Predictive modeling for insurance claim: Dealing with the challenge of precise prediction through an example

Alexandre Carbonell,
Mathieu Hoff,
Clémence Marcaillou

Supervisor: Prof. Peter Bühlmann

Department of Mathematics
Eidgenössische Technische Hochschule Zürich

D MATH

July 2023

Abstract

In the dynamic landscape of insurance, the ability to anticipate whether a client will make a claim is of paramount importance. It enables insurers to assess potential risks, allocate resources efficiently, and provide personalized services to their clients. It is of prior importance to ask premiums to clients that are in line with their behaviors. It is fair that a client with a good behavior and no claims will benefit in a lower premiums to pay. This is why, we delve into the intricate task of predicting if a client will make a claim, focusing specifically on automobile and home insurance.

With a dataset encompassing a huge amount of clients, we confront the stark reality that a small amount of them have claimed insurance for motor and/or home. The vast majority of data consists of non-claiming clients, highlighting the inherent imbalance that plagues the prediction problem. Such a skewed distribution creates a profound challenge in constructing a model that can accurately discern claim-prone clients from the vast pool of non-claimants. To better understand the challenges associated with this concept, we have studied the article [4].

This paper aims, through an example, to explore the complexities of predictive modeling for insurance claim incidence, shedding light on the methods and techniques employed to overcome the inherent difficulty of precise prediction.

Table of contents

Introduction

In the intricate realm of insurance, the fundamental objective of every insurance company is to strike a delicate balance between risk and reward. The ability to accurately assess the risk of a customer making a claim or remaining claim-free lies at the heart of this equilibrium. By effectively predicting claim likelihood, insurers can tailor their premium pricing to reflect the individual risk profiles of their diverse clientele. Insurance companies operate in a dynamic landscape, where they continuously encounter the ever-changing patterns of risks associated with various policyholders. These risks may arise from factors such as personal characteristics, driving habits, health conditions, location, and even lifestyle choices. Insurance companies use statistical models, actuarial sciences, and advanced data analytics to navigate this intricate landscape and uncover valuable insights from historical data. Accurate risk assessment is pivotal for the sustainable growth and profitability of insurance providers. An essential component of this process is the precise calculation of premiums for each customer. The premium amount serves not only as a financial safeguard for insurers but also as an equitable pricing mechanism that must consider each individual's potential risk exposure. For example, consider two automobile insurance applicants: Alice, a cautious driver with a clean driving record, and Bob, a young driver with a history of accidents. It would be unjust for Alice to pay the same premium as Bob since their risk levels vastly differ. By accurately assessing the risk associated with each customer, the insurance company can charge Alice a lower premium, reflecting her low likelihood of making a claim, while charging Bob a higher premium due to his increased risk.

We propose, in this paper, a project aimed at predicting insurance claim for a leading insurance company in Spain. In this endeavor, we will leverage a dataset [2] comprising 40,284 insurance private customers, each holding a combination of motor and homeowners insurance contracts. These customers have been diligently tracked from 2010 to 2014, providing valuable insights into their behavior and claim patterns over the course of five years. The dataset combines different information on the customers that are more or less important for our goal of predicting if a given client with the same type of information will make a claim or not.

To achieve our objective we will proceed as follow: In the Chapter 1, we will provide a comprehensive overview of the mathematical model we have chosen for predicting insurance claims. We will revisit the underlying principles and key concepts of the selected model, elucidating its strengths and limitations. Emphasis will be placed on the model's suitability for handling imbalanced datasets, a critical aspect given the vast majority of non-claiming clients in our dataset. Furthermore, we'll underscore the importance of making the selected model easy to understand and explain. This is crucial for building trust among stakeholders and promoting openness in the claim prediction process.

Then, in Chapter 2, we embark on a thorough exploration of our entire dataset, delving into its characteristics, distribution, and potential patterns. We will conduct an exploratory data analysis (EDA) to gain valuable insights into the dataset's structure and discern any relationships between variables.

Lastly, in Chapter 3, we will delve into the heart of our prediction endeavor by implementing the chosen mathematical models to our dataset. To prepare the data for modeling, we will undertake a rigorous variable selection process, identifying the most influential features that significantly impact claim incidence. We will explore various techniques for feature selection, such as correlation analysis and feature importance scores. We will meticulously describe the model application process, including parameter tuning and cross-validation strategies to achieve optimal performance. Then, we will present the results of our claim prediction experiments, evaluating the model's performance metrics, such as accuracy, precision, recall, F_β-score, and area under the receiver operating characteristic curve (AUC-ROC).
Finally, to establish a robust foundation for decision-making, we will conduct a comparative analysis of multiple predictive models. Through a systematic evaluation, we will determine which model best addresses the challenge of predicting claim incidence, taking into account the unique characteristics of our imbalanced dataset. Insights from this comparison will guide us in selecting the most suitable model to employ in real-world insurance scenarios.

Chapter 1

Mathematical definitions and properties recall

In this chapter, we delve into the mathematical foundations of the selected predictive models for our purpose. The goal of our research is to aid insurance companies in accurately predicting whether a client will make a claim or not, while taking into account the inherent challenges posed by imbalanced data. We begin by introducing the chosen models, which include Linear Models such as Generalized Linear Model (GLM) and Logistic Regression, Ensemble Learning Models like XGBoost and Random Forest, and Neural Networks. Each model's fundamental principles and properties will be explored to provide a solid understanding of their capabilities and limitations.

1.1 Linear models

1.1.1 Generalized Linear Models

Generalized Linear Models are a versatile class of statistical models that extend the concept of linear regression to a broader range of data distributions, allowing for more flexible and appropriate modeling of various types of responses. Unlike the ordinary linear regression, which assumes that the response variable follows a Gaussian distribution, GLMs can handle responses that follow different probability distributions, such as binary (Bernoulli), count (Poisson), or multinomial (categorical) distributions.

The fundamental idea behind GLMs is to relate the linear predictor, a linear combination of the predictor variables, to the expected value of the response using a link function and a variance function. The link function establishes the relationship between the linear predictor

and the expected value of the response, while the variance function describes how the variance of the response is related to its mean.

The general form of a GLM for a single observation can be represented as follows:

Linear Predictor:

$$\eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p \tag{1.1}$$

where η is the linear predictor, β_0 is the intercept, $\beta_1, \beta_2, \ldots, \beta_p$ are the coefficients for the predictor variables $x_1, x_2, \ldots, x_p$, respectively.

Link Function: The link function denoted by $g(\cdot)$ relates the linear predictor to the expected value of the response variable μ. It can be different for each distribution. For example, for binary data, the logit link is commonly used, while for count data, the log link is often employed.

$$g(\mu) = \eta \tag{1.2}$$

Variance Function: The variance function denoted by $V(\mu)$ specifies the relationship between the variance of the response variable and its mean.

$$Var(Y) = V(\mu) \tag{1.3}$$

Distribution: The probability distribution function determines the nature of the response variable (e.g., Gaussian, Bernoulli, Poisson) and plays a crucial role in estimating the model parameters.

Given a dataset $\mathscr{D}$ with N observations and predictor variables $x_1, x_2, \ldots, x_p$, GLMs aim to estimate the model parameters $\beta_0, \beta_1, \ldots, \beta_p$ by maximizing the likelihood function of the data under the assumed probability distribution.

GLMs can be particularly effective for predicting whether a client will make a claim or not, even when dealing with imbalanced datasets such as the one we have.

- **Appropriate Probability Distribution:** GLMs can accommodate various probability distributions for the response variable, allowing them to handle binary (Bernoulli) data, which is commonly used for binary classification problems like claim prediction. By assuming the appropriate distribution and using the correct link function (e.g., logit), GLMs can model the probabilities of different outcomes accurately.

- **Interpretability:** GLMs provide interpretable coefficients, making it easier to understand how each predictor variable influences the likelihood of making a claim. These coefficients indicate the direction and magnitude of the relationships, enabling risk assessors or insurance experts to identify key factors affecting the likelihood of a claim.

- **Bias Handling:** With imbalanced datasets, standard classifiers may have a tendency to favor the majority class, resulting in biased predictions. GLMs, however, focus on estimating the model parameters to maximize the likelihood of the data, considering the entire dataset, thus minimizing the impact of class imbalance on parameter estimation.
- **Model Robustness:** GLMs are generally robust to outliers and noisy data, which can be beneficial in real-world scenarios where data quality may vary. This robustness helps prevent the model from being overly influenced by extreme or rare instances, which could disproportionately affect other classifiers.
- **Weighted Classifications:** GLMs can be adapted to handle imbalanced data through weighted classifications. By assigning higher weights to instances of the minority class during model training, GLMs become more sensitive to the minority class and improve their ability to predict it accurately.
- **Regularization Techniques:** GLMs can also be enhanced using regularization techniques, such as L1 (Lasso) or L2 (Ridge) regularization, which help prevent overfitting and improve generalization. Regularization is particularly useful in imbalanced datasets where overfitting can be a concern.

In summary, Generalized Linear Models (GLMs) offer a powerful framework for modeling a wide range of response variables and are a fundamental tool in statistics and machine learning for both interpretability and prediction tasks.

However, some drawbacks exist.

- **Sensitivity to Outliers:** GLMs can be sensitive to extreme outliers in the data, which may lead to biased coefficient estimates and affect the model's predictive performance.
- **Assumption of Linearity:** GLMs assume a linear relationship between the predictors and the log-odds (in the case of binary classification) or the expected value (in other distributions). This assumption may not always hold for complex real-world data, leading to suboptimal model fits.
- **Limited Flexibility:** While GLMs can handle a wide range of probability distributions, they may not be as flexible as more advanced machine learning algorithms, such as neural networks or gradient boosting machines, in capturing complex non-linear relationships in the data.
- **Inability to Capture Feature Interactions:** GLMs typically model each predictor independently, which means they may struggle to capture interactions between multiple predictor variables. Interactions can be crucial in predicting certain outcomes accurately.

- **Overfitting with Complex Models:** While GLMs can be regularized to prevent overfitting, when using more complex GLM variants, such as polynomial or interaction terms, there is still a risk of overfitting if the regularization is not appropriately tuned.
- **Imbalanced Data Challenges:** Although GLMs can handle imbalanced datasets to some extent, they may still struggle to achieve optimal predictive performance in highly imbalanced scenarios. Other techniques like resampling or specialized algorithms for imbalanced data might be more effective.

1.1.2 Logistic regression

Logistic Regression is a popular statistical and machine learning algorithm used for binary classification tasks, where the response variable takes only two possible classes (e.g., "yes" or "no," "0" or "1"). Despite its name, logistic regression is a classification algorithm, not a regression algorithm.

The fundamental concept behind logistic regression is to model the relationship between the predictor variables and the probability of the positive class (e.g., "1" or "yes") using the logistic function (sigmoid function). The output of the logistic function lies between 0 and 1, which makes it suitable for representing probabilities.

Model Representation:

Given a dataset with binary response variable y (0 or 1) and predictor variables $x_1, x_2, \ldots, x_p$, the logistic regression model can be represented as:

Linear Predictor:

$$\eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p \tag{1.4}$$

where η is the linear predictor, β_0 is the intercept, and $\beta_1, \beta_2, \ldots, \beta_p$ are the coefficients associated with the predictor variables.

Logistic (Sigmoid) Function: The logistic function, denoted by $g(\cdot)$, maps the linear predictor to the probability of the positive class.

$$\mathbb{P}(y = 1 \mid x) = g(\eta) = \frac{1}{1 + e^{-\eta}} \tag{1.5}$$

Prediction Rule: To make a prediction, a probability threshold $p_{threshold}$ (e.g., 0.5) is set. If $\mathbb{P}(y = 1 \mid x) \geq p_{threshold}$, the class is predicted as "1" (positive class); otherwise, it is predicted as "0" (negative class).

Model Training:

The logistic regression model is trained using a maximum likelihood estimation approach. The goal is to find the optimal values for the coefficients $\beta_0, \beta_1, \ldots, \beta_p$ that maximize the likelihood of observing the data given the model.

There are a lot of good points in using Logistic regression.

- **Simple and Interpretable**: Logistic regression is a straightforward algorithm that produces interpretable results. The coefficients can be easily understood in terms of the direction and strength of the relationship between predictors and the probability of the positive class.
- **Efficient for Small Datasets:** Logistic regression is computationally efficient, making it suitable for small to moderately sized datasets.
- **Probabilistic Outputs:** The model provides probabilistic outputs, allowing for the estimation of the likelihood of each class, enabling more nuanced decision-making.
- **Low Variance:** Logistic regression is less prone to overfitting, especially when the number of predictors is small and the dataset is balanced.

However, as for each model, some drawbacks need to be put in light.

- **Linear Decision Boundary:** Logistic regression assumes a linear decision boundary, which may not be appropriate for complex datasets with non-linear relationships.
- **Imbalanced Data Challenges:** Like other algorithms, logistic regression can face challenges in handling imbalanced datasets. Additional techniques like class weighting or resampling may be necessary.
- **Feature Engineering:** Logistic regression relies on appropriate feature engineering to capture meaningful relationships between predictors and the response variable. This requires domain knowledge and data preprocessing.
- **Multicollinearity:** When predictor variables are highly correlated, multicollinearity can affect the stability and interpretability of the coefficient estimates.
- **Outliers Impact:** Logistic regression can be sensitive to outliers, which might influence the model's predictions.

1.2 Ensemble learning

1.2.1 Random Forest

Random Forests is an influential ensemble learning method. This technique constructs multiple decision trees during the training phase. Each decision tree is built from a different subset of the training data, which is randomly selected with replacement (bootstrapping). Additionally, rather than considering all features for each split in the tree, only a random subset of features is considered. This introduces randomness into the model and decorrelates the trees, leading to less overfitting and variance than individual decision trees. When it comes to making a prediction, for a classification task, each tree 'votes' for a class, and the class with the most votes is output. For regression, the model output is the mean prediction of the individual trees. This strategy of combining the predictions helps to improve the robustness and predictive power of the model over individual decision trees or bagging techniques, by reducing both bias and variance.

Decision trees are fundamental building blocks of a Random Forest model. A single decision tree $T(x;\Theta_k)$ is represented as:

$$T(x;\Theta_k) = \sum_{m=1}^{M} c_m I(x \in R_m), \tag{1.6}$$

where x is the input vector, $\Theta_k = \{R_m, c_m\}_{m=1}^{M}$ are the parameters of each tree, R_m is the region of the feature space corresponding to the leaves of the tree, c_m are the predicted constants for each region, and M is the total number of leaves in the tree.

In Random Forests, a set of decision tree models $\{T(x;\Theta_k)\}_{k=1}^{B}$ are independently grown on bootstrap samples from the training data. The Random Forest predictor $\hat{f}_{rf}(x)$ is obtained by averaging the predictions from these B trees, formulated as:

$$\hat{f}_{rf}(x) = \frac{1}{B}\sum_{k=1}^{B} T(x;\Theta_k). \tag{1.7}$$

For a regression problem, $T(x;\Theta_k)$ is the predicted response for x by the kth tree and for a classification problem, $T(x;\Theta_k)$ is the class prediction of the kth tree for x.

To de-correlate the trees, when building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors. A fresh sample of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$.

Algorithm

The algorithm for Random Forests can be summarized as follows:

Algorithm 1 Random Forests

procedure RANDOM FORESTS($\mathscr{D}$, B, m)
  for $k = 1$ to B **do**
    Draw a bootstrap sample $\mathscr{D}_k^*$ of size N from $\mathscr{D}$.
    Grow a random forest tree T_k to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      Select m variables at random from the p variables.
      Pick the best variable/split-point among the m.
      Split the node into two daughter nodes.
  end for
  Output the ensemble of trees $\{T_k\}_{k=1}^{B}$.
end procedure

Pros

- **Robust to Overfitting:** Random Forests create a set of decision trees from a randomly selected subset of the training set. It then combines votes from different decision trees to decide the final class of the test object which makes it less prone to overfitting.
- **Handle Large Data Sets:** Random Forests can handle large datasets with high dimensionality. They can handle thousands of input variables and identify most significant ones.
- **Parallelizable:** The process of building trees is easily parallelizable as each tree is fully independent of the others.
- **Versatility:** Random Forests can be used for both regression and classification tasks.

Cons

- **Model Interpretability:** Unlike decision trees, where the learned sequence of if-then-else rules can be interpreted by humans, Random Forest models are more like black boxes.
- **Predictive Speed:** Prediction could be slow for large number of trees. The model needs to run the input data through many decision trees and make a majority vote.

- **Memory Usage:** Random Forests can require more memory if the number of trees is large.
- **Not Ideal for Linear Relationships:** If the data contain linear relationships, linear models are more appropriate. Random Forests do not handle this well because they operate by establishing non-linear decision boundaries.

Random Forests, by virtue of being an ensemble model, decreases the model variance without increasing the bias. This makes the model more robust to overfitting. Although relatively simple, it is one of the most powerful machine learning algorithms available today.

1.2.2 Gradient Boosting

Gradient boosting is a machine learning technique which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. The underlying idea is to add new models to the ensemble sequentially. Each new model gradually minimizes the loss function (the difference between the predicted and true values) of the whole system using Gradient Descent method. The process continues until a pre-set number of trees is reached, or no further improvement can be made.

Unlike Random Forests, which constructs a collection of independent decision trees, Gradient Boosting builds its decision trees sequentially. Each newly added tree is designed to correct the errors and inadequacies of its predecessors. As a consequence, with each additional tree, the model incrementally enhances its ability to capture complex patterns in the data, thereby becoming more expressive and potentially more accurate

XGBoost, a scalable machine learning system for tree boosting, uses a more regularized model formalization to control over-fitting, which gives it better performance. It is equipped to handle sparse data and missing values, and can do parallel computations on a single machine.

The objective function to be optimized for XGBoost is given as:

$$\text{obj}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i), \tag{1.8}$$

where l is the differentiable convex loss function that measures the difference between the target y_i and the prediction $\hat{y}_i^{(t)}$, Ω is the regularization term which penalizes the complexity of the model, f_i is the i-th tree and t is the number of iterations.

The algorithm for XGBoost can be summarized as follows:

Algorithm 2 XGBoost

procedure XGBOOST($\mathscr{D}$, num_rounds)
  Initialize model with a constant prediction $\hat{y}_i^{(0)} = \frac{1}{2}\log\left(\frac{\sum_{i=1}^{n} w_i I(y_i=1)}{\sum_{i=1}^{n} w_i I(y_i=-1)}\right)$
  for $t = 1$ to num_rounds **do**
    Compute first and second-order gradients $g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ and $h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ for all i
    Build a tree model that predicts g_i with regularization Ω
    Compute the optimal weight w_j^* for each leaf node j in the tree
    Update prediction $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta w_j^*$, where j is the leaf node that instance i falls into and η is the learning rate
  end for
  Output the final prediction $\hat{y}_i^{(t)}$.
end procedure

Pros

- **High Performance:** often delivers high predictive accuracy compared to other machine learning algorithms. It is known for winning numerous Kaggle competitions.
- **Parallelizable:** The construction of decision trees can be parallelized, which makes XGBoost faster than other gradient boosting techniques.
- **Regularization:** has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents the model from overfitting.
- **Handling Missing Values:** has an in-built routine to handle missing values.

Cons

- **Requires Careful Tuning:** requires careful parameter tuning to give good results, which can be time-consuming.
- **Overfitting:** If not properly regularized or if the data is noisy, XGBoost can overfit the training data.
- **Computationally Intensive:** can be computationally intensive for large datasets, especially with a high number of features.
- **Not Ideal for Linear Relationships:** If the data contain linear relationships, linear models are more appropriate. XGBoost operates by establishing non-linear decision boundaries.

XGBoost is an effective machine learning model even on datasets where the features are of different types. Its built-in L1 and L2 regularization terms prevent the model from overfitting, and it can handle missing values, making it a robust and powerful machine learning algorithm.

1.3 Deep learning

1.3.1 Neural Network

Neural networks are machine learning algorithms inspired by the human brain's neural connections. They consist of interconnected artificial neurons organized in layers. Each neural network comprises multiple layers, with interconnected neurons in each layer. These neurons receive input data, process it using learned weights, and transmit the information to the next layer. Neural networks iteratively adjust the connection weights during training to minimize prediction errors.

A single neuron's operation can be mathematically represented as:

$$z = \sum_{i=1}^{n} w_i \cdot x_i + b$$

with x_i is the input to the neuron, w_i represents the corresponding weights, b is the bias term, and n is the number of inputs. The neuron's output is then passed through a specific mathematical function (referred to as the activation function, for instance, Sigmoid Function, ReLU, etc.) to introduce non-linearity in the system: $a = f(z)$. With in the case of the sigmoid function:

$$f(z) = \frac{1}{1+e^{-z}}$$

and for the ReLU function:

$$f(z) = \max(0,z) = \begin{cases} z & \text{if } z \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

In a neural network, these interconnected neurons are arranged in layers, including an input layer, one or more hidden layers, and an output layer. The presence of hidden layers allows the network to grasp intricate data representations.

The output of the first layer becomes the input to the second layer and so forth, ultimately leading to the final output layer. The computation of the entire neural network's output can be described as:

$$\hat{y} = f_{\text{NN}}(z)$$

where $\hat{y}$ represents the predicted output and $f_{\text{NN}}(\cdot)$ denotes the neural network's transformation.

An algorithm of Neural network in our situation can look like:

Algorithm 3 Neural Network Training and Prediction

procedure NEURALNETWORK($X_{\text{train\_scaled}}, y_{\text{train}}, X_{\text{test\_scaled}}$)
    Create the neural network model
    Initialize a Sequential model.
    Add a Dense layer with 64 units with a ReLU activation function as the input layer, using the input shape of $X_{\text{train\_scaled}}$.
    ▷ Add x hidden layers with activ_f activation function
    for $i = 1$ to x **do**
        Add a Dense layer with n_i units and activ_i activation function.
    end for
    Add a Dense layer with 1 unit and sigmoid activation as the output layer.
    Compilation
    Compile the model with a optimizer, a loss function, and a metric.
    Training
    Train the model on $X_{\text{train\_scaled}}$ and y_{train} for n_e epochs with batch size b_s.
    Predictions
    Use the trained model to predict the target variable for $X_{\text{test\_scaled}}$.
    Store the predictions in the variable *pred*.
    Output the predictions *pred*.
end procedure

We will see in the result part of the paper, how using gridsearch, we found the best parameters for the number epochs, the batch size, the optimizer and the number of units in the hidden layers.

Neural Networks offer numerous advantages:

- **Flexibility**: Neural networks can learn complex relationships in data, making them suitable for a wide range of tasks.
- **Non-Linear Transformations**: They can approximate non-linear functions, making it possible to model complex patterns in the data.

- **Feature Learning**:Neural networks can automatically learn relevant features from raw data, reducing the need for manual feature engineering.
- **Scalability**: They can be scaled up to handle large datasets and complex problems.

As for every models there are some drawbacks:

- **Training Complexity**: Deep neural networks may require significant computational resources and time for training.
- **Overfitting**: Neural networks can overfit on small datasets, necessitating regularization techniques.
- **Interpretability**: Understanding the reasoning behind neural networks' decisions can be challenging due to their black-box nature.
- **Data Requirements**: Training deep networks effectively often requires large amounts of labeled data.
- **Hyperparameter Tuning**: Selecting the appropriate architecture and hyperparameters can be a tedious process.

Chapter 2

Exploratory data analysis (EDA)

In this EDA, we seek to identify relevant trends and relationships within our data set. To do this, we will use a diverse set of visualizations, including histograms, correlation matrices and graphs. With histograms, we will have a clear understanding of the distribution of individual variables, allowing us to identify key and central trends. The correlation matrix will help us explore the strength and direction of relationships between different variables, highlighting potential dependencies.

Dataset Description: The dataset comprises information on 40,284 Spanish insurance policy holders tracked from 2010 to 2014. Each policy holder had motor and homeowners insurance contracts. Not every policy holder stayed the full five years in the dataset due to some not renewing their policies.
The dataset includes 122,935 rows with each row uniquely identified by a policy ID ("PolID") and contains data for each customer, policy, and yearly claims by contract type.

- **Key Fields:**
  - **gender:** 1 for male and 0 for female.
  - **Age_client:** The age of the customer.
  - **age_of_car_M:** Age of the car since its purchase by the customer.
  - **Car_power_M:** Power of the car.
  - **Car_2ndDriver_M:** 1 if a second driver is present, 0 otherwise.
  - **num_policiesC:** Total number of policies held by the customer.
  - **metro_code:** 1 for urban or metropolitan, 0 for rural.
  - **Policy_PaymentMethodA/H:** Payment method for motor/homeowners policy (1 for annual payment, 0 for monthly).

- **Insuredcapital_content_re:** Value of content in homeowners insurance.
- **Insuredcapital_continent_re:** Value of building in homeowners insurance.
- **appartment:** 1 if the homeowners insurance is for an apartment, 0 otherwise.
- **Client_Seniority:** Number of years the customer has been with the company.
- **Retention:** 1 if the policy is renewed, 0 otherwise (we will not use that information for prediction as this information is not supposed to be available for the insurer at the begining of the year when we want to predict claims).
- **NClaims1:** Number of claims for the motor insurance in the corresponding year.
- **NClaims2:** Number of claims for the home insurance in the corresponding year.
- **Claims1:** Total claims cost for the motor insurance in the corresponding year.
- **Claims2:** Total claims cost for the home insurance in the corresponding year.
- **Types:** Claims type (1: no claims, 2: auto claim only, 3: home claim only, 4: both auto and home claims). We will not use this variable as it is redundant with Claims1 and Claims2.

We decided to create four additional variables. We introduced two new variables: BClaim1 for motor insurance and BClaim2 for home insurance, they are derived from **NClaims1** and **NClaims2**. These binary variables are assigned a value of 0 if the clients have made no claims and 1 if they have made one or more claims. These varaibles will be the target varaibles in our predictive models.

This decision was motivated by the relatively small number of clients making claims, which made it challenging to differentiate between clients who made single or multiple claims within a year. Thus, to facilitate a more robust analysis, we opted to aggregate all claim-making clients into a single group. By introducing these variables, the objective is to build a classifier that can distinguish between clients who are likely to make a claim and those who are not.

We also created the variables "**avg_past_claims1**" and "**avg_past_claims2**", which are features that represent the ratio of the number of claims made by a client through the previous years for the motor and home sub dataset respectively. These features are introduced to capture the claim history of clients and assess their overall claim propensity. However, there is a challenge in the dataset regarding new clients (who have no history). Since there is no data available for what happened in the previous year for clients in the initial year, calculating the ratio directly would lead to undefined or inaccurate values for this group. To address this issue, a practical approach is taken. For new clients we set the "**avg_past_claims1**" and "**avg_past_claims2**" to the overall mean (over the dataset) of these varaibles. This approach helps in filling the data

gap for the initial year and provides a neutral estimate of the average claim propensity for those clients. While this approximation may not be as precise as actual historical data, it allows the model to utilize the available information and still make predictions for the target variables "BClaims1" and "BClaims2".

We split the datasets in two, one for the motor model and one for the home model, keeping some of the variables in both datasets. The "motor dataset" contains the following specific features: "age_of_car_M", "Car_power_M", "Car_2ndDriver_M", "Policy_PaymentMethodA", "BClaims1" as well as the common features: "gender", "Age_client", "year", "PolID", "num_policiesC", "metro_code", "Client_Seniority", "Retention", "Types", "avg_past_claims1", "avg_past_claims2". The "home dataset" includes: "Policy_PaymentMethodH", "Insuredcapital_content_re", "Insuredcapital_continent_re", "appartment", "BClaims2" along with the common features as listed above.

In the following plots, we observe that the distribution of the age of the clients appears to be normal, with the clients' ages concentrated around 50 years. Similarly, for car power and insured capital, they mostly have a normal distribution and ranges between 0 and 200 for the car power and between 7 and 12 for the insured capital. However, concerning client seniority, there is a peak between 0 and 10 years, which then decreases, suggesting a majority of recent clients, similar to the age distribution of the cars, where there is a peak for new vehicles, leading to a majority of recent cars. And for NClaims1 and NClaims2 that logically is pretty similar to the distribution of Claims1 and Claims2, we see a similar pattern, with a majority of clients having few claims, and mainly 0 claims. Note that we use a logarithmic scale for these two graphs, otherwise it would not be possible to see the distribution for clients with one or more claims.

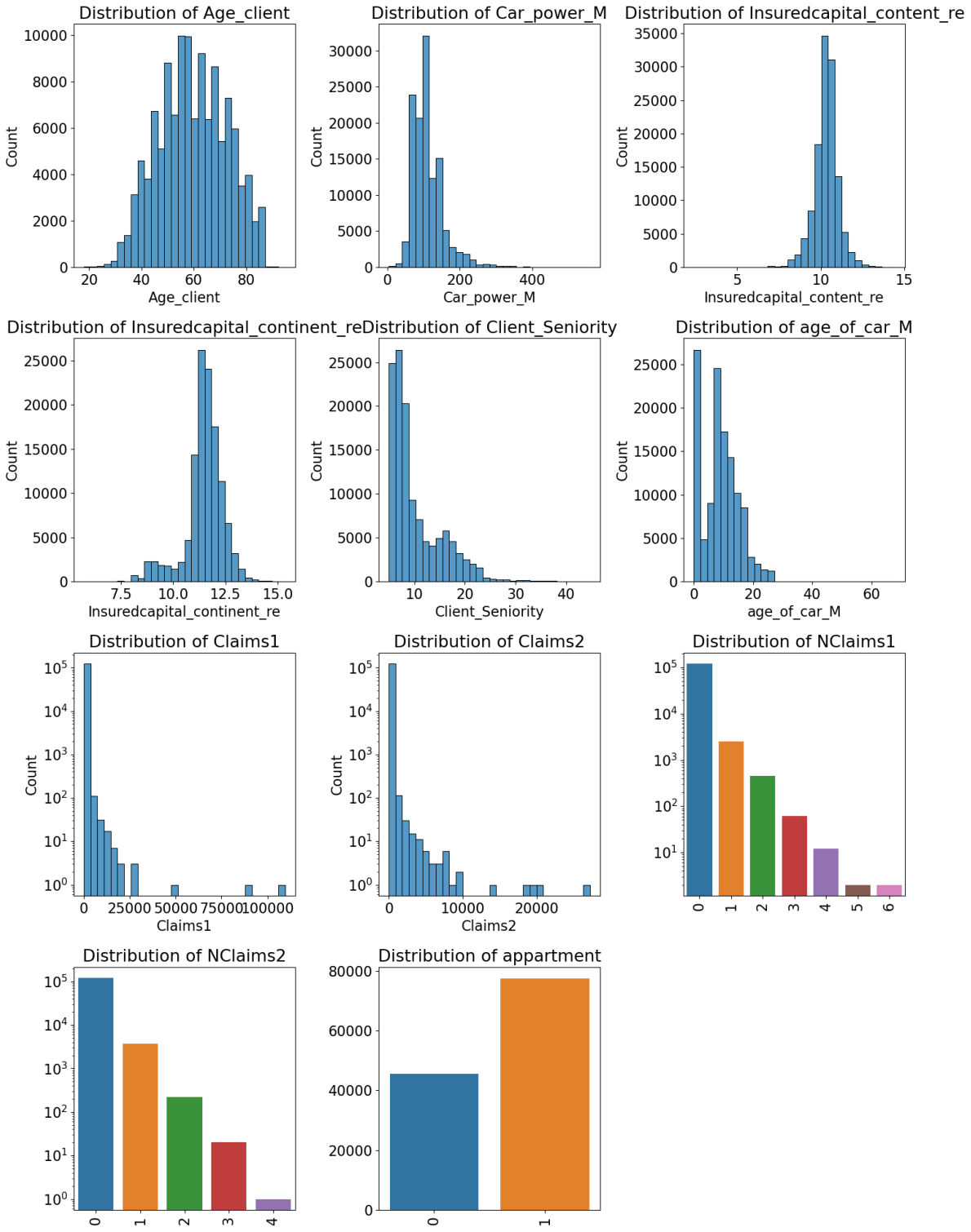

Distribution of Age_client
Count
Age_client
Distribution of Car_power_M
Count
Car_power_M
Distribution of Insuredcapital_content_re
Count
Insuredcapital_content_re
Distribution of Insuredcapital_continent_re
Count
Insuredcapital_continent_re
Distribution of Client_Seniority
Count
Client_Seniority
Distribution of age_of_car_M
Count
age_of_car_M
Distribution of Claims1
Count
Claims1
Distribution of Claims2
Count
Claims2
Distribution of NClaims1
Distribution of NClaims2
Distribution of appartment

Fig. 2.1

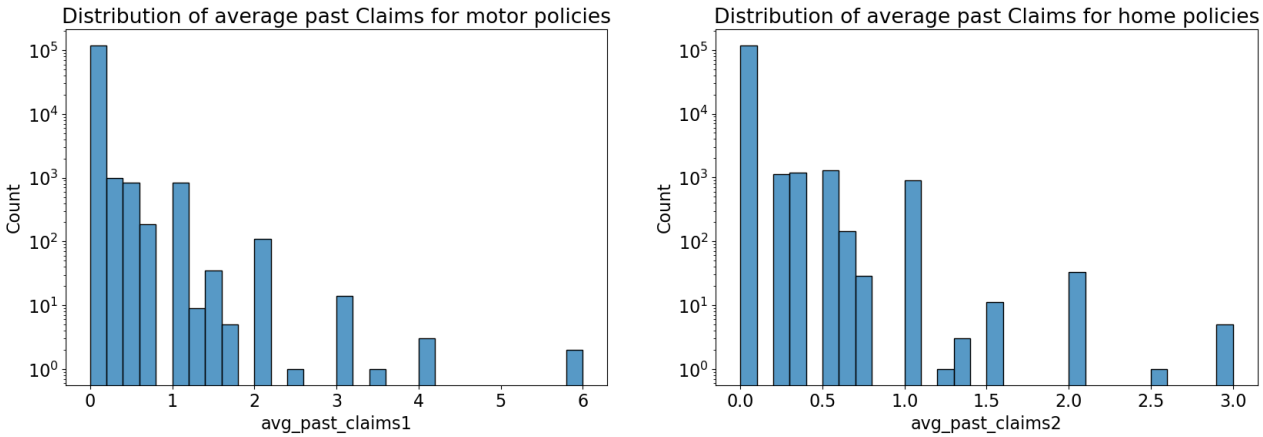


Fig. 2.2

In the same way as for the distribution of claims1 and claims2, we use a logarithmic scale for these graphs, to visualize the distribution of these variables despite their striking imbalance.

2.1 EDA on the motor related variables

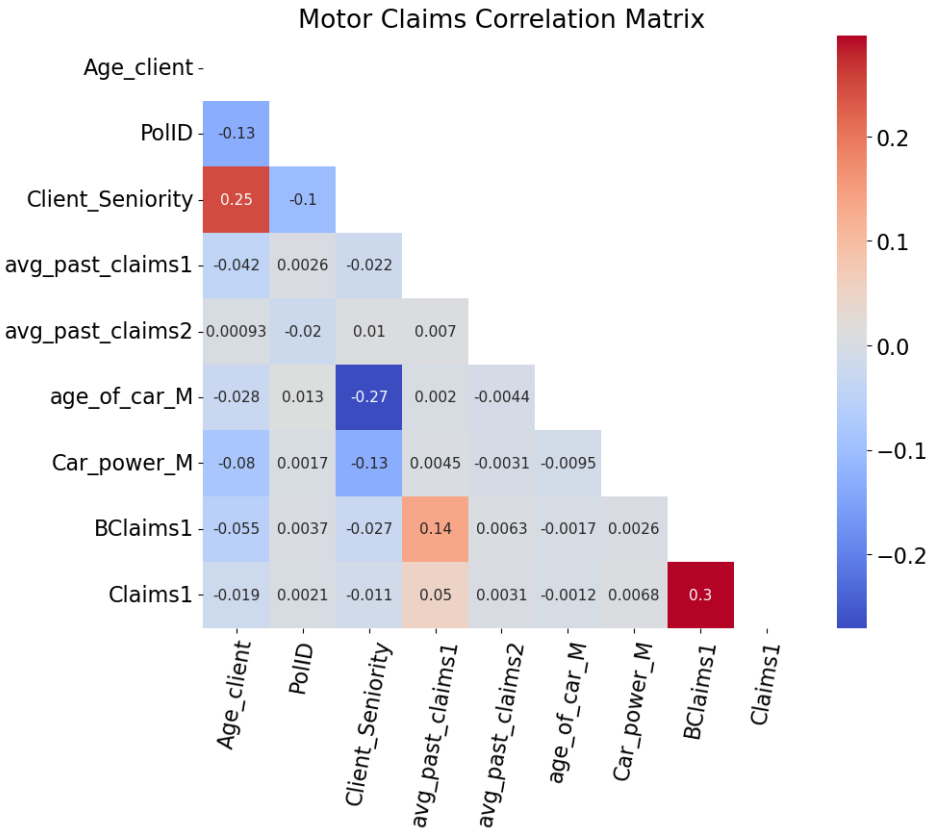


Fig. 2.3

Let us recall that in a correlation matrix, values close to $+1$ mean a perfect positive linear relationship. On the other hand, a correlation coefficient of -1 indicates a perfect negative linear relationship. And a value close to zero indicates a weak correlation. In the correlation matrix of the motor claims, the variables most correlated with the variable BClaims1 are Claims1, as well as Avg_Past_Claims1, suggesting a significant and consistent association between these variables. In contrast, the variables least correlated with BClaims1 are car age, client seniority and PolID, implying that BClaims1 is less influenced by these factors. Given that no pair of variables have a correlation above 0.5, we expect that ther won't be any colinearity problems when fitting the models.

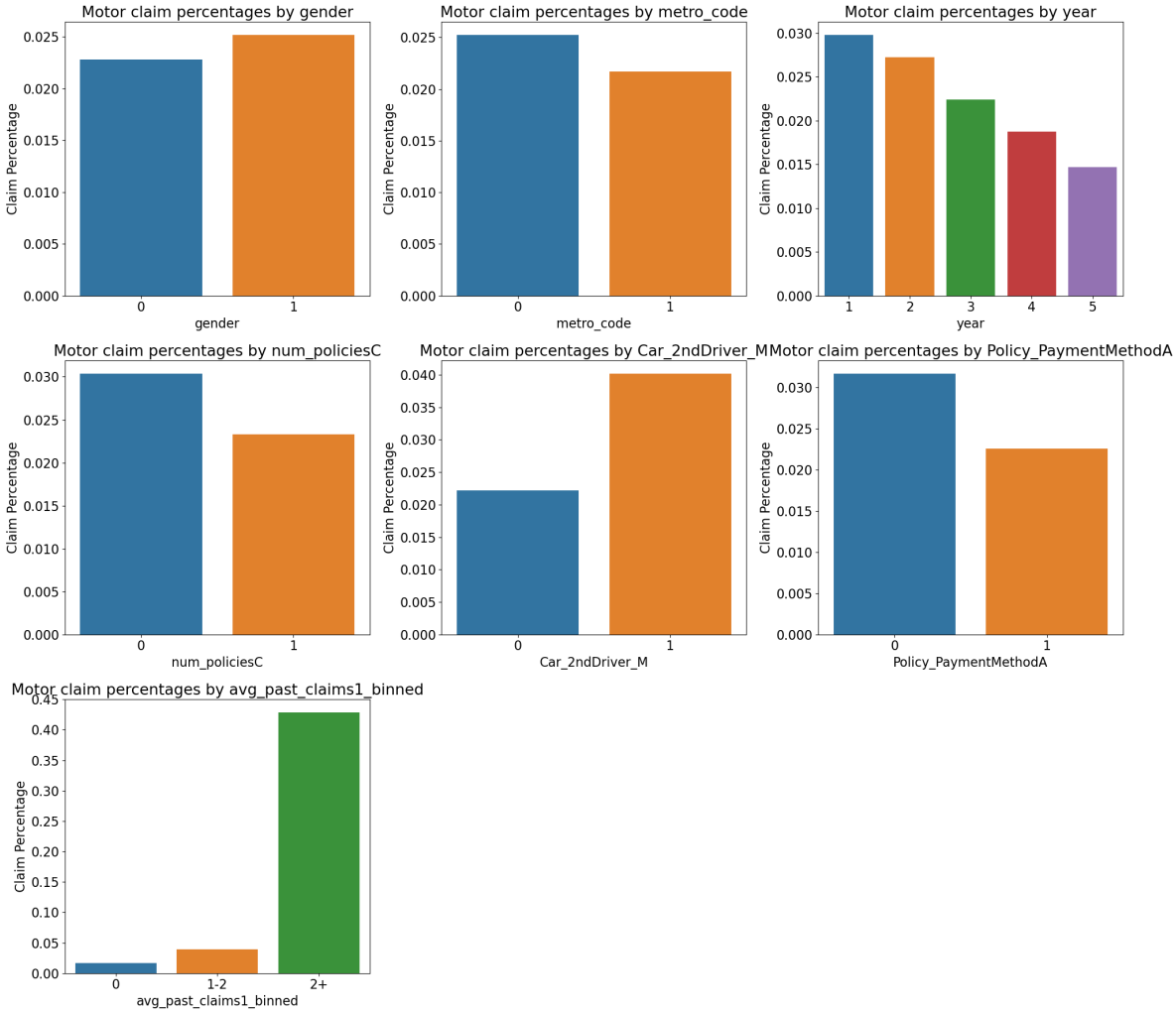


Fig. 2.4

We decided to examine claim percentages as a function of certain variables. What we observed is that the percentage of claims is roughly the same for both genders and for both metropolitan codes. The varaibles with the more discrepancies seem to be "car_2ndDriver" and "avg_past_claims1_binned". Surprisingly, with our data set, we find that there are fewer and fewer claims over the years.

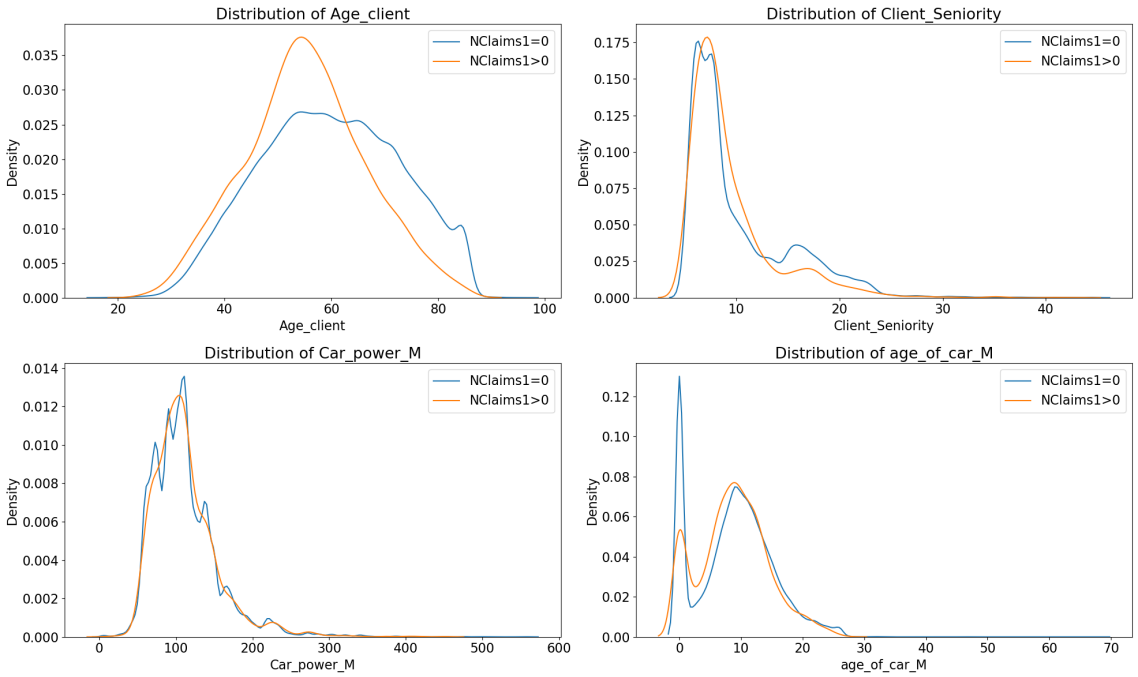


Fig. 2.5

Thanks to these plots, we observe a similar distribution for the different value of some variables such as car_power_M and the client seniority. There is only a difference between the distrubution of the age of the clients having more than 0 claims versus whose with zero claims, and with the age of car. We can see that the distribution of age of client for a positive number of claims is almost normal, whereas for the zero number of claims, it is not. With the plot of the distribution of the age of the car we observe that there are very few claims for recent cars, but except from this peak the distribution looks the same.

2.2 EDA on the home related variables

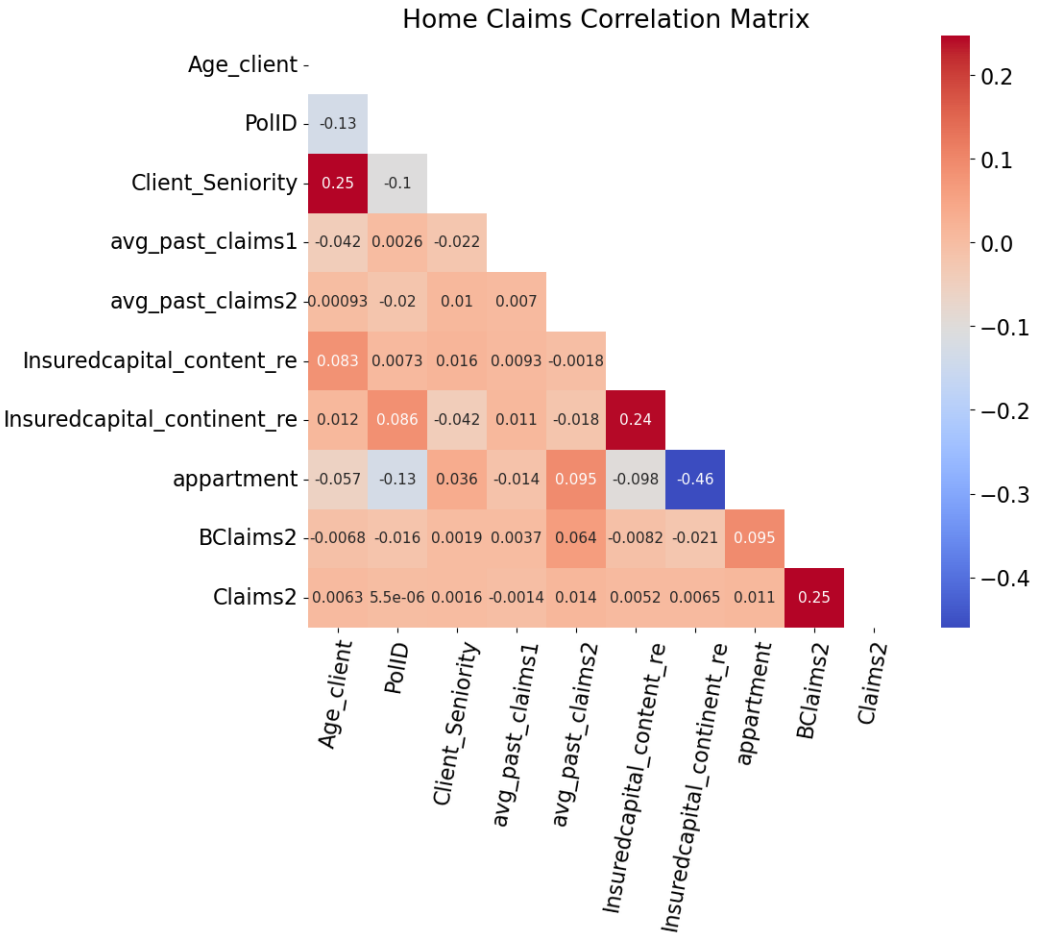


Fig. 2.6

In the correlation matrix of the home claims, the variables most correlated with the variable BClaims2 are Claims2, as well as avg_past_claims2 and appartment suggesting a significant and consistent association between these variables. In contrast, the variables least correlated with BClaims2 are Age_client, Client_Seniority and avg_past_claims1, implying that BClaims2 is less influenced by these factors.

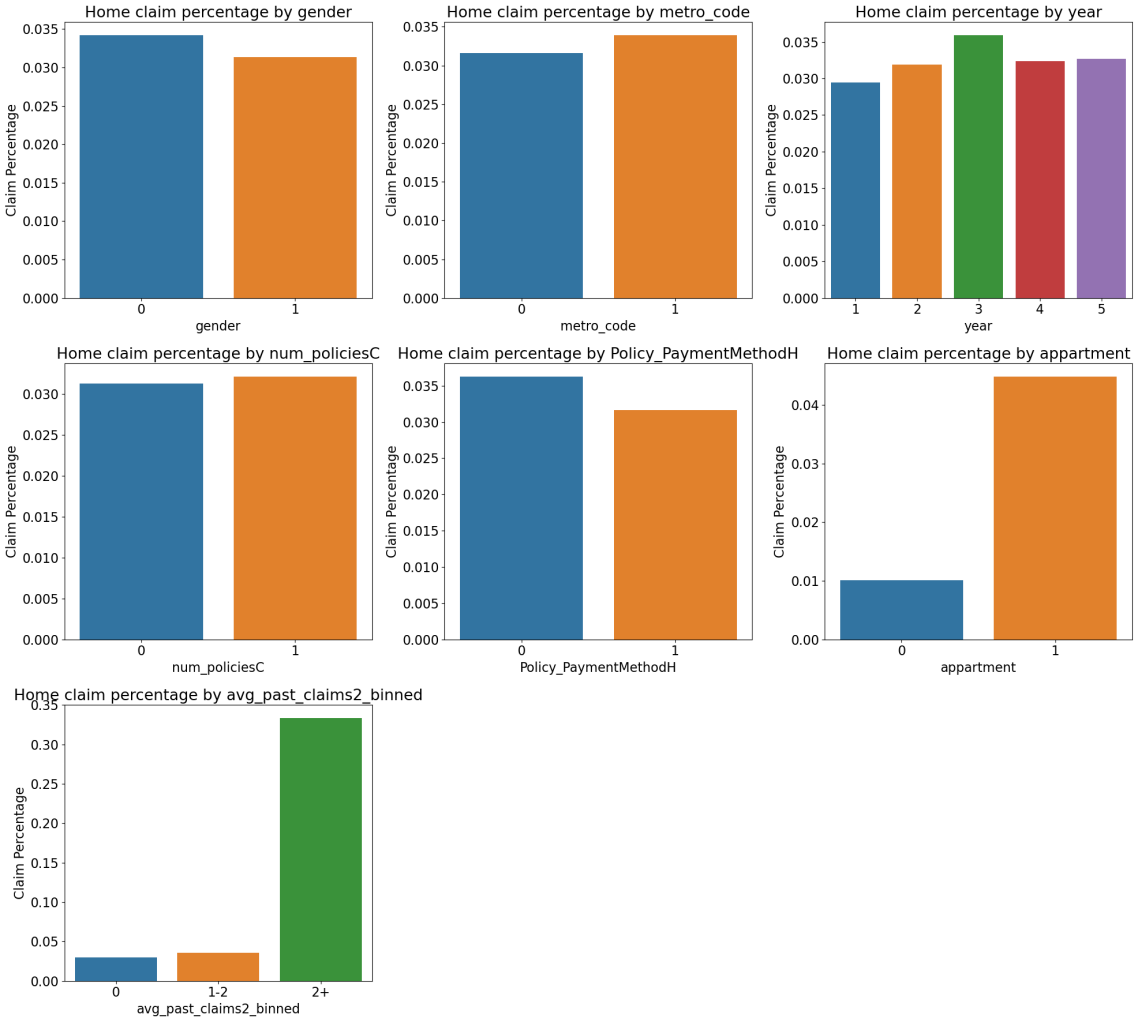


Fig. 2.7

Here at the opposite from the data from the motor model, we have similar percentage of claims for different num of policies, different gender, different metro code, different year and different policy payment method. This may mean that these variables will not have a significant impact on the number of claims. Indeed, we cannot directly understand a relationship between these variables and the number of claims. However the more people have an apartment the more they make claims.

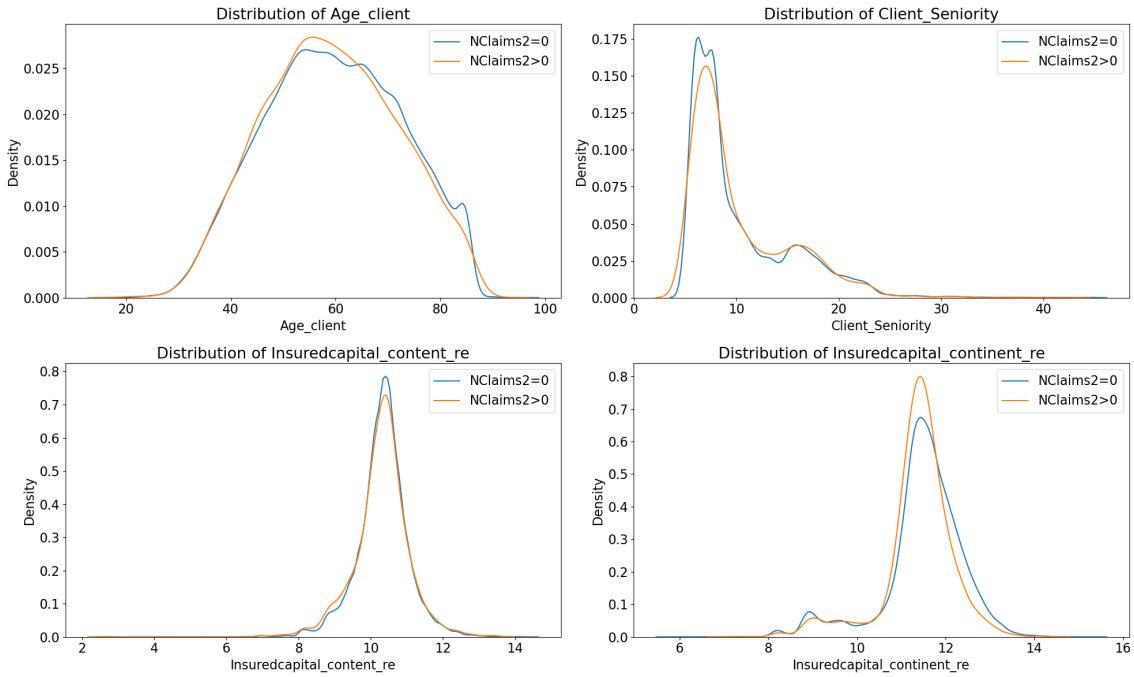


Fig. 2.8

Here for all of the 4 plots, the distributions of the variables are almost exactly the same whether the number of claims is more than 0 or not. For Insured capital continent re, the peak is a bit higher for a positive number of claims, same for age client , in contrast to the distribution insured capital content re and to the distribution of client seniority .

Chapter 3

Model results and comparisons

3.1 Methodology

To optimize the hyperparameters of our model, we perform a two-step cross-validation grid search. The first grid search is conducted with a relatively wide range of possible values. This step serves as a coarse search to identify a promising range for the optimal hyperparameters. Once a promising range is identified, a second grid search is performed with a more refined list of values that are closer to the optimal values found in the first grid search. This step is like a fine search, which allows us to pinpoint the optimal value more accurately. This two-step process proves to be beneficial because it makes the search over the hyperparameter space more efficient. Starting with a broad range helps identify the promising areas, and then narrowing down the search to those areas saves computational resources, avoiding detailed searches in less promising regions.

When dealing with a highly imbalanced dataset, such as in predicting insurance claims where the instances of positive outcomes (claims) are very scarce, traditional metrics like accuracy or precision may not provide an accurate evaluation of a model's performance. A model might achieve a high accuracy rate simply by always predicting the majority class, thus underrepresenting the minority (positive) class.
The F-beta score provides a more appropriate metric in such scenarios. It is a weighted harmonic mean of precision and recall, the exact contribution of which is determined by the β parameter. When $\beta = 1$, the F-beta score equates to the F1 score, with precision and recall contributing equally. However, when $\beta > 1$, the F-beta score places more emphasis on recall over precision. We chose the value of $\beta = 1.2$ in this case, meaning that the model is more incentivized to correctly identify as many positive instances (claims) as possible, which is usually the intended goal in an insurance setting.

We used F-beta score in the cross-validation to optimize our hyperparameters and during the features selection.

After performing the two-step cross-validation grid search and identifying the optimal hyperparameters, we proceed to build the model using these best parameters. Then, we evaluate the model's performance using various metrics to gain insights into its effectiveness in predicting the target variable. Once the model is trained, we can extract feature importances. Feature importances help us understand the relative importance of each input variable in making predictions. This information is crucial as it allows us to identify which features have the most significant impact on the model's output and which ones might be less influential. The least important features are omitted to improve model accuracy and efficiency. The metrics that we will use to assess the models' performance are the following:

- **Confusion Matrix**: The confusion matrix is a fundamental tool for evaluating the model's classification performance. It provides a breakdown of the model's predictions in a tabular format, comparing them to the true labels in the dataset. From the confusion matrix, we derive key metrics such as True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN).
- **Precision, Recall, and F-beta Score**: Precision and recall are important metrics when dealing with imbalanced datasets, where one class may dominate the other. Precision represents the proportion of true positive predictions among all positive predictions, while recall (also known as sensitivity) measures the proportion of true positive predictions among all actual positive instances. The F-beta score combines both precision and recall, with the parameter beta controlling the balance between the two. In our case, using a beta of 1.2 places more emphasis on recall, which is valuable when minimizing false negatives is a priority. Mathematically speaking, we have:
  - Precision = $\frac{TP}{TP+FP}$
  - Recall = $\frac{TP}{TP+FN}$
  - F-beta= $\frac{(1+\beta^2)\cdot(Precision\cdot Recall)}{(\beta^2\cdot Precision)+Recall}$
- **Accuracy and AUC-ROC**: Accuracy, a more general metric, indicates the overall correctness of the model's predictions. It is the ratio of correct predictions to the total number of predictions. On the other hand, the AUC-ROC metric evaluates the model's ability to discriminate between positive and negative instances across different classification thresholds. A higher AUC-ROC score signifies better model performance.

3.2 Linear methods

3.2.1 GLM

- **Hyperparameter optimization**: A cross-validation grid search is performed to find the optimal hyperparameters for a TweedieRegressor model. In our case, we chose to optimize the following hyperparameters:
  - **alpha**: Controls the regularization strength in the GLM model.
  - **link**: Determines the link function used to connect the linear predictor with the mean of the distribution.
  - **power**: Controls the variance function of the distribution and influences the shape of the distribution. Lower values of "power" (close to 1) correspond to distributions with higher variance, capturing data with more excessive zeros and heavy-tailed behavior. Higher values of "power" (greater than 1) correspond to distributions with lower variance, suitable for modeling data with fewer zeros and lighter tails

  By specifying a wide range of values for `alpha` and including multiple link functions, we aim to explore different combinations to identify the hyperparameters that optimize the model's performance on the training data. The optimal values found are as follows:
  - For the motor model: `{'alpha': 0.001, 'link': 'auto', 'power': 0}`
  - For the home model: `{'alpha': 0.001, 'link': 'auto', 'power': 0}`

  Given the context of an imbalanced dataset and the goal of assessing whether a client is in a risk group rather than making precise predictions, the selected hyperparameters seem to have balanced the model's performance in capturing actual claimants while still allowing some flexibility in the model's complexity. We observe, without surprise that the optimal power is 1.

- **Feature selection**: In GLMs, the feature importance is typically determined by analyzing the estimated coefficients assigned to each feature in the model. These coefficients represent the impact of each feature on the target variable's prediction. Features with higher absolute coefficients are considered more important as they contribute more significantly to the model's predictions. Below Figure 3.1 and 3.2 shows the importance of each variables for predicting the clients that will make motor claims or home claims.

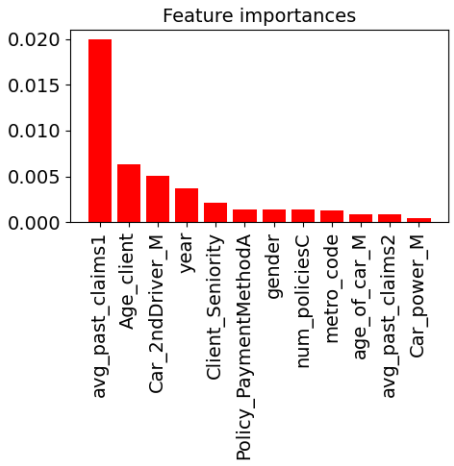


Fig. 3.1 Motor claims

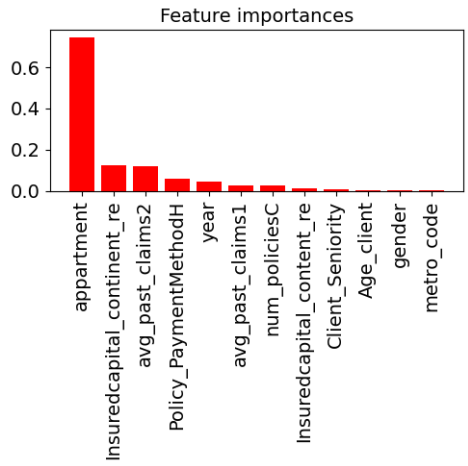


Fig. 3.2 Home claims

Based on this ranking, we conducted a step-wise elimination of the least important variables and observed the impact on the F-beta score. Since the number of variables was relatively small, we opted to remove variables only when doing so led to an enhancement in the F-beta score, ensuring a continuous improvement in the model's performance.

This methodology resulted in removing the following variables in the case of the motor claim prediction : "avg_past_claims2" (which is the average number of past claims for the home policy) and "Car_power_M".

For the home claims prediction, following the same process, we removed the variables "metro_code" and "gender".

- **Model training**: After obtaining the predictions using the trained TweedieRegressor model, we apply a threshold to convert the predicted probabilities into binary predictions (1 or 0) for claims or no claims respectively. Specifically, we set the threshold as `threshold_p` which is simply define as:

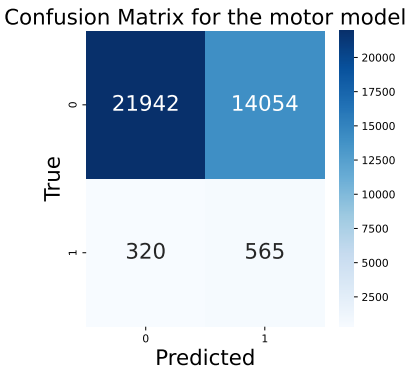
$$\texttt{threshold\_p} = \frac{\#(BClaims_i^{train} = 1)}{\#(BClaims_i^{train})}, \quad \text{where } i \in 1,2. \tag{3.1}$$

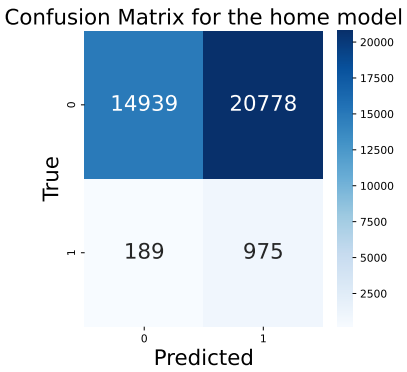
Thus, the binary predictions are determined as follows:

```
predictions = np.where(predictions > threshold_p, 1, 0)
```

Here, the value of `threshold_p` plays a crucial role in the classification process. It allows us to control the trade-off between precision and recall, as well as false positives and false negatives. Setting a higher threshold (closer to 1) results in a more conservative prediction approach, leading to fewer positive predictions (claims). This can increase precision but may lower recall. Setting a lower threshold (closer to 0) leads to a more aggressive prediction approach, resulting in more positive predictions. This can increase recall but may lower precision. The choice of the optimal threshold value depends on the specific requirements and objectives of the prediction task. To determine the best threshold, we though it was a good idea to understand the ratio of claims and to take this ratio to be the threshold.

- **Results**:





Motor Model:

- **F-beta Score:** 0.087
- **Precision:** 0.039
- **Recall:** 0.638
- **Accuracy:** 0.610
- **AUC-ROC:** 0.624

Home Model:

- **F-beta Score:** 0.102
- **Precision:** 0.049
- **Recall:** 0.827
- **Accuracy:** 0.439

  - **AUC-ROC:** 0.627

Based on the provided results for the GLM model, it seems that the performance of the model is not satisfactory for predicting clients who make claims versus those who do not, particularly in identifying the risky group that may make claims. More explicitly, the low F-beta suggests that the model has low performance in correctly classifying positive samples. With a precision of 0.039 for motor and 0.045 for home, the model is making very few true positive predictions compared to the overall positive predictions, which is not desirable. While the recall is better than precision for both motor and home, it is still not high enough, indicating as pointed before that the model is missing a substantial number of actual positive cases. The accuracy quite good is driven by the large number of true negatives which is not a good point after all. The AUC of 0.624 and 0.628 are only slightly better than random guessing (AUC = 0.5), indicating that the model's discriminatory power is weak.

3.2.2 Logistic Regression

- **Hyperparameter optimization**: As for the GLM (TweedieRegressor) model, we apply GridSearchCV to find the optimal hyperparameters for a Logistic Regression model. We have chosen the following parameters to optimize:
  - **C**: The regularization parameter, which controls the inverse of the strength of regularization. Smaller values specify stronger regularization.
  - **penalty**: The regularization norm to be used.
  - **solver**: The algorithm to use for optimization during model training.
  - **max_iter**: The maximum number of iterations allowed for the solver to converge during training.
  - **class_weight**: Allows to assign weights to classes in the dataset, which is helpful when dealing with imbalanced.

  The optimal values found are as follows:

  - For the motor model:
    `{'C': 0.01, 'class_weight': 'balanced', 'max_iter': 10000, 'penalty': 'l2', 'solver': 'sag'}`
  - For the home model:
    `{'C': 10, 'class_weight': 'balanced', 'max_iter': 1000, 'penalty': 'l2', 'solver': 'saga'}`

For the regularization parameter **C**, the motor model performed best with a value of 0.01, indicating that a smaller regularization penalty was more suitable for this particular model. On the other hand, the home model achieved its best performance with a higher **C** value of 10, suggesting that a stronger regularization penalty was preferred in this case. Not surprisingly, both models utilized the same class weight strategy, opting for **balanced** weights. However, the choice of solver algorithm differed between the two models. The motor model prefer the **sag** solver, known for its efficiency with large datasets, while the home model selected the **saga** solver, an extension of **sag** well-suited for large datasets also.

- **Feature selection**: In Logistic regression models, the feature importance is determined the same way than for GLMs. Below Figure 3.3 and 3.4 shows the importance of each variables for predicting the clients that will make motor claims or home claims.

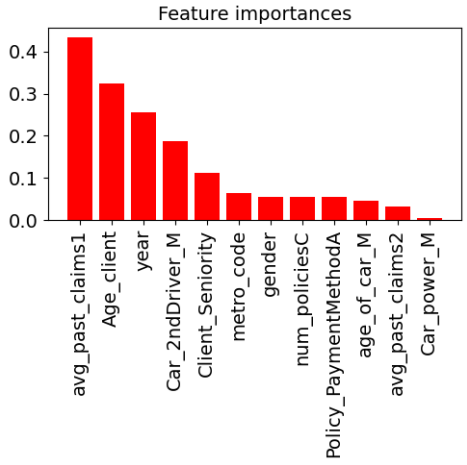


Fig. 3.3 Motor claims

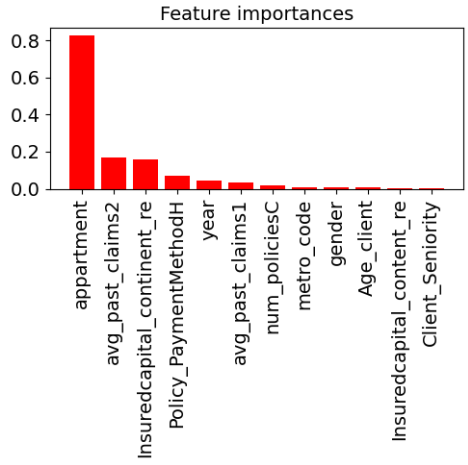


Fig. 3.4 Home claims

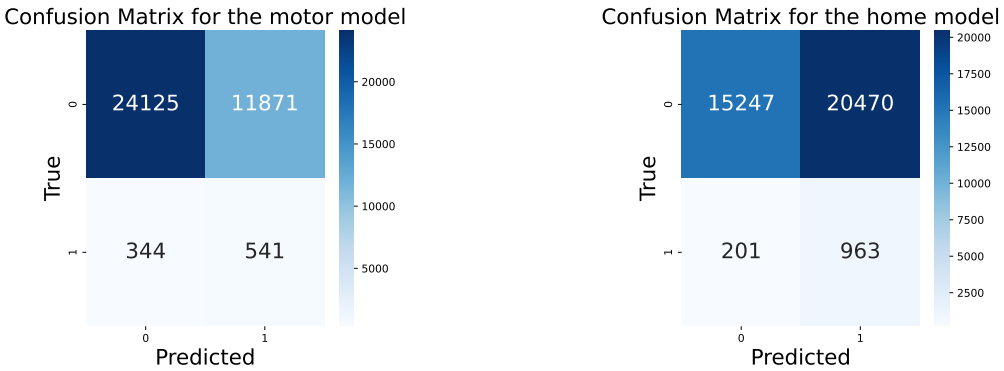
Applying the same methodology than for the GLM subsection, we find the following variables to remove:

In the case of the motor claim prediction : "avg_past_claims2" (which is the average number of past claims for the home policy) and "Car_power_M".

For the home claims prediction we removed the variables "metro_code" and "gender".

- **Model training**: After obtaining the predictions using the trained LogisticRegression model, we apply the same threshold than defined above (3.1).

- **Results**:



Motor Model:

- **F-beta Score:** 0.086
- **Precision:** 0.039
- **Recall:** 0.638
- **Accuracy:** 0.610
- **AUC-ROC:** 0.624

Home Model:

- **F-beta Score:** 0.102
- **Precision:** 0.045
- **Recall:** 0.828
- **Accuracy:** 0.440
- **AUC-ROC:** 0.628

Once again, the results are inconclusive. However, it is crucial to consider the challenge of precisely predicting claims for an imbalanced dataset.

3.3 Ensemble methods

3.3.1 Random Forest

- **Hyperparameter optimization**: A cross-validation grid search is performed to find the optimal hyperparameters for a RandomForestClassifier model. In our case, we chose to optimize the following hyperparameters:

- **max_depth**: Defines the maximum depth of each tree. A higher depth can lead to overfitting.
- **min_samples_leaf**: Specifies the minimum number of samples required at a leaf node. Higher values prevent overfitting by avoiding splits that result in too small leaf nodes.
- **min_samples_split**: Indicates the minimum number of samples required to split an internal node. Higher values avoid overfitting by restricting splits that result in nodes with too few samples.
- **n_estimators**: Determines the number of trees in the forest. More trees usually increase performance but slow down computation.

The optimal values found are as follows:

- For the motor model: `max_depth: 20`, `min_samples_leaf: 1`, `min_samples_split: 2`, `n_estimators: 1`.
- For the home model: `max_depth: 40`, `min_samples_leaf: 1`, `min_samples_split: 5`, `n_estimators: 1`.

It is surprising to see that the best number of estimators is 1 as we usually expect that more estimators lead to higher performance of the model but in our case it seems to lead to overfitting, because of the particular nature of the data wich is very imbalance.

- **Feature selection**: Random Forest, as a tree-based model, can provide a measure of feature importance. The feature importance provided by Random Forest comes from the fact that the tree-based strategies used by Random Forest naturally ranks by how well they improve the purity of the node. Nodes with the greatest decrease in impurity happen at the start of the trees, while notes with the least decrease in impurity occur at the end of trees. Thus, by pruning trees below a particular node, we can create a subset of the most important features. Below Figure 3.13 and 3.14 shows the importance of each variables for predicting the clients that will make motor claims or home claims.

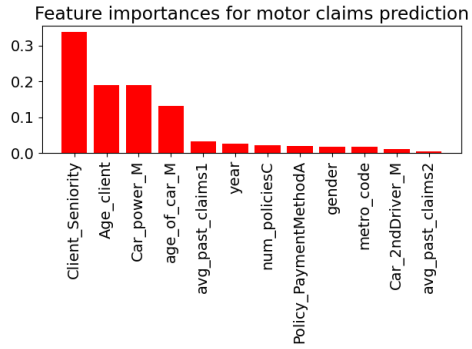


Fig. 3.5

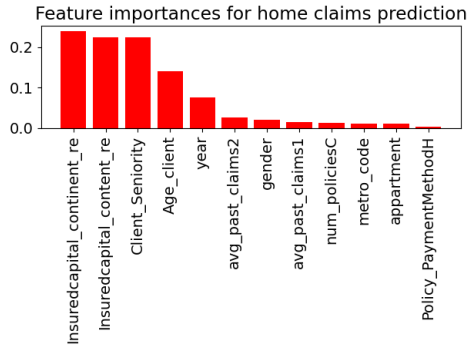


Fig. 3.6

Given this ranking, we removed, one by one, the least important variables and checked if the F-beta score was improving, decreasing or the same. As the number of variables is relatively small, we decided to remove variables only if this was resulting in a better F-beta score.

This methodology resulted in removing the following variables in the case of the motor claim prediction : "avg- past claims2" (which is the average number of past claims for the home policy), "Car 2ndDriver M", "gender", "Policy PaymentMethodA". Note that, by removing variables one by one, the ranking of importance changed, indeed as some variables are removed, other variables can become relatively more important, this is the case of "metro code".

For the home claims prediction, following the same process, we only removed the variable "Policy PaymentMethodH".

- **Model training**: Once the optimal set of features and hyperparameters are found, a RandomForestClassifier model is trained on the training data and used to predict the target variable on the test set.
- **Results**:

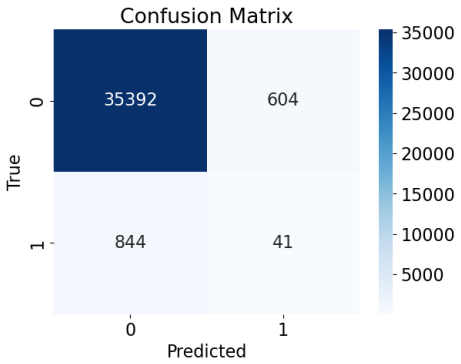


Fig. 3.7 Motor model

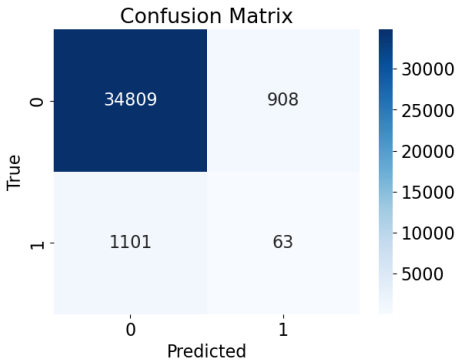


Fig. 3.8 Home model

Motor Model:

- **F-beta Score:** 0.096
- **Precision:** 0.044
- **Recall:** 0.611
- **Accuracy:** 0.669
- **AUC-ROC:** 0.640

Home Model:

- **F-beta Score:** 0.102
- **Precision:** 0.045
- **Recall:** 0.838
- **Accuracy:** 0.431
- **AUC-ROC:** 0.628

Once again, the results are inconclusive. However, it is crucial to consider the challenge of precisely predicting claims for an imbalanced dataset.

3.3.2 Gradient Boosting

As our target variable is binary we naturally chose to build a logistic XGBoost model. Considering the fact that we have a very imbalanced dataset, we used the parameter **scale_pos_weight** to balance the positive and negative weights, we set it to the ratio of the number of negative class to the positive class. Additionally we set the parameter **eval_metric** to F_β with $\beta = 1.2$ as it is our metric of interest.

- **Hyperparameter optimization**: The cross validation grid search is performed to find the optimal hyperparameters, we chose to optimize the following :
  - `learning_rate`: Often referred to as the "step size," this parameter scales the contribution of each tree when it is added to the current ensemble of trees. A smaller learning rate requires more boosting rounds, which makes the model training slower but can yield more refined models.
  - `max_depth`: Controls the maximum depth of any tree in the ensemble. Larger values can capture more complex interactions but also increase the risk of overfitting.
  - `subsample`: Specifies the fraction of the total instances that should be used to train each individual tree. By randomly selecting a subset for each tree, this can help prevent overfitting. Values typically range from 0.5 to 1.
  - `colsample_bytree`: Specifies the fraction of features (columns) to sample (with replacement) when constructing each tree. Reducing this value can add randomness and help prevent overfitting.
  - `n_estimators`: The total number of gradient boosted trees to construct. More trees can help model more complex patterns but also increase the risk of overfitting.
  - **Motor Model:** The optimal values found were:
    - `colsample_bytree`: 1.0,
    - `learning_rate`: 0.2,
    - `max_depth`: 7,
    - `n_estimators`: 200,
    - `subsample`: 0.7
  - **Home Model:** The optimal values found were:
    - `colsample_bytree`: 1.0,
    - `learning_rate`: 0.2,
    - `max_depth`: 7,
    - `n_estimators`: 200,
    - `subsample`: 0.7
- **Feature selection**: Similarly to Random Forests, XGBoost also provides us with a ranking of the most important features. We therefore used the same selection process as for Random Forests. Figures 3.13 and 3.14 shows the importance of each variables for predicting the clients that will make motor claims or home claims.

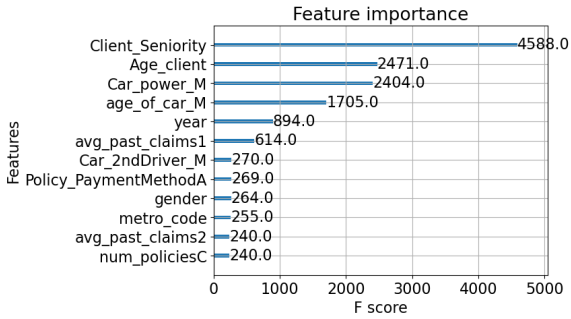


Fig. 3.9

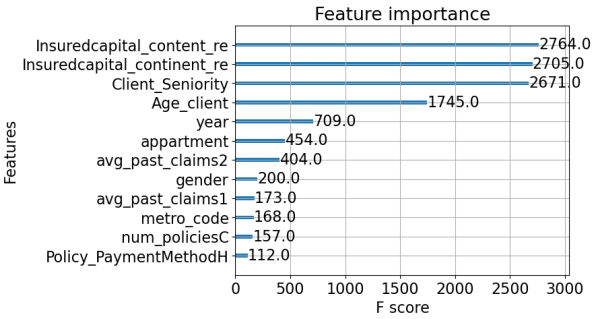


Fig. 3.10

This methodology resulted in removing "num_policiesC" and "avg_past_claims2" variables in the case of the motor claim prediction and "num_policiesC" and "Policy_PaymentMethodH" for the home claims prediction.

- **Model training**: Once the optimal set of features and hyperparameters are found, the XGBoost model is trained on the training data and used to predict the target variable on the test set.
- **Results**:

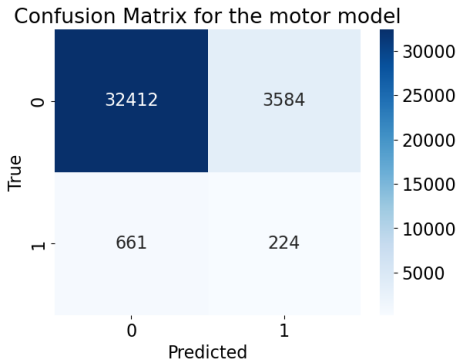


Fig. 3.11

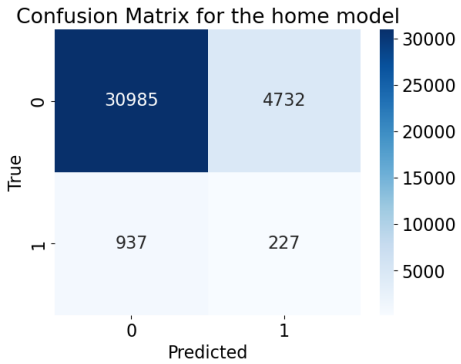


Fig. 3.12

Motor Model:

- **F-beta Score:** 0.10754
- **Precision:** 0.05882
- **Recall:** 0.2531
- **Accuracy:** 0.8849
- **AUC-ROC:** 0.57677

Home Model:

- **F-beta Score:** 0.08348
- **Precision:** 0.04578
- **Recall:** 0.1950
- **Accuracy:** 0.8463
- **AUC-ROC:** 0.5313

Our results indicate that the XGBoost model significantly outperforms the Random Forest model in terms of F-beta scores both for the home and the motor datasets. It's important to note, however, this enhanced predictive performance is achieved at the expense of increased computational time.

3.4 Neural Network

We will use grid search to find the best hyperparameters for our custom neural network model. The hyperparameters to be chose are the following:

- **Batch Size:** The number of samples used in each training iteration.
- **Number of Epochs:** The number of times the entire dataset is passed through the neural network during training.
- **Number of Units in Hidden Layers:** The number of neurons in each hidden layer.
- **Optimizer:** The algorithm used to update the model parameters during training.

We decide not to include the number of hidden layers, the loss function and the metric in this grid search. Indeed given the size of the dataset, two hidden layers is a reasonable choice. Moreover the loss function binary_crossentropy and the metric accuracy are logical choices as we have a binary target.

By conducting the grid search over these hyperparameters, we will be able to identify the optimal combination that results in the highest performance of our custom neural network model for the specific task at hand.

We get the following result:

- For the **motor model**:
  - Batch Size: 20

  - Number of Epochs: 100
  - Number of Units in Hidden Layers: 32
  - Optimizer: adam

- For the **home model**:
  - Batch Size: 70
  - Number of Epochs: 100
  - Number of Units in Hidden Layers:32
  - Optimizer: adam

After performing the gridsearch we are able to use our model. Contrarly to the other models presented in this paper, our Neural Network model, does not contain a feature importance function, so instead of this, we will perform a backward selection to get rid of the least important features for our model. After our backward selection, we got rid of the variables: "avg_past_claims2" and "metro_code" for the motor model and only of the variable: "Client_Seniority" for the home model.

Then as previously, we use a threshold to convert the probabilities into binary, in the same way as described in the GLM section. Finally we get the following results:

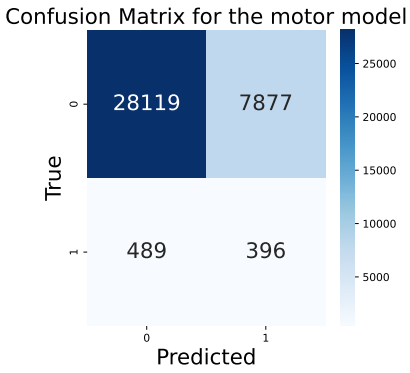


Fig. 3.13

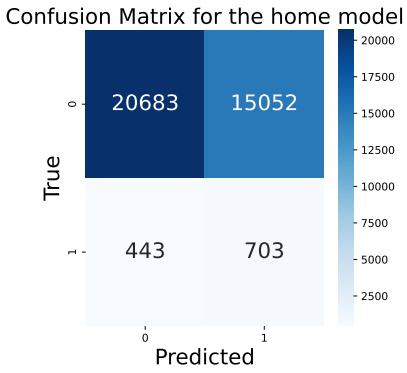


Fig. 3.14

Motor Model:

- **F-beta Score**: 0.105
- **Precision**: 0.0501
- **Recall**: 0.432

- **Accuracy**: 0.790
- **AUC-ROC**: 0.61

Home Model:

- **F-beta Score**: 0.0985
- **Precision**: 0.0446
- **Recall**: 0.613
- **Accuracy**: 0.579
- **AUC-ROC**: 0.59

3.5 Comparison of models

Motor Model				
Method	Accuracy	Precision	Recall	F-beta Score
Logistic Regression	0.610	0.039	**0.638**	0.086
GLM	0.610	0.039	**0.638**	0.087
XGB	**0.885**	**0.059**	0.253	**0.107**
Random Forest	0.669	0.044	0.611	0.096
Neural Network	0.790	0.0501	0.432	0.105

Home Model				
Method	Accuracy	Precision	Recall	F-beta Score
Logistic Regression	0.628	0.045	0.828	**0.102**
GLM	0.439	**0.049**	0.827	**0.102**
XGB	**0.846**	0.046	0.195	0.0835
Random Forest	0.431	0.045	**0.838**	**0.102**
Neural Network	0.579	0.0447	0.613	0.098

With these tables for the motor model and the home model, we can make the following observations:

Motor model :
The F-beta score, which balances precision and recall and is in our case the most important metric, is the best for XGBoost with a value of 0.107, suggesting that it offers a good compromise between precision and recall. The highest accuracy is achieved by the XGBoost (XGB) method, which achieves an accuracy of 0.885. It outperforms the other methods in terms of overall correct predictions. XGBoost also has the highest precision of all methods, indicating that its false positive rate is relatively low. Logistic regression and GLM have the highest recall (0.638), which means they better identify positive instances, but their precision is rather low.

Home model :
For the Home model, there are three models having the best F-beta score: Random Forest, Logistic Regression and GLM with 0.102. XGBoost again achieves the best precision with a value of 0.846. As with the motor model, it also scores well in terms of overall correct predictions for the home model. The highest accuracy for the domestic model is achieved by

the GLM method, with a value of 0.049. The accuracy of the GLM method is higher than that of the other methods, indicating that its false positive rate is relatively low. And finally, Random Forest has the highest recall of 0.838, suggesting that they are better able to correctly identify positive instances in the home model.

Overall, XGBoost performs well in both the auto and home models, for its accuracy and precision but has the worst recall of all the methods. Logistic regression, Random Forest and GLM also perform competitively, particularly in terms of recall, while Neural Network appear to be more balanced between accuracy, precision and recall, without outperforming the other methods.

Conclusion

In this paper, we have presented an analysis of claims prediction for an insurance company in Spain, using various models such as Generalized Linear Models, Logistic Regression, Random Forest, eXtreme Gradient Boosting, and Neural Networks.
To understand the relationships between variables we realized an Exploratory Data Analysis (EDA). We then performed a cross-validation grid search for each models to find the best hyperparameters. We then used backward selection to find the best subset of features.
In order to evaluate the performance of the models, we focused our comparisons on the F-Beta metric (with a beta of 1.2) to ensure the right balance between false positive and false negative. We also used other measures, such as accuracy, precision, recall, F-beta score and AUC-ROC, to understand the strengths and the drawbacks of each models.
Analyzing the results, we observed that XGB was the best in terms of F-beta score (and also for the precision and the accuracy) for the motor model, while it was Logistic Regression, GLM and Random Forest that had the best F-beta score for the home model. For the motor model, even though Logistic regression and GLM do not have the best F-beta score, they present the best recall. For the Home model, XGB while not having the best F-beta score, presents the best accuracy. Finally, the neural network doesn't have the best F-beta score for either model, but it does have the second best F-beta score and the second best scores for accuracy and precision for the motor models. On the other hand, it is one of the worst methods for the home model. It should be noted, however, that the choice of the best model is subjective and depends on the user's preferences, whether one favors precision, avoids false negatives or pursues other specific objectives.
In conclusion, this paper provides valuable insights into claims prediction in the insurance industry, offering an in-depth exploration of different models and their performance measures. The results underline the importance of tailoring predictive models to specific objectives, to enable decision-makers to make informed choices in risk assessment and policy planning.

References

[1] Chen, T., . G. C. (2016). *XGBoost: A Scalable Tree Boosting System*. ArXiv.

[2] Guillen, Montserrat; Bolancé, C. F. E. W. V. E. A. (2021). *Insurance data for homeowners and motor insurance customers monitored over five years*. Mendeley Data, V1, doi: 10.17632/vfchtm5y7j.1.

[3] Jordan, J. (2017). *Neural networks: representation*. https://www.jeremyjordan.me/intro-to-neural-networks.

[4] Paula Branco, L. T. and Ribeiro., R. P. (2016). *A survey of predictive modeling on imbalanced domains*. ACM Comput. Surv. 49, 2, Article 31 (August 2016), 50 pages.