Lisbon School of Economics & Management
Universidade de Lisboa

MASTER
ACTUARIAL SCIENCE

MASTER'S FINAL WORK
INTERNSHIP REPORT

LAPSE MODELING IN HOME INSURANCE USING MACHINE LEARNING

MARÍA DEL CARMEN SACHEZ NGUEMA

JUNE 2025

Lisbon School of Economics & Management
Universidade de Lisboa

MASTER
ACTUARIAL SCIENCE

MASTER'S FINAL WORK
INTERNSHIP REPORT

LAPSE MODELING IN HOME INSURANCE USING MACHINE LEARNING

MARÍA DEL CARMEN SACHEZ NGUEMA

SUPERVISION:
NUNO SOBREIRA
MIGUEL SEIXAS

JUNE 2025

ACKNOWLEDGEMENTS

First, I want to thank my supervisors, Nuno Sobreira and Miguel Seixas, for trusting in my abilities and supporting me throughout this experience. Thank you for your patience, your willingness to help, and for guiding me every step of the way.

I also want to thank Allianz Portugal for opening its doors to me and giving me the opportunity to carry out this internship. A special thanks to the Pricing team for welcoming me from day one, making me feel like part of the team, helping me adapt to the daily routine, and creating an environment where I always felt comfortable to learn, make mistakes, and keep moving forward.

To my masters colleagues, thank you for sharing this chapter with me, for listening, supporting, and encouraging me during the difficult moments, and for sharing every joy and accomplishment along the way.

I also want to thank all the teachers who have taught me throughout these two years of the masters program, providing me with the tools and knowledge I needed to build and develop my future as an actuary.

Finally, and most importantly, I want to thank my family, especially my parents, for their love and for always being by my side through every challenge I have faced. Thank you for supporting me during the tough days and for always believing in me and my abilities.

To everyone who is a part of my life, thank you from the heart. This journey would not have been the same without you.

ABSTRACT

This thesis focuses on developing a predictive model to anticipate home insurance policy cancellations (lapses) following the issuance of a renewal offer with a revised price. Lapse behavior is a major concern for insurance companies, as it directly affects premium income and portfolio stability. Accurately identifying which customers are at higher risk of cancellation enables more effective and personalized retention strategies.

The model is built using LightGBM, a gradient boosting machine learning algorithm that handles complex interactions and correlated variables efficiently. It is trained on a dataset that includes commercial data, customer behavior indicators, and product characteristics. Feature selection is supported by techniques such as Boruta, PCA, and SHAP values.

Model performance is evaluated using metrics such as the AUC (Area Under the ROC Curve) and the Brier Score, measuring both discriminative power and probability calibration. Hyperparameter optimization is conducted using the Optuna framework to fine-tune model performance. SHAP values are also used to provide interpretability at both the global and individual level, making the model more actionable in a business context.

Results show that the proposed machine learning model outperforms traditional actuarial methods, such as Generalized Linear Models (GLMs), in both accuracy and calibration. Furthermore, the model delivers interpretable results that support informed decision-making. Overall, this thesis contributes a robust and practical tool for enhancing customer retention strategies in the insurance industry.

RESUMO

Esta dissertação tem como objetivo desenvolver um modelo preditivo para antecipar cancelamentos de apólices de seguro residencial (lapses) após o envio de uma proposta de renovação com um novo preço. O comportamento de cancelamento representa um desafio relevante para as seguradoras, pois afeta diretamente a receita com prêmios e a estabilidade da carteira de clientes. A capacidade de prever com antecedência quais clientes têm maior probabilidade de cancelar permite a definição de estratégias de retenção mais eficazes e personalizadas.

O modelo é construído com o algoritmo LightGBM, uma técnica moderna de machine learning baseada em árvores de decisão, capaz de lidar com interações complexas e variáveis altamente correlacionadas. O treinamento é realizado com um conjunto de dados que inclui informações comerciais, indicadores de comportamento do cliente e características do produto. A seleção de variáveis é apoiada por métodos como Boruta, PCA e valores SHAP.

A avaliação do desempenho do modelo é feita por meio de métricas como a AUC (Área sob a Curva ROC) e o Brier Score, que medem tanto a capacidade discriminativa quanto a calibração das probabilidades previstas. A otimização dos hiperparâmetros é conduzida utilizando a biblioteca Optuna, com o objetivo de maximizar o desempenho preditivo. Os valores SHAP também são utilizados para fornecer interpretabilidade das previsões em nível global e individual, facilitando a aplicação prática do modelo.

Os resultados demonstram que o modelo proposto supera abordagens atuariais tradicionais, como os Modelos Lineares Generalizados (GLMs), tanto em termos de precisão quanto na calibração das previsões. Além disso, o modelo fornece estimativas interpretáveis que podem ser usadas diretamente no suporte à decisão. Em resumo, esta dissertação oferece uma ferramenta robusta e aplicável para apoiar estratégias de retenção de clientes no setor de seguros.

DISCLAIMER

This masters internship report was prepared in full compliance with the academic integrity standards of ISEG, Universidade de Lisboa. All content presented reflects my personal research, reasoning, and authorship, except where proper references are provided.

To ensure transparency, I acknowledge that certain AI tools were utilized throughout the development process, specifically to assist with tasks such as translating content, exploring academic literature, and managing formatting in LaTeX. Nonetheless, the interpretation of findings, critical thinking, and final written output are entirely my own.

TABLE OF CONTENTS

List of Figures

List of Tables

ABBREVIATIONS

AUC Area Under the ROC Curve

ROC Receiver Operating Characteristic

GLM Generalized Linear Model

IRLS Iteratively Reweighted Least Squares

EDA Exploratory Data Analysis

SHAP SHapley Additive exPlanations

PCA Principal Component Analysis

ML Machine Learning

LGBM / LightGBM Light Gradient Boosting Machine

BIC Bayesian Information Criterion

AIC Akaike Information Criterion

Brier Score Metric to assess probability calibration

EIOPA European Insurance and Occupational Pensions Authority

PCA Principal Component Analysis

CatBoost Categorical Boosting

FANOVA Functional Analysis of Variance

QQ plot Quantile-Quantile plot

TPE Tree-structured Parzen Estimator

SMBO Sequential Model-Based Optimization

1 INTRODUCTION

1.1 Main objective of the thesis

The main goal of this thesis is to build and explain a model that predicts when customers will cancel their home insurance after receiving a renewal offer with a new price. The idea is to identify patterns and important factors that explain customer decisions. The analysis only looks at cancellations that happen at the time of renewal and excludes cancellations during the policy period (mid-term lapses). This helps keep the focus on cancellations caused by price changes.

The study also compares the machine learning model with traditional actuarial methods, like Generalized Linear Models (GLMs). Machine learning is used because it can find more complex patterns and relationships in the data. This allows for better predictions and more useful results for insurance companies.

Interpretability is also a core objective. The model will leverage SHAP values to determine the relative contribution of each feature to the predicted lapse probability. This makes the model more actionable in a business environment, enabling clearer communication with stakeholders and more targeted retention and pricing strategies.

Finally, the thesis includes a comparison between the machine learning model and a Generalized Linear Model (GLM) built using Emblem, which is a market-standard pricing software used in the insurance industry. The goal is to measure the difference in performance and show how modern predictive methods can improve the detection of cancellations and help insurers make better business decisions.

2 THEORETICAL FRAMEWORK

2.1 The lapse concept in the insurance sector

In the insurance industry, a lapse refers to a situation where a policy becomes inactive because the customer doesn't take the necessary steps to renew it, such as failing to pay the premium or not accepting a new renewal price. Unlike an active cancellation, where the policyholder clearly informs the insurer of their decision to end the contract, a lapse happens passively. The customer simply does nothing, and as a result, the policy expires automatically without any formal cancellation request.

For insurers, these lapses aren't a minor issue. Each policy that isn't renewed represents lost revenue and a possible reduction in their client base. In extreme cases, it can even affect the financial stability of the company. This is especially relevant in insurance such as home insurance, where, in times of economic hardship, many people may see it as a non-essential expense.

To measure how many policies are lost in this way, a key metric is used: the lapse rate. Basically, it is calculated by dividing the number of policies that weren't renewed in a given period by the total number of active policies at the beginning of that same period. In mathematical terms, it is represented as follows:

$$L_t = \frac{1}{n_t} \sum_{i=1}^{n_t} Y_{it}$$

where:

- t is a fixed time period (one month or one year) during which lapse behavior is measured.
- L_t is the lapse rate in the time period t.
- n_t is the total number of active policies at the start of time period t.
- $Y_{it} = 1$ if policy i was cancelled during time period t, and $Y_{it} = 0$ if it remained active during t.

This formula allows us to understand lapse as an average of individual decisions: each client may or may not renew his policy, and this is reflected in the total rate.

In this thesis, the analysis focuses only on cancellations that occur after the customer receives a new renewal offer. Other types of cancellations, such as mid-term lapses due to changes in the contract, death, or internal administrative decisions, are not included. The objective is to focus on cases where the customer decides not to renew because they are unhappy with the new price, a behavior known as price sensitivity lapse.

Applying this definition to the survey data, the observed lapse rate was found to be 8.61%. That is, almost 1 in 12 customers who received a renewal offer decided not to continue with their insurance.

Although this percentage may not seem so high, its economic impact can be very significant. That is why it's essential to have predictive models that help identify which customers are closest to cancelling. This way, insurers can act before this happens, designing more effective retention strategies.

2.2 Factors associated with the lapse

One of the biggest challenges insurers face is understanding how their clients react when they receive a new price proposal when renewing their policy. In the case of home insurance, this is further complicated by the fact that two homes that appear similar at first glance can be priced very differently due to a number of factors (Galilea, 2025; Seguros, 2025).

When it comes time to renew, the client's decision doesn't depend only on the new premium value. There are many elements that come into play, and they can be grouped into five broad categories:

- **Location and environment of the property:** Factors such as the area where the property is located, whether there is a risk of natural disasters, whether the area has high crime rates, or whether it is close to emergency services such as fire brigades, directly affect the level of risk and thus the price (Galilea, 2025).
- **Profile of the insured:** Credit history, previous claims, how long the insured has been insured with the company, and even marital status can influence how their level of risk is perceived (Seguros, 2025).
- **Physical condition of the property:** The age of the house, the construction materials, whether or not it has been remodeled, and the estimated replacement value are key aspects that also impact the rate (Galilea, 2025).
- **Contract details:** Elements such as the type of deductible chosen, the limits of coverage, whether there are additional risks such as swimming pools or pets, or whether commercial activities are conducted at the home, can also adjust the price (Seguros, 2025).
- **General economic factors:** Variables such as inflation, the rising cost of building materials, supply chain problems, and the state of the reinsurance market can force up renewal prices, creating a negative customer perception (de Supervisão de Seguros e Fundos de Pensões (ASF), 2024; of Insurance, 2022).

When customers feel that the new price does not match the value they receive or what they consider fair, they are more likely to decide not to renew their policy. In addition, strong competition in the insurance market increases this risk, as customers can easily compare prices and switch to another company offering a better deal. Therefore, understanding these factors and being able to model them effectively is key to anticipating which customers are at risk of cancelling in response to a price adjustment (de Supervisão de Seguros e Fundos de Pensões (ASF), 2024).

2.3 Introduction to traditional GLM models

Within insurance data analysis, generalized linear models, or GLMs, are one of the most widely used tools for predicting binary events such as lapse that is, when a customer decides not to renew his or her policy after receiving a new price proposal. These models are highly valued because they are flexible enough to incorporate many factors at once and adapt to insurance industry data, which rarely follow ideal patterns such as normality or constant variance.

As De Jong and Heller (2008) explain, the key to GLMs is that they extend linear regression by allowing the dependent variable to follow other distributions within the exponential family. This includes the binomial distribution (ideal for variables that can only be 0 or 1), the Poisson (for counts), or the gamma (for continuous positive values). In the case of lapse, where the customer either cancels (1) or renews (0), it is most appropriate to use a binomial distribution with only one observation per customer (i.e., $n = 1$).

In this context, the variance of the response variable is calculated as:

$$\text{Var}(Y_i) = \mu_i(1 - \mu_i)$$

where μ_i is the probability that customer i will cancel his policy.

A GLM model is built from three key components:

1. The distribution of the response variable.
2. A link function that connects the response variable to the predictors.
3. A linear combination of the explanatory variables.

To predict the lapse, the most commonly used link function is the logit, which transforms the probability of cancellation into log-odds, and is expressed as follows:

$$\log\left(\frac{\mu_i}{1 - \mu_i}\right) = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}$$

This transformation allows us to use a logistic function to estimate the probability of cancellation as a function of the predictors. Each β_j indicates how the probability changes (in terms of log-odds) when the variable x_{ij} changes by one unit, holding everything else constant. For example, if the coefficient associated with percentage increase in premium is positive, it means that as the price rises, the probability of cancellation increases.

The coefficients β are estimated using the maximum likelihood method, whose log-likelihood function for a sample of n observations is:

$$\ell(\beta) = \sum_{i=1}^{n} [y_i \log(\mu_i) + (1 - y_i) \log(1 - \mu_i)]$$

As this formula cannot be solved directly, an algorithm known as IRLS is used, which adjusts the values step by step using quadratic approximations until the best solution is found (De Jong & Heller, 2008; Ohlsson & Johansson, 2010).

One of the major advantages of GLMs is that they allow many relevant variables to be incorporated into the model. For example:

- How much the price increased compared to the previous year.

- How many years the customer has been with the insurer.
- Whether they contracted online or through an agent.
- Their claims history.
- The type of property.
- Or the geographical area.

All this allows the calculation of a customised churn probability, which can be used to segment customers, design retention campaigns or adjust prices.

To know if the model works well, indicators such as the deviancy, the AIC or predictive metrics such as the ROC curve and the AUC can be used. Residuals are also analysed to detect possible fitting errors. For example, the residual deviance in a binomial model is expressed as follows:

$$D = 2\sum_{i=1}^{n}\left[y_i \log\left(\frac{y_i}{\hat{\mu}_i}\right) + (1-y_i)\log\left(\frac{1-y_i}{1-\hat{\mu}_i}\right)\right]$$

Where $\hat{\mu}_i$ is the estimated probability of churn for each customer.

As Ohlsson and Johansson (2010) point out, another great advantage of GLMs is that they allow the construction of multiplicative tariffs, where each variable adjusts the base price in a clear and simple way. This is especially useful in insurance, where tariff structures must be transparent and auditable.

However, GLMs also have their limits. They assume that the relationship between predictors and probability is linear (on the scale of the link), which may not reflect reality in all cases. Furthermore, they don't automatically capture complex interactions and non-linear relationships, unless they are manually added to the model. For example, if the effect of property type changes by geographical area, this interaction will only be taken into account if it is explicitly included.

2.4 Introduction to Machine Learning

For years, traditional models such as GLMs have been a key tool in the insurance world for predicting customer behaviours, such as when they are likely to cancel their policy. They are robust, well known, and relatively easy to interpret. But they also have their limits. They work under certain assumptions, such as that the relationships between the variables and the response must be specified in advance and that those relationships are linear (on a transformed scale, at least). In the real world, where data is huge, complex, and full of unexpected interactions, this can fall short.

Within the wide world of ML, there are several models that are used especially for predicting policy cancellations (lapse). Some of the most common are decision trees, Random Forests, and boosting models such as GBM and XGBoost.

Decision trees work as a series of yes or no decisions that split the dataset based on specific rules. They are intuitive and easy to interpret. However, when used as a single model, they can overfit the training data

which reduces their accuracy on new case. Random Forests improve on this by generating many different trees and averaging their results, which helps avoid errors and improves stability. On the other hand, models such as GBM and XGBoost build the trees one by one, learning from the mistakes of the previous one, which gives them a great ability to identify complex patterns. According to Friedman et al. (2001), this type of model is ideal when high accuracy is sought and a rich database of variables is available.

Recent studies, such as Henckaerts et al. (2022), have shown that these boosting models, and in particular XGBoost, outperform GLMs when it comes to predicting policy cancellations. They achieve better results in key metrics such as accuracy, sensitivity, and AUC (area under the ROC curve). This reinforces the idea that, in complex environments, ML can make a big difference.

However, this improvement in accuracy comes at a price. One of the challenges with Machine Learning is that the models are much less transparent. Unlike a GLM, where each variable has a clear effect that can be directly interpreted, ML models often operate as a black box. We know they predict well, but it isn't always clear why they make a decision. And that can be a problem, especially in insurance, where decision justification is key for both the customer and regulators.

Fortunately, in recent years, tools have been developed to help interpret these complex models. One of the most widely used is SHAP, which explains how a model arrives at its decisions by showing the contribution of each variable to individual predictions. SHAP relies on concepts from game theory to assign a fair importance value to each feature. Molnar (2022a) provides a comprehensive guide on how to apply SHAP in the context of advanced classification models.

In summary, Machine Learning offers a clear improvement in predictive ability to understand why customers cancel their policies. It is true that these models are less interpretable than GLMs, but their analytical power and flexibility make them a valuable complement. Throughout this thesis, we will explore how different ML models applied to lapse perform in home insurance, comparing them with the GLM both in terms of accuracy and their potential for real-world environments.

2.5 Evaluation metrics for prediction models

Assessing the quality of a predictive model is a key part of any supervised analytics, and is particularly relevant when it comes to anticipating household policy cancellations, where what really matters is to accurately identify which customers are likely to not renew. For that, evaluation metrics help us to measure how well the model is working and to compare it with other alternatives. Of course, there is no universally best metric: it all depends on the type of variable we are predicting, whether the classes are balanced or not, and the objective of the model (ranking, estimating probabilities, or simply prioritizing cases) (Géron, 2019).

In this case, lapse is a binary classification problem: the customer either cancels or doesn't cancel. The most commonly used metrics to evaluate it are accuracy, precision, recall, specificity, F1-score, area under the ROC curve (AUC-ROC), and gain and lift curves. Each of these gives us a different look at the behavior of the model (Géron, 2019).

The most basic metric is accuracy, which indicates the proportion of correctly classified cases. However, if cancellations are rare, a model that always predicts 'didn't cancel' may show high accuracy but be useless for

identifying real cancellations.

More informative metrics are based on the confusion matrix: TP (True Positives), FP (False Positives), TN (True Negatives), and FN (False Negatives). From these, we calculate:

TABLE I: Confusion matrix-based evaluation metrics

Metric	**Formula**	**Interpretation**
Precision	$\frac{TP}{TP+FP}$	Percentage of predicted cancellations that were correct.
Recall (Sensitivity)	$\frac{TP}{TP+FN}$	Proportion of actual cancellations correctly identified.
Specificity	$\frac{TN}{TN+FP}$	Ability of the model to correctly identify non-cancellations.
F1-Score	$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$	Harmonic mean of precision and recall.

For a more global view, especially when varying the decision threshold, the ROC curve is used, which shows how the true positive and false positive rates change across different cut-off points. The AUC-ROC (area under this curve) summarises the model's ability to distinguish between cancellers and non-cancellers. An AUC of 0.5 indicates a random model, while one close to 1.0 indicates very good discriminatory ability. In studies on cancellations, such as Henckaerts et al. (2022), the AUC is often used to compare the performance of traditional models such as GLM with more advanced options such as XGBoost.

Another very useful metric in the real world is the gain and lift curves. These tell us how many lapses are concentrated in the most likely segments according to the model. This is key when you want to apply retention actions to only a portion of the portfolio, for example, offer incentives to the 10% most likely to cancel. The lift tells us how many times more likely it is to find a lapse in that group compared to the overall average (Fawcett, 2006).

Calibration quality is also relevant and is assessed in detail through the Brier Score, discussed in Section 5.3.1.

3 DATA AND PREPARATION

3.1 Description of the dataset

For this study, an internal database of Allianz Portugal, an insurer operating in the multi-risk home insurance market in Portugal, was used. The particularity of this dataset is that it only includes cases where, at renewal time, the client received a new premium proposal. This is very important, as it allows to focus on those cancellations that occur directly in response to a price change, a key aspect in both retention strategies and pricing processes.

In total, the dataset contains around 118,000 observations, each representing a renewal opportunity over an annual period. For each of these policies, it is recorded whether the customer accepted the new premium or decided not to renew. This information is reflected in a binary variable indicating whether a lapse occurred or not.

The variables in the dataset are grouped into five main blocks:

- **Commercial variables:** These describe aspects of the policy, such as the type of product, deductibles, current and previous premium, and the sum insured. These variables help to understand the economic and contractual dimension of the renewal.
- **Customer variables:** Include information such as tenure , claims history, number of policies purchased, and country of residence. This group makes it possible to profile the relationship between the customer and the company.
- **Distribution channel variables:** These provide data on the broker or agent linked to the policy, including their experience and the number of policies they manage. This makes it possible to analyse whether the channel through which the policy was taken out influences the renewal decision.
- **Geographic risk variables:** These variables describe the physical and social environment of the insured property, including exposure to natural hazards, population density, social vulnerability, and characteristics of the territory (such as land use or vegetation cover).
- **Derived variables:** These are transformations or combinations of the previous ones. Of note here is the percentage increase of the premium with respect to the insured capital, a critical variable for understanding the client's reaction to changes in price.

From a technical point of view, the dataset includes both continuous numerical and categorical variables, allowing for a diverse and comprehensive analysis combining structural, socio-economic and commercial factors. To ensure confidentiality, all variables have been anonymised and are presented under generic names, without reference to specific internal or commercial labels.

Taken together, this dataset represents a solid and representative basis for lapse analysis in household insurance. The combination of business information and external data on the geographical and social context allows us to approach the phenomenon from a holistic perspective, aligned with current challenges in retention models and customer behaviour prediction.

3.2 Exploratory Data Analysis

Before building any predictive model, it is essential to take the time to explore the data. This step, known as EDA, allows us to understand what we have on our hands: how the variables are distributed, what relationships exist between them, whether there are data that don't make sense, outliers, or errors that could affect the analysis.

In this study, the EDA focused on preparing the dataset for modelling policy cancellations in home insurance, specifically in the multi-risk homeowners business. The main objective was to leave the data in optimal conditions to apply predictive techniques, but also to take advantage of this stage to identify interesting patterns that could help to better understand the behaviour of customers when deciding whether or not to renew their policy.

In addition to detecting possible data quality issues, this preliminary analysis served as a first approach to the lapse phenomenon, allowing us to uncover early signals about which factors might be most related to the decision to cancel a policy after receiving a new price offer.

3.2.1 Data preparation, visualisation and cleaning

The analysis process started with the import of the dataset, stored in a SAS format file. Subsequently, the categorical formats defined in the internal technical dictionaries were applied in order to facilitate the readability and comprehension of the available variables.

As part of the variable engineering tasks, the derived variable 'Variable 1' was generated, defined as the ratio of the pre-renewal premium to the total sum insured. This ratio was considered an approximation of the level of economic burden assumed by the policyholder in relation to the value of the contracted cover.

In parallel, information on dates, specifically the month of renewal of the policy, was treated. This variable, originally in date format, was transformed into a categorical variable 'Variable 2' by extracting the month number (1 to 12) and then labelling it with the corresponding name. This transformation allows for capturing possible seasonal effects on customer renewal behaviour.

Once these transformations were performed, the overall cancellation rate was calculated, yielding a value of 8.61%. This percentage indicates that, on average, one out of every twelve policies in the sample was cancelled during the period analysed.

Next, a univariate analysis was carried out on the categorical variables ('Variable 3' to 'Variable n'). At this stage, the following were evaluated:

- The absolute frequencies of each category.
- The number and proportion of cancellations within each group.

The results were represented by bar plots combined with lines, which allowed a clear visualisation of the different levels of cancellation according to specific characteristics of the policies or policyholders.

3.2.2 Elimination of variables unsuitable for modelling

Before starting to reduce the number of variables through correlation analysis, it was necessary to perform an initial cleaning of the dataset. This stage focused on eliminating those variables that, because of their nature or how they are used within the insurance process, weren't suitable for building a reliable predictive model.

Many of these variables corresponded to information that is only known after the policy term ends, such as the actual duration of coverage or events that occur after the contract has started or even ended. Including such variables would have introduced a bias, as it would be using future information to predict an event that, in practice, hasn'tt yet occurred. In a real scenario, this type of data wouldn't be available at the time of decision making, so its use would have compromised the validity of the model.

We also removed variables that, although technically available, were too closely related to the cancellation itself. For example, we excluded fields that showed the exact end date of the policy or directly said whether it was cancelled. Using these variables would break the rule that predictors must be known before the result happens, which is very important in supervised models. In real life, this kind of information is only known after the customer makes their decision, so it cannot be used to predict it.

This initial clean up not only helped ensure the model's consistency but also simplified subsequent analysis by reducing the number of variables to work with. The result was a data set that was clearer, more manageable, and better aligned with a real insurance decision-making context.

3.2.3 Collinearity analysis and reduction of redundancies using a correlation matrix

After finishing the initial data cleaning, a collinearity analysis was carried out among the numerical variables. This step was important to ensure that the variables used in the model weren't too closely related to each other. When variables are highly correlated, it can cause problems such as making the model less stable, harder to interpret, and even more likely to overfit the data.

Collinearity, or multicollinearity when it involves more than two variables, happens when two or more variables give almost the same information. To detect this, a Pearson correlation matrix was used. This matrix shows the strength of the linear relationship between pairs of variables. The correlation coefficient goes from -1 to 1: values close to 1 or -1 show a strong relationship, while values near 0 mean there is little or no linear connection. In this case, a threshold of 0.80 was chosen: if two variables had a correlation higher than that, they were considered too similar to keep both in the model.

The analysis was done only on numerical variables, including both the original ones and the ones created during feature engineering. However, a high correlation between two variables did not automatically mean one had to be removed. Each case was reviewed carefully, considering how well each variable explained the target variable (policy cancellation), how clear its meaning was, and the quality of its data. If a variable was just a transformation of another and didnt add any new information, it was removed in favor of the more useful one.

A clear example was found with two variables related to the insured capital and the total premium, which had a correlation above 0.90. In that case, a ratio variable was kept instead, since it also showed stronger predictive power in previous analyses.

Separately, we also removed variables that were redundant, that is, those that didn't provide new or independent information. This included highly correlated fields or features derived from the same base metric. Keeping only one variable in such cases helped simplify the model without losing predictive power. As a result, the final set of predictors was smaller, more practical for real-world use, and better aligned with the goals of interpretability and stability.

The full correlation matrix and details of the decisions made are included in the technical annex at the end of this document, to ensure transparency and traceability of the process.

3.3 Feature selection for modeling

After completing the initial data cleaning and variable reduction steps, the next phase focused on defining the final set of attributes to be used as predictors in the machine learning model aimed at estimating policy cancellation (lapse). This stage, known as feature selection, plays a critical role in the modeling process, as it directly impacts the models accuracy, robustness, and interpretability. Additionally, it helps reduce the risk of overfitting and improves computational performance.

In this context, two types of variables were combined: those that had already been cleaned and validated during the earlier steps, and the categorical variables identified based on their data type in the Dataframe, specifically, those that weren't numeric (neither float nor int). Categorical variables are especially valuable in this domain, as they capture structural and behavioral dimensions such as building type, payment frequency, geographic location, or sales channel, which are often closely linked to cancellation patterns.

The resulting dataset, includes only the variables selected for modelling. It excludes variables that were removed for technical reasons, such as those only available after the policys renewal or due to high collinearity. This Dataframe serves as the operational base for the next steps in the process: data imputation, encoding, and model training.

Despite the careful selection of variables, many of them contain missing values, which is common in datasets built from administrative records or mixed data sources. This situation requires a clear imputation strategy, which will be addressed in the next section. This step is essential to ensure the completeness and structural consistency of the input matrix before training the predictive model.

3.4 Data Cleaning, Imputation, and Transformation

Handling missing values is an important part of preparing the data. Some algorithms, like LightGBM, can work with missing data, but it is still a good idea to treat them carefully. This helps us control how the model learns from missing values and avoid unexpected effects on the predictions. In this study, several variables were found to contain missing values, both numeric and categorical. To preserve the informational value of the dataset and avoid losing records, a structured imputation strategy was implemented.

The first step involved identifying which variables had missing data. In some cases, these were external variables, such as socio-environmental indicators, while in others, they referred to specific characteristics of certain policies or clients, which were not always available.

The imputation approach combined different methods based on the type and nature of each variable:

- For a specific subset of variables impact or logically interpretable-missing values were replaced with zero. This decision was based on the fact that, in these cases, the absence could be reasonably interpreted as a null or non-applicable value.
- For the remaining numeric variables, missing values were imputed using `-1`, following best practices for tree-based algorithms like LightGBM and XGBoost. These models treat `-1` as an additional category rather than a true numeric value, allowing the structure of the data to be preserved without introducing bias.
- In the case of categorical variables, missing values were replaced with a synthetic category called `"99_Missing"`, allowing the model to treat missing data as a valid and distinguishable class, without losing any underlying information.

After completing the imputation process, all categorical variables were transformed to be compatible with LightGBM. Specifically, all variables of type object were converted to the categorical type. This transformation improved training efficiency, reduced memory usage, and enabled the model to internally manage categorical variables.

Finally, a full validation was performed to ensure that the dataset was free of any remaining null values. This check confirmed that the dataset was fully prepared for the training and validation of both linear and non-linear predictive models.

3.5 Dataset Partitioning (Train-Test Split)

After completing the data imputation and transformation steps, the next phase involved splitting the dataset into training and test subsets.

The main goal of this split is to train the model on one portion of the data (training set) and evaluate its performance on a separate portion (test set). This is a standard approach to prevent overfitting and ensure that the model has the ability to generalize to new, unseen data.

Figure 1 shows a simple visual example of this process.

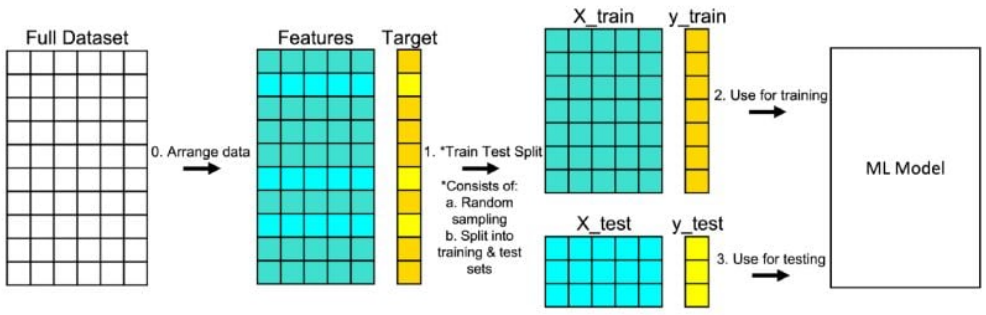

FIGURE 1: Train-test split: separation of data for model training and evaluation. Source: Built In

In this study, the dataset was partitioned into a training and a test set using a predefined split that ensures

both subsets maintain a similar distribution of the target variable. This technique is conceptually similar to bootstrapping and helps preserve the average cancellation rate across both groups. As a result, the evaluation of model performance remains fair and reproducible.

Approximately 80% of the data was allocated to the training set and 20% to the test set, following standard practice in supervised machine learning. The target variable was stored separately from the input features in both subsets, and auxiliary columns used for the split were excluded from the modeling process.

This partitioning is essential for the next steps of the analysis, as it allows the model to be trained on one part of the data and evaluated on a separate, unseen portion. This setup supports reliable performance measurement, unbiased hyperparameter tuning, and accurate computation of key evaluation metrics, such as accuracy, recall, specificity, and area under the ROC curve (AUC), which will be discussed in the following sections.

4 FILTERING

4.1 Selection of explanatory variables: combining PCA and Boruta

One of the most important steps before building a predictive model is deciding which variables to use as input. In complex problems, such as the one addressed in this thesis, which focuses on predicting the cancellation of home insurance policies after a new renewal price is offered, it is common to work with a large number of variables. Many of these may be redundant or have limited predictive power on their own. For this reason, selecting the most relevant variables becomes essential to improve the accuracy of the model and make it easier to interpret.

To tackle this task, a two-phase approach was used. First, a dimensionality reduction technique called Principal Component Analysis was applied. PCA generates new variables by combining the original ones in a way that captures underlying patterns in the data, allowing the most important information to be condensed into fewer dimensions.

Next, the Boruta algorithm was used, which is a variable selection method that automatically identifies which features, whether original or derived from PCA, carry useful predictive information. Boruta works by comparing the importance of each variable to that of randomly shuffled versions of itself (called shadow features), making it a robust tool for feature selection (Kursa & Rudnicki, 2010).

This combined approach not only reduces the number of variables used in the model but also ensures that only those with real predictive value are retained. The result is a more efficient model, better performance, and easier interpretation in practical applications.

4.1.1 Dimensionality Reduction with Principal Component Analysis

Principal Component Analysis is a widely used statistical technique that transforms a set of possibly correlated numerical variables into a smaller set of new uncorrelated variables called principal components. These new variables are linear combinations of the original ones and are ordered so that the first components capture the largest amount of variability in the data.

In this study, the main goal of applying PCA was not to directly reduce the number of variables, but rather to create new synthetic variables that could capture underlying relationships among the original features, especially those that might have limited predictive power when considered individually. By adding these derived components to the pool of candidate features, the model was given the opportunity to uncover hidden patterns that might not be evident when using only the original variables.

Figure 2 shows a graphical representation of the PCA process, illustrating how the original variables are projected onto new orthogonal axes (principal components), which better capture the structure of the data.

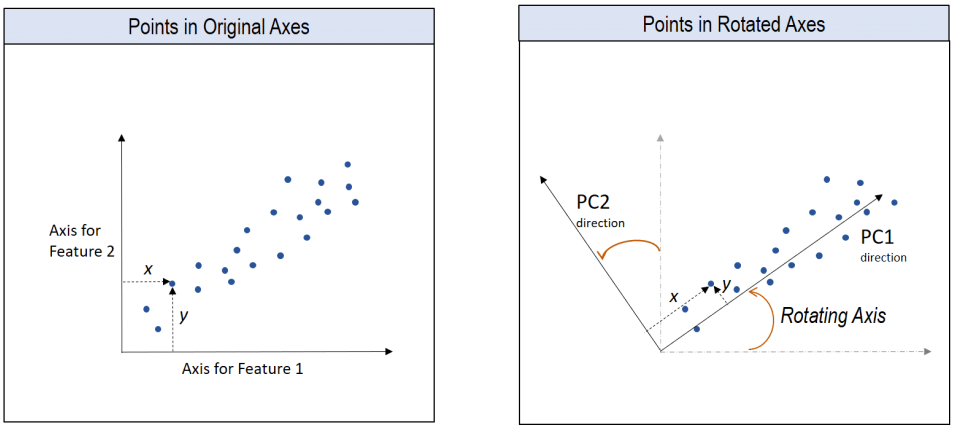

FIGURE 2: Principal Component Analysis: transformation from original axes to principal components. Source: Machine Learning Plus

In practical terms, PCA was applied only to continuous numerical variables, excluding those subject to monotonicity constraints. The number of components generated matched the number of original variables, and these new features were then included in the data set used in the next phase of variable selection.

4.1.2 Automatic Variable Selection with Boruta

After expanding the dataset with the principal components generated through PCA, the Boruta algorithm was used to automatically select the most important variables. Boruta, created by Kursa and Rudnicki (2010), is based on Random Forest models. Unlike other methods that try to find a minimal set of optimal variables, Borutas goal is to keep all variables that carry useful information for prediction.

Boruta works by comparing the importance of real variables with that of 'shadow features', which are created by randomly shuffling the values of the original variables to remove any relationship with the target. A Random Forest model is then trained on both real and shadow features, and for each variable, an importance score is computed, typically as a Z-score (standardized mean decrease in impurity across trees). In each iteration, Boruta identifies the maximum Z-score among all shadow features, referred to as the best shadow feature, and uses this as a threshold. A real variable is considered important if its Z-score is significantly higher than this threshold, irrelevant if it is significantly lower, and tentative if the difference is not statistically significant. This comparison is repeated in many iterations to ensure stability and robustness in the selection process.

In this thesis, Boruta was applied to the complete set of variables, including both original and PCA-derived ones. Only numeric variables (or those converted into numeric format) were used. The algorithm was set to run 100 iterations, using all available CPU cores to speed up the process.

Figure 3 shows the main steps of the Boruta algorithm, highlighting how it uses shadow variables and importance scores to filter out irrelevant features.

In the end, Boruta selected 45 important variables. These included:

- Original variables related to customer characteristics, product details, premium history, and geographic context.
- Principal components created from socioeconomic and territorial data.

This method made it possible to keep both explicit information and hidden patterns in the data, giving the predictive model a strong and well-rounded set of explanatory features.

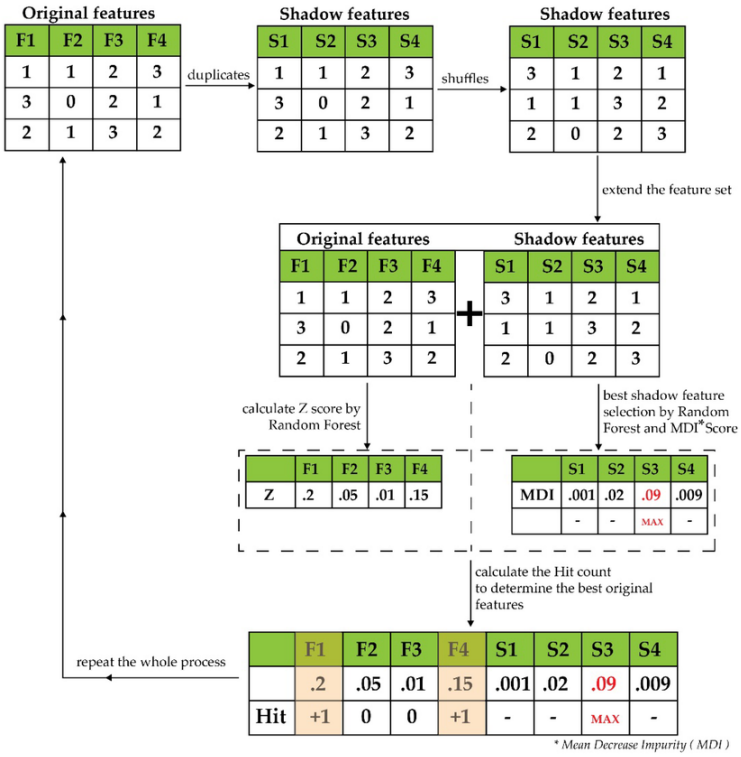

FIGURE 3: Steps of the Boruta feature selection algorithm. Source: ResearchGate

After identifying the most relevant predictive variables using the Boruta algorithm, one final filtering step was added, this time focused on understanding which variables actually help explain the models decisions. The goal was twofold: to improve performance while also making the model easier to interpret and more practical for real business use.

For this purpose, the SHAP technique was used. SHAP is a tool that shows how much each variable contributes to an individual prediction, using principles from game theory. In simple terms, SHAP helps answer the question: How much did this variable influence the models decision in this specific case? (Molnar, 2022a)

In this thesis, SHAP was applied to the test dataset after the final model was trained. Based on the analysis, the 20 variables with the highest explanatory impact were selected. This approach brought several advantages:

- It simplified the model, making it easier to manage.

- It improved performance, especially in terms of AUC (area under the ROC curve), from 69.47% to 69.83%.
- It made the results more understandable for both technical and business teams.

To protect confidentiality, the exact variable names have been anonymized. However, the selected variables include commercial factors (such as changes in the renewal price), client attributes (like customer tenure or policy profile), and geographic or socioeconomic features, either from external sources or generated through statistical transformations.

Since SHAP is still relatively new in the insurance industry and not widely known outside the machine learning field, its logic and benefits will be explained in more detail in the chapter focused on model interpretation.

5 MODELLING

5.1 LightGBM: Practical Implementation and Results

In this study, the LightGBM algorithm was used to predict whether a customer would cancel their home insurance policy after receiving a new renewal price. This model was selected due to its strong performance in handling complex, high-dimensional datasets, which is common in the insurance industry.

The training process involved a dataset combining commercial information, customer behavior data, and product characteristics.The model output was a score between 0 and 1, representing the estimated probability of cancellation for each customer. Since LightGBM produces well-calibrated scores when properly tuned, this output can be interpreted as a probability and used to guide business decisions. These probabilities were then used to support targeted decision-making and the development of retention strategies.

Details about the algorithms internal workings, advantages over traditional models, and the specific training configuration can be found in Appendix C.

5.2 Evaluation of the Predictive Model: ROC Curve and AUC

To assess model performance, the ROC curve (Receiver Operating Characteristic) and the AUC (Area Under the Curve) metric were used. These tools evaluate the model's ability to distinguish between two outcomes: cancellation and non-cancellation.

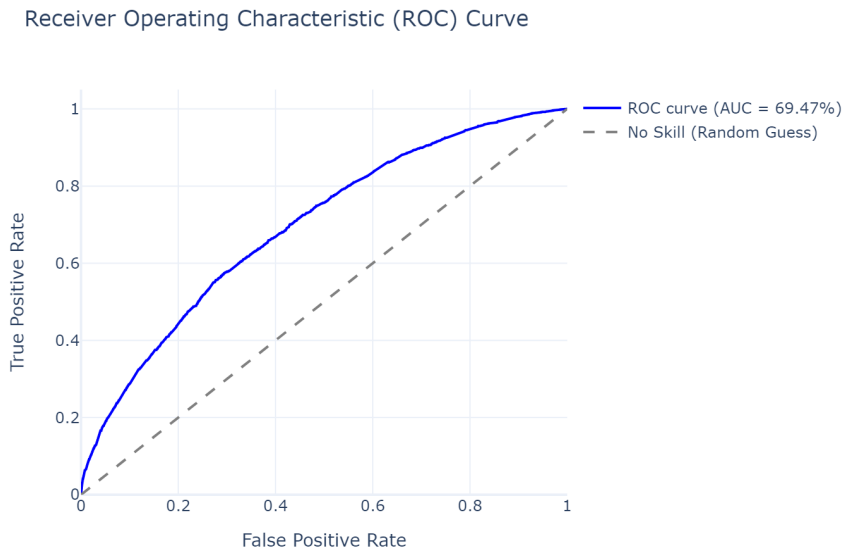

FIGURE 4: ROC curve for the LightGBM model trained on selected variables.

The ROC curve in Figure 4 shows that the model performed significantly better than random guessing, with

an AUC score of 69.47%. This means that the model can correctly assign a higher cancellation probability to a customer who actually canceled in nearly 70% of cases.

Although not perfect, this score is well above the baseline of 50% and reflects the models ability to identify meaningful patterns in the data. From a business perspective, this allows Allianz Portugal to focus retention efforts on high-risk customers, improving commercial efficiency and reducing portfolio loss.

AUC is particularly useful in this context because it doesn't depend on a fixed classification threshold, offering a comprehensive view of model performance across all decision boundaries.

5.3 Model Calibration Assessment

After training the model and analyzing its ability to distinguish between customers who cancel and those who dont, its equally important to check whether its predictions are reliable in absolute terms. In other words, it isn't enough for the model to detect risky cases, it should also assign probabilities that match what actually happens. This process is called model calibration, and it plays a particularly important role in the insurance industry, where decisions are often based on probabilities, not just binary classifications (Géron, 2019; Molnar, 2022b).

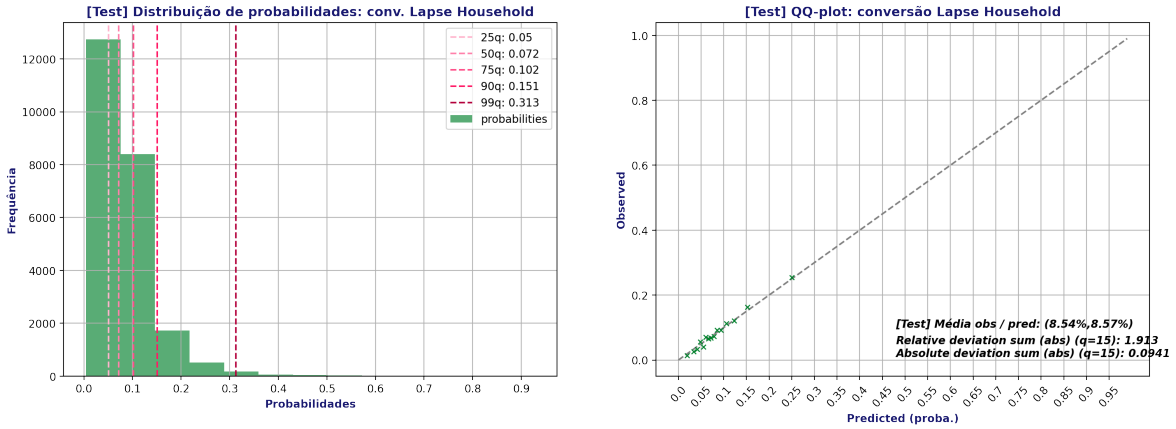

FIGURE 5: Distribution of predicted probabilities

To assess this, two key visualizations were generated. Figure 5 shows the first visualization: a histogram of predicted probabilities, which shows how the model distributes its risk estimates across the dataset. As expected, most predictions fall below 10%, which aligns with the actual low cancellation rate in the portfolio. However, the model also assigns higher probabilities to some specific cases, suggesting that it is indeed identifying profiles with a higher likelihood of cancellation Molnar (2022b).

The second visualization in Figure 5 is a QQ plot, which gives a more precise view of calibration. This graph compares the predicted probabilities with the actual cancellation rates across different segments of the test set (grouped by percentiles). If the points fall close to the diagonal line, it means the models predictions closely match the observed outcomes. In this case, the results are promising: the average predicted probability was 8.57%, compared to an observed cancellation rate of 8.54%. This small difference suggests that the model

is well-calibrated on average.

Most points on the QQ plot cluster around the diagonal, reinforcing the idea of good calibration. Although there are minor deviations in some percentiles, the overall calibration errors remain low: the absolute error was 0.09 and the cumulative relative error was 1.91. These are reasonable values, especially considering how complex it is to predict cancellations.

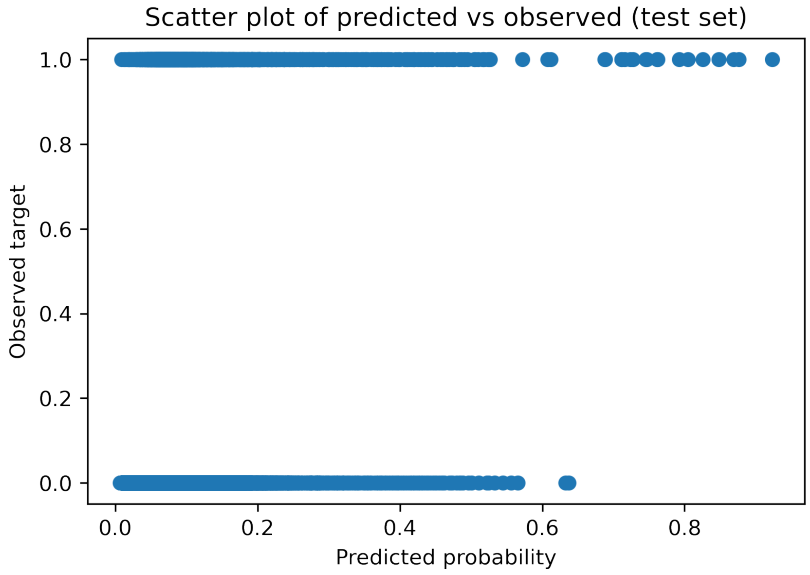

FIGURE 6: Scatter plot: predicted probabilities vs. actual lapse outcomes

Additionally, Figure 6 presents a scatter plot was created to observe how predictions are distributed in relation to actual outcomes. It shows that most non-cancelled policies (real value 0) have low predicted probabilities, while cancelled ones (real value 1) tend to have slightly higher values. However, only a few cases exceed a 50% predicted probability. This suggests that the model is somewhat conservative, that is, it rarely assigns very high probabilities of cancellation. From a business perspective, this can be beneficial, as it helps avoid unnecessary interventions with low-risk customers (Molnar, 2022b).

Overall, the analysis shows that the model not only has decent discriminatory power, but also performs well in terms of calibration. This means it not only ranks customers correctly by risk level, but also provides probability estimates that align closely with actual behavior. As a result, the model becomes a reliable tool for supporting risk-based business decisions.

5.3.1 Numerical Calibration Assessment: Brier Score

In addition to visual tools, its helpful to have a single number that summarizes how well the models predicted probabilities match reality. One of the most widely used metrics for this is the Brier Score. This score, common in probabilistic classification problems, measures how far off the predicted probabilities are from the actual outcomes. Simply put, it evaluates whether the probabilities assigned by the model reflect the real frequency of the event (Brier, 1950).

The formula for the Brier Score is:

$$\text{BS} = \frac{1}{N}\sum_{t=1}^{N}(f_t - o_t)^2$$

where:

- f_t is the predicted probability for observation t,
- o_t is the actual outcome (1 for a cancellation, 0 otherwise),
- N is the total number of observations.

The score ranges from 0 to 1. A value close to 0 indicates excellent calibration, meaning the predicted probabilities closely match the observed outcomes, while higher values reflect greater discrepancies.

In this case, the LightGBM model achieved a Brier Score of 0.0741 on the test set. According to widely accepted standards, this result reflects good calibration. For example, Machine Learning Plus (2022) suggests that scores below 0.08 indicate a low average difference between the predicted and actual outcomes, which supports the reliability of the model.

From a practical point of view, this result is especially meaningful for an insurance company like Allianz Portugal. A model that not only identifies which customers are more likely to cancel, but also estimates how likely they are to do so, allows for more refined retention strategies. Instead of relying on fixed thresholds (such as only acting when the predicted risk exceeds 50%), the company can focus its efforts where the economic value of preventing a cancellation is higher, considering both the probability and the policys value.

In summary, the Brier Score confirms that this model isn't just effective at distinguishing between high- and low-risk customers, it also provides well-calibrated probabilities. This makes it a robust tool for supporting business decisions based on actual risk, rather than generic rules.

6 ADVANCED OPTIMIZATION OF THE LIGHTGBM MODEL

6.1 Introduction to Advanced Modeling Techniques with LightGBM

Once a solid base model has been established with good initial performance, the natural next step is to go further and apply techniques that can help enhance its accuracy, stability, and ability to generalize to new data. This section focuses on more advanced strategies for optimizing and training the LightGBM model, including the use of monotonic constraints, automated hyperparameter tuning with Bayesian algorithms, and cross-validation to ensure more robust results.

What makes these techniques especially valuable is that they not only refine the model from a technical standpoint. They also offer a way to bring business knowledge directly into the modeling process, allowing the algorithm to learn not just from the data, but also from the logic and experience of the industry. This leads to models that are more useful, more reliable, and more in line with the operational reality of an insurance company.

6.2 Hyperparameter Optimization

Once the model structure is in place and key constraints like monotonicity have been applied, the next crucial step is hyperparameter optimization. These hyperparameters are external settings that must be defined before training, and they play a decisive role in shaping how the model learns. Unlike internal parameters (which are automatically adjusted during training) hyperparameters are set in advance and directly influence the models behavior and final performance.

In boosting algorithms like LightGBM, hyperparameters control essential aspects of the training process and model complexity. A detailed description of the most relevant hyperparameters adjusted in this project is available in Appendix E.5, Table III.

Choosing the right values can make a big difference in terms of accuracy, calibration, and generalization to new data. Given the size and complexity of the search space, it isn't feasible to tune these values manually or randomly. Instead, this project used Optuna, a library for automated hyperparameter tuning based on Bayesian optimization, especially efficient for fine-tuning complex models (Akiba et al., 2019; Optuna.org, 2024).

Before launching the optimization process, it was essential to establish a robust evaluation method. This is where cross-validation comes into play, helping ensure that the results aren't dependent on just one particular split of the data.

6.2.1 Hyperparameter Optimization with Optuna

After identifying the key hyperparameters for the LightGBM model, the goal was to find the best possible combination. Rather than using manual tuning or traditional methods like grid search or random search, which can be inefficient and time-consuming, this project employed Optuna, an automated optimization framework based on Bayesian methods (Akiba et al., 2019).

Optunas flexible structure allows the hyperparameter search space to be defined dynamically within the code,

which is especially helpful for exploring complex models with conditional parameter relationships. A technical description of how Optuna operates, including the algorithm used and its sampling strategy, is available in Appendix E.4.

In this study, an objective function was created that trained and evaluated a LightGBM model using K-Fold cross-validation. For each hyperparameter combination, Optuna calculated a performance metric to guide the search. Instead of optimizing for AUC, which is common in binary classification problems, this project aimed to minimize the Brier Score Loss (BSL). The reasoning was that beyond distinguishing classes, the model should also produce well-calibrated probabilities that reflect the actual cancellation risk (Machine Learning Plus, 2022).

Optuna was configured to run 70 iterations, testing various hyperparameters such as `n_estimators`, `max_depth`, `learning_rate`, `lambda_l1`, `lambda_l2`, and `colsample_bytree`. The best configuration achieved a minimum Brier Score of 0.0749, slightly better than the base model. A detailed description of these hyperparameters and their roles in model performance is provided in Appendix E.4.

With this optimal configuration identified, a final LightGBM model was trained on the full training set and then evaluated on the test set. The results showed improvements in both calibration and discrimination:

- Brier Score Loss improved from 0.0741 to 0.0737
- AUC increased from 0.6947 to 0.7008

Although these gains may seem small in absolute terms, they are consistent and aligned with the business objective. In a business context, even modest improvements in prediction accuracy can lead to more effective and profitable decision-making. Optuna proved to be a powerful tool for automating the tuning process, removing the need for manual searches, and significantly contributing to the robustness of the final model.

6.2.2 Optimization History

The Optimization History Plot shows how the Brier Score evolved over the trials carried out by Optuna. Each point represents a trial using a different set of hyperparameters.

As shown in Figure 7, the higher error values appear mostly in the early stages. Over time, the algorithm hones in on better configurations and the Brier Score stabilizes near its lowest value. This trend demonstrates the efficiency of the Bayesian approach in narrowing down the search to the most promising areas (Akiba et al., 2019; Optuna.org, 2024).

This kind of visualization helps confirm that the optimization process converged and that the number of trials was sufficient to reach a high-performing configuration.

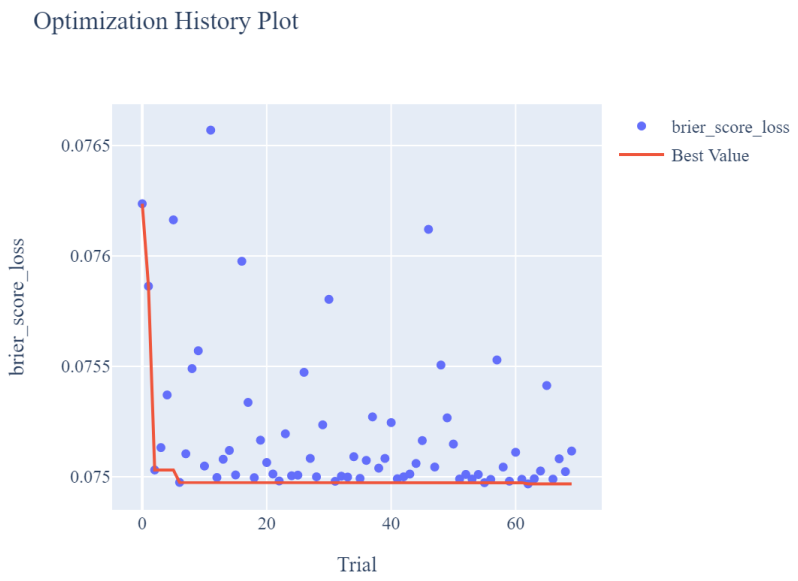

FIGURE 7: Optimization history plot: Brier score across trials

6.2.3 Hyperparameter Importance

Figure 8 shows the Hyperparameter Importance plot, another important tool for understanding model tuning. Using the default FANOVA method, Optuna estimates how much each hyperparameter contributed to differences in model performance Hutter et al. (2014).

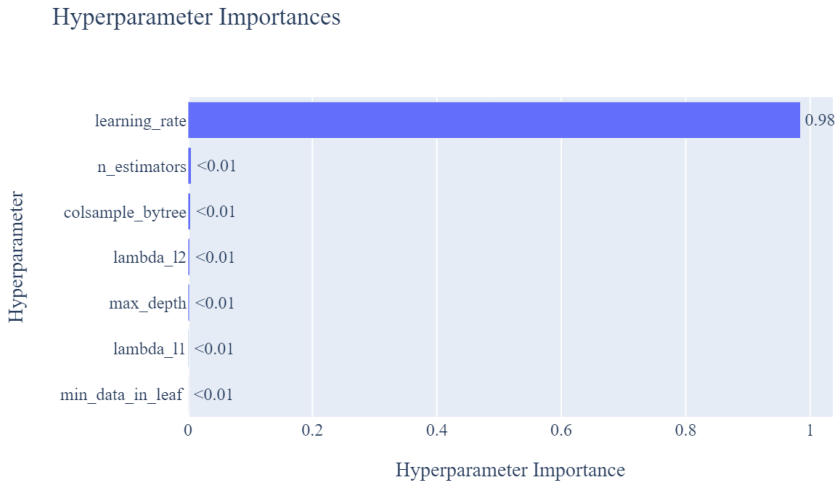

FIGURE 8: Hyperparameter importance based on optimization impact

In this case, the `learning_rate` clearly stood out as the most influential parameter accounting for almost 98% of the performance variation. Other parameters like `max_depth`, `colsample_bytree`, or `lambda_l1` had much smaller impacts. This ranking confirms what is widely known in boosting models: tuning the learning rate is crucial for achieving good results.

This insight also helps narrow down future search spaces by focusing on the parameters that matter most.

6.2.4 Slice Plot: Parameter Sensitivity

Figure 9 shows the Slice Plot, a useful visualization that illustrates how the Brier Score changes with different values of each hyperparameter, one at a time. It plots the performance of all trials against their respective hyperparameter values, making it easier to spot patterns or stable regions (Optuna.org, 2024).

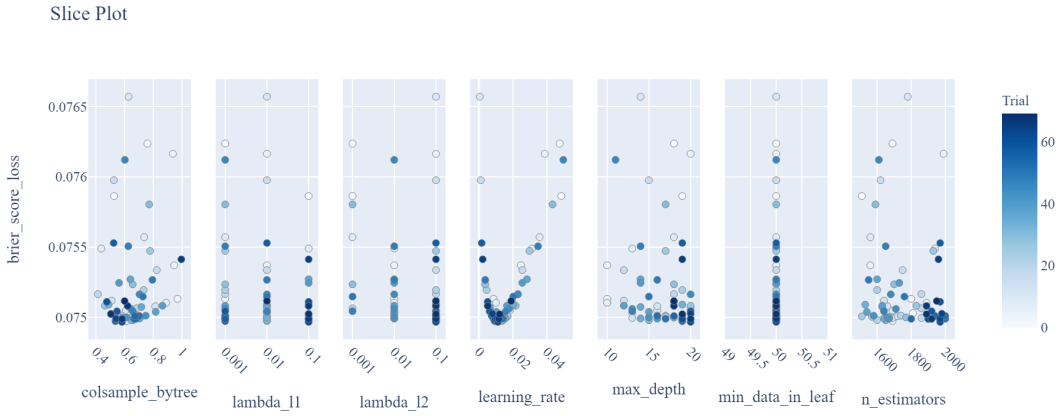

FIGURE 9: Optuna slice plot: impact of hyperparameter values on Brier score loss

For example, the plot shows that `learning_rate` values around 0.01 consistently achieved better scores. On the other hand, changes in `lambda_l2` or `min_data_in_leaf` had little effectmatching their low importance in the previous analysis.

These visualizations go beyond raw numbers, helping to interpret the optimization process more clearly and guiding better decisions when simplifying or improving the final model.

A detailed explanation of the code implementation can be found in Appendix E.6.

7 MODEL INTERPRETABILITY: SHAP ANALYSIS

7.1 Introduction to the SHAP Framework

In recent years, the use of machine learning in regulated sectors like insurance has created a growing need for models that aren't only accurate but also easy to understand. Although models like LightGBM offer excellent predictive performance, their structure as tree ensembles makes it hard to explain how decisions are made. Thats where SHAP comes in a powerful and reliable method to interpret 'black box' models.

SHAP is based on Shapley values from cooperative game theory. These values help explain how much each feature (or variable) contributes to a models prediction. It does this in a fair and consistent way, giving credit to each variable for its role in the prediction Lundberg and Lee (2017).

One of SHAPs biggest strengths is that it provides both global and local explanations. Globally, it shows which features are the most important across the whole dataset. Locally, it can explain a single prediction, showing exactly how each feature helped increase or decrease the predicted value. SHAP works well with complex models like decision trees, thanks to a tool called `TreeExplainer` (Lundberg et al., 2020).

In this thesis, we used `TreeExplainer` with LightGBM. This tool lets us not only see which variables are important, but also how they interact and behave in non-linear ways. The goal isn't to replace traditional statistical validation but to add an extra layer of insight, helping businesses understand how the model makes decisions.

In this project, the SHAP analysis helped answer key questions, such as: Which factors most affect the risk of cancellation? How does this risk change with different values? And which combinations of features increase risk together?

7.2 SHAP Analysis Results and Interpretation

The SHAP analysis gives a clear view of how each feature affects the models predictions. It breaks down predictions into smaller parts, showing both overall (global) importance and individual (local) effects. Below are the main visual results.

7.2.1 Summary Plot

Figure 10 shows how much each feature influences the predictions. Features are listed vertically from most to least important. On the horizontal axis, we see the SHAP values, that is, how much each feature moved the prediction up or down. Each point is one observation, and its color shows the features value (blue for low, red for high). The specific structure of each variable cannot be revealed to protect business confidentiality.

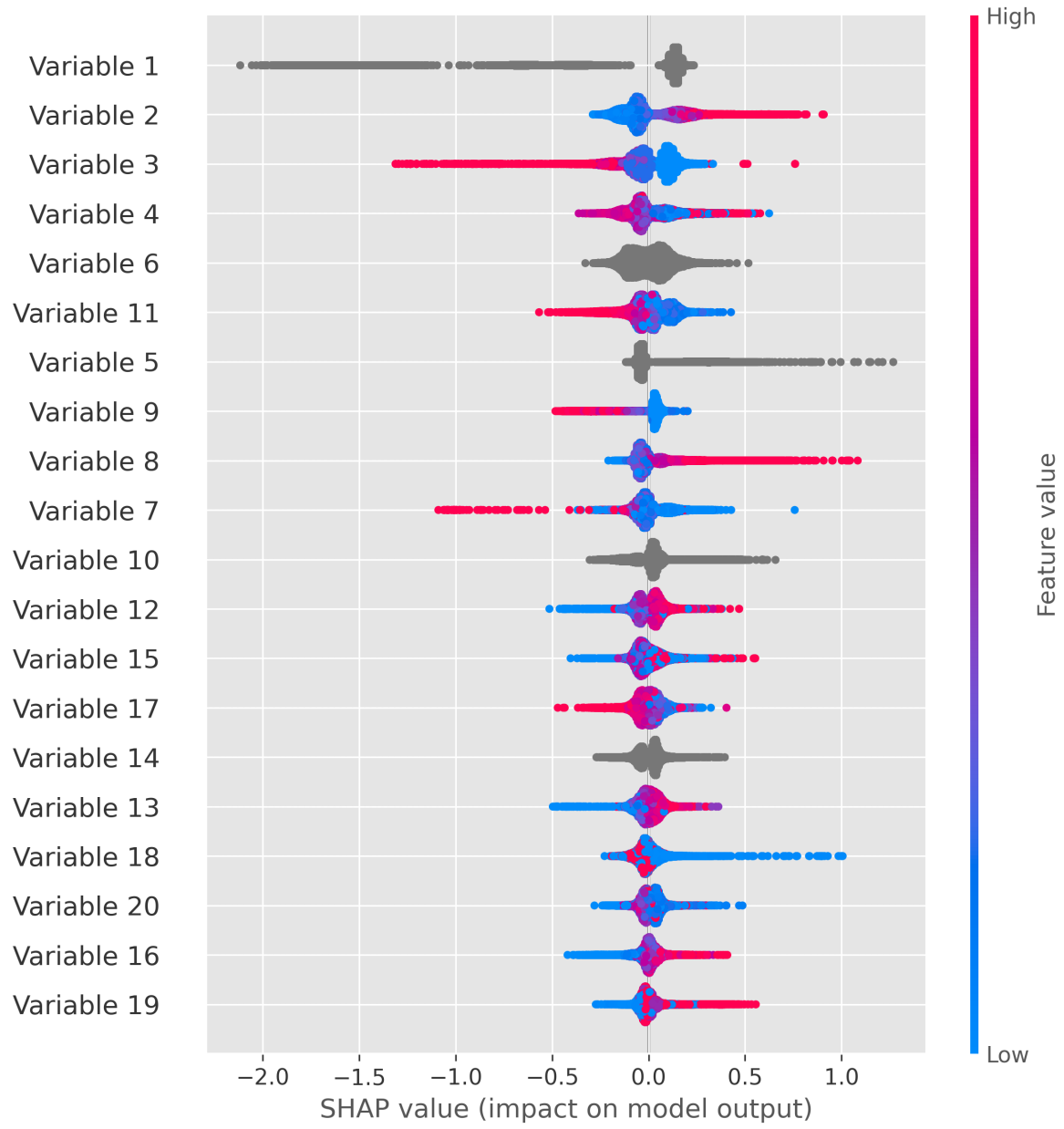

FIGURE 10: Variable importance and SHAP-based interpretability of the LightGBM model

In our plot, one variable clearly stands, it has a big influence on the cancellation risk. When the value of this variable increases, the SHAP value also increases, which means a higher risk of cancelation. Other variables have smaller effects, and some show complex or non-linear patterns (SHAP Contributors, 2024).

7.2.2 Bar Plot of Feature Importance

Figure 11 shows the average importance of each feature, regardless of whether it increases or decreases the prediction. It gives a clear ranking of the most important features.

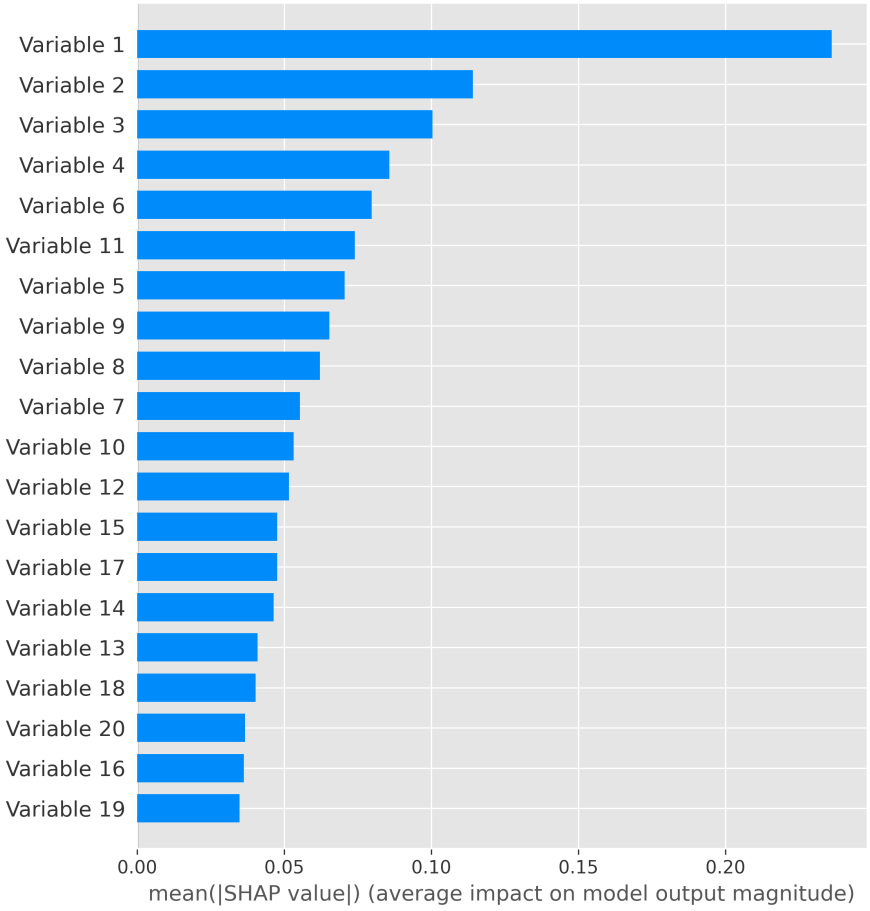

FIGURE 11: Mean SHAP values: feature importance in the LightGBM model

Again, the top variable is clearly more important than the others, confirming what we saw in the summary plot. This tells us that the model relies mainly on one or two key variables, which is useful because it means that fewer features explain a large part of the result (Lundberg et al., 2020).

7.2.3 Dependence Plot

Figure 12 shows how the SHAP value for one feature changes with its original value. In our example, we see a clear positive relationship: as the feature value increases, so does the predicted risk of cancellation.

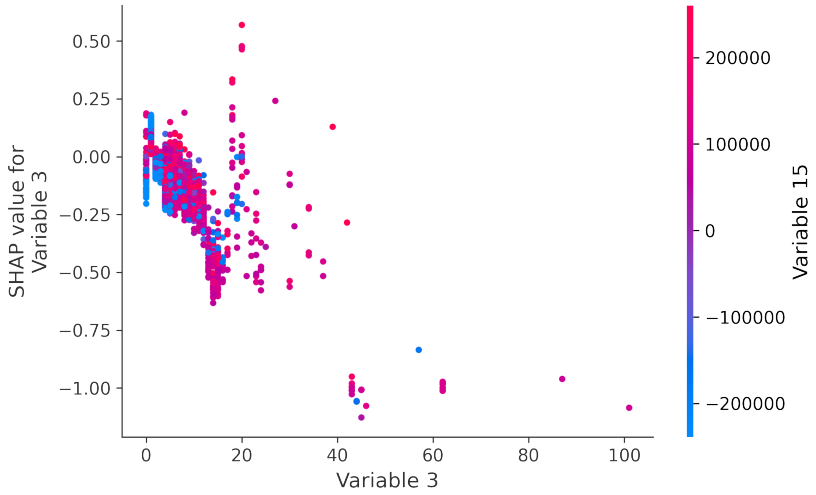

FIGURE 12: SHAP value distribution for feature

The color shows a second feature, suggesting a possible interaction between the two. This plot helps us spot non-linear relationships or dependencies that we might not notice with standard analysis (Molnar, 2022a).

7.2.4 Force Plot

Figure 13 shows the force plot, which gives a visual explanation of a single prediction. It shows which features pushed the prediction up (increased risk) or down (decreased risk).

FIGURE 13: SHAP force plot: feature contribution to individual lapse prediction

In our case, for one client in the test set, we can see how a few key features pushed the prediction toward a higher cancellation risk, while others had a protective effect. The models average prediction is adjusted based on the values of these features. This is very useful in business settings, as it helps explain why a certain risk score was given, even to people without technical knowledge (Lundberg & Lee, 2017).

8 COMPARISON WITH TRADITIONAL MODELS

8.1 Introduction to the Comparison

Despite the rise of machine learning models, Generalized Linear Models remain widely used in the insurance industry due to their interpretability, statistical robustness, and ease of implementation in regulated environments De Jong and Heller (2008) and Ohlsson and Johansson (2010). To assess the potential advantages of the LightGBM-based approach, an empirical comparison is conducted with a GLM model developed using Emblem software, utilizing the same policy cancellation dataset.

This comparison focuses on two fundamental metrics evaluates probabilistic prediction models: the AUC, which measures the models discriminative ability, and the Brier Score, which assesses the calibration of the estimated probabilities (Brier, 1950; Fawcett, 2006).

8.2 Current GLM Model in Production

The cancellation model developed in Emblem represents the solution currently used in production by the pricing department. It is a Generalized Linear Model, a technique traditionally employed in the insurance sector for its balance between predictive capability, interpretability, and regulatory compliance (De Jong & Heller, 2008).

This model was internally calibrated in Emblem (a specialized actuarial software widely used in insurance pricing to build GLMs), using a selection of variables considered standard and appropriate from a technical-actuarial perspective. Although the variable names have been anonymized in this thesis to protect business confidentiality, it can be noted that they include characteristics such as premium factors, the insureds historical behavior, policy tenure, and product features.

Unlike the LightGBM model proposed in this research, the GLM model was not built on the same variable base. The selection in Emblem was guided by criteria of parsimony, interpretability, and internal validation, and did not include complex transformations or automatic selection techniques like Boruta Kursa and Rudnicki (2010). In fact, many variables used in the machine learning model aren't available in the current Emblem environment, limiting its ability to capture nonlinear patterns or interactions between variables.

Mathematically, the model is based on the logistic function, expressed as:

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}$$

where:

- p_i is the estimated probability of cancellation for observation i.
- β_0 is the model intercept.
- β_j are the estimated coefficients for each explanatory variable x_j.

- x_{ij} is the value of variable j for observation i.

From this transformation, the estimated cancellation probability is obtained using the inverse logistic function:

$$p_i = \frac{1}{1 + \exp\left(-(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij})\right)}$$

The predicted probabilities were then evaluated using the same test set defined at the beginning of the modeling process. This allowed for an objective assessment of model performance using data that had not been seen during training. The results were:

- AUC: 66.20%
- Brier Score: 0.0757

These results reflect acceptable performance in production, although, as will be seen in the next section, the LightGBM model achieves significant improvements in both discriminative ability and calibration.

8.3 Graphical Comparison between Models: GLM vs. Machine Learning

To visually and thoroughly compare the performance of the traditional GLM model with the machine learning model developed using LightGBM, various graphical representations have been created. These allow examination of both models behavior from multiple perspectives: calibration by categories, alignment by probability quantiles, and relative differences between predictions. All of this analysis has been conducted using anonymized data, without revealing specific variable names, to protect business confidentiality.

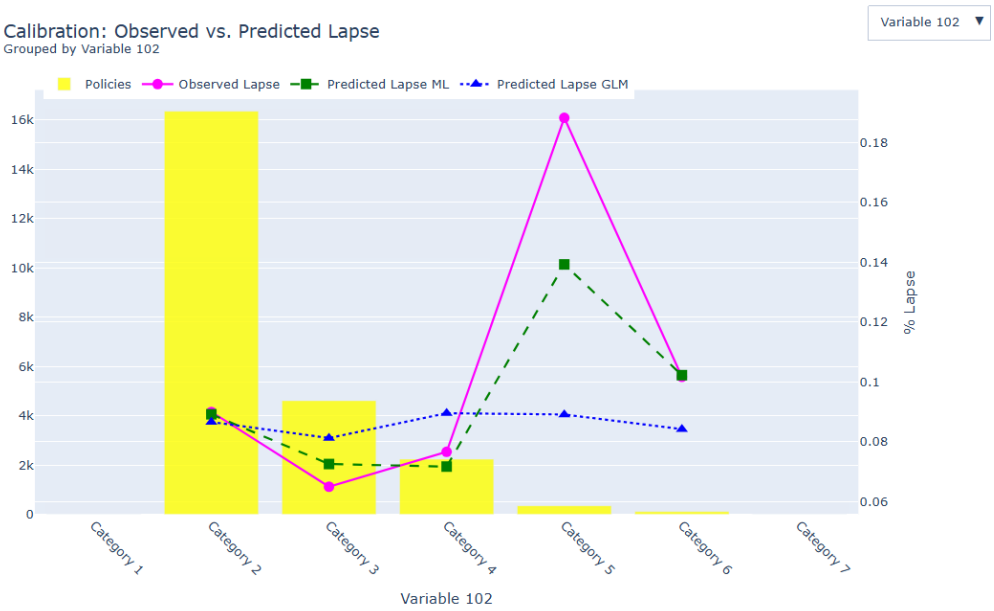

FIGURE 14: Calibration plot: observed vs. predicted lapse by lapse_est bins

8.3.1 Calibration of Observed vs. Predicted Lapse by Segmentation Variable

Figure 14 shows the first visualization, which is an interactive tool built with Plotly, a Python library for building interactive charts. This function allows analysis, for each categorical variable available in the dataset, of the calibration of both models concerning the observed lapse.

In this graph, the left axis represents the volume of policies per category (yellow bars), while the right axis shows the observed cancellation rate (pink line), along with the rates predicted by the machine learning model (green dashed line) and the GLM (blue dotted line).

By exploring the different categories, one can visually detect which model better fits the actual behavior of the insured. Generally, it is observed that the LightGBM model achieves greater fidelity in its estimates, aligning more closely with the observed rates, while the GLM presents a more smoothed trend, leading to underestimation or overestimation of risk in specific segments. This difference reflects the superior capacity of ML to capture complex and non-linear interactions.

8.3.2 Calibration by Probability Quantiles Estimated by the GLM Model

Figure 15 shows the second visualization, where the test set observations are grouped based on the quantiles generated by the probabilities predicted by the GLM model. The goal is to assess whether the model adequately discriminates between increasing levels of risk.

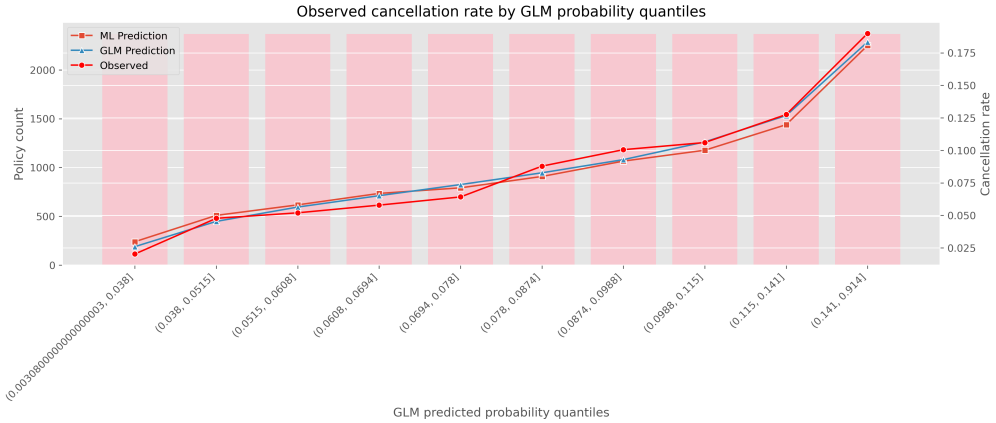

FIGURE 15: Observed lapse rate by GLM-predicted probability quantiles

The pink bars represent the number of policies in each decile, while the lines indicate the average observed cancellation rates and the predictions of both models. The GLM curve shows a progressive increase, indicating that the model correctly distinguishes between higher and lower risk groups. However, the line corresponding to the ML model aligns more closely with the observed data, especially at the distributions extremes, where precise prediction is more challenging.

8.3.3 Calibration by Probability Quantiles Estimated by the ML Model

Figure 16 repeats the logic of the previous graph, but groups the observations based on the quantiles derived from the probabilities predicted by the LightGBM model. This allows evaluation of the ML models calibration in its own prediction space.

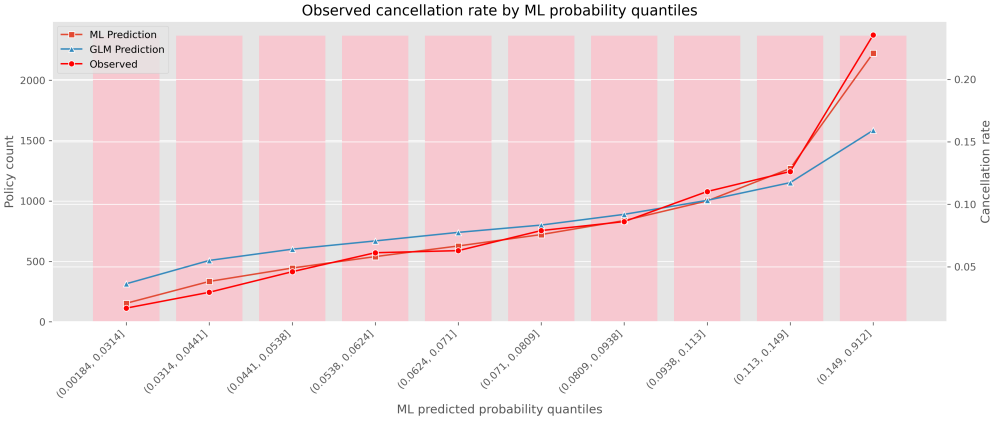

FIGURE 16: Observed lapse rate by ML-predicted probability quantiles

The results show a remarkably precise correspondence between the rates predicted by ML and the actual cancellation rates. The GLMs prediction line also remains ascending, although with a lower slope, especially in the higher quantiles. This difference suggests that the GLM tends to underestimate risk in high-probability

situations, while the ML model maintains greater consistency with the observed reality.

The graph reinforces the robustness of the ML model in stratifying risk and adequately segmenting policyholders, a key aspect in retention policies and differentiated pricing.

8.3.4 Analysis by Probability Difference (ML - GLM)

Figure 17 shows a direct comparison between models. The difference between the cancellation probabilities estimated by the ML model and the GLM is calculated for each observation, and the results are grouped into quantiles.

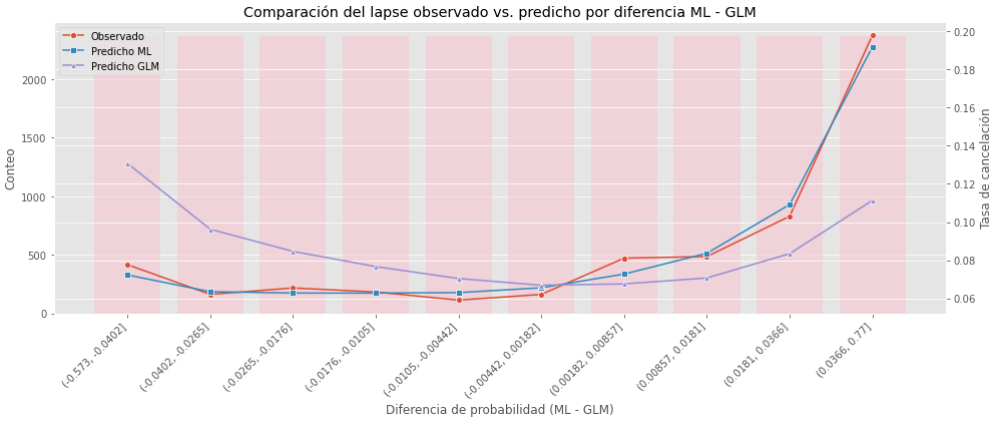

FIGURE 17: Comparison of observed lapse vs. predicted lapse by probability difference between ML and GLM models

The horizontal axis represents the levels of difference between models: negative values imply that the GLM predicts more risk than the ML; positive values indicate that the ML anticipates higher risk. The pink bars show the volume of policies per bin, while the lines reflect the observed cancellation rates and the predictions of each model.

The results reveal interesting patterns. In groups where the GLM predicts more risk than the ML, both models tend to overestimate the actual cancellation rate. In the intermediate ranges, both models align with the observed data. However, when the ML predicts more risk than the GLM (that is, when the difference is positive), the ML model aligns more precisely with the real data.

9 CONCLUSION

This thesis focused on predicting lapse behavior in home insurance, especially after customers receive a new renewal price. The goal was to identify patterns and risk factors that lead to cancellations, allowing insurers to take action and improve customer retention.

The main result is that the ML model performed better than the traditional GLM. This conclusion is based on two key evaluation metrics: the AUC and the BSL. While both metrics showed strong performance for the machine learning model, the Optuna optimization process focused on improving the BSL. This gave the model better calibration and prediction quality, which is essential in a business context where accurate risk estimates are needed.

The explainability of the model, supported by SHAP values, also helped identify the most relevant variables. These results offer valuable business insights and confirm that the model can be useful in real-life decision-making.

Although existing variables already produced solid results, there were further opportunities for feature engineering. For example, we could create interaction variables that combine geographic region with policy characteristics, or built indicators that measure price increases relative to previous years. Create customer segmentation flags, could also improve the models ability to detect risk patterns. All these could have been explored using the current framework without changing the main structure.

From a modeling perspective, we could test other algorithms, such as CatBoost or neural networks, or introduced time series validation techniques to handle seasonality. These changes would require more setup but could be implemented using the same tools and dataset already available.

Overall, this work shows that machine learning not only improves accuracy compared to GLMs but also allows for deeper understanding and flexibility. With the right tools and data, insurers can better predict which customers are at risk of leaving and take steps to keep them. Future improvements could include using time-aware models, enriching the dataset, or applying alternative optimization objectives. Still, even in its current form, the model provides strong, useful results.

APPENDIX

A CORRELATION-BASED FEATURE FILTERING

The following Python code was used to identify and remove highly correlated features based on a defined threshold (0.8). This step helps reduce redundancy and multicollinearity in the dataset, improving model generalizability. Business-relevant variables were prioritized for retention when deciding which variable to keep.

```
from tqdm.notebook import tqdm
import numpy as np

# List to store features to be removed
features_highCorr_toBeRemoved = []

# Correlation threshold
threshold = 0.8

# Business-relevant prefixes
priority_prefixes = ['Aumento_', 'pr_', 'cap_', 'AP_', 'Cover_', 'Cap_']

# List to store high correlation pairs
high_corr_pairs = []

# Loop through correlation matrix
for i in tqdm(range(len(correlation_matrix.columns))):
    for j in range(i + 1, len(correlation_matrix.columns)):
        col1 = correlation_matrix.columns[i]
        col2 = correlation_matrix.columns[j]
        correlation = correlation_matrix.iloc[i, j]

        if abs(correlation) >= threshold:
            high_corr_pairs.append((col1, col2, correlation))

# Step 1: Print all high correlation pairs
print("\n>> Pairs with High Correlation (> 0.8):")
for pair in high_corr_pairs:
    print(f"({pair[0]}, {pair[1]}) -> Correlation: {pair[2]:.2f}")

# Step 2: Elimination logic
features_to_keep = set(correlation_matrix.columns)

for col1, col2, correlation in high_corr_pairs:
    if col1 in features_highCorr_toBeRemoved or col2 in features_highCorr_toBeRemoved:
        continue

    corr_still_in_matrix = correlation_matrix.loc[
        [col for col in correlation_matrix.columns if col not in features_highCorr_toBeRemoved],
        [col for col in correlation_matrix.columns if col not in features_highCorr_toBeRemoved]
    ]

    count_col1 = len(corr_still_in_matrix[corr_still_in_matrix[col1] >= threshold])
    count_col2 = len(corr_still_in_matrix[corr_still_in_matrix[col2] >= threshold])

    if count_col1 != count_col2:
        feat_to_keep = col1 if count_col1 < count_col2 else col2
    else:
        feat_to_keep = col1
        for prefix in priority_prefixes:
```

```
            if col1.startswith(prefix):
                feat_to_keep = col1
                break
            if col2.startswith(prefix):
                feat_to_keep = col2
                break

    feat_to_be_removed = col2 if feat_to_keep == col1 else col1
    assert feat_to_be_removed is not None, "[ERROR] feat_to_be_removed is None"

    features_highCorr_toBeRemoved.append(feat_to_be_removed)
    features_to_keep.discard(feat_to_be_removed)

    print(f"\n Pair ({col1}, {col2}) has a correlation of {correlation:.2f}")
    print(f"    Keeping: '{feat_to_keep}'")
    print(f"    Removing: '{feat_to_be_removed}'")

# Final summaries
print("\n Final List of Removed Features:")
print(features_highCorr_toBeRemoved)
print(f"Total removed features: {len(features_highCorr_toBeRemoved)}")

print("\n Final List of Kept Features:")
features_to_keep = list(features_to_keep)
print(features_to_keep)
print(f"Total kept features: {len(features_to_keep)}")
```

B FEATURE SELECTION WITH BORUTA AND PCA

The following Python code describes the process used to apply feature selection using Boruta, after enriching the training dataset with principal components. The goal was to capture additional variance and assess feature relevance with a Random Forest-based wrapper method.

```
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from boruta import BorutaPy
import pandas as pd

# If these variables are selected, they get a monotone constraint:
boruta_monotone_constraints_logic = ['Aumento_enviado4', 'Aumento_enviado_abs2', 'pr_enviado3', '
    ncl_Increase']
boruta_in_X_train = [boruta_monotone_constraints_logic[i] for i in range(0,4) if
    boruta_monotone_constraints_logic[i] in X_train]

current_numerical_features = [col for col, dt in zip(X_train.columns, X_train.dtypes) if dt in [float,
    int]]

X_train_no_monotonic = X_train[current_numerical_features].drop(boruta_in_X_train, axis=1)
X_test_no_monotonic  = X_test[current_numerical_features].drop(boruta_in_X_train, axis=1)

# Define number of principal components
n_components = X_train_no_monotonic.shape[1]

# PCA transformation
pca = PCA(n_components=n_components, random_state=42)
X_train_pca = pca.fit_transform(X_train_no_monotonic)
X_test_pca = pca.transform(X_test_no_monotonic)

```

```
# Name components
pca_columns = [f'PCA_{col}' for col in X_train_no_monotonic.columns]
X_train_pca_df = pd.DataFrame(X_train_pca, columns=pca_columns, index=X_train.index)
X_test_pca_df = pd.DataFrame(X_test_pca, columns=pca_columns, index=X_test.index)

# Combine original and PCA features
X_train_combined = pd.concat([X_train, X_train_pca_df], axis=1)
X_test_combined  = pd.concat([X_test,  X_test_pca_df], axis=1)

# Boruta feature selection
rf = RandomForestClassifier(n_jobs=-1, max_depth=8, random_state=42)
boruta_selector = BorutaPy(rf, n_estimators='auto', verbose=2, random_state=42)

numeric_columns = [col for col, dt in zip(X_train_combined.columns, X_train_combined.dtypes) if dt in [
    float, int]]
boruta_selector.fit(X_train_combined[numeric_columns].values, y_train)

# Extract selected features
selected_features = [col for col, support in zip(numeric_columns, boruta_selector.support_) if support]
print("Selected Features:", selected_features)
```

C TECHNICAL BACKGROUND ON LIGHTGBM

This appendix expands on the theoretical background of the LightGBM algorithm, its training configuration, and internal workings, which support the predictive model presented in the main thesis.

LightGBM is a supervised machine learning algorithm, which means it learns from examples where the outcome is already known. In this case, it learns whether a policy was canceled or not, and then uses that knowledge to predict what might happen in similar future situations. In practical terms, LightGBM estimates the likelihood that a customer will cancel their policy after receiving a new renewal price Microsoft (2024).

What sets it apart from traditional models like logistic regression is its ability to handle more complex relationships between variables. It can uncover subtle patterns and interactions without the need to define the relationships explicitly. This capability makes it especially suitable for large and diverse datasets, typical in the insurance industry Ke et al. (2017).

The concept of *boosting* helps explain how LightGBM works. It starts with a simple model that makes initial predictions. A second model is then trained to correct the errors of the first, followed by a third model that improves upon the second, and so on. Together, the series of small models create a stronger, more accurate prediction system Ke et al. (2017).

Each model in this sequence is a decision tree, structured as a series of questions. For example:

Is the customer holding more than three policies? $\rightarrow$ Yes / No $\rightarrow$ next question $\rightarrow \ldots \rightarrow$ outcome.

Figure 18 shows a visualization of a trained decision tree structure used in this process. LightGBM improves on traditional tree-building methods by introducing efficient techniques that enable faster training on large datasets without compromising accuracy. These methods are detailed in both the original LightGBM paper and Microsofts documentation Ke et al. (2017) and Microsoft (2024).

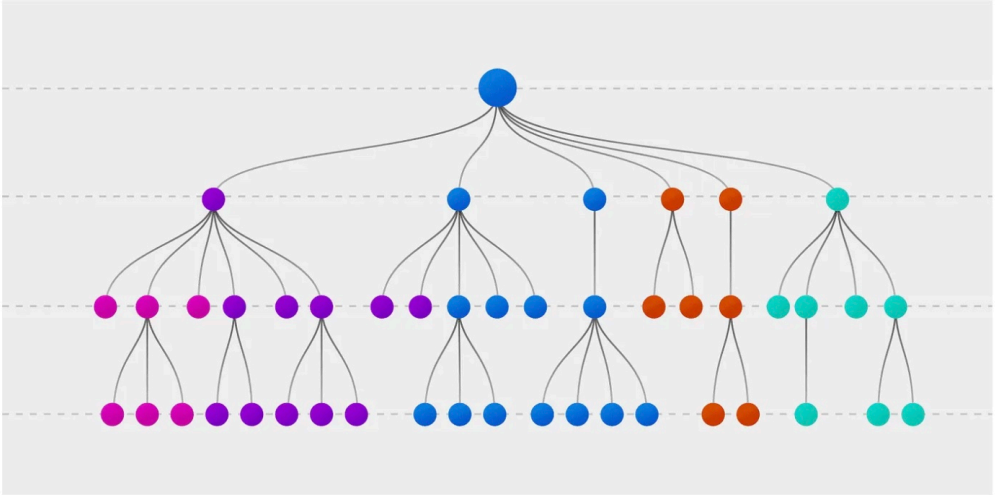

FIGURE 18: Visualization of a trained decision tree structure. Source: Medium

D CALIBRATION ANALYSIS: QQ-PLOT

The following Python function was developed to generate QQ-plots for evaluating the calibration of predicted probabilities against observed lapse rates. It segments data into quantiles and plots both histograms and observed-vs-predicted calibration curves, providing insight into model reliability at different risk levels.

D.1 Python Implementation of `qq_plot()` Function

```python
from datetime import datetime

def qq_plot(partition, model, data, target_vec, qq_n_quantile, prob_cal, manual_bol):
    if manual_bol == False:
        tik = datetime.now()
        y_prob = model.predict_proba(data)[:,1]
        tak = datetime.now()
        print(">> Time to compute predictions:", tak - tik)
    else:
        y_prob = prob_cal

    df_temp = pd.DataFrame({'y_prob': y_prob, 'y_obs': target_vec.values})
    df_temp['qcut'] = pd.qcut(df_temp['y_prob'], qq_n_quantile, duplicates='drop', precision=4)
    df_temp['qcut_r'] = df_temp['qcut'].apply(lambda x: str(x.right))

    group = df_temp['qcut_r'].value_counts().index.sort_values()
    dicty = {}
    tag = 0
    for right in group:
        tag += 1
        key = str(tag).zfill(2) if len(group) < 100 else str(tag).zfill(3)
        dicty[float(right)] = key
    df_temp['qcut'] = df_temp['qcut'].apply(lambda x: dicty[x.right] + "_<=" + str(x.right))

    df_temp2 = df_temp.groupby('qcut').agg({'y_obs': 'mean', 'y_prob': 'mean', 'qcut': 'count'})

    title_tag = "Lapse Household"
```

```
    color_prob = '#00802b'

    fig = plt.figure(figsize=(18,6), dpi=200)
    ax1 = fig.add_subplot(121)
    ax1.hist(y_prob, bins=13, label='probabilities', color=color_prob, alpha=0.65)

    for q, color in zip([0.25, 0.5, 0.75, 0.9, 0.99],
                        ['#ffb3cc', '#ff80aa', '#ff4d88', '#ff1a66', '#b3003b']):
        plt.axvline(x=np.quantile(y_prob, q), color=color, linestyle='--',
                    label=f'{int(q*100)}q: {round(np.quantile(y_prob, q),3)}')

    ax1.set_xticks(np.arange(0,1,0.1))
    ax1.grid()
    ax1.set_xlabel('Probabilidades')
    ax1.set_ylabel('Frequência')
    ax1.set_title(f'[{partition}] Distribuição de probabilidades: conv. {title_tag}')
    ax1.legend()

    ax2 = fig.add_subplot(122)
    xx = yy = np.arange(0,1,0.01)
    ax2.plot(xx, yy, linestyle='--', color='gray')
    ax2.plot(df_temp2.y_prob.tolist(), df_temp2.y_obs.tolist(), marker='x', color=color_prob, linewidth
        =0, markersize=4)
    ax2.set_xticks(np.round(np.arange(0,1,0.05), 2))
    ax2.set_xticklabels(np.round(np.arange(0,1,0.05), 2), rotation=48)
    ax2.grid()
    ax2.set_xlabel('Predicted (proba.)')
    ax2.set_ylabel('Observed')
    ax2.set_title(f'[{partition}] QQ-plot: conversão {title_tag}')

    y_prob_plot = df_temp2.y_prob.tolist()
    y_obs_plot = df_temp2.y_obs.tolist()
    rel_dev, abs_dev, abs_dev_record = 0, 0, []
    for i in range(len(y_prob_plot)):
        abs_err = abs(y_prob_plot[i] - y_obs_plot[i])
        abs_dev += abs_err
        abs_dev_record.append(abs_err)
        rel_dev += abs_err / y_obs_plot[i] if y_obs_plot[i] != 0 else abs_err

    max_abs_dev = np.max(abs_dev_record)
    y_obs_mean_str = f"{round(df_temp.y_obs.mean()*100,2)}%"
    y_prob_mean_str = f"{round(df_temp.y_prob.mean()*100,2)}%"
    ax2.annotate(f'[{partition}] Média obs / pred: ({y_obs_mean_str},{y_prob_mean_str})', (0.48,0.10))
    ax2.annotate(f'Relative deviation sum (q={qq_n_quantile}): {round(rel_dev,3)}', (0.48,0.05))
    ax2.annotate(f'Absolute deviation sum (q={qq_n_quantile}): {round(abs_dev,4)}', (0.48,0.01))

    plt.show()
    return y_obs_mean_str, y_prob_mean_str, round(rel_dev,2), round(abs_dev,2), round(max_abs_dev,3)

# Example call
qq_plot(
    partition = "Test",
    model = model,
    data = X_test_final,
    target_vec = y_test,
    qq_n_quantile = 15,
    prob_cal = None,
    manual_bol = False
)
```

E HYPERPARAMETER OPTIMIZATION CODE

For this project, an initial set of hyperparameters was defined and stored in a dictionary called `lgbm_support_params`. This configuration provided a well-balanced starting point, aligning accuracy, computational efficiency, and domain knowledge. These hyperparameters were later refined through automated tuning.

E.1 Initial Hyperparameter Setup

Hyperparameter	**Description**
`objective`	Specifies a binary classification task (cancellation = 1, no cancellation = 0), determining the appropriate loss function.
`boosting`	Uses the Gradient Boosting Decision Trees algorithm to iteratively correct model errors.
`n_jobs`	Utilizes all processor cores available, speeding up training.
`random_state`	Ensures reproducibility of results by setting a fixed random seed.
`verbosity`	Minimizes training log output for cleaner reports.
`bagging_freq`	Performs bagging every 5 iterations to reduce overfitting and improve generalization.
`monotone_constraints`	Applies business logic by enforcing monotonic relationships on selected features.
`monotone_method`	Enables an advanced mode for handling monotonic constraints effectively.

TABLE II: Initial configuration of LightGBM hyperparameters

These settings were defined in Python as follows:

```
lgbm_support_params = {
    "objective": "binary",
    "boosting": "gbdt",
    "n_jobs": -1,
    "random_state": 42,
    "verbosity": -1,
    "bagging_freq": 5,
    "monotone_constraints": monotone_constraints_vector,
    "monotone_method": "advanced"
}
```

E.2 Cross-Validation and Optimization Strategy

To evaluate the performance of each hyperparameter configuration in a consistent and robust way, this project used the *K-Fold Cross Validation* technique during the Optuna optimization process.

In this method, the training dataset is split into K equally sized parts. The model is then trained K times, each time leaving out one fold for validation and using the remaining $K-1$ folds for training. This ensures that every data point is used both for training and for validation, resulting in a more accurate and stable estimate of model performance. Figure 19 shows a visual representation of this process.

For this study, $K = 4$ was selected, offering a balance between statistical robustness and computational efficiency. The mean performance across the four folds was used as the objective metric (Brier Score Loss) to be minimized by Optuna.

This validation process was embedded directly into the objective function of the optimization, so that every hyperparameter set was evaluated under the same cross-validation conditions. This approach helps reduce the risk of overfitting to a single train-test split and is recognized as a best practice in supervised machine learning Bergstra et al. (2011) and Optuna.org (2024).

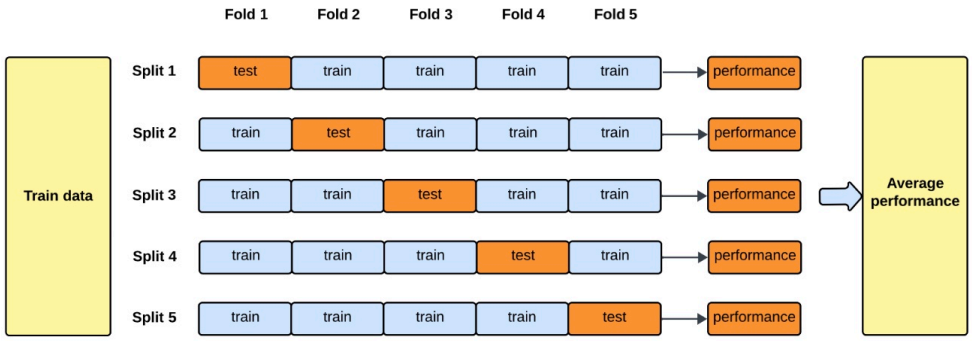

FIGURE 19: K-fold cross-validation: evaluation process based on repeated train/test splits. Source: Towards Data Science

E.3 Hyperparameter Tuning with Optuna

The following Python code was used to perform hyperparameter tuning of the LightGBM model using Bayesian optimization with Optuna. The optimization objective was evaluated using 4-fold cross-validation and the Brier Score Loss.

```
from sklearn.model_selection import KFold
from sklearn.metrics import roc_auc_score, brier_score_loss
import lightgbm as lgbm
import optuna
import numpy as np

# Define the objective function to minimize
def objective(trial, X_train, y_train, n_splits, lgbm_support_params, score_metric='brier_score_loss'):
    allowed_metrics = ['auc', 'brier_score_loss']
    assert score_metric in allowed_metrics, f"[ERROR] {score_metric} not identified (must be {
        allowed_metrics})"

    n_estimators = trial.suggest_int('n_estimators', 1500, 2000, step=10)
    learning_rate = trial.suggest_float('learning_rate', 1e-4, 0.05)
    max_depth = trial.suggest_int('max_depth', 10, 20, step=1)
```

```
    min_data_in_leaf = trial.suggest_int('min_data_in_leaf', 50, 60, step=25)
    colsample_bytree = trial.suggest_float('colsample_bytree', 0.4, 1)
    lambda_l1 = trial.suggest_categorical("lambda_l1", [0.001, 0.01, 0.1])
    lambda_l2 = trial.suggest_categorical("lambda_l2", [0.001, 0.01, 0.1])

    model = lgbm.LGBMClassifier(
        **lgbm_support_params,
        n_estimators=n_estimators,
        learning_rate=learning_rate,
        max_depth=max_depth,
        min_data_in_leaf=min_data_in_leaf,
        colsample_bytree=colsample_bytree,
        lambda_l1=lambda_l1,
        lambda_l2=lambda_l2
    )

    kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)
    scores = []

    for fold, (train_index, valid_index) in enumerate(kf.split(X_train)):
        X_train_cv, X_valid_cv = X_train.iloc[train_index, :], X_train.iloc[valid_index, :]
        y_train_cv, y_valid_cv = y_train.iloc[train_index], y_train.iloc[valid_index]
        model.fit(X_train_cv, y_train_cv)
        y_prob_valid_cv = model.predict_proba(X_valid_cv)[:, 1]

        if score_metric == 'auc':
            metric = roc_auc_score(y_valid_cv, y_prob_valid_cv)
        else:
            metric = brier_score_loss(y_valid_cv, y_prob_valid_cv)

        scores.append(metric)

    return np.mean(scores)

sampler = optuna.samplers.TPESampler(seed=42)
study = optuna.create_study(direction='minimize', sampler=sampler)

study.optimize(
    lambda trial: objective(
        trial, X_train, y_train, n_splits=4, lgbm_support_params=lgbm_support_params,
        score_metric='brier_score_loss'),
    n_trials=70
)

print("Best value (minimum):", study.best_value)
print("Best parameters:", study.best_params)

try:
    import matplotlib.pyplot as plt
    optuna.visualization.matplotlib.plot_optimization_history(study)
    plt.show()
except ImportError:
    print("Install matplotlib and optuna[visualization]")
```

E.4 Technical Details of Optuna

Optuna is an automated hyperparameter optimization framework that uses Bayesian methods to explore the hyperparameter space efficiently Akiba et al. (2019). In this project, Optuna was selected to tune the LightGBM models parameters due to its flexibility and its capacity to adapt the search dynamically depending on the

performance observed during the trials.

One of the distinguishing features of Optuna is its *define-by-run* approach, which allows the definition of the hyperparameter search space to be built dynamically as part of the execution flow. This is especially helpful for models with conditional relationships between parameters, making the optimization process both more flexible and more powerful compared to grid or random search.

Internally, Optuna employs the Tree-structured Parzen Estimator (TPE) as its default sampling algorithm. This method is a form of sequential model-based optimization (SMBO) that models the objective function using probability distributions. At each step, TPE selects the most promising set of hyperparameters to evaluate next, based on how likely they are to improve the performance metric.

In this project, Optuna was integrated with a K-Fold cross-validation setup, where for each trial (i.e., hyperparameter configuration), the LightGBM model was trained and evaluated using four distinct data folds. Instead of the more typical AUC metric used for classification problems, the Brier Score Loss was used as the objective to minimize. This decision reflects the importance of producing well-calibrated probabilities, not just correct classifications, when predicting the likelihood of policy cancellation Machine Learning Plus (2022). Figure 20 shows the overall tuning process and how training and validation are repeated in each trial.

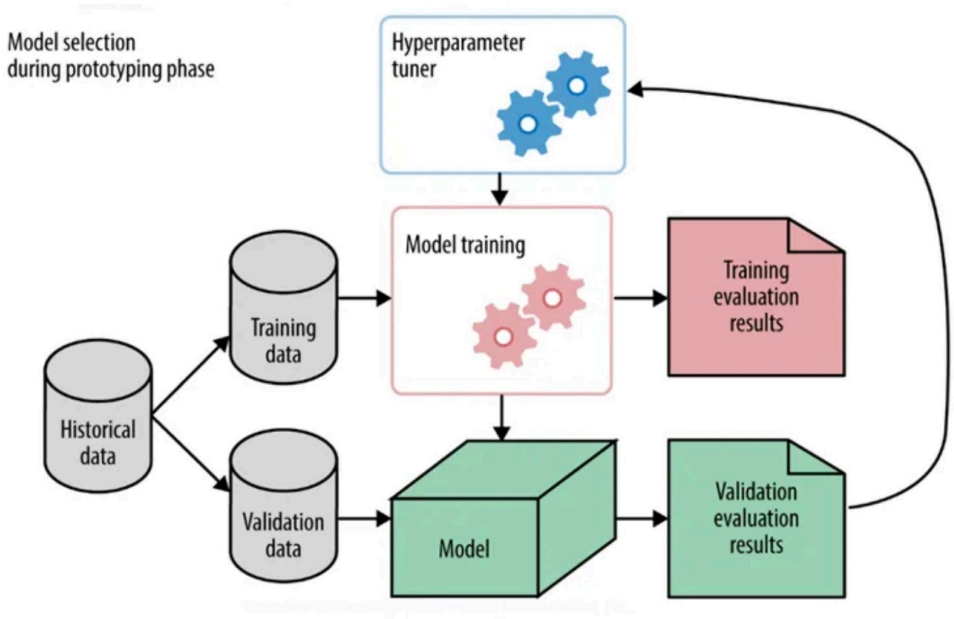

FIGURE 20: Hyperparameter tuning workflow: interaction between training, validation, and evaluation during the model selection phase. Source: AlmaBetter

The configuration, implementation, and execution of the optimization loop with Optunaincluding how the objective function and hyperparameters were handledare detailed in Appendix E.3.

E.5 Explanation of the Tuned Hyperparameters

During the optimization process with Optuna, several key hyperparameters in LightGBM were automatically adjusted. These settings, defined before training begins, significantly impact the models accuracy and generalization ability. Table III summarizes the most relevant ones along with their descriptions.

TABLE III: Key hyperparameters adjusted by Optuna and their function

Hyperparameter	**Description**
`n_estimators`	Number of boosting rounds (trees). More trees can improve accuracy but may increase overfitting if not properly balanced.
`learning_rate`	Controls how much the model adjusts in each iteration. Lower values usually lead to more robust models but require more trees.
`max_depth`	Sets the maximum depth of each tree. Helps manage model complexity and reduces overfitting risk.
`min_data_in_leaf`	Minimum number of data points required in a leaf. Prevents overly specific splits, serving as a regularization mechanism.
`colsample_bytree`	Fraction of features randomly selected per tree. Encourages variety and limits similarity across trees.
`lambda_l1` / `lambda_l2`	L1 and L2 regularization terms. Penalize complexity, useful when dealing with many or correlated features.

These hyperparameters were not arbitrarily selected. They were chosen based on their capacity to improve both classification performance and probability calibration. As discussed in the main text (see Section 5.3), the `learning_rate` parameter emerged as the most impactful, underscoring its critical role in driving convergence and enhancing model stability Optuna.org (2024).

E.6 Technical Details of Optuna

```python
import os

# Create output folder if it doesn't exist
os.makedirs("images_tesis", exist_ok=True)

# --- Optimization History ---
fig = optuna.visualization.plot_optimization_history(study=study, target_name=opt_metric)
fig.update_layout(width=550, height=450, font=dict(family="Allianz Neo"))
fig.show()
fig.write_image("images_tesis/optuna_optimization_history.png", scale=2)
fig.write_html("images_tesis/optuna_optimization_history.html")

# --- Parameter Importance ---
fig = optuna.visualization.plot_param_importances(study=study, target_name=opt_metric)
fig.update_layout(width=650, height=400, font=dict(family="Allianz Neo"))
fig.show()
fig.write_image("images_tesis/optuna_param_importance.png", scale=2)
```

```
fig.write_html("images_tesis/optuna_param_importance.html")

# --- Slice Plot for hyperparameter tuning visualization ---
fig = optuna.visualization.plot_slice(
    study=study,
    params=list(study.best_params.keys()),
    target_name=opt_metric
)
fig.update_layout(width=950, height=400, font=dict(family="Allianz Neo"))
fig.update_xaxes(tickangle=45)
fig.show()
fig.write_image("images_tesis/optuna_slice_plot.png", scale=2)
fig.write_html("images_tesis/optuna_slice_plot.html")
```

To generate the figures included in Chapter 6, a Python script was implemented using the visualization functionalities provided by the Optuna library. The script produced three different plots to support the interpretation of the hyperparameter tuning process: the optimization history, the hyperparameter importance ranking, and the slice plot for parameter sensitivity.

First, the script creates a folder to store all generated images. Then, the optimization history plot is generated, which shows how the models Brier Score evolved throughout the trials performed by Optuna. This visualization helps assess whether the optimization process converged and whether additional trials would likely bring further improvements.

Next, the hyperparameter importance plot is created. This figure provides an estimate of how much each hyperparameter contributed to the variation in model performance. Internally, Optuna uses the fANOVA (Functional ANOVA) method to evaluate the marginal impact of each parameter while averaging over the others. This allows the identification of the most influential hyperparameters and supports the prioritization of tuning efforts.

Finally, the script generates a slice plot, which displays how the performance metric (Brier Score) changes as a function of individual hyperparameter values across all completed trials. This helps visualize stable regions or thresholds where model performance tends to improve or deteriorate, supporting the interpretation of parameter sensitivity.

All plots were customized to match the visual identity of the thesis and exported in both image and interactive HTML formats to ensure reproducibility and traceability of results.

F Appendix: SHAP Interpretation Code

F.1 SHAP-Based Model Interpretation

The following code was used to analyze and visualize the contributions of individual features to the models predictions using SHAP. The SHAP package allows global and local interpretability of machine learning models, helping to explain why certain predictions are made and which variables influence them the most.

```
import shap

# Create a SHAP Explainer
explainer = shap.TreeExplainer(best_model)

```

```
# Get Shapley values for the test set
shap_values = explainer.shap_values(X_test_final)
```

F.1.1 Global Interpretability

```
# 1. Summary Plot (Overall Feature Importance)
shap.summary_plot(shap_values[1], X_test_final)

# 2. Bar Plot of Feature Importance
shap.summary_plot(shap_values[1], X_test_final, plot_type="bar")
```

F.1.2 Feature Interaction Analysis

```
# 3. Dependence Plot for a selected feature (example: "FP")
shap.dependence_plot("FP", shap_values[1], X_test_final)
```

F.1.3 Local Interpretability

```
# 4. Force Plot for a Single Prediction
shap.initjs()
shap.plots.force(
    explainer.expected_value[1],
    shap_values[1][0],
    features=X_test_final.iloc[0:1]
)
```

Note: The force plot in Jupyter renders as an interactive JavaScript object. In PDF format, it can be captured as a static image for reporting purposes.

REFERENCES

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631. https://doi.org/10.1145/3292500.3330701

Bergstra, J. S., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems 24 (NeurIPS 2011)*, 2546–2554. https://proceedings.neurips.cc/paper/2011/hash/86e8f7ab32cfd12577bc2619bc635690-Abstract.html

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, *78*(1), 1–3.

De Jong, P., & Heller, G. Z. (2008). *Generalized linear models for insurance data*. Cambridge University Press.

de Supervisão de Seguros e Fundos de Pensões (ASF), A. (2024). Consumer trends report 2024 heatmap [Recuperado el 24 de mayo de 2025]. https://www.asf.com.pt/documents/d/site-asf/consumer-trends-report-2024-heatmap

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Friedman, J., Hastie, T., & Tibshirani, R. (2001). The elements of statistical learning: Data mining, inference, and prediction (1st ed.)

Galilea, G. (2025). Revisar los riesgos en el seguro de hogar [Recuperado el 24 de mayo de 2025]. https://www.grupogalilea.com/revisar-los-riesgos-en-el-seguro-de-hogar

Géron, A. (2019). *Hands-on machine learning with scikit-learn, keras, and tensorflow* (2nd). OReilly Media.

Henckaerts, R. J., Verbelen, R., & Antonio, K. (2022). Explainable machine learning models in insurance claims prediction. *European Actuarial Journal*, *12*, 69–101. https://doi.org/10.1007/s13385-021-00274-4

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2014). An efficient approach for assessing hyperparameter importance. *International Conference on Machine Learning (ICML)*, 754–762. http://proceedings.mlr.press/v32/hutter14.html

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*. https://papers.nips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

Kursa, M. B., & Rudnicki, W. R. (2010). Feature selection with the boruta package. *Journal of Statistical Software*, *36*(11), 1–13. https://doi.org/10.18637/jss.v036.i11

Lundberg, S. M., Erion, G., Chen, H., et al. (2020). From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, *2*(1), 56–67. https://www.nature.com/articles/s42256-019-0138-9

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *30*. https://arxiv.org/abs/1705.07874

Machine Learning Plus. (2022). Brier score explained [Recuperado el 24 de mayo de 2025]. https://machinelearningplus.com

Microsoft. (2024). Lightgbm documentation. https://lightgbm.readthedocs.io/

Molnar, C. (2022a). Interpretable machine learning: A guide for making black box models explainable (2nd). https://christophm.github.io/interpretable-ml-book/

Molnar, C. (2022b). *Interpretable machine learning: A guide for making black box models explainable*. Self-published. https://christophm.github.io/interpretable-ml-book/

of Insurance, T. D. (2022). Outside factors affect home and auto insurance costs [Recuperado el 24 de mayo de 2025]. https://www.tdi.texas.gov/column/outside-factors-affect-home-and-auto-insurance-costs-sp.html

Ohlsson, E., & Johansson, B. (2010). *Non-life insurance pricing with generalized linear models*. Springer.

Optuna.org. (2024). Optuna documentation. https://optuna.org

Seguros, B. (2025). £qué influye en el precio del seguro de hogar? [Recuperado el 24 de mayo de 2025]. https://breezyseguros.com/es/que-influye-precio-seguro-hogar/

SHAP Contributors. (2024). Shap python library documentation. https://shap.readthedocs.io/en/latest/