed that this approach is highly flexible because it can optimize any differentiable loss function. For classification tasks like churn prediction, the logistic loss is common:

$$L(y, F(x)) = \log\big(1 + \exp(-yF(x))\big),$$

where $y \in \{-1, +1\}$ are class labels. By minimizing this loss iteratively, Gradient Boosting produces a sequence of trees that progressively reduce misclassification.

Gradient Boosting offers several strengths. Its sequential nature allows it to capture complex, non-linear patterns and subtle interactions between predictors. It supports a wide range of loss functions, which makes it highly versatile. In addition, regularization techniques such as shrinkage, subsampling , and tree constraints help control overfitting and improve generalization (Natekin and Knoll, 2013).

However, these same advantages come with challenges. Gradient Boosting is computationally more demanding than Random Forests, as trees are built sequentially rather than in parallel. Its performance is also highly sensitive to hyperparameter tuning, often requiring cross-validation and extensive experimentation. Moreover, while Gradient Boosting models can provide variable importance measures, their interpretability is lower than single decision trees, and even lower than Random Forests when boosted ensembles become very large.

2.5 Support Vector Machines

Support Vector Machines (SVMs) are powerful supervised learning models designed for classification and regression tasks. An SVM seeks to find a decision boundary, a hyperplane in feature space, that separates classes with the largest possible margin. The margin is defined as the distance from the hyperplane to the nearest data point of any class (Campbell and Ying, 2011). Maximizing this margin helps achieve better generalization, which is especially useful in churn prediction, where the cost of false negatives can be high.

Given a training set:

$$\{(x_i, y_i)\}_{i=1}^{n}, \quad x_i \in \mathbb{R}^d,\ y_i \in \{-1, +1\},$$

we want to find parameters (w, b) defining a hyperplane $w^\top x + b = 0$ so that:

$$y_i(w^\top x_i + b) \geq 1 \quad \text{for all } i$$

in the ideal case. The margin in this case is $\frac{2}{\|w\|}$. As most real-world data are not perfectly separable, for example due to noise or overlapping classes, SVM introduces slack variables $\xi_i \geq 0$ to allow some misclassification. The optimization problem becomes:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{subject to } y_i(w^\top x_i + b) \geq 1 - \xi_i,\ \forall i$$

Here $C > 0$ is a regularization parameter that controls the trade-off between maximizing the margin keeping and minimizing the classification error in the training data. A large C penalizes misclassification's heavily, small C allows more errors for a wider margin.

A particularly powerful aspect of SVMs is their ability to handle non-linear separations through what is known as the kernel trick. As Campbell and Ying (2011) explain, the idea is to map the original input data into a higher-dimensional feature space where a linear separation becomes possible. Rather than computing this mapping explicitly, which would often be computationally infeasible, kernel functions provide a way to calculate inner products in the transformed space directly. This implicit representation allows SVMs to capture complex relationships in the data without ever leaving the original feature space.

Several kernels have been proposed in the literature, each offering different inductive biases. One of the most widely used is the radial basis function (RBF) kernel, which measures similarity between points as an exponentially decaying function of their Euclidean distance:

$$K(x_i, x_j) = \exp\left(-\gamma\|x_i - x_j\|^2\right).$$

Here, the parameter γ determines how quickly similarity decays with distance, effectively controlling the locality of the decision boundary. Another common choice is the polynomial kernel, which captures interactions between features up to a specified degree, making it suitable when relationships among predictors are multiplicative rather than purely additive.

Incorporating kernels fundamentally changes the structure of the optimization problem. When expressed in its dual formulation, the decision function of an SVM can be written as:

$$f(x) = \text{sign}\left(\sum_{i=1}^{n}\alpha_i y_i K(x_i, x) + b\right),$$

where the α_i are Lagrange multipliers obtained by solving a quadratic programming problem. Importantly, only a subset of the training points, those with $\alpha_i > 0$, known as the support vectors, contribute to the final classifier. This property highlights the efficiency

and parsimony of SVMs: rather than relying on all data points, the decision boundary is determined only by those examples that lie closest to it, which makes the model particularly effective at focusing on the most informative cases.

One of the key strengths of Support Vector Machines lies in their ability to capture complex, non-linear decision boundaries through the use of kernel functions. By focusing on maximizing the margin between classes, SVMs also achieve strong generalization, reducing the risk of overfitting even in high-dimensional spaces where traditional methods may struggle. Another advantage is their robustness in situations where the number of features exceeds the number of observations, provided that appropriate regularization is applied.

However, these benefits come with notable limitations. SVMs are highly sensitive to the choice of hyperparameters, such as the regularization constant C, the kernel type, and kernel-specific parameters such as γ. Selecting the right configuration often requires extensive cross-validation and can be computationally demanding. Scalability presents another challenge: While linear SVMs can be trained relatively efficiently, non-linear variants that rely on kernel methods involve solving quadratic programming problems that grow prohibitively expensive as datasets become larger. Finally, interpretability is a persistent concern. Unlike logistic regression and decision trees, which produce coefficients that can be directly linked to predictors, SVM decision functions depend on support vectors and kernel evaluations. This dependency makes it difficult to derive clear managerial insights about the relative importance of individual variables.

2.6 Extremely Randomized Trees (ExtraTrees)

Extremely Randomized Trees, or ExtraTrees, were introduced by Geurts et al. (2006) as an extension of the Random Forest methodology. Like Random Forests, they are an ensemble of decision trees designed to improve predictive accuracy and reduce variance compared to individual trees. However, ExtraTrees increase the level of randomness introduced during tree construction in order to further diversify the ensemble and prevent overfitting.

The two key differences between Random Forests and ExtraTrees lie in sampling and split selection. First, while Random Forests are built on bootstrap samples of the original dataset, ExtraTrees often use the entire training sample, reducing the variance introduced by resampling. Second, and more importantly, ExtraTrees do not choose the best split among all candidate thresholds for a given feature. Instead, they generate random split thresholds and select the one that yields the best separation among this random subset.

This procedure significantly increases the diversity among trees in the ensemble, reducing correlation and thereby improving generalization.

Formally, for each candidate feature at a node, a random split point is drawn uniformly from the range of feature values, and the split is chosen based on the best criterion (e.g., Gini impurity) among these random thresholds. This mechanism reduces computational cost, since fewer candidate splits need to be evaluated, and increases randomness, which often yields improved performance in practice.

ExtraTrees offer several advantages. They are computationally efficient, since random thresholding is faster than exhaustively searching for the best split, and they often achieve predictive accuracy comparable to or higher than Random Forests. Furthermore, they retain many of the desirable properties of tree ensembles, such as handling mixed variable types, providing feature importance measures, and being relatively robust to outliers and noise. However, the high level of randomness of the model can sometimes lead to slightly higher bias compared to Random Forests, particularly in small datasets. Interpretability also remains limited relative to single decision trees.

2.7 Extreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting, commonly referred to as XGBoost, was introduced by Chen and Guestrin (2016) and has quickly become one of the most widely adopted machine learning algorithms due to its exceptional performance in a variety of domains. XGBoost builds on the framework of Gradient Boosting by incorporating several algorithmic and system-level optimizations that make it both more efficient and more regularized.

At its core, XGBoost is an additive model in which new trees are trained to correct the residual errors of the current ensemble, as in standard Gradient Boosting. However, XGBoost introduces a regularized objective function that penalizes both the number of leaves and the magnitude of leaf weights in each tree. This regularization term helps to control model complexity and reduce overfitting, a common issue in boosting methods. The objective function can be written as:

$$\mathcal{L}(t) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t),$$

where l is the differentiable loss function, f_t is the t-th tree, and $\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_j w_j^2$ is the regularization term, with T being the number of leaves and w_j the leaf weights.

In addition to regularization, XGBoost incorporates shrinkage (learning rate), column subsampling, and efficient handling of sparse data, which collectively improve its scalability and robustness. From a systems perspective, it is highly optimized for parallel and distributed computing, allowing it to handle very large datasets efficiently.

XGBoost has demonstrated state-of-the-art performance in a wide range of structured data problems, often outperforming traditional Gradient Boosting implementations. However, this power comes at the cost of increased complexity: the model has many hyperparameters that must be tuned carefully to achieve optimal performance. Its interpretability is also relatively low, though the importance metrics of the features can provide some insight.

2.8 Hyperparameter Tuning

Machine learning models have the potential to capture complex patterns in data, but their predictive power depends heavily on the choice of hyperparameters. Hyperparameters govern the learning process itself, for instance, the depth of decision trees, the regularization constant in support vector machines, or the learning rate in gradient boosting. Unlike model parameters, which are estimated from the data during training, hyperparameters must be specified in advance and calibrated to suit the properties of the dataset. Poorly chosen hyperparameters may result in models that are underfit, overfit, or fail to generalize, while well-tuned hyperparameters can significantly enhance performance.

Because hyperparameters cannot usually be derived analytically, a range of search and optimization strategies have been developed to identify effective configurations. These methods vary in their computational demands, efficiency, and ability to explore the search space. In the following subsections, we discuss the most widely used approaches, grid search, random search, and Bayesian optimization, each of which illustrates different trade-offs between thoroughness, efficiency, and sophistication in the tuning process.

2.8.1 Grid Search

Grid search is one of the most straightforward and commonly used approaches to hyperparameter tuning. In this method, the practitioner specifies a discrete set of candidate values for each hyperparameter and evaluates the model performance for every possible combination. As described by Agrawal (2020), grid search is attractive due to its simplicity and deterministic nature: all configurations within the predefined grid are evaluated, making the method easy to implement and interpret.

Despite these advantages, grid search quickly becomes computationally expensive as the number of hyperparameters or candidate values increases. This exponential growth in evaluations, often referred to as the curse of dimensionality, makes grid search impractical for complex models. Moreover, grid search allocates equal computational effort to all hyperparameters, regardless of their relative importance. As noted by Agrawal (2020), this can result in inefficient use of resources, particularly when only a small subset of hyperparameters significantly influences model performance.

2.8.2 Random Search

To address the inefficiency of grid search, Bergstra and Bengio (2012) introduced random search as a more resource-effective alternative. Instead of exhaustively evaluating all parameter combinations, random search samples hyperparameter configurations at random from predefined distributions. Each configuration is independently evaluated, and the sampling process does not take into account the results of previous runs, even when certain configurations yield particularly strong or weak performance. Although this approach may seem less rigorous, it often outperforms grid search in practice because it explores a larger portion of the search space when only a limited number of evaluations are feasible, a point also emphasized in the practical discussion of hyperparameter tuning by Agrawal (2020).

Random search is particularly advantageous when only a subset of hyperparameters substantially influences model performance. By sampling more broadly, it increases the probability of identifying promising regions of the search space without the computational burden of evaluating every combination. As argued by Bergstra and Bengio (2012), this behavior arises because random search allocates more trials to influential hyperparameters than grid search under a fixed evaluation budget. Its main limitation is that performance is inherently stochastic: different runs may yield different results, and there is no guarantee that the optimal configuration will be discovered. Nevertheless, the balance between efficiency and effectiveness has made random search a popular baseline for hyperparameter tuning.

2.8.3 Bayesian Optimization

A more sophisticated strategy is Bayesian optimization, which aims to model the relationship between hyperparameters and model performance using probabilistic surrogate functions. Rather than selecting configurations blindly, Bayesian optimization iteratively

chooses new hyperparameters to evaluate based on both prior knowledge and the outcomes of previous trials. Typically, a Gaussian process or another surrogate model is used to approximate the performance surface, and an acquisition function determines where to sample next.

Formally, Bayesian optimization proceeds iteratively by fitting a probabilistic surrogate model to all observed evaluations of the objective function. At iteration t, the surrogate provides a predictive distribution for the performance y at a candidate configuration $\boldsymbol{\lambda}$, typically assumed to be Gaussian with mean $\mu(\boldsymbol{\lambda})$ and standard deviation $\sigma(\boldsymbol{\lambda})$. Based on this predictive distribution, an acquisition function is used to quantify the utility of evaluating $\boldsymbol{\lambda}$. A commonly used acquisition function is the expected improvement (EI), defined as

$$\mathbb{E}[I(\boldsymbol{\lambda})] = \mathbb{E}\big[\max(f_{\min} - y, 0)\big], \tag{2.1}$$

where $f_{\min}$ denotes the best observed value of the objective function so far. When the predictive distribution of y is normal, the expected improvement can be computed in closed form as

$$\mathbb{E}[I(\boldsymbol{\lambda})] = (f_{\min} - \mu(\boldsymbol{\lambda}))\,\Phi\left(\frac{f_{\min} - \mu(\boldsymbol{\lambda})}{\sigma(\boldsymbol{\lambda})}\right) + \sigma(\boldsymbol{\lambda})\,\phi\left(\frac{f_{\min} - \mu(\boldsymbol{\lambda})}{\sigma(\boldsymbol{\lambda})}\right), \tag{2.2}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ denote the probability density function and cumulative distribution function of the standard normal distribution, as discussed by Hutter et al. (2019).

This approach is far more efficient than random or grid search, as it prioritizes exploring regions of the hyperparameter space that are likely to improve performance while avoiding unpromising areas. As a result, Bayesian optimization often identifies near-optimal hyperparameters with far fewer evaluations. Its main drawbacks are the increase in algorithmic complexity and the computational overhead associated with maintaining and updating the surrogate model. Furthermore, the method can become less effective in very high-dimensional search spaces. Despite these limitations, Bayesian optimization represents the state of the art for hyperparameter tuning, particularly when training models is expensive and each evaluation must be used judiciously.

3

Literature Review

3.1 Churn

Customer churn, the voluntary or involuntary ending of a relationship between a customer and a firm, remains one of the most important challenges for organizations operating in competitive and subscription-driven markets. A long-established principle in marketing is that retaining customers is considerably more cost-effective than acquiring new ones (Reichheld and Sasser, 1990). This cost asymmetry has been repeatedly confirmed in empirical studies. In telecommunications, for example, churn prediction models based on call detail records have shown that even small improvements in retention accuracy can yield substantial savings in marketing expenditure (Wei and Chiu, 2006). As acquisition costs continue to rise, companies across industries increasingly prioritize customer retention as a key strategic objective (Qureshi et al., 2013).

Predicting churn remains challenging because it is influenced by a multifaceted interplay of behavioral, service-related, and market-driven factors. Traditional statistical models often struggle to capture the nonlinear structure of these relationships, which has driven a widespread adoption of machine-learning approaches. Machine learning models can uncover subtle and complex patterns in customer engagement that would otherwise remain undetected (Huang et al., 2012). Ensemble methods, in particular, have been found to improve predictive stability by reducing bias and variance while leveraging diverse feature sets that represent customer transactions, service usage, and demographic characteristics. Empirical comparisons further show that advanced ensemble techniques such as XGBoost can outperform traditional tree-based algorithms, random forests, and gradient boosting machines in telecom based churn prediction (Ahmad, 2019). Similarly, in subscription contexts such as newspaper services, optimally tuned support vector machines have outper-

formed logistic regression and random forests in predictive accuracy and AUC (Coussement and den Poel, 2009).

However, model development involves more than just choosing an algorithm. The importance of feature engineering is strongly emphasized across the literature. High-quality behavioral and temporal features often contribute more to predictive performance than the choice of algorithm itself (Lalwani et al., 2019; Verbeke et al., 2012). By integrating domain knowledge into feature design, such as metrics of investor activity, portfolio interactions, or service usage patterns, models can more accurately differentiate between customers likely to churn and those likely to remain loyal. In banking, advanced techniques including rough set modeling and genetic programming have also been used to capture complex loyalty dynamics (Eiben and Michalewicz, 2003). A key insight from this work is that loyal customers not only exhibit higher profitability but are also less sensitive to price fluctuations and more likely to provide referrals, amplifying the long-term value of retention efforts.

Beyond methodological considerations, the prediction of churn has important managerial implications. Predictive models enable firms to deploy targeted retention strategies that are both cost-effective and customer-centric. Burez and den Poel (2007) demonstrated in the pay-TV sector that targeted interventions significantly outperform untargeted campaigns by focusing resources on customers who are both likely to churn and likely to respond. More broadly, research emphasizes that predictive models should not only maximize accuracy, but also inform actionable retention decisions (Neslin et al., 2006). This evidence aligns with CRM theory, which argues that firms that systematically manage customer initiation, maintenance, and retention processes perform better financially (Reinartz and Kumar, 2005).

Although the telecommunications industry has historically provided the richest datasets for churn research, the conceptual and methodological insights extend well beyond telecom. For organizations such as Meewind, which operates in the sustainable investment domain, the ability to predict and prevent churn is equally valuable. Applying the lessons from the telecommunications sector, Meewind can leverage machine learning models to identify at-risk investors, design targeted interventions, and ultimately improve customer lifetime value. In this sense, the methodologies reviewed not only provide a foundation for model selection but also highlight the broader strategic role of churn prediction as a driver of sustainable growth.

3.2 Class Imbalance in Churn Prediction

Customer churn prediction frequently faces the challenge of class imbalance, in which the number of non-churning customers far exceeds the number of churners. This imbalance causes predictive models to be biased towards the majority class, resulting in high overall accuracy but poor identification of the minority class, which is often of greatest business interest (Burez and Van den Poel, 2009). Addressing class imbalance is therefore critical for developing models that can accurately identify at-risk customers and support effective retention strategies.

Several strategies have been proposed to mitigate the effects of class imbalance, broadly categorized into data-level, algorithm-level. At the data level, classical methods include random under-sampling of the majority class, random over-sampling of the minority class, and more advanced synthetic sampling approaches such as SMOTE (Guo et al., 2008). Under-sampling reduces the dominance of the majority class, but risks discarding potentially informative data, which can reduce overall model generalizability. Over-sampling the minority class, on the other hand, can balance the dataset without losing information, but may lead to overfitting, especially when the number of minority instances is small. Synthetic approaches like SMOTE generate artificial samples by interpolating between existing minority instances, mitigating overfitting but introducing the risk of creating unrealistic or noisy samples.

Algorithm-level solutions incorporate the imbalance directly into the learning process. Cost-sensitive learning, for example, assigns higher misclassification costs to the minority class, encouraging models to focus on accurately predicting churners (Burez and Van den Poel, 2009). Ensemble methods, such as weighted random forests or boosting, combine multiple models and can integrate resampling or cost adjustments to improve minority class prediction. The main advantage of these approaches is that they exploit the full dataset while explicitly addressing imbalance, but they can increase model complexity and computational cost.

Evaluation metrics play a critical role when the imbalance problem is addressed. Standard accuracy is often misleading, as it can mask poor minority-class performance. Metrics such as AUC, F1-score, precision–recall curves, and lift provide a more informative assessment (Guo et al., 2008; Vafeiadis et al., 2015). Additionally, Vafeiadis et al. (2015) highlight that carefully combining resampling, cost-sensitive learning, and domain-informed feature engineering substantially improves predictive performance for minority-class churners.

Recent studies also underscore the interplay between feature engineering and imbalance. Subtle churn signals can be overwhelmed by the majority-class patterns, making domain knowledge crucial. Features that reflect declines in engagement, portfolio interactions, or service satisfaction can strengthen minority-class signals prior to resampling or algorithmic adjustment (Lalwani et al., 2019).

Taken together, the literature indicates that the most robust churn prediction pipelines integrate complementary strategies: domain-informed feature engineering, appropriate resampling or cost-sensitive adjustments, and evaluation metrics tailored to minority-class detection. For organizations such as Meewind, where churn events are infrequent but economically significant, addressing class imbalance is central to building models that reliably identify at-risk investors and support effective, targeted retention strategies.

3.3 Machine learning techniques for detecting churn

Modern data-driven approaches have transformed the way organizations analyze and predict customer behavior. Rather than relying solely on fixed statistical models or expert-defined rules, contemporary methods leverage algorithms that can automatically detect patterns and relationships within large datasets and adapt as more information becomes available. This paradigm, widely referred to as machine learning (ML), enables the construction of predictive systems that improve performance over time by learning directly from data (Bishop, 2006).

In the context of customer churn prediction, such adaptability is particularly valuable. Churn is rarely the result of a single factor; instead, it emerges from a combination of behavioral, demographic, and contextual influences that interact in complex and often nonlinear ways. Traditional techniques often struggle to capture these dynamics, whereas ML methods excel at uncovering subtle signals that may serve as early indicators of churn.

One of the key strengths of ML lies in the diversity of available approaches. On one end of the spectrum are interpretable models, such as logistic regression and decision trees, which can highlight the main drivers of churn in a transparent manner. On the other end are more advanced approaches, such as ensemble methods, which are capable of capturing intricate dependencies in high-dimensional datasets. Together, these techniques provide both explanatory insight and strong predictive performance.

Previous literature on churn prediction has focused primarily on logistic regression, decision trees, random forests, gradient boosting, and support vector machines (Huang et al.,

2012; Vafeiadis et al., 2015). These models provide strong baselines and cover a range from simple, interpretable techniques to more complex, accuracy-oriented approaches.

In addition to these established methods, the present study also considers several related algorithms that, while less common in the churn prediction literature, are widely recognized in machine learning. These include Extremely Randomized Trees (ExtraTrees), which increase diversity in tree ensembles through randomized splits; and Extreme Gradient Boosting (XGBoost), a highly efficient and regularized variant of boosting that has shown strong performance in many applied prediction tasks.

Together, these methods broaden the evaluation beyond the models most frequently cited in prior research, allowing for a more comprehensive assessment of predictive approaches for churn prediction at Meewind.

4

Data Exploration and Preparation

4.1 Raw Data Structure

The company maintains a relational database designed primarily to support operational processes and regulatory compliance. Consequently, a substantial portion of the stored information is meant for front-end functionality and legal verification rather than for analytical or predictive purposes. This distinction is crucial because it determines which variables can be directly utilized for modelling and which require transformation through feature engineering to become analytically meaningful.

A notable example of non-analytical data concerns the binary variables maintained for compliance verification. Each account in the system must meet strict legal requirements, such as validated banking information and verified identity documentation. This is reflected in the Person table, which contains demographic and regulatory attributes, including multiple binary indicators confirming whether mandatory documents have been submitted. Although essential for operational integrity, such variables contribute little explanatory value to churn prediction and are therefore excluded from further analysis.

At the core of the database lies the Account table, which serves as the primary linking entity across all data sources. Each account is associated with at least one individual, although joint ownership is also common, allowing multiple persons to be connected to a single account. An individual may have several accounts, including shared ones. For analytical purposes, the focus is placed on the first registered person in each account to maintain a consistent one-to-one mapping between customers and accounts.

Surrounding this central entity, several complementary tables record different aspects of customer behavior and account activity. These include transaction histories, order placements, investment patterns, and dividend reinvestment actions. Together, they form

the foundation from which behavioral and temporal features are engineered in subsequent steps.

- *Holdings*: current portfolio compositions,
- *Incassos*: records of direct debit instructions,
- *Orders*: transactions related to the sale of participations,
- *Transactions*: transactions related to the purchase of participations,
- *Login logs*: timestamps of account access,
- *SGG Transactions*: all cash and participation movements associated with accounts, provided by executor IQEQ,
- *Portfolio history*: monthly account valuations.
- *Users*: registers activities on the site through API's
- *Fund Values*: prices of all funds through the years

From an exploratory perspective, some variables can be used directly to generate descriptive insights. For example, demographic information allows the visualization of age distributions, while geographical information (e.g., postal codes) can be mapped to assess regional differences in the customer base (Figures 4.1 and 4.2).

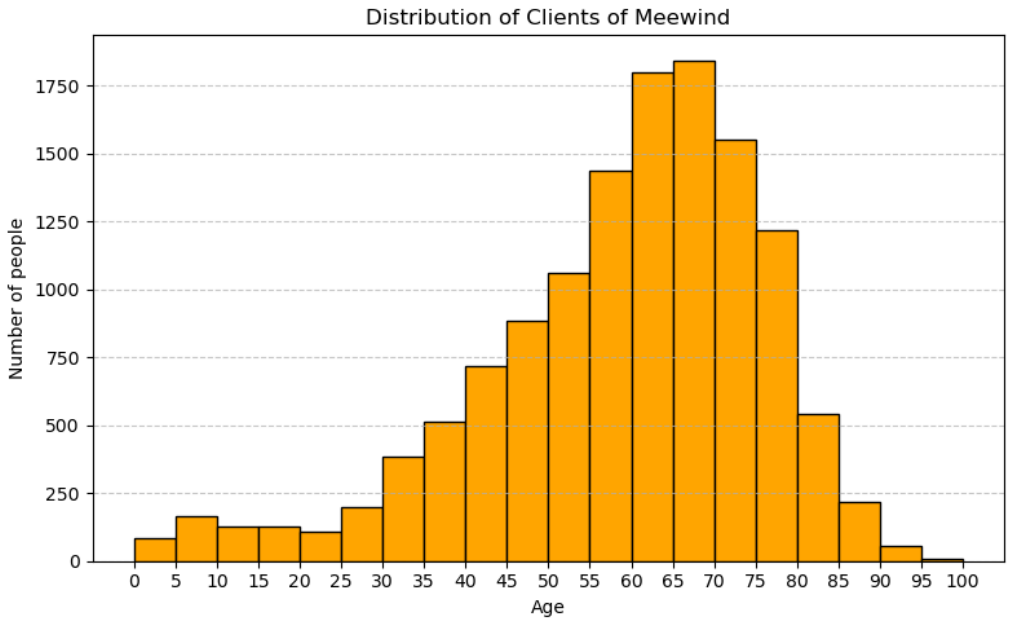

Figure 4.1: Distribution of customer ages based on the *Person* table.

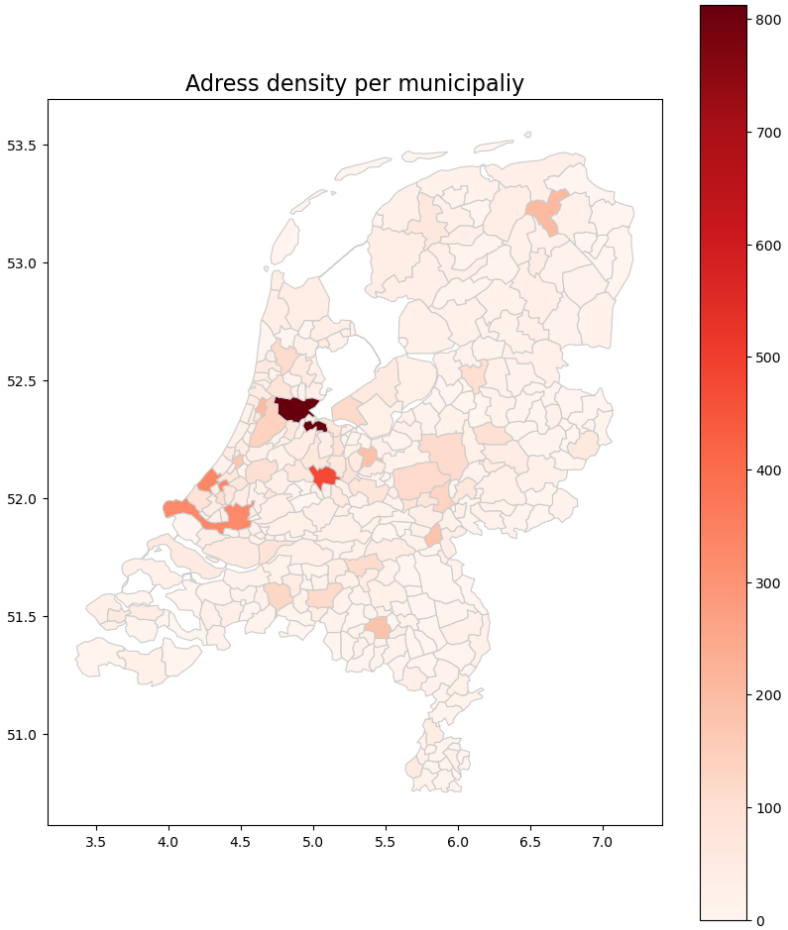

Figure 4.2: Geographical dispersion of customers by postal code region.

Time plays an equally crucial role in the dataset. Nearly all records are event-based, which means that they are collected when an action occurs, such as a trade, logging in, or cash movement. This temporal granularity makes it possible to reconstruct behavioral trajectories and capture dynamic processes such as engagement intensity, account growth, and churn. At the same time, the database design imposes limitations. For example, the Holdings table reflects only the current portfolio composition, thereby omitting historical snapshots. This gap requires reliance on the Portfolio history and Transactions tables to approximate longitudinal changes.

4.2 Feature Engineering

After the first exploratory steps, the dataset was systematically transformed to construct variables suitable for empirical analysis implementing feature engineering. Feature engi-

neering is the process of extracting relevant variables and information from raw data using domain knowledge and the task context. The final dataset spans the period from 2019 to the present, covering a changing set of customers over time. To align with the company's operational cycle, all features are aggregated and updated quarterly, corresponding to the execution of sales orders.

The feature engineering process builds on the Person, Account, Orders, and related tables described above, with transformations designed to capture demographic, behavioral, and transactional dimensions of customer activity. The main derived features are as follows:

Static Features Static features represent characteristics that remain constant over time and are determined at the point of account creation. These include personal and account-level information that provides a structural understanding of the customer base. First, the geographical information is derived from the participant's postal code, which is transformed to a Dutch province. This categorical variable is easier to interpret for a model and captures potential regional heterogeneity in investor behavior, as financial preferences and engagement levels can vary between regions.

Second, the type of account is included to differentiate between various forms of participation at Meewind. Accounts can be individual, joint, child, or corporate. Since these account types represent distinct user groups with different investment goals and decision patterns, distinguishing between them allows the model to learn from behavioral variation across customer segments.

Finally, each customer is assigned a risk profile determined by an external investment advisor. This classification reflects the general assessment with respect to investment risk tolerance and compliance of a participant, incorporating factors such as country of origin, exposure to sanction lists, potential fraud indicators and invested capital. Including this variable helps contextualize customer behavior within the broader framework of regulatory and risk-based segmentation.

Customer Age and Account Tenure Calculated by measuring the difference between a relevant datetime field (e.g., date of birth, account creation date) and a reference point in time, expressed in years.

Login Activity User engagement is quantified by counting logins within the three months preceding each reference date. Additionally, the feature captures changes in login behavior

by comparing activity in the most recent period to the prior period, expressed as a percentage change. This approach allows the model to account for both the level and trend of engagement, highlighting customers whose activity is changing and who may be at higher risk of churn.

Account Performance Differences in monthly account values relative to a reference date, computed for multiple lag periods (3, 6 and 12 months), capturing both short-term and long-term performance trends, while ensuring no data leakage occurs. The data leakage is prevented but looking at the portfolio value of an account the month before the sale date.

Systematic Investments Binary indicators that indicate whether a customer had an active systematic investment agreement on a given reference date. If this was the case, the amount of the agreement is also stored; otherwise, this is set to 0.

Investment and Dividend Actions Binary values indicating whether customers purchased new shares, reinvested dividends, or withdrew dividends in cash during the six months preceding each reference date. These features provide a fine-grained distinction between different investment behaviors and preferences.

Trading Activity To capture medium-term investment behavior, the number of specific actions was counted for each customer within the two years preceding the reference date. This feature provides insight into trading frequency and intensity, allowing differentiation between passive investors, who rarely adjust their portfolios, and more active traders. By incorporating both the type and volume of actions, it complements short-term indicators such as dividend reinvestments and purchase flags, while emphasizing patterns of longer-term engagement.

Quarterly Churn Trend Tracking sentiment at an aggregate level is also a powerful tool when looking at behaviors in general. To record this, a feature was engineered that records the number of churn events in the quarter preceding each reference date. By systematically tracking this quarterly churn activity, the model is able to incorporate broader behavioral dynamics and identify potential sentiment-driven patterns. For example, an increase in churn in the most recent quarter can signal declining customer confidence or market-related stress, which in turn, could increase the likelihood of further churn in the subsequent period.

Together, these engineered features integrate demographic, transactional, and behavioral information into a unified dataset, enabling the analysis of long-term patterns as well as short-term customer dynamics.

4.3 Data Cleaning

Before further analysis, the new dataset was subjected to a series of cleaning steps to ensure consistency and plausibility of the values. Since the database is primarily designed for operational purposes, certain anomalies were present that distorted descriptive analyses and feature construction.

The first correction concerned the sell date variable. In some cases, sales were registered on 31 December instead of 1 January for tax reasons, even though they were part of the same transaction. To ensure temporal consistency, all such cases were adjusted so that these transactions were uniformly recorded as occurring on the first day of the month.

Another prominent issue was found in the age variable, which contained implausible values, including negative numbers and ages higher than 100 years. As these values are unlikely to represent valid records, the corresponding rows were removed. Similarly, duplicate entries were identified and dropped; these often arise because every fund requires its own order registration. Customers within Meewind often have participations in multiple funds, but actions that impact several funds are still documented multiple times.

In addition, several binary indicator variables contained missing values. To maintain interpretability and ensure that the absence of confirmation was not mistaken for a positive signal, all missing values in these columns were recoded as zero. Furthermore, the categorical features were also cleaned. Several steps were taken to achieve this. First, NaN entries were placed in their own category labelled “Missing”, as the absence of a value can carry informative meaning. Rare categories were also removed, as they were often the result of input errors or inconsistencies.

For the remaining variables, rows with missing values were excluded to prevent biases introduced by incomplete records. These preprocessing steps result in a dataset that is both cleaner and more reliable for subsequent analysis while retaining sufficient coverage of the customer population.

4.4 Descriptive Statistics and Correlation Structure

Following the cleaning and feature engineering procedures described above, the dataset comprises 262,370 quarterly account-level observations. Table 4.1 presents the summary statistics of the main numeric and binary variables.

Variable	Count	Mean	Std	Min	Median	Max
Account Type ID	262,370	1.12	0.37	1	1	5
Newsletter (binary)	262,370	0.93	0.25	0	1	1
Amount direct debit	262,370	15.43	74.67	0	0	2,575
Active direct debit (binary)	262,370	0.12	0.33	0	0	1
DK Dividend reinvested (last 6 months) (binary)	262,370	0.41	0.49	0	0	1
DU Dividend paid out (last 6 months) (binary)	262,370	0.28	0.45	0	0	1
KS New investments (last 6 months) (binary)	262,370	0.13	0.34	0	0	1
K Total buy orders (last 2 years)	262,370	4.06	4.02	0	3	214
V Total sell orders (last 2 years)	262,370	2.29	2.73	0	1	22
Account Tenure (years)	262,370	3.48	1.99	1	3	8
Age (years)	262,370	56.65	16.34	0	59	100
Login Count (recent 3 months)	262,370	0.08	0.69	0	0	107
Login Count (previous 3 months)	262,370	0.07	0.83	0	0	154
Login Count (% change)	262,370	-1.47	12.23	-100	0	600
Account Value (ref.)	262,370	27,517	139,683	1.09	10,823	13,927,510
Value Change (last 3 months)	262,370	300	8,908	-1,227,332	58	969,306
Value Change (last 6 months)	262,370	768	12,356	-1,692,679	131	1,035,394
Value Change (last 12 months)	262,370	1,773	16,849	-2,140,862	311	1,333,117
Quartile Count (previous quarter)	262,370	301.12	205.12	40	263	1,052
Sell order (binary)	262,370	0.019	0.14	0	0	1

Table 4.1: Summary statistics of the engineered dataset.

Several patterns emerge. Customers have a relatively long average account tenure of 3.5 years, while the mean age is 56.7 years, indicating a relatively mature investor base. Most clients are subscribed to the company newsletter (93%). The account balances are highly skewed, with a median value of €10,823 compared to a mean of €27,517, reflecting a small group of very large investors. Engagement, measured through logins, is low on average, but exhibits substantial variation, with some customers logging in more than 100 times in a three-month period.

To explore potential multicollinearity and the underlying relationships, a correlation matrix was also constructed (Figure 4.3). The strongest correlations are observed among financial performance measures, such as value changes across different horizons and account value. Expectedly, buy and sell activity (K and V) are also correlated, while dividend reinvestment (DK), dividend payout (DU) and new investments (KS) cluster together with moderate associations. Login activity shows only weak correlations with financial

and demographic variables, but login counts across periods are naturally interrelated. Age and tenure remain largely independent of financial actions, suggesting that demographic, transactional, and behavioral features provide complementary information to the analysis.

The most relevant indicator for predicting churn is *is_ref_date*, since it records if an account churns in the given period. Although its pairwise correlations are modest overall, the highest associations are found with account value, recent value changes (3, 6, and 12 months), login activity, and the previous quartile count. This suggests that both financial performance and engagement patterns carry predictive information about churn. Although these linear relationships are relatively weak in isolation, they provide valuable input features for subsequent machine learning models, which can exploit nonlinear interactions and combinations of variables to improve predictive accuracy.

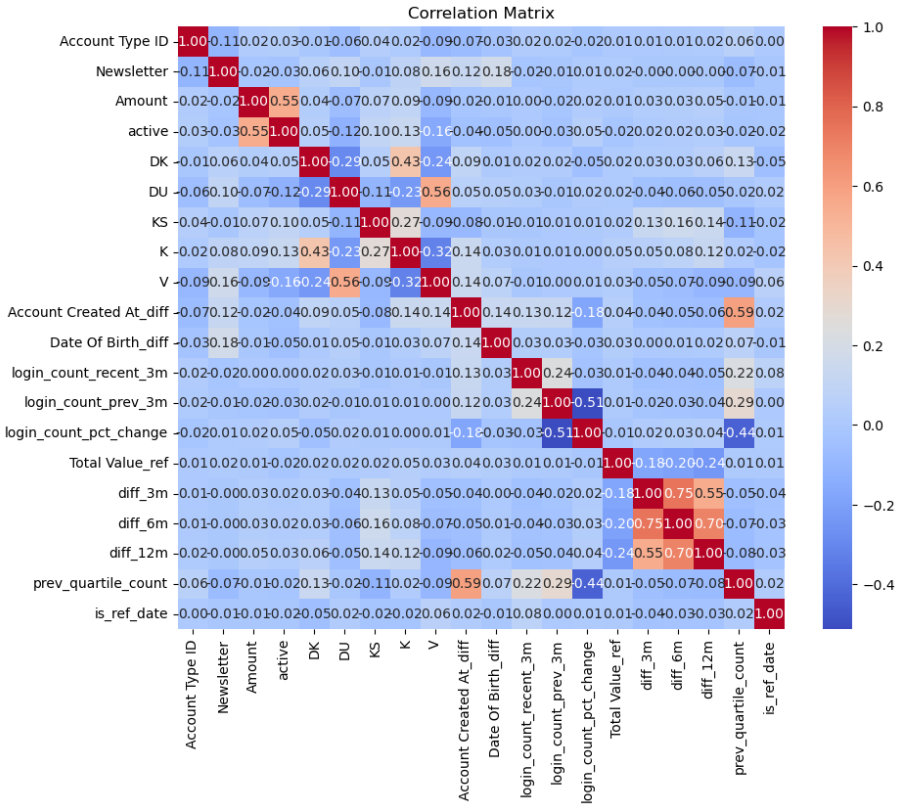

Figure 4.3: Correlation matrix of main features.

4.5 Class Imbalance in the case of Meewind

In the Meewind dataset, class imbalance is a particularly pronounced challenge: only 4903 of 262370 instances correspond to churn, representing just 1.9% of the total. Such an

extreme imbalance poses difficulties for machine learning models, which tend to be biased toward the majority class. As highlighted by Burez and Van den Poel (2009), standard algorithms trained on imbalanced data will achieve high overall accuracy while failing to identify the minority class, in this case the churners who are of primary business interest. Similarly, Guo et al. (2008) emphasize that class imbalance can compromise generalization and reduce model robustness, particularly when predicting rare events.

Several strategies can be implemented to address this issue in practice. At the data level, resampling techniques are commonly used. Oversampling approaches, such as the Synthetic Minority Over-sampling Technique (SMOTE), generate artificial minority class examples to balance the dataset and reduce bias toward the majority class. In contrast, undersampling the majority class simplifies the dataset, but risks discarding valuable information; in the Meewind context, this approach alone would result in a significant loss of data. Although oversampling is generally more suitable, its impact should be carefully evaluated to ensure that it improves minority class recognition without introducing excessive noise.

Algorithm-level solutions complement these data-level adjustments. Cost-sensitive learning, which penalizes misclassification of minority class instances more heavily, encourages models to focus on correctly predicting churners and can be applied to a wide range of algorithms, including logistic regression, decision trees, and ensemble methods. In practice, this is often implemented by assigning higher weights to minority-class samples during training. The weights are automatically scaled inversely proportional to class frequencies using the formula

$$w_j = \frac{n_{\text{samples}}}{n_{\text{classes}} \cdot \text{count}(y_j)},$$

so that each class contributes more equally to the learning process. In the Meewind dataset this adjustment means that churn instances are assigned a weight roughly thirty times higher than non-churn instances. This rebalancing ensures that the learning algorithm treats churn cases as equally important during optimization, rather than being overwhelmed by the majority class. Consequently, cost-sensitive learning directly addresses the shortcomings of baseline models, where churners are most probably ignored, and improves the likelihood that churn prevention strategies can be targeted at the right customers.

Addressing class imbalance is critical in the Meewind case, as failing to identify potential churners may result in substantial revenue loss and missed opportunities for targeted retention strategies. By comparing data- and algorithm-level techniques, the l process can be tailored to improve minority class prediction while preserving the integrity and richness of the dataset.

5

Methodology

Once the data had been cleaned and prepared, the next step involved developing and evaluating the machine learning models. All algorithms described in Section 3.3 were first implemented in a baseline configuration to establish reference performance levels. In this baseline setting, no model-specific optimisation or imbalance correction was applied. The only preprocessing performed was the categorisation of features into numerical, categorical, and binary variables, ensuring that all models received appropriately formatted and actionable input data. These baseline models serve as a benchmark against which the impact of subsequent modelling decisions can be assessed.

The modelling process follows a deliberately sequential structure consisting of three stages: baseline estimation, class imbalance correction, and hyperparameter optimisation. This stepwise design allows each methodological intervention to be evaluated in isolation, making it possible to assess the incremental contribution of imbalance handling and tuning relative to the baseline. By structuring the analysis in this way, improvements in predictive performance can be attributed directly to specific modelling choices rather than to interacting effects that would be difficult to disentangle in a single, fully optimised model.

In the second stage, the issue of severe class imbalance is explicitly addressed. Two distinct approaches are examined: data-level imbalance correction and algorithm-level adjustments. Data-level correction is implemented using the Synthetic Minority Oversampling Technique (SMOTE), which is incorporated directly into the modelling pipeline. Algorithm-level correction is achieved by adjusting class weights within the classifier itself, thereby penalising misclassification of the minority class more heavily during training. For each model, these approaches are implemented as separate pipeline variants, while all other preprocessing steps remain unchanged. This design enables a direct comparison between imbalance correction strategies and ensures that the effectiveness of each method

can be evaluated independently for different algorithms. Importantly, addressing class imbalance at this stage allows validation that the minority class is adequately captured before introducing further performance enhancements.

The final stage of the modelling process focuses on hyperparameter optimisation. This step is intended to ensure that each model is well aligned with the underlying data structure and operates under its most suitable configuration. For each algorithm, a search space of relevant hyperparameters is defined based on existing literature and methodological best practices. Hyperparameter tuning is performed using three complementary strategies: Bayesian optimisation, grid search, and random search. Bayesian optimisation is implemented using the `skopt` library, while grid search and random search are conducted using `scikit-learn`. Each tuning strategy is applied through a pipeline that preserves the same preprocessing and evaluation framework used in earlier stages. As with imbalance correction, tuned models are accessed through explicit variant calls, ensuring consistency and reproducibility across experiments.

5.1 Model selection

The choice of predictive models in this study is guided by both methodological and practical considerations. Investor churn prediction at Meewind constitutes a binary classification problem characterized by heterogeneous feature types, complex non-linear relationships, temporal dynamics, and a pronounced class imbalance. In addition, the resulting model is intended to support operational decision-making within the organization, which places constraints on interpretability and transparency. To address these requirements, three broad categories of machine learning models are considered: regression-based models, tree-based models, and boosting-based ensemble methods. Within each category, multiple specific algorithms are evaluated to ensure robustness and to avoid reliance on a single modelling paradigm.

Regression-based models, such as logistic regression, provide a natural starting point for churn prediction. These models are well-established in both academic literature and industry applications, and offer a clear probabilistic interpretation of churn risk. Their linear structure makes them particularly suitable as baseline models, as estimated coefficients directly reflect the direction and relative magnitude of feature effects. This "glass box" property is especially valuable in a financial context, where stakeholders often require transparent explanations of model behaviour. Although linear models may struggle to capture complex non-linear interactions present in behavioural and transactional data,

they serve an important role as reference points against which more flexible models can be compared.

Tree-based models, including decision trees and ensemble variants such as random forests and extremely randomized trees, extend this framework by allowing for non-linear relationships and interaction effects without requiring explicit feature engineering. These models are well suited to tabular data with mixed feature types, as they rely on recursive partitioning rather than distance-based calculations. As a result, they are robust to monotonic transformations and less sensitive to feature scaling. From an interpretability perspective, tree-based methods retain a relatively transparent structure: individual decision trees can be visualized directly, and ensemble models allow for global interpretability through feature importance measures. This makes them a strong candidate for churn prediction, where behavioural thresholds and rule-like patterns may naturally arise.

Boosting-based models, most notably gradient boosting methods such as XGBoost, represent the most flexible category considered in this study. These models iteratively combine weak learners to correct previous errors, enabling them to capture subtle patterns in the data, including non-linearities and higher-order interactions. Boosting methods have been shown to perform particularly well on imbalanced classification problems and structured tabular datasets, aligning closely with the characteristics of the Meewind data. Although these models are often perceived as less interpretable than simpler alternatives, recent advances in model-agnostic explanation techniques, such as SHAP values, allow their predictions to be decomposed into feature-level contributions. As a result, boosting models can still satisfy the requirement of transparency while offering superior predictive performance.

By selecting models from these three categories, this study balances predictive power with interpretability and methodological rigor. Regression models provide transparency and a strong baseline, tree-based models introduce flexibility while remaining interpretable, and boosting methods offer state-of-the-art performance for complex, imbalanced data. Evaluating multiple models within each category further reduces the risk that results are driven by idiosyncratic algorithmic choices. Together, this structured model selection strategy ensures that the final conclusions are robust, comparable, and relevant for both academic analysis and practical deployment.

5.2 Model Implementation and Preprocessing

The first step is to separate the features (X) from the target variable (y), where the target indicates whether or not a customer churned. The features are then categorized into three groups: numerical variables, categorical variables, and binary variables. Some additional preprocessing steps were required to ensure compatibility with specific models. In particular, logistic regression and support vector machines are sensitive to differences in scale and cannot directly handle categorical variables. Consequently, all categorical features were transformed by one-hot encoding, while numeric variables such as age were normalized to avoid distortions caused by varying magnitudes. These transformations are crucial, as both SVMs and logistic regression rely on distance-based calculations that are strongly influenced by feature scaling. Without normalization, features measured on larger scales could dominate the model, leading to biased results. By applying these preprocessing steps consistently, the models were better able to capture patterns in the data and produce meaningful results.

For tree-based methods, such as Decision Trees, Random Forests, Gradient Boosting, Extreme Randomized Trees, and XGBoost, the requirements are less restrictive. These algorithms can inherently handle numerical features without normalization, as they split the data based on threshold values rather than relying on distance calculations. Therefore, standardization or normalization of the numeric input is not strictly necessary. However, categorical features still need to be converted to a numerical representation. Here, one-hot encoding was also applied to ensure consistency across all models, even though some implementations of tree-based methods can directly process categorical variables.

This distinction in preprocessing highlights the importance of tailoring data preparation to the underlying model architecture. While linear and kernel-based methods require careful scaling to function correctly, tree-based ensembles are more robust to raw feature distributions, instead focusing on partitioning the feature space. By applying one-hot encoding universally but normalizing only where needed, the dataset was made suitable for a diverse range of algorithms while preserving comparability across experimental setups.

These preprocessing steps are implemented within a Column Transformer, which organizes the transformations for each feature type. The Column Transformer is then combined with the classifier into a single pipeline. This approach not only simplifies the code, but also reduces the risk of data leakage, since all transformations are applied consistently to both training and test sets.

The complete modelling pipeline can be summarized in pseudocode form, Logistic Regression is taken as an example and shown in Algorithm 1.

Algorithm 1 Logistic Regression Modelling Pipeline

Require: Training data $(X_{\text{train}}, y_{\text{train}})$, test data $(X_{\text{test}}, y_{\text{test}})$
Require: Feature sets: numeric, categorical, binary
Require: Model variant $v \in \{\text{baseline}, \text{cost-sensitive}, \text{SMOTE}, \text{Bayesian}, \text{grid}, \text{random}\}$
Define numeric transformer: `StandardScaler`
Define categorical transformer: `OneHotEncoder(handle_unknown = ignore)`
Define binary transformer: passthrough
Construct preprocessing block using ColumnTransformer
Define base classifier: Logistic Regression with class weighting
if $v =$ baseline **then**
    Build pipeline: preprocessing $\rightarrow$ Logistic Regression
else if $v =$ cost-sensitive **then**
    Build pipeline: preprocessing $\rightarrow$ Logistic Regression (balanced class weights)
else if $v =$ SMOTE **then**
    Build imbalanced pipeline: preprocessing $\rightarrow$ SMOTE $\rightarrow$ Logistic Regression
else if $v =$ Bayesian **then**
    Define hyperparameter search space for C, penalty, and solver
    Wrap pipeline in Bayesian optimization with cross-validation
else if $v =$ grid **then**
    Define discrete hyperparameter grid
    Wrap pipeline in grid search with cross-validation
else if $v =$ random **then**
    Define hyperparameter distributions
    Wrap pipeline in randomized search with cross-validation
else
    Raise error: invalid model variant
end if
Fit pipeline on $(X_{\text{train}}, y_{\text{train}})$
if pipeline contains hyperparameter search **then**
    Select best estimator from cross-validation
end if
Evaluate final model on $(X_{\text{test}}, y_{\text{test}})$ using predefined metrics

Although classifiers differ between models, the structure of the pipeline remains the same. Only the classifier object is exchanged, while the preprocessing and evaluation framework is

kept identical. This consistency is crucial for a fair comparison of model performance as it ensures that differences in predictive outcomes can be attributed to the models themselves rather than to inconsistencies in data preparation.

Once the pipeline is constructed, the dataset is divided into training and test sets. To ensure that both sets reflect the original distribution of churners and non-churners, stratified sampling is used. This is particularly important in the presence of severe class imbalance, as non-stratified sampling could result in training or test sets that contain too few churn cases to allow for reliable estimation and evaluation. The models are trained on the training set and subsequently evaluated on the unseen test set. Evaluation is a critical step, as it determines not only overall predictive performance, but more importantly the extent to which models successfully identify the minority class of churners.

5.3 Evaluation Strategy

To assess the predictive performance of the churn models, several evaluation metrics are applied, each reflecting a different aspect of classification quality. These measures originate from the confusion matrix, which records how often the model correctly identifies churners (true positives) and non-churners (true negatives), as well as the instances where churners are incorrectly classified as staying (false negatives) or where non-churners are incorrectly flagged as churners (false positives). Building on these four fundamental outcomes, more informative performance indicators can be computed, allowing a nuanced interpretation of model behavior, particularly under class imbalance.

All evaluation metrics are computed consistently across models and experimental stages. This uniformity ensures that observed performance differences are driven by modelling choices rather than by changes in evaluation criteria.

- **Accuracy** measures the overall proportion of correctly classified observations. While widely used, it can be misleading in churn prediction: a model that simply predicts every customer as non-churner would achieve high accuracy due to class imbalance, but would completely fail to identify the minority class of interest.

- **Precision** measures the proportion of predicted churners that are truly churners, reflecting the cost of false alarms.

- **Recall** (or sensitivity) measures the proportion of actual churners that were correctly identified, highlighting the ability to capture the minority class.

- **F1-score** is the harmonic mean of precision and recall, providing a balanced measure when false positives and false negatives are of concern.
- **ROC-AUC** (Area Under the Receiver Operating Characteristic Curve) evaluates the trade-off between true positive rate and false positive rate across different thresholds, providing a single-number summary of discriminative ability.

To give an overview of how these metrics are defined, Table 5.1 presents the corresponding formulas based on the entries of the confusion matrix.

Table 5.1: Overview of evaluation metrics derived from the confusion matrix

Metric	**Definition**	**Formula**
Accuracy	Overall correctness	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	Correctness among predicted positives	$\frac{TP}{TP+FP}$
Recall (Sensitivity)	Coverage of actual positives	$\frac{TP}{TP+FN}$
F1-score	Balance between precision and recall	$2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}$
ROC-AUC	Ranking ability across thresholds	Area under ROC curve

To illustrate why accuracy alone is not sufficient in an imbalanced dataset, consider the following hypothetical example. Suppose that we have a dataset of 1,000 customers, of which only 50 are churners (5%). A very naive model predicts that no customers will churn. The resulting confusion matrix would look like Table 5.2.

Table 5.2: Example confusion matrix for a naive churn model

	Predicted Non-churn	Predicted Churn
Actual Non-churn (950)	950 (TN)	0 (FP)
Actual Churn (50)	50 (FN)	0 (TP)

In this case, the model achieves an accuracy of $\frac{950}{1000} = 95\%$, which at first sight appears to be excellent. However, precision, recall, and F1-score for the churn class are all equal to zero, as the model fails to identify a single churner. This simple example illustrates why accuracy can be highly misleading in churn prediction and underlines the importance of considering complementary metrics such as recall, precision, F1-score, and ROC-AUC.

In the specific context of churn prediction at Meewind, the primary objective is to correctly identify as many churners as possible. From a business perspective, false positives (non-churners incorrectly classified as churners) are less costly than false negatives (churners incorrectly classified as non-churners). A higher number of false positives may result

in contacting some customers unnecessarily, but this is preferable to the risk of overlooking actual churners who could otherwise have been retained through timely intervention. Consequently, evaluation metrics that emphasize recall and the balance between recall and precision are of particular importance in this setting.

6

Experimental Setup

The goal of this section is to discuss the implementation of multiple different test, with as goal to determine the most suitable model to predict customer churn at Meewind. Based on previous research on churn prediction that is presented in Section 3.1, five widely used classification methods were selected: logistic regression, decision trees, random forests, gradient boosting, and support vector machines (SVM). These methods represent a diverse set of approaches, ranging from interpretable linear models to more complex ensemble-based techniques. In addition to the established methods, two closely related techniques, Extremely Randomized Trees (ExtraTrees) and XGBoost, were included in the assessment but were implemented only at a later stage of the process as they have similar characteristics to the established models.

A major challenge in this task is the pronounced class imbalance in the dataset, where only 1.9% of the observations correspond to churners. This imbalance can bias models toward the majority class, reducing their ability to correctly identify customers at risk of leaving. To address this issue, two strategies were applied: Synthetic Minority Over-sampling Technique (SMOTE) and cost-sensitive learning (Guo et al., 2008). These approaches ensure that model evaluation is not dominated by the majority class, allowing for a fairer assessment of predictive performance.

In addition, the performance of the model is influenced by the selection of hyperparameters. Each model determines which methods are appropriate. The three tuning strategies used were grid search, random search and Bayesian optimization. These methods differ in computational costs and efficiency, which is particularly relevant for complex models such as random forests, gradient boosters and SVMs, where extensive adjustment may become impractical. Therefore, feasibility considerations regarding training time are included in the assessment alongside forecast performance.

By combining multiple models, imbalance correction strategies, hyperparameter tuning, cross-validation, and evaluation strategies, this chapter provides a comprehensive comparison of predictive techniques. The analysis not only identifies the best-performing model in terms of accuracy and minority class detection, but also highlights trade-offs between predictive power, computational efficiency, and interpretability, all of which are essential in selecting a practically useful churn prediction model for Meewind.

6.1 Data Selection

A critical component of developing a reliable churn prediction model is ensuring that performance is evaluated in a way that reflects both methodological rigor and real-world applicability. Because different evaluation strategies can yield different conclusions, especially in time-dependent settings such as investor behaviour, this study adopts a dual approach. The overarching goal is to balance statistical robustness with operational realism, allowing for a fair comparison of models during experimentation while also testing how well the final model performs under conditions resembling practical deployment.

Therefore, two complementary train–test splitting strategies were implemented, each serving a distinct purpose in the analysis:

- **Random Split:** For the main evaluation, the data was randomly divided into training (80%) and test (20%) sets, stratified by churn status to preserve the original class distribution. This procedure, implemented using a fixed random seed, provides a balanced and statistically valid basis for comparing different models, imbalance corrections, and hyperparameter settings.
- **Temporal Split:** To simulate the practical deployment of churn prediction models, a second evaluation was conducted using a temporal split. Specifically, the most recent quartile of the data was set aside as the test set, while the preceding quartiles were used for training. This approach mimics the real-world application of predicting future churn from historical data, ensuring that model performance is not inflated by information leakage over time.

The random split is used throughout the modelling pipeline, including baseline testing, imbalance correction, and hyperparameter tuning, because it offers a controlled and repeatable environment in which different modelling decisions can be compared directly. Cross-validation within the training portion further stabilizes these comparisons and reduces variance in the performance estimates.

However, a model that performs well in random splits does not necessarily generalize to future data, especially when the underlying customer behaviour may shift over time. For that reason, the temporal split is applied only to the final selected model configuration. This final step acts as a form of validation, demonstrating whether the model's learned patterns remain predictive when evaluated under realistic chronological constraints.

By combining these two evaluation perspectives, one optimized for experimental consistency and one designed for real-world plausibility, the study ensures that the selected model is both empirically well-founded and operationally reliable.

6.2 Baseline Performance

Before applying imbalance correction and hyperparameter tuning, each of the five established models was first evaluated in its baseline configuration. Establishing these baseline results is an essential methodological step: it provides a neutral reference point against which the effectiveness of later interventions can be measured. Without such a benchmark, improvements achieved through resampling strategies, cost-sensitive learning, or parameter optimization would be difficult to attribute or interpret.

In this initial phase, the models were trained using default hyperparameters and without any form of class imbalance handling. This setup reflects how the algorithms behave when exposed directly to the raw data structure, offering insights into their inherent sensitivity to skewed class distributions and their ability to capture the underlying patterns without tuning. Logistic regression serves as a linear, highly interpretable benchmark; decision trees reveal how simple nonlinear partitioning performs; random forest and gradient boosting illustrate the baseline capabilities of ensemble methods; and support vector machines (SVM) provide a contrast through margin-based classification.

Evaluating the models in this unadjusted state also highlights which algorithms are naturally more robust to class imbalance or noise, thereby informing which approaches may benefit most from targeted improvements in later stages.

6.3 Impact of Class Imbalance Corrections

Building on the baseline findings, this section evaluates the impact of two complementary strategies to address the strong class imbalance in the churn dataset. Since only a small fraction of investors discontinue their participation, standard models tend to overwhelmingly favor the majority class, resulting in poor churner recall. To counter this, two imbalance-correction pipelines were implemented, one integrated directly within the model training process and one applied externally before fitting.

The first pipeline relies on cost-sensitive learning, where the misclassification of churners is penalized more. This adjustment modifies the objective function of the model from within, effectively shifting the decision boundary toward the minority class without altering the underlying data distribution. Because this correction occurs inside the estimator, it offers a lightweight and computationally efficient way to highlight minority-class samples, making it particularly suitable for models that naturally accommodate class weights, such as logistic regression, tree-based models, and gradient boosting.

The second pipeline applies the Synthetic Minority Oversampling Technique (SMOTE) as a preprocessing step. SMOTE creates synthetic churn instances based on existing minority-class observations, thereby augmenting the training data before the model is fitted. Unlike cost-sensitive learning, which affects the loss function, SMOTE alters the feature space itself and can help models that rely heavily on geometric relationships, especially algorithms such as k-nearest neighbors or margin-based methods that are sensitive to sample density. However, because SMOTE operates outside the model, it requires a more explicit pipeline to avoid data leakage and ensure that only the training folds are oversampled during cross-validation.

Evaluating both approaches provides insight into the relative benefits of modifying the learning process versus modifying the data. Together, they form a structured progression beyond the baseline, enabling a clearer understanding of how different imbalance mitigation strategies influence each model's ability to detect churners.

6.4 Hyperparameter Tuning and Optimized Models

Following the baseline and imbalance-corrected evaluations, the next methodological step involved systematically tuning the model hyperparameters. Hyperparameter optimization is a crucial component of machine learning workflows, as many algorithms are highly sensitive to configuration choices that influence model complexity, regularization strength, and decision boundaries. However, tuning can also be computationally demanding, especially for models with large search spaces or continuous parameters. To ensure that the chosen optimization strategy was both methodologically sound and computationally feasible, the tuning process was deliberately structured into two phases.

6.4.1 Selecting the search strategy

The first phase focused on logistic regression, which provides an ideal controlled environment for comparing tuning strategies. Unlike more complex ensemble or kernel-based models, logistic regression has a compact and clearly defined hyperparameter space consisting of both discrete parameters (e.g., the type of regularization penalty, solver choice) and a continuous parameter (the inverse regularization strength, C). This combination makes logistic regression well suited for evaluating different search methodologies, as it captures both forms of hyperparameter variation without introducing the combinatorial explosion seen in more complex models.

In this phase, three common tuning strategies; grid search, random search, and Bayesian optimization, were applied to logistic regression. Comparing these approaches on a simple, interpretable model helps ensure that the chosen search method is appropriate for the structure of the churn dataset before applying it to models with substantially larger or more irregular search spaces. This step is particularly important because tree-based models, gradient boosting methods, and SVMs involve numerous interacting hyperparameters, many of which are continuous, making exhaustive search approaches impractical.

6.4.2 Relevant Hyperparameters per Model

Based on this methodological comparison, the second phase proceeded by selecting a single tuning strategy for all machine learning models. This will be a strategy that will perform best for all the different methods, while also taking computational limitations into account. The effectiveness of hyperparameter tuning depends not only on the search strategy, but also on selecting parameters that meaningfully influence performance. Different algorithms are sensitive to different aspects of their configuration and therefore focusing

the search on the most impactful hyperparameters is crucial. Additionally, hyperparameters can belong to discrete or continuous search spaces. Discrete hyperparameters, such as categorical choices for kernel type or splitting criteria, are selected from a finite set of options, whereas continuous hyperparameters, such as regularization strength or learning rate, can be explored over a continuous range. Recognizing this distinction is important because it informs the choice of search strategy: grid search is well-suited for small discrete spaces, while random or Bayesian search is more efficient for continuous or high-dimensional spaces.

The specific hyperparameters considered for each model in this study, including their type (categorical, integer, or continuous) and respective search ranges or values, are summarized in Table 6.1. This table provides a concise overview of the tuning search spaces, enabling a structured comparison across models and serving as a reference for the detailed discussion of the most relevant hyperparameters for each model provided below.

- **Logistic Regression:** The most influential hyperparameter is the regularization strength (C), which controls the penalty applied to large coefficients. Smaller values of C enforce stronger regularization, reducing variance but potentially introducing bias. Searching C on a logarithmic scale is a common and empirically supported approach that helps balance model complexity and generalization performance. The choice between $L1$ (lasso) and $L2$ (ridge) regularization affects sparsity and feature selection, as discussed in the literature on penalized regression (Friedman et al., 2010; Ng, 2004). Finally, the solvers *liblinear* and *saga* are widely recommended for efficiently optimizing logistic models with these regularization schemes.

- **Decision Trees:** Key hyperparameters include the maximum depth of the tree, the minimum number of samples required to split a node ($min\_samples\_split$), the minimum number of samples required at a leaf node ($min\_samples\_leaf$), and the number of features considered at each split ($max\_features$). These parameters directly influence the complexity of the model and its tendency to overfit. Shallow trees or high $min\_samples\_leaf$ values may underfit, while overly deep trees with unrestricted splits can capture noise. Tuning these parameters is therefore essential to balance bias and variance (Breiman et al., 1984). Limiting tree depth and controlling the minimum number of samples per node are well-established methods to improve generalization (Rokach and Maimon, 2010). Similarly, varying the number of features considered at each split helps reduce the correlation between nodes and encourages a more robust model performance.

- **Random Forest:** As an ensemble of decision trees, random forests inherit all tree-level hyperparameters (e.g., depth, $min\_samples\_split$, $min\_samples\_leaf$, $max\_features$) but also introduce the number of trees ($n\_estimators$). Increasing $n\_estimators$ improves stability and reduces variance, but increases computational cost, as discussed in the original formulation of the algorithm (Breiman, 2001). Performance is particularly sensitive to the number of trees and the interaction between tree depth and $max\_features$, parameters that control the trade-off between bias and variance. Empirical studies further suggest that expanding the ensemble size improves model generalization up to a point of diminishing returns (Oshiro et al., 2012).

- **Gradient Boosting:** Gradient boosting models are highly sensitive to the learning rate and the number of boosting iterations ($n\_estimators$). A small learning rate combined with a larger number of iterations typically yields better generalization but increases training time. Additional parameters such as tree depth, $min\_samples\_leaf$, and $min\_samples\_split$ influence the model's complexity, while subsampling introduces stochasticity that acts as a regularizer and helps prevent overfitting (Friedman, 2001, 2002). Because boosting methods tend to overfit when learning rates are high or trees are too deep, careful tuning of these interacting hyperparameters is essential for robust performance.

- **Support Vector Machines (SVM):** The main hyperparameters are the regularization constant (C) and the kernel width (γ). C and γ are continuous hyperparameters that regulate the trade-off between margin maximization and model complexity. Exploring these parameters on a logarithmic scale is a standard and empirically validated approach Hsu et al. (2010). High values of C and γ can lead to overfitting, while smaller values promote smoother and more generalizable decision boundaries (Smola and Schölkopf, 2004). In this study, only the RBF kernel was used, although other kernel types are available, because including multiple kernels would substantially increase computational cost and was not deemed necessary or feasible for the comparative analysis.

- **Extremely Randomized Trees (ExtraTrees):** Similar to Random Forests, ExtraTrees build an ensemble of decision trees, but introduce additional randomness by selecting random split thresholds for each candidate feature (Geurts et al., 2006). The main hyperparameters are therefore also similar, including the number of trees ($n\_estimators$), maximum tree depth (max_depth), minimum number of samples required to split a node ($min\_samples\_split$), minimum number of samples required at a leaf node ($min\_samples\_leaf$), the number of features considered at each split ($max\_features$), and whether to bootstrap samples when building trees ($bootstrap$). These parameters control the balance between bias and variance, model complexity, and computational efficiency.

- **XGBoost:** XGBoost is an optimized implementation of gradient boosting that incorporates both $L1$ and $L2$ regularization to improve generalization and reduce overfitting (Chen and Guestrin, 2016). Its predictive performance is highly sensitive to several hyperparameters, including the number of trees ($n\_estimators$), maximum tree depth (max_depth), learning rate, subsample ratio, column sampling ratio, minimum loss reduction required for further partitioning (γ), and the $L2$ regularization term (reg_lambda). These parameters jointly control the learning dynamics, complexity, and regularization strength of the model. Careful tuning of these interacting hyperparameters has been shown to significantly improve performance, particularly in heterogeneous and imbalanced datasets, where XGBoost's stochastic sampling and regularization mechanisms enhance robustness and predictive stability.

Understanding the role of individual hyperparameters and whether they operate in discrete or continuous search spaces allows the tuning strategy to allocate computational resources more effectively. By focusing on the parameters that have the greatest influence on model performance, even complex models such as XGBoost and SVM can be optimized efficiently, balancing predictive accuracy with practical constraints on computation time. At the same time, tailoring the search space to the characteristics of each algorithm while applying a consistent optimization procedure ensures that the tuning process remains both efficient and theoretically well-founded.

Table 6.1: Hyperparameter search spaces for all models

Model	Hyperparameter	Type	Values / Range
Logistic Regression	C	Categorical	{0.01, 0.1, 1, 10, 100}
	Penalty	Categorical	{L1, L2}
	Solver	Categorical	{liblinear, saga}
Decision Tree	Max depth	Integer	[2, 20]
	Min samples split	Integer	[2, 20]
	Min samples leaf	Integer	[1, 10]
	Max features	Categorical	{sqrt, log2, None}
	Criterion	Categorical	{gini, entropy}
Random Forest	Number of trees	Integer	[100, 500]
	Max depth	Integer	[2, 30]
	Min samples split	Integer	[2, 100]
	Min samples leaf	Integer	[1, 10]
	Max features	Categorical	{sqrt, log2, None}
	Bootstrap	Categorical	{True, False}
Gradient Boosting	Number of estimators	Integer	[50, 500]
	Learning rate	Continuous (log-uniform)	[0.01, 0.3]
	Max depth	Integer	[2, 10]
	Min samples split	Integer	[2, 100]
	Min samples leaf	Integer	[1, 50]
	Subsample	Continuous	[0.5, 1.0]
SVM	C	Continuous (log-uniform)	$[10^{-3}, 10^{3}]$
	γ	Continuous (log-uniform)	$[10^{-4}, 10^{1}]$
	Kernel	Categorical	{rbf}
Extra Trees	Number of trees	Integer	[100, 500]
	Max depth	Integer	[2, 30]
	Min samples split	Integer	[2, 20]
	Min samples leaf	Integer	[1, 10]
	Max features	Categorical	{sqrt, log2, None}
	Bootstrap	Categorical	{True, False}
XGBoost	Number of estimators	Integer	[100, 500]
	Max depth	Integer	[3, 12]
	Learning rate	Continuous (log-uniform)	[0.01, 0.3]
	Subsample	Continuous	[0.5, 1.0]
	Column sample by tree	Continuous	[0.5, 1.0]
	Gamma	Continuous	[0, 5]
	λ (L2 regularization)	Continuous (log-uniform)	[0.01, 10]

6.4.3 Evaluation

Within the training portion of the data, k-fold cross-validation was used to guide the hyperparameter search and to obtain a more reliable estimate of model performance. The training set was divided into $k = 5$ equally sized folds, with each fold serving once as the validation subset while the remaining folds were used for fitting. For each candidate hyperparameter configuration, performance was averaged across the five folds and the best-performing configuration was selected. The model was then retrained on the full training data using these optimal hyperparameters before being evaluated on the held-out test set. This procedure reduces the risk of overfitting to a particular split of the data and provides a more stable estimate of the generalization error (Hastie et al., 2009).

Overall, structuring the hyperparameter tuning into a controlled preliminary phase followed by a full-model optimization phase, supported by 5-fold cross-validation, ensures that the chosen tuning strategy is well justified and that each model is evaluated under conditions designed to maximize its potential, without introducing unnecessary computational overhead or methodological inconsistencies. The best preforming method can then be used for further evaluation, such as feature importance or reliability control.

6.5 Statistical testing

Although cross-validation provides a robust estimate of predictive performance, the differences observed between models may still arise from randomness in data partitioning rather than from genuine methodological superiority. To ensure that performance comparisons are statistically well-founded, an additional layer of statistical testing was introduced. This step aims to assess whether observed differences in evaluation metrics are consistent across repeated samples and unlikely to be caused by chance.

Instead of relying on a single train–test split with a fixed random seed, a repeated stratified cross-validation framework was employed. Specifically, the dataset was evaluated using 5-fold stratified cross-validation, repeated 10 times with varying random seeds, resulting in a total of 50 paired performance measurements per model. Stratification ensures that each fold preserves the original class distribution, which is particularly important given the strong class imbalance present in the churn dataset. Repeating the procedure with different random initializations further reduces the sensitivity to a specific partitioning of the data and produces a more reliable empirical distribution of performance scores.

For each repetition and fold, models were trained and evaluated on identical data splits, producing paired observations of the chosen performance metric. This pairing is essential,

as it allows performance differences to be attributed solely to the modelling approach rather than to variation in the underlying data. The resulting distributions therefore consist of matched samples, where each score reflects performance under the same experimental conditions.

To formally compare these paired performance distributions, the Wilcoxon signed-rank test was applied. This non-parametric test evaluates whether the median of the paired differences between two methods differs significantly from zero. The Wilcoxon test is particularly well suited to this setting for several reasons. First, performance metrics obtained from cross-validation, such as average precision or recall, are not guaranteed to follow a normal distribution, especially under severe class imbalance. Second, the sample size, although increased through repetition, remains relatively modest, making parametric assumptions less reliable. Finally, the paired nature of the evaluation violates the independence assumption required by unpaired tests.

By operating on the ranks of the paired differences rather than on their raw values, the Wilcoxon signed-rank test provides a robust and distribution-free assessment of whether one method consistently outperforms another across repeated evaluations. In addition to the test statistic and associated p-value, summary statistics of the paired differences are retained to support interpretation and to assess the magnitude and direction of any observed effects.

6.6 Business-Oriented Perspective

Beyond the standard cross-validated evaluation, a secondary assessment was conducted to reflect the conditions under which a churn prediction model would be deployed in practice. As discussed in Section 6.1, a temporal split provides a more realistic testing scenario by preserving the chronological order of customer observations and ensuring that predictions are generated only for data that occurs after the training period. This avoids information leakage and better captures the forward-looking nature of real-world use cases.

To incorporate this business-oriented perspective, the best, performing model from the earlier tuning stage, based on cross-validated performance, was re-evaluated using this temporally ordered train–test setup. This structure emulates operational deployment, where the model is trained on historical behavior and subsequently applied to new or incoming customer activity.

An additional goal of this step was to assess the reliability and robustness of the selected model by comparing its temporal-split performance to the results obtained through

cross-validation. If the model performs consistently across both evaluation strategies, this suggests that the learned patterns generalize well and are not a product of sampling variation. In contrast, large discrepancies between cross-validated and temporal performance may indicate sensitivity to time-dependent shifts or potential overfitting to earlier periods. To support this reliability check, the internal cross-validation metrics obtained during hyperparameter tuning were also examined, ensuring that the chosen hyperparameters were stable across folds and that performance did not depend heavily on any single subset of the data.

From a business point of view, this evaluation framework is crucial. It tests the model's ability to generalize under realistic temporal dynamics, including evolving customer behavior, seasonality, and gradual market shifts. By validating performance under these time-aware constraints, the analysis offers a more meaningful indication of how reliably the model would function in a live churn-monitoring environment.

7

Results

This chapter presents the empirical findings derived from the modelling and evaluation procedures outlined in the previous section. Each set of results corresponds directly to one stage of the analysis pipeline: baseline model assessment, the impact of class imbalance corrections, the effects of different hyperparameter tuning strategies, and the final evaluation using both random and temporal splits. By following the structure of the methodological framework, the results illustrate how each modelling decision influences predictive performance and contributes to the construction of a reliable churn prediction model.

The chapter begins by examining the baseline behaviour of all candidate algorithms, providing a reference point for subsequent improvements. Then it evaluates how imbalance correction techniques affect the ability of the model to detect churners, followed by an assessment of how optimized hyperparameters influence predictive accuracy and stability across cross-validation folds. Finally, the best-performing model configuration is tested under the temporal evaluation scheme to assess its practical reliability in forecasting future churn.

Together, these results allow for a systematic comparison of modelling strategies and form the basis for the discussion and business-oriented interpretation presented in the next chapter.

7.1 Baseline results

As a first step, all five candidate models were first evaluated in their baseline configuration, as described in Section 6.2. This initial assessment provides a reference point for understanding how subsequent modelling decisions influence performance, and it highlights the

challenges posed by the severe class imbalance in the churn dataset. The results can be found below in Table 7.1.

Table 7.1: Baseline performance metrics across models.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Logistic Regression	0.98	0.12	0.00	0.00	0.71
Decision Tree	0.97	0.12	0.12	0.12	0.57
Random Forest	0.98	0.12	0.05	0.07	0.82
Gradient Boosting	0.98	0.41	0.03	0.05	0.86
SVM	0.98	0.00	0.00	0.00	0.62

Across all baseline models, accuracy remains extremely high (around 98%). However, as discussed in Section 5, this metric is dominated by the overwhelming proportion of non-churners in the dataset and therefore provides little insight into the practical usefulness of the models. The more relevant indicators for Meewind's churn prediction objective, recall and F1-score for the minority class, are uniformly poor.

Logistic regression achieves a recall close to zero, detecting only 2 of the 981 churners in the test set. The decision tree shows modest improvement, identifying more churners and getting a recall of 0.12, though its precision remains low at 0.12 due to many false positives. Random forests perform slightly better on some metrics, with a recall of 0.05 and a comparatively strong ROC-AUC of 0.82, indicating that the model captures some underlying separation between churners and non-churners even if it fails to translate this into effective classification. Gradient boosting exhibits the highest precision (0.41) but still identifies very few churners (recall 0.03), resulting in a low F1-score.

The support vector machine performs the weakest on minority detection, failing to identify a single churner and producing zero precision, recall, and F1-score. Its ROC-AUC of 0.62 suggests limited discriminative capability under the baseline configuration.

Overall, these findings show that although several models achieve ROC-AUC scores notably above random guessing, none meaningfully detect churners in their baseline form. Therefore, high accuracy masks a fundamental inability to identify the minority class, the opposite of what is required for Meewind's churn prediction use case. The baseline results clearly demonstrate that imbalance correction and more advanced modelling strategies are essential before reliable conclusions can be drawn or a model can be considered for deployment.

7.2 Class imbalance results

Having established the substantial performance limitations of the baseline models, the next step is to evaluate how the correction strategies described in Section 6.3 influence the models' ability to detect churners. Since the dataset is heavily skewed toward non-churners, addressing this imbalance is essential for achieving meaningful predictive performance. The following subsections present the results for each correction method, organized by model class. The evaluation metrics can be found in Table 7.2 below the comparison of methods.

7.2.1 Cost-Sensitive Learning

Introducing class weights generally improved recall for churners, though often at the cost of precision. Logistic regression improved from nearly zero recall in the baseline to 63% under cost-sensitive training, while maintaining a precision of 0.04 and a ROC-AUC of 0.72. This represents a substantial improvement in the identification of churners, despite a higher false positive rate.

The decision tree showed limited gains, with recall improving modestly to 0.14 and precision at 0.11, indicating that cost-sensitive weighting had only a small impact. Random forests achieved similar results, with recall of 0.05 and precision of 0.11, showing slightly better discrimination (ROC-AUC 0.83) but still limited minority-class detection.

However, SVM's benefited more substantially. The cost-sensitive SVM achieved 66% recall with 0.06 precision and a ROC-AUC of 0.82, demonstrating a much better balance between recall and discrimination compared to tree-based models. Gradient boosting, however, does not natively support class weights in the same straightforward manner as logistic regression or tree-based models like decision trees and random forests. The algorithm iteratively fits trees to minimize a specified loss function, directly incorporating misclassification costs into this iterative gradient-based optimization. Although some implementations allow for sample weighting, these are often less stable or require custom loss functions, which were beyond the scope of this evaluation. Consequently, cost-sensitive learning could not be applied to gradient boosting in this study.

7.2.2 Resampling Approaches

The application of SMOTE generally led to more stable and conservative results compared to cost-sensitive learning. For logistic regression, the recall was 65% with precision 0.03, nearly identical to the cost-sensitive results, but with slightly smoother decision boundaries due to balanced training data.

Tree-based models demonstrated limited benefit from SMOTE. The decision tree achieved a recall of 0.11 and a precision of 0.10, while the random forest reached a recall of 0.04 and a precision of 0.10, both similar to their cost-sensitive counterparts. This suggests that synthetic oversampling alone does not sufficiently overcome the bias toward the majority class in these models.

Gradient boosting, where only SMOTE could be applied, achieved 0.33 precision and 0.03 recall (ROC-AUC 0.82), showing good discriminatory potential but a tendency to miss most churners. The SMOTE-adjusted SVM achieved 61% recall and 0.07 precision (ROC-AUC 0.80), performing comparable to the cost-sensitive version but with slightly reduced recall.

Overall, both methods improved churn detection relative to the baseline, but with differing trade-offs. Cost-sensitive learning emphasizes recall, ensuring that more churners are identified, while SMOTE offers a more temperate improvement, yielding better precision and stability. For Meewind's churn prediction use case, where identifying potential churners is the priority, cost-sensitive approaches, particularly logistic regression and SVM, appear to have improved the most with the implementation of class rebalancing. However, these come with higher false positive rates that must be managed operationally in downstream retention strategies.

Table 7.2: Comparison of imbalance correction methods across models. Metrics are reported for the minority class (churners).

Model	Correction	Accuracy	Precision	Recall	F1-score	ROC-AUC
Logistic Regression	Cost-sensitive	0.67	0.04	0.63	0.07	0.72
	SMOTE	0.65	0.03	0.65	0.06	0.72
Decision Tree	Cost-sensitive	0.96	0.11	0.14	0.13	0.56
	SMOTE	0.96	0.10	0.11	0.10	0.55
Random Forest	Cost-sensitive	0.98	0.11	0.05	0.06	0.83
	SMOTE	0.98	0.10	0.04	0.06	0.82
Gradient Boosting	Cost-sensitive	–	–	–	–	–
	SMOTE	0.98	0.33	0.03	0.06	0.82
SVM	Cost-sensitive	0.81	0.06	0.66	0.12	0.82
	SMOTE	0.83	0.07	0.61	0.12	0.80

7.3 Hyperparameter tuning results

The following section presents the results of the hyperparameter optimisation procedures described previously in Section 6.4. Whereas the earlier section outlined the methodological choices and tuning strategy, the focus here shifts to the empirical results obtained from applying these procedures to each model. The results are structured in two parts. First, the tuning strategies themselves are compared to determine which approach is most suitable for the subsequent modelling stages. Second, the performance of each tuned model is reported to illustrate the effect of hyperparameter optimisation on predictive quality. Together, these results form the basis for selecting the most reliable modelling approach for churn prediction.

7.3.1 Determining the search strategy

The first step in the tuning stage was to determine which optimisation strategy should be used for the remainder of the modelling pipeline. Before tuning any of the models in depth, the three candidate approaches were applied to a single reference model to assess their practical differences. The results of this comparison showed that all three tuning methods achieved nearly identical predictive performance, with minimal variation in precision, recall, or ROC-AUC (Table 7.3). However, the computational times differed substantially. Random search completed the tuning process the fastest, followed by Bayesian optimization, while grid search was by far the slowest. This outcome can be explained by the exhaustive nature of grid search and the relatively small search space of the logistic regression model, where Bayesian optimization's adaptive exploration offers limited time savings.

Table 7.3: Comparison of hyperparameter tuning strategies using logistic regression.

Tuning Method	Accuracy	Precision	Recall	F1-score	ROC-AUC	Runtime
Grid Search	0.67	0.04	0.63	0.07	0.7223	103m 8.7s
Random Search	0.68	0.04	0.63	0.07	0.7205	28m 31.6s
Bayesian Opt.	0.67	0.04	0.63	0.07	0.7225	48m 0.8s

Despite its moderate runtime, Bayesian optimization was selected for all subsequent models. Its adaptive nature makes it more scalable and computationally efficient than grid search. Especially when dealing with larger and more complex parameter spaces, as found in algorithms such as random forests, gradient boosting, and support vector machines.

Model optimization was performed with respect to the average precision (AP) score, which summarizes the area under the precision–recall curve. This metric is particularly suitable for imbalanced classification problems, such as churn prediction, as it emphasizes correctly identifying minority class instances, in this case, churners. By maximizing AP, the tuning process aligns directly with Meewind's business objective: prioritizing the detection of churners while maintaining an acceptable level of precision.

7.3.2 Hyperparameter tuning on the models

After selecting the preferred tuning strategy, the next step is to assess how hyperparameter optimisation affected the performance of each individual model, highlighting improvements relative to earlier modelling stages and clarifying differences in model behaviour under the same optimisation criterion. Table 7.4 reports the performance of the tuned models when Bayesian hyperparameter tuning was applied.

The results show a consistent improvement over the baseline and imbalance-corrected stages, particularly in the recall for the minority class. Logistic regression maintained a strong recall of 63% (precision 0.04, ROC-AUC 0.72), confirming its ability to identify a large share of churners despite high false positives. The decision tree and the random forest improved markedly, with recall values of 69% and 43%, respectively. Random forest in particular achieved much higher precision (0.19) and F1-score (0.27), suggesting a better trade-off between churn detection and reliability.

Gradient boosting demonstrated the highest precision (0.46) among all models, correctly identifying a smaller but more accurate set of churners (recall 0.09, ROC-AUC 0.88). Although its recall remained moderate, the model produced fewer false positives, making it a strong candidate for scenarios requiring higher precision. However, the tuning process was computationally demanding, taking approximately 600 minutes to converge.

The additional ensemble models, ExtraTrees and XGBoost, provided the best overall balance between recall and precision. ExtraTrees achieved 0.41 recall and 0.19 precision (ROC-AUC 0.87), while XGBoost achieved the highest recall (0.76) with precision 0.09 and a ROC-AUC of 0.88. Thus, XGBoost emerged as the strongest model in terms of churn detection, capturing most churners while retaining reasonable discriminatory power.

In contrast, Support Vector Machines (SVM) presented significant computational challenges. Given the size of the dataset and the algorithmic complexity of SVMs, training times quickly became long. To ensure feasibility, a time limit was imposed on the hyperparameter tuning process, allowing the search to terminate once the predefined runtime was reached. In addition, the number of cross-validation folds was reduced to accelerate the

evaluation. If the optimization process ended before full convergence, the best-performing parameter configuration found within the allotted time was retained, even if it potentially represented a local optimum rather than the global optimum. This trade-off enabled the inclusion of SVM results within a practical time frame, though performance should be interpreted with caution due to these computational constraints.

The hyperparameter tuning process ran approximately 14,594 minutes (just over ten days) to compute these results, completing 27/100 iterations. The resulting model achieved a recall of 66% and a precision of 0.06 for churners, with a ROC-AUC of 0.80. While this represents a substantial improvement over the baseline SVM and demonstrates that the model can effectively identify churners when adequately tuned, its performance remains comparable to logistic regression and inferior to the best-performing ensemble methods. Given the extreme computational cost relative to the marginal performance gains, SVM is considered impractical for operational deployment in this context.

Table 7.4: Performance of tuned models applying Bayesian hyperparameter tuning (optimized on Average Precision). Minority class (churners) metrics are reported.

Model	Precision	Recall	F1	ROC-AUC
Logistic Regression	0.04	0.63	0.07	0.72
Decision Tree	0.06	0.69	0.11	0.78
Random Forest	0.19	0.43	0.27	0.89
Gradient Boosting	0.46	0.09	0.15	0.88
SVM	0.06	0.66	0.11	0.80
ExtraTrees	0.19	0.41	0.26	0.87
XGBoost	0.09	0.76	0.16	0.88

In summary, hyperparameter tuning significantly improved model performance, particularly for ensemble methods. Although precision remains low across most models, this trade-off is acceptable for churn prediction, where missing a churner is costlier than incorrectly flagging a loyal customer. Overall, XGBoost and ExtraTrees offered the best balance between recall, precision, and ROC-AUC, while Gradient Boosting stood out for precision-oriented applications. However, computational feasibility remains an important consideration, especially for resource-intensive models such as gradient boosting and SVM. Thus, the choice for use in practise will be made for XGBoost.

7.4 Feature importance

To better understand how the churn model makes predictions, feature importance was examined using both XGBoost's built-in gain metric (Figure 7.1) and SHAP values (Figure 7.2). Although both visualisations aim to explain which features matter most, they rely on different underlying principles and therefore offer complementary perspectives. XGBoost's built-in feature importance is based on how often and how effectively a variable is used to split decision trees during training. Features that repeatedly reduce classification error receive higher scores. This makes the method fast and intuitive, but it can also introduce bias. For example, variables with many possible split points (often numerical features) may appear more important simply because they offer more opportunities for the model to use them, while one-hot encoded categorical variables or correlated features may be undervalued. In addition, the gain metric does not communicate whether a feature increases or decreases the risk of churn, nor how large that effect is.

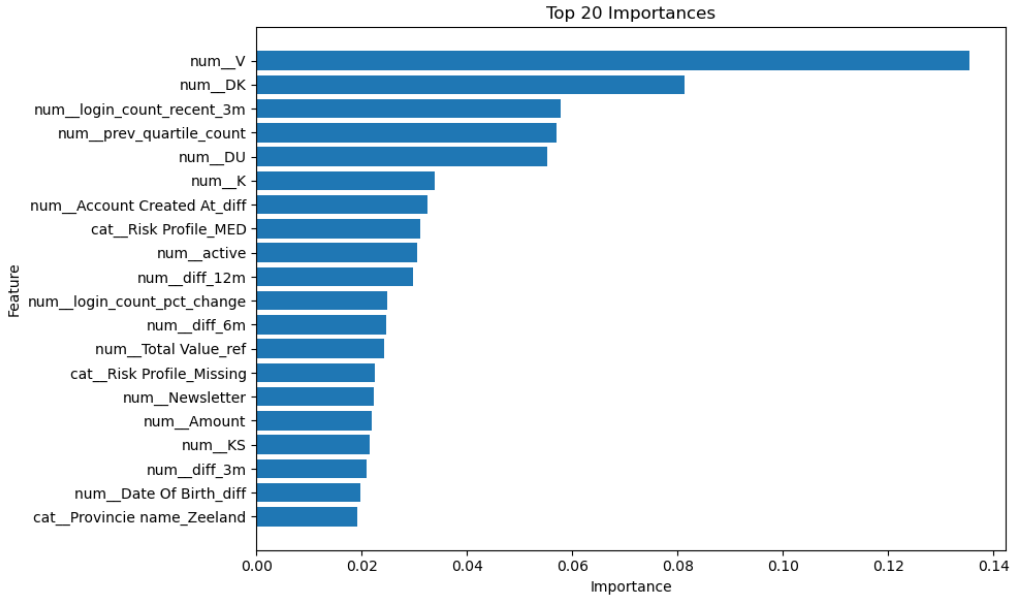

Figure 7.1: Top 20 feature importances based on the tuned XGBoost model.

SHAP, on the other hand, approaches feature importance from a prediction perspective. Instead of focusing on how the model is built, it looks at how much each feature contributes to individual predictions by estimating its marginal effect when combined with all other variables. As a result, SHAP provides both magnitude and direction: positive SHAP values indicate a push towards predicting churn, while negative values lower the predicted risk. SHAP also treats all features on a more equal footing, reducing the structural biases

that can occur in tree-based importance scores. This makes SHAP particularly useful for understanding how the model behaves for different types of investors rather than only at the model-level.

Figure 7.2: SHAP summary plot for the tuned XGBoost model.

Even with these methodological differences, both approaches show a consistent narrative in this study. Transaction-related features remain among the most influential predictors of churn. For instance, *num_ V*, representing the number of sell orders in the last two years, shows that higher values are associated with an increased likelihood of churn, while lower sell activity corresponds to more stable behaviour. Similarly, *num_ K*, the number of buy orders over the same period, exhibits the opposite trend: investors with fewer purchases are less likely to churn, suggesting that frequent trading activity may reflect higher volatility in investor engagement.

Dividend-related behaviour also plays a role. The binary feature *num_ DK*, indicating whether a dividend reinvestment occurred in the past six months, has a clear protective effect: investors who reinvest dividends are less likely to churn. In contrast, the feature *num_ prev_ quartile_ count*, capturing the number of churners in the previous period, shows

no clear directional effect, with SHAP values fluctuating widely, reflecting substantial noise and limited predictive power in isolation.

Investor demographics and tenure are similarly informative. *num_ Date Of Birth_ diff*, a proxy for age, indicates that younger investors have a higher probability of churn, while longer-tenured accounts (*num_ Account Created At_ diff*) are associated with lower churn risk. This highlights the stabilizing influence of experience and familiarity with the platform.

Account value dynamics are captured through the *num_ diff_ ** series, which measures changes in account value over multiple intervals. Across these features, larger increases in account value tend to reduce the likelihood of churn, consistent with the idea that financially engaged or growing accounts reflect more satisfied or committed investors.

Engagement metrics also contribute meaningfully. *num_ login_ count_ recent_ 3m*, the number of logins in the past three months, and *num_ login_ count_ pct_ change*, capturing changes in login frequency between periods, both show that lower recent activity corresponds to lower churn probability. While this may seem counterintuitive at first, it suggests that highly active investors may be more attuned to market conditions and more willing to exit positions if dissatisfied, whereas more passive investors exhibit inertia.

Finally, completeness of account information has predictive value. The categorical feature *cat_ Frequency_ Missing*, indicating missing values for the incasso amount, shows that when the variable is present, churn probability decreases, suggesting that investors who provide complete data are more engaged. Most other binary features similarly indicate that positive or active states tend to lower churn probability, while negative or absent signals often do not meaningfully change churn risk.

Taken together, the SHAP analysis complements traditional feature importance by providing directional insights, showing not only which features matter but also how specific values influence churn risk. Transaction patterns, engagement levels, account longevity, and demographic factors interact in nuanced ways, painting a rich and interpretable picture of the drivers behind investor churn at Meewind. The agreement between XGBoost's importance ranking and SHAP values strengthens confidence in the model, while SHAP improves transparency by clarifying the behavioral mechanisms that underlie churn predictions.

7.5 Statistical testing

Following the evaluation framework described in Section 6.5, a formal statistical comparison was conducted between the two best-performing ensemble models identified earlier in the analysis: XGBoost and Extremely Randomized Trees (ExtraTrees). These models were selected for direct comparison because they consistently demonstrated the strongest predictive performance during hyperparameter tuning.

The repeated stratified cross-validation procedure yielded a total of 50 paired performance evaluations for each model. Across these evaluations, XGBoost achieved a mean Average Precision (AP) of 0.2049, compared to 0.1903 for ExtraTrees. This corresponds to a mean performance difference of 0.0146 in favor of XGBoost. The median difference between the paired AP scores was 0.0135, closely aligning with the mean difference and indicating that the observed performance advantage is not driven by a small number of extreme cases.

Although the individual evaluation scores varied across folds and repetition, as expected given the randomized data splits, the consistently positive mean and median differences suggest a stable performance advantage for XGBoost across the majority of evaluations. This variability highlights that the two models do not produce identical results on every split, but that XGBoost tends to outperform ExtraTrees in a systematic manner rather than sporadically.

To formally assess whether the observed performance difference is statistically meaningful, the Wilcoxon signed-rank test was applied to the paired Average Precision (AP) scores. The null hypothesis of this test states that there is no systematic difference in performance between XGBoost and ExtraTrees, meaning that the median of the paired AP differences is equal to zero. Under this hypothesis, any observed differences would be attributable solely to random variation arising from different data splits.

The resulting p-value of 1.78×10^{-14} provides overwhelming evidence against this null hypothesis. This extremely small probability suggests that the observed differences are highly unlikely under the assumption of equal model performance. Consequently, the null hypothesis can be rejected with high confidence, implying that the observed advantage of XGBoost over ExtraTrees is not due to random chance, but due to consequent better performance by XGBoost.

Importantly, statistical significance does not imply that XGBoost outperforms ExtraTrees in every individual evaluation. Rather, the Wilcoxon test assesses whether one model tends to yield higher performance than the other across repeated paired observations. In

this case, the results show that XGBoost consistently achieves higher average precision scores across the 50 evaluations, despite some overlap in individual fold outcomes.

Taken together, the higher mean AP (0.2049 versus 0.1903), the positive median difference (0.0135), and the highly significant Wilcoxon test result jointly support the conclusion that XGBoost offers a statistically and practically superior performance compared to ExtraTrees within the experimental setup of this study. This evidence strengthens the robustness of the model selection process and provides a sound empirical basis for prioritizing XGBoost in subsequent analysis and business-oriented evaluation.

7.6 Business-oriented results

The final step in the evaluation pipeline is to assess how well the best-performing model operates under business-realistic conditions. Based on the earlier experiments, XGBoost emerged as the strongest candidate, combining high recall with competitive ROC-AUC performance. In this section, its predictive ability is examined using the temporal split introduced in Section 6.6, which allows the model to be tested on genuinely future observations rather than randomly mixed data. The results are presented in direct comparison with the standard random split in Table 7.5 and are complemented by the internal cross-validation outcomes to evaluate the reliability and stability of the model's predictions.

Table 7.5: Performance of XGBoost under random and temporal split evaluations. Minority class (churners) metrics are reported.

Evaluation	Precision	Recall	F1-Score	ROC-AUC
Random Split	0.09	0.76	0.16	0.88
Temporal Split	0.09	0.77	0.16	0.85

The temporal evaluation reveals that XGBoost maintains stable performance when predicting future data, with the recall remaining high at 77% and only a minor drop in ROC-AUC to 0.85. This demonstrates the robustness and generalizability of the model under more realistic, time-dependent conditions. The precision remains low (0.09), consistent with the random split, reflecting the inherent difficulty of predicting churn given the substantial class imbalance.

From a business perspective, these results are highly relevant. A recall of 77% implies that the model successfully identifies the majority of potential churners, enabling

Meewind to take proactive retention actions. Although low precision indicates some over-identification of at-risk customers, this trade-off is acceptable in practice: the cost of contacting a few additional loyal clients is outweighed by the benefit of retaining actual churners.

To further assess the reliability of these findings, the internal cross-validation (CV) results from the Bayesian optimization procedure were examined. The mean CV recall score was 0.153 (standard deviation 0.022) for the random split and 0.072 (standard deviation 0.008) for the temporal split. These relatively small standard deviations indicate consistent model behavior across folds, confirming that the observed results are not driven by random fluctuations in data partitions. Together with the external evaluation, this reinforces confidence in the stability and suitability of the mode for deployment in Meewind's production environment.

Overall, the temporal split analysis, supported by the internal CV results, confirms that the tuned XGBoost model generalizes well and can be effectively used for early churn detection, providing actionable insights aligned with Meewind's strategic objectives.

8

Conclusion and Discussion

8.1 Summary of Research

The objective of this thesis was to develop and evaluate a machine-learning framework to predict investor churn in sustainable investment funds, using Meewind as a case study. Investor churn poses a significant strategic challenge in the asset management industry, where maintaining long-term client relationships is crucial to ensuring stable capital inflows and fostering trust in the organization. Using behavioral and transactional data available within Meewind's systems, this research aimed to identify investors at risk of terminating their participation and to support proactive retention efforts.

Beyond the immediate business relevance, this study contributes to the academic literature on churn prediction in financial contexts. Sustainable investment platforms such as Meewind are characterized by relatively small datasets, long investment horizons, and infrequent churn events, which complicates the application of standard predictive modelling approaches. By explicitly addressing these challenges, this thesis demonstrates how machine learning can be applied in practice to support decision-making in sustainable finance.

8.2 Method and Results

The central methodological challenges in this study were data creation and the severe class imbalance inherent in the dataset, where churn events constituted only a small fraction of all investor observations. Data creation was particularly challenging due to the structure of the database and the temporal nature of the available information. Many features had to be reconstructed retrospectively to ensure that the models only used information

that would have been available at the time of prediction. This step was crucial to avoid look-ahead bias and to ensure the validity of the empirical results.

To address the class imbalance problem, two complementary approaches were applied: algorithm-level adjustment through cost-sensitive learning and data-level balancing using the Synthetic Minority Oversampling Technique (SMOTE). Both methods significantly improved the ability of the models to detect observations belonging to the minority churn class. In general, cost-sensitive learning resulted in higher recall while maintaining greater model stability, highlighting the importance of explicitly accounting for imbalance when the business objective prioritizes the detection of rare but high-impact events, such as investor churn.

A comprehensive set of machine learning algorithms was implemented and compared, ranging from interpretable linear models such as Logistic Regression to more complex ensemble methods, including Random Forest, Gradient Boosting, Extremely Randomized Trees, and Extreme Gradient Boosting (XGBoost). Model performance was evaluated using both random and temporal train–test splits, allowing for an assessment under traditional cross-validation settings as well as more realistic, time-aware scenarios. Evaluation metrics such as recall, precision, F1-score, and ROC-AUC were selected to align with Meewind's strategic priorities, where failing to identify a churner is more costly than incorrectly identifying a non-churner.

Among all models tested, XGBoost emerged as the best-performing algorithm. After hyperparameter optimization using Bayesian search, it achieved a recall of approximately 77% and a ROC-AUC of 0.85 under temporal validation. Statistical testing confirmed that this performance was consistently superior to that of the other models. These results demonstrate XGBoost's strong capacity to capture complex, nonlinear relationships in investor behavior while remaining robust to noise and class imbalance. Although other ensemble-based methods such as Random Forest and Extremely Randomized Trees also performed well, XGBoost provided the most favorable trade-off between predictive power, computational efficiency, and generalization performance.

8.3 Managerial and Operational Added Value

Beyond predictive performance, this research provides clear managerial and operational value by illustrating how churn prediction can be integrated into Meewind's existing business processes. Before this study, retention efforts were largely reactive. Investors were

only contacted after registering a sell order, initially through an externally sourced call-center and later via generic marketing emails. Although these approaches aimed to retain customers, they often proved ineffective and, in some cases, counterproductive, as investors perceived the outreach as intrusive or unwanted. These strategies were also initiated only after investors had already made the decision to leave, limiting their potential impact.

The churn prediction framework developed in this thesis enables a fundamental shift from reactive to proactive retention management. By identifying investors who exhibit early warning signs of disengagement, Meewind can intervene before a sell order is placed. This allows for more targeted and timely communication, such as providing personalized performance updates, emphasizing the societal impact of investments, or offering tailored incentives. Contacting investors while they are still considering their options is likely to be more effective than trying to reverse a decision that has already been made.

From an operational perspective, the model allows Meewind to allocate retention resources more efficiently by prioritizing investors with the highest predicted churn risk. This reduces unnecessary outreach and supports more focused, data-driven marketing campaigns. In this sense, the added value of this thesis extends beyond predictive accuracy: it demonstrates how machine learning can meaningfully change the way retention strategies are designed and executed. By embedding predictive analytics into its customer relationship management processes, Meewind can strengthen long-term investor relationships, improve operational efficiency, and reinforce its position within the competitive market for sustainable investments.

8.4 Limitations

Nevertheless, several limitations should be acknowledged. First, the predictive models were developed on historical data, which means that their performance depends on the stability of investor behavior over time. Significant market or policy shifts, such as changes in interest rates or energy subsidies, could alter these behavioral patterns and require model retraining. Second, the analysis treated observations as independent, even though investor decisions may exhibit temporal dynamics. Future research could therefore explore models capable of capturing sequential dependencies, such as Long Short-Term Memory (LSTM) networks or other time-aware architectures. Third, while the study focused primarily on model performance, interpretability remains a challenge for complex ensemble algorithms. Some insights are provided through measures such as feature importance, but decision-makers may require more transparent explanations in practice. Finally, integrating external

data sources, including macroeconomic indicators or sentiment measures, could enhance predictive performance by capturing broader contextual influences on investor decisions.

In conclusion, this thesis demonstrates that machine learning can serve as a powerful tool for predicting investor churn in sustainable investment funds. By combining ensemble modeling, class imbalance correction, and careful evaluation design, the study provides both methodological rigours and practical value. For Meewind, the implementation of such a model represents a significant step toward data-driven customer relationship management, allowing the firm to anticipate investor behavior and strengthen its position as a leader in sustainable finance. More broadly, the findings underscore the potential of predictive analytics to support the long-term viability of sustainable investment strategies, contributing to a more resilient and forward-looking financial ecosystem.

References

Tanay Agrawal. *Hyperparameter Optimization in Machine Learning: Make Your Machine Learning and Deep Learning Models More Efficient*. Apress, Cham, 2020. ISBN 978-1-4842-6579-6. doi: 10.1007/978-1-4842-6579-6. 16, 17

Abdelrahim Kasem Ahmad. Customer churn prediction in telecom using machine learning in big data platform. *International Journal of Advanced Computer Science and Applications*, 10(8):220–228, 2019. 19

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012. 17

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006. ISBN 978-0-387-31073-2. 22

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324. 10, 47

Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984. 46

J. Burez and D. Van den Poel. Handling class imbalance in customer churn prediction. *Expert Systems with Applications*, 36(3):4626–4636, 2009. doi: 10.1016/j.eswa.2008.05.027. 21, 32

Jonathan Burez and Dirk Van den Poel. Crm at a pay-tv company: Using analytical models to reduce customer attrition by targeted marketing for subscription services. *Expert Systems with Applications*, 32(2):277–284, 2007. 20

Colin Campbell and Yiming Ying. *Learning with Support Vector Machines*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011. ISBN 9781608456161. doi: 10.2200/S00324ED1V01Y201102AIM010. 12, 13

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016. 15, 48

Kristof Coussement and Dirk Van den Poel. Churn prediction in subscription services: An application of support vector machines with parameter selection. *European Journal of Operational Research*, 193(3):412–424, 2009. 20

A. E. Eiben and Zbigniew Michalewicz. Modelling customer retention with statistical techniques, rough data models, and genetic programming. *Computational Management Science*, 1(1):27–45, 2003. 20

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. 46

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001. 11, 12, 47

Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002. 47

Louis Geiler and Thomas Klein. A survey on machine learning methods for churn prediction. *ACM Computing Surveys*, 55(4):1–36, 2022. doi: 10.1145/3524104. 3

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006. 14, 48

X. Guo, Y. Yin, C. Dong, G. Yang, and G. Zhou. On the class imbalance problem. In *2008 Fourth International Conference on Natural Computation*, volume 4, pages 192–201. IEEE, 2008. doi: 10.1109/ICNC.2008.871. 21, 32, 41

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, NY, 2nd edition, 2009. doi: 10.1007/978-0-387-84858-7. 50

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2010. 47

REFERENCES

Bingquan Huang, Mohand Tahar Kechadi, and Brian Buckley. Customer churn prediction in telecommunications. *Expert Systems with Applications*, 39(2):1414–1425, 2012. doi: 10.1016/j.eswa.2011.08.024. 19, 22

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Springer, Cham, 2019. ISBN 978-3-030-05317-8. doi: 10.1007/978-3-030-05318-5. 18

Praveen Lalwani, Manish Kumar Mishra, Jasjeet Singh Chadha, and Prashant Sethi. Customer churn prediction system: A machine learning approach. *Computational Intelligence in Data Mining*, pages 221–232, 2019. doi: 10.1007/978-981-13-8676-3_20. 3, 20, 22

Awais Manzoor, Rauf Ahmad, and Ali Ahmad. A review on machine learning methods for customer churn prediction and recommendations for business practitioners. *IEEE Access*, 9:150064–150088, 2021. doi: 10.1109/ACCESS.2021.3124652. 3

Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, 7:21, 2013. doi: 10.3389/fnbot.2013.00021. 11, 12

Scott A Neslin, Sunil Gupta, Wagner A Kamakura, Junxiang Lu, and Charlotte H Mason. Defection detection: Measuring and understanding the predictive accuracy of customer churn models. *Journal of Marketing Research*, 43(2):204–211, 2006. doi: 10.1509/jmkr.43.2.204. 20

Andrew Y Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004. 46

Thais Mayumi Oshiro, Patricia Souto Perez, and Jose Augusto Baranauskas. How many trees in a random forest? *Machine Learning and Data Mining in Pattern Recognition*, pages 154–168, 2012. 47

S. A. Qureshi, A. S. Rehman, A. M. Qamar, A. Kamal, and A. Rehman. Telecommunication subscribers' churn prediction model using machine learning. *International Journal of Advanced Computer Science and Applications*, 4(11):1–6, 2013. 19

Frederick F. Reichheld and W. Earl Sasser. Zero defections: Quality comes to services. *Harvard Business Review*, 68(5):105–111, 1990. 1, 19

Werner Reinartz and V. Kumar. The impact of customer relationship management on firm performance. *Journal of Marketing*, 69(1):76–80, 2005. 20

Lior Rokach and Oded Maimon. Decision trees. In *Data Mining and Knowledge Discovery Handbook*, pages 165–192. Springer, Boston, MA, 2005. doi: 10.1007/0-387-25465-X_9. 9

Lior Rokach and Oded Maimon. Decision trees in data mining. *Data Mining and Knowledge Discovery Handbook*, pages 165–192, 2010. 46

Glory Sam and Ramkumar Kannan. Customer churn prediction using machine learning models. *International Journal of Data Science and Analytics*, 15(2):175–188, 2023. doi: 10.1007/s41060-022-00333-5. 3

Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004. 47

Sandro Sperandei. Understanding logistic regression analysis. *Biochemia Medica*, 24(1): 12–18, 2014. doi: 10.11613/BM.2014.003. 7, 8

Thanasis Vafeiadis, Konstantinos I Diamantaras, Georgios Sarigiannidis, and Konstantinos Ch Chatzisavvas. A comparison of machine learning techniques for customer churn prediction. *Simulation Modelling Practice and Theory*, 55:1–9, 2015. doi: 10.1016/j.simpat.2015.03.003. 21, 23

Wouter Verbeke, Kris Dejaeger, David Martens, Bart Baesens, and Jan Vanthienen. New insights into churn prediction in the telecommunication sector: A profit-driven data mining approach. *Expert Systems with Applications*, 39(1):1–10, 2012. doi: 10.1016/j.eswa.2011.07.027. 20

Chih-Ping Wei and I-Tsang Chiu. Turning telecommunications call details to churn prediction: A data mining approach. *Expert Systems with Applications*, 23(2):103–112, 2006. 19