

Medical Insurance Claim Prediction

Big Data & Databases Group 1 Project

Akhmetova Zhansaya – 3259293

Favata Martina – 3258750

Giambra Lorenzo – 3247686

Lutz Jakob – 3253325

Russo Martina – 3213229

Sultanov Ayan – 3257705



Table of contents



Introduction

Overview



Data Description

Univariate and Bivariate



Data Preparation

Missing values, outliers, data cleaning & feature engineering



Modelling

8 Regression and Classification Models



Outcomes

Results & Comparison of models

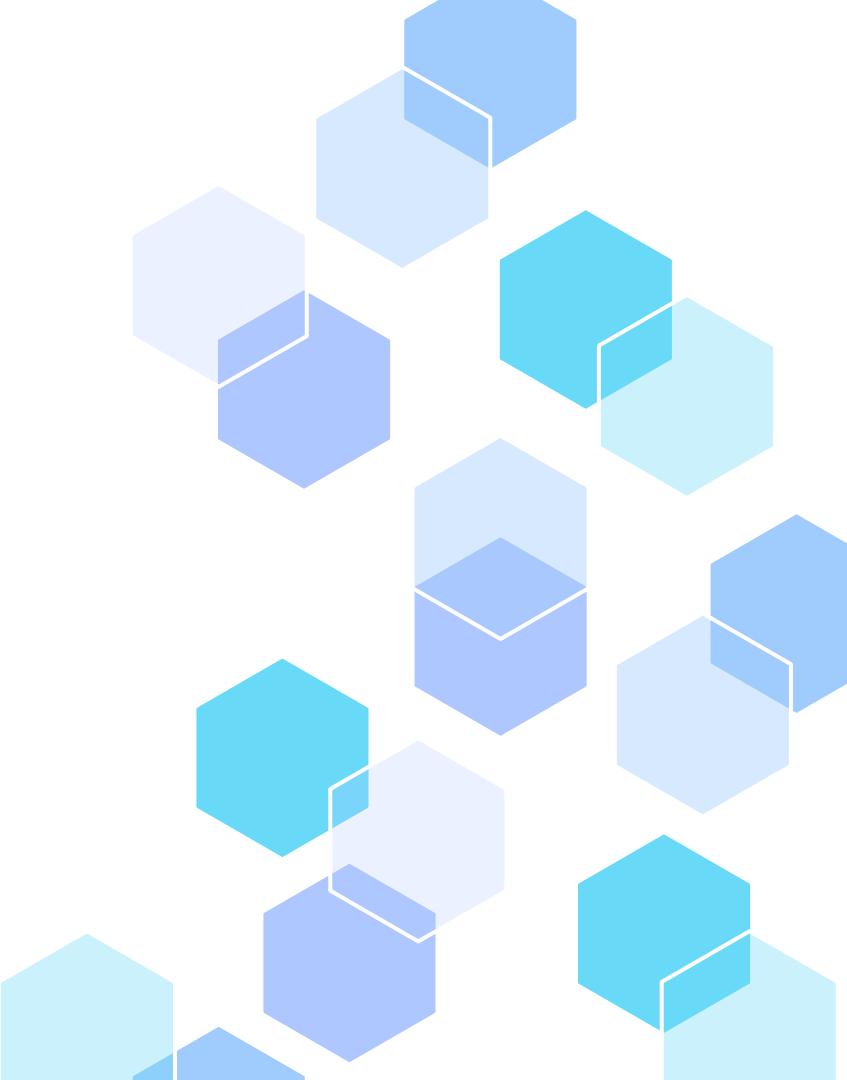


Managerial Implications

Conclusions

01

Introduction



INTRODUCTION

In this project we analyse the **“Medical Insurance Cost Prediction”** dataset from Kaggle to study claim behaviour within a health-insurance context.

The dataset includes **100000 observations** and **54 columns**, providing detailed information on policyholders, together with their annual claim outcomes.

Our goal is to identify factors linked to medical expenditure and assess whether claim outcomes can be reliably predicted.

These insights are relevant from an **insurance perspective**, as they support risk assessment, pricing strategies and the identification of policyholders who may require targeted interventions.

To guide this analysis, we focus on the following research questions:

- Can we predict what factors drive high claim frequency and amounts?
- Can we predict which policyholders will file claims?

Medical Insurance Cost Prediction

100K individuals with health, lifestyle, insurance, claims, and medical costs



To address these questions, we rely on two target variables:

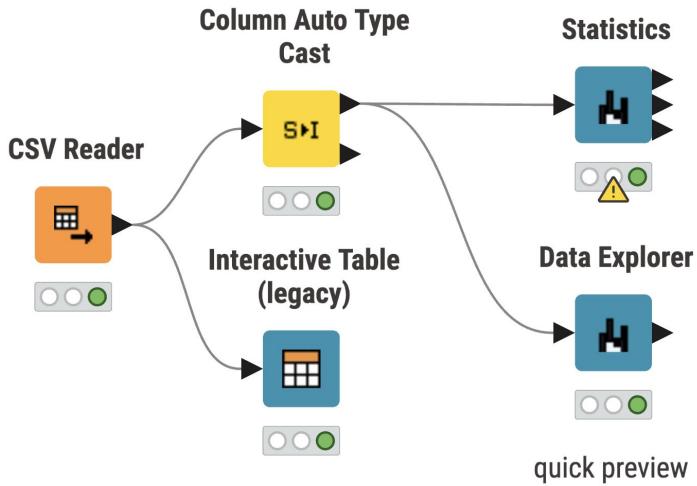
- **filed_claim**, indicating whether a policyholder submitted at least one claim;
- **total_claims_paid**, measuring the total amount reimbursed by the insurer over the year.

02

Data Description



INITIAL STEPS



CSV READER	COLUMN AUTO TYPE CAST	STATISTICS
We started here to load all the raw policy and claim data into our workspace.	We used automatically this clean up data types.	We ran this to generate the official descriptive statistics
DATA EXPLORER	INTERACTIVE TABLE (LEGACY)	
We used this for visual inspection		We performed a quick spot-check on the actual values after type conversion.

UNIVARIATE ANALYSIS

Total_claims_paid

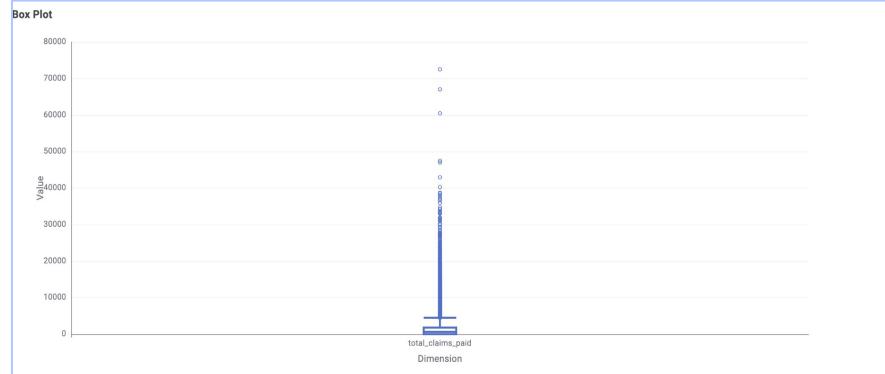
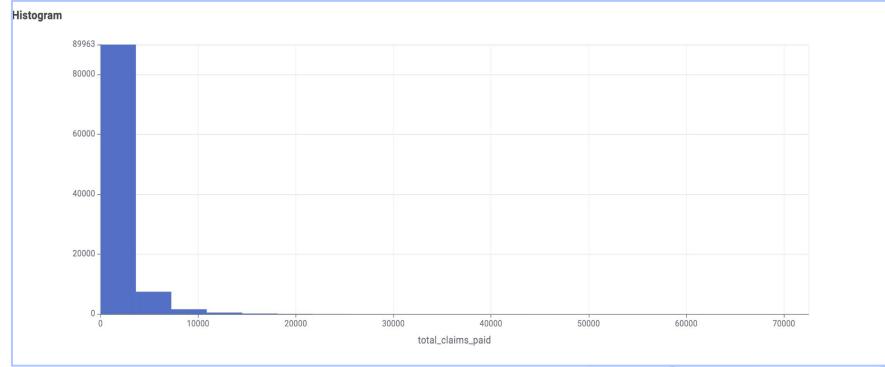
Target variable for regression

Nature: Numeric, continuous

Description: Total value of claims paid in USD.

Range: from 0 to 72517.95

Insights: Heavily right-skewed distribution, with most policyholders having zero or very low total claims paid and a small group generating very large costs. High variance and pronounced outliers are typical of insurance severity data. The mean is substantially higher than the median, confirming strong asymmetry and a heavy tail. No missing or infinite values.



Filed_claim

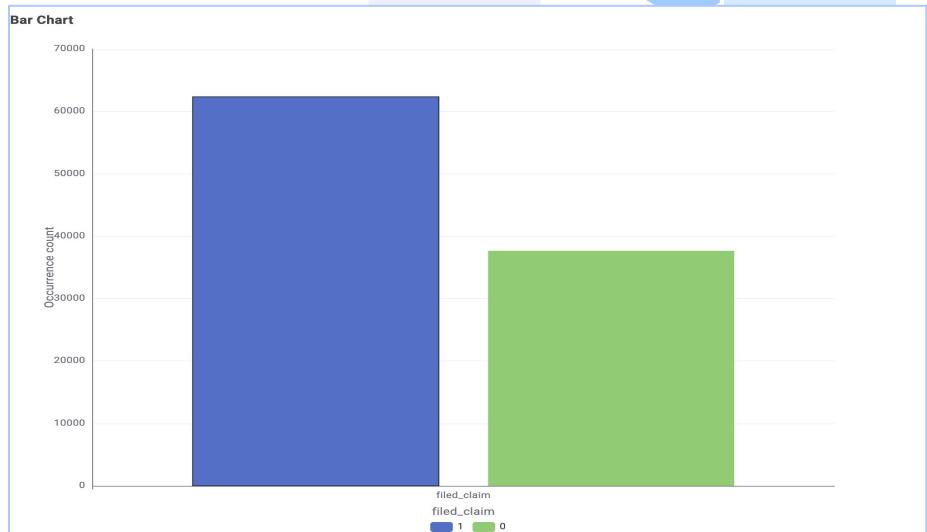
Target variable for classification

Nature: Categorical, binary

Description: is a binary indicator derived from *claims_count*, taking the value 1 when a policyholder submitted at least one claim, and 0 otherwise.

Categories: 0 (no filed claim) or 1 (filed claim)

Insights: Imbalanced class distribution, with around 62% of policyholders filing at least one claim and 38% filing none. This imbalance may influence model performance and requires appropriate evaluation metrics. The variable is clean, with no missing values, and provides a clear indicator of claim occurrence.



Age

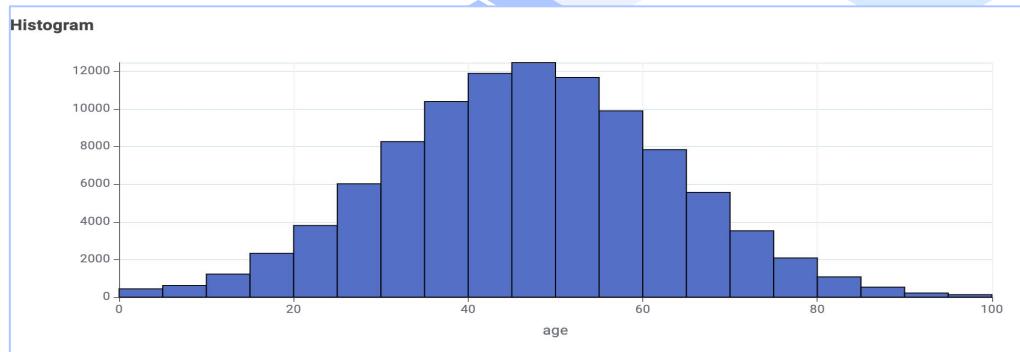
Target feature (predictor variable)

Nature: Numeric, discrete

Description: Age of the individual in years. Represents demographic risk; older individuals typically have higher medical needs and associated costs.

Range: from 0 to 100 years

Insights: The distribution is approximately normal, centered around a mean age of ~47.5 years. Low skewness, no missing or infinite values. Most policyholders are middle-aged individuals.



Sex

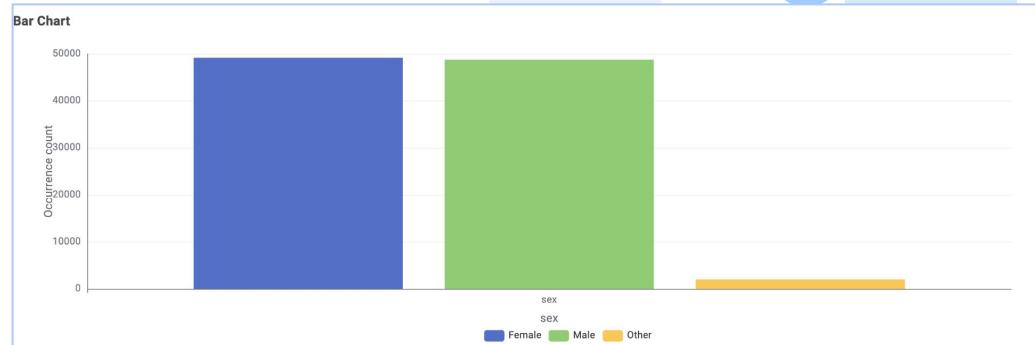
Target feature (predictor variable)

Nature: Categorical, nominal

Description: Gender of the individual

Categories: Female, Male, Other

Insights: The dataset is highly dominated by Female and Male, which together account for ~98% of all records. The *Other* category represents only ~2%, indicating a highly imbalanced categorical distribution. No missing values.



Income

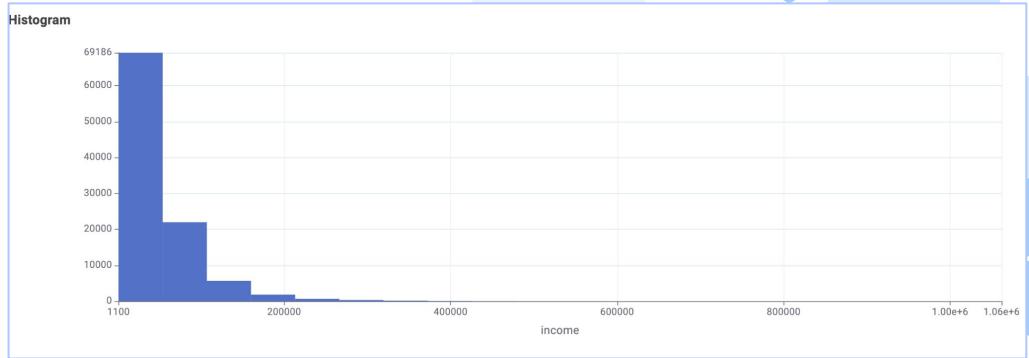
Target feature (predictor variable)

Nature: Numeric, continuous

Description: Annual income of the individual in USD.

Range: from 1100 to 1061800 USD

Insights: The distribution is heavily right-skewed, with the vast majority of individuals earning relatively low to moderate incomes and a small number of extremely high-income outliers. Mean income $\approx 49,874$ USD. High variance and extreme outliers reflect a wide socioeconomic spread in the dataset. No missing or infinite values.



BMI

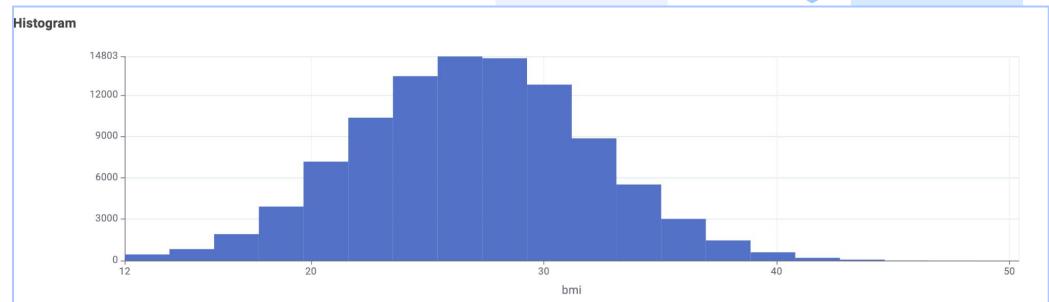
Target feature (predictor variable)

Nature: Numeric, continuous

Description: Body Mass Index (BMI), calculated as weight in kilograms divided by height in meters squared.

Range: from 12 to 50.4

Insights: The distribution is roughly symmetric and bell-shaped, centered around a mean BMI of ~27.0. Most individuals cluster between BMI 20–35, with few underweight or severely obese cases. No missing or infinite values.



Smoker

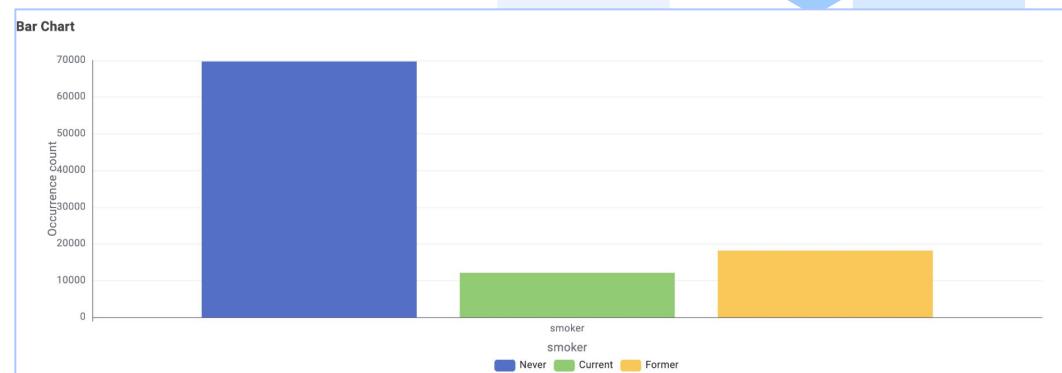
Target feature (predictor variable)

Nature: Categorical, ordinal

Description: Smoking status of the individual.

Categories: Never, Former, Current

Insights: Never-smokers are the dominant group, with smaller proportions for Current and Former smokers. No missing values. Smoking status is a high-impact health risk factor, commonly associated with cardiovascular disease, respiratory illness, and higher medical costs.



Systolic_bp

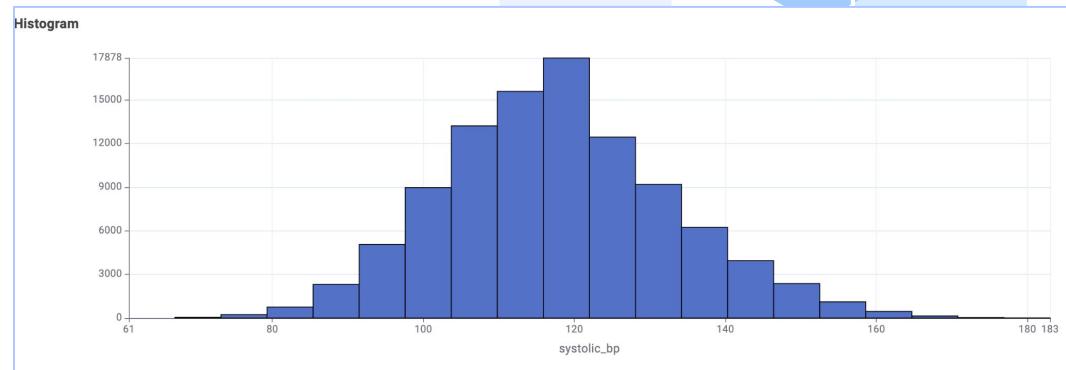
Target feature (predictor variable)

Nature: Numeric, continuous

Description: Systolic blood pressure measured in mmHg (millimeters of mercury).

Range: from 61 to 183 mmHg

Insights: The distribution is approximately normal, centered around a mean systolic BP of ~117.8 mmHg. Very low skewness and kurtosis indicate a symmetrical distribution with no heavy tails. The dataset includes both low and high-risk individuals. No missing or infinite values.



HbA1c

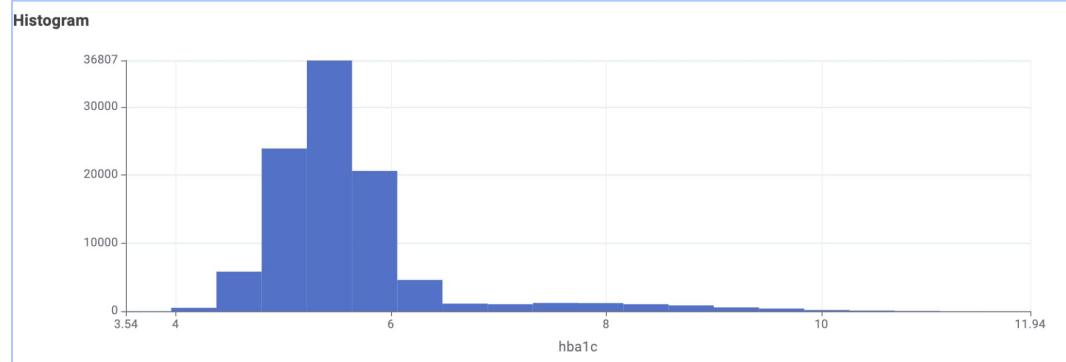
Target feature (predictor variable)

Nature: Numeric, continuous

Description: Represents the percentage of glycosylated hemoglobin in the blood.

Range: from 3.54 to 11.94

Insights: The distribution is heavily right-skewed, with the majority of values clustering in the 5.0% to 6.0% range, which suggests a predominantly healthy population. The small but clinically significant tail of values at 7% or higher reflects individuals in prediabetic or diabetic ranges. Low overall variation suggests most individuals have similar glucose regulation, with a few meaningful outliers contributing to the skewness.



Ldl

Target feature (predictor variable)

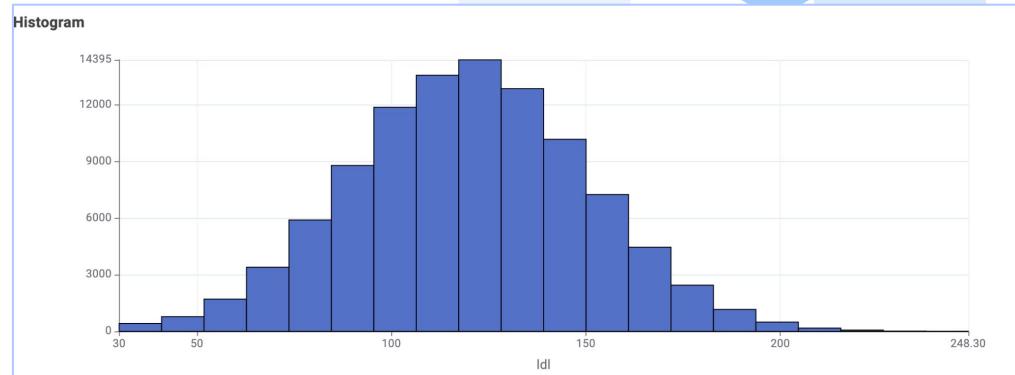
Nature: Numeric, continuous

Description: Standard lipid profile indicator.

Higher values increase cardiovascular risk, whereas lower values are generally healthier.

Range: from 30 to 248.3 mg/dL

Insights: LDL levels in the dataset are nearly normally distributed, indicated by almost zero skewness and kurtosis. Most observations fall between roughly 95–140 mg/dL, representing near-optimal to borderline-high cholesterol levels. A smaller number of individuals show very low or very high LDL values, but still without strong tail behavior.



Chronic_count

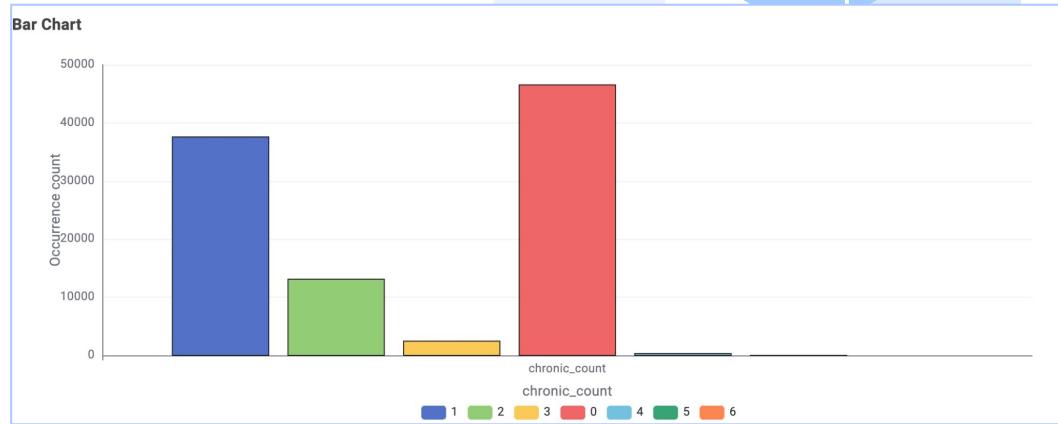
Target feature (predictor variable)

Nature: Numeric, discrete

Description: Represents how many chronic medical conditions each individual has.

Ranges: from 0 to 6

Insights: The number of chronic conditions is heavily concentrated at the lowest values, with 0 and 1 conditions making up the overwhelming majority of the population. A much smaller group has 2 or 3 conditions, and only a tiny fraction has 4–6 conditions. This pattern shows a strong imbalance where most individuals are relatively healthy, while only a small subgroup has multiple chronic illnesses.



Hypertension

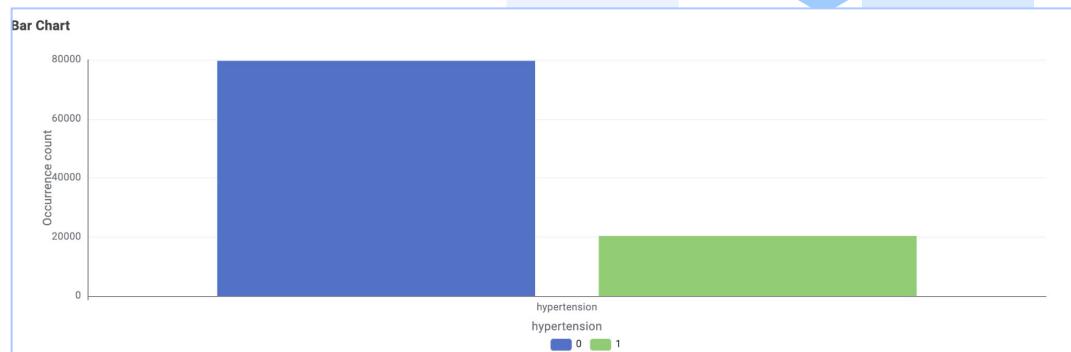
Target feature (predictor variable)

Nature: Categorical, binary

Description: Indicates whether the individual has a diagnosed case of hypertension.

Categories: 0 (no hypertension) or 1 (hypertension diagnosed)

Insights: The variable is highly imbalanced, with approximately 79.7% of individuals not having hypertension and 20.3% having a diagnosis. This imbalance is typical in population-level health datasets. No missing or infinite values.



Diabetes

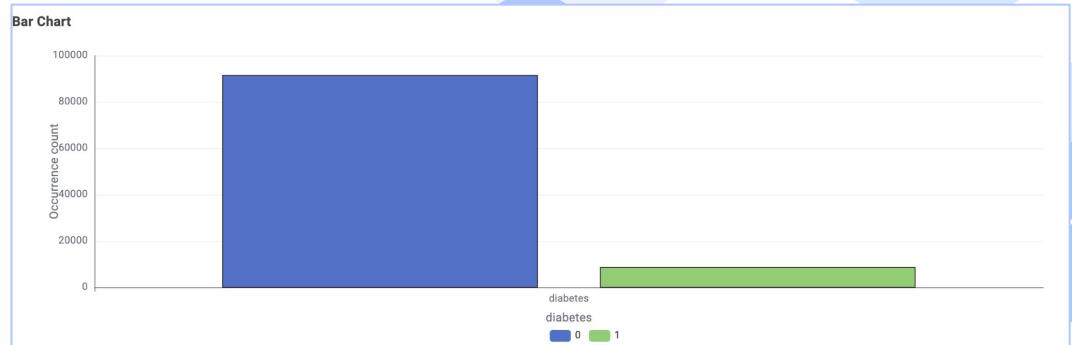
Target feature (predictor variable)

Nature: Categorical, binary

Description: Indicates whether the individual has been diagnosed with diabetes.

Categories: 0 (no diabetes) or 1 (diabetes diagnosed)

Insights: This variable is highly imbalanced, with only 8.6% of individuals diagnosed with diabetes. Diabetes is a major chronic condition and is strongly associated with increased healthcare utilization, complications, and long-term medical costs. No missing or infinite values.



Cardiovascular_disease

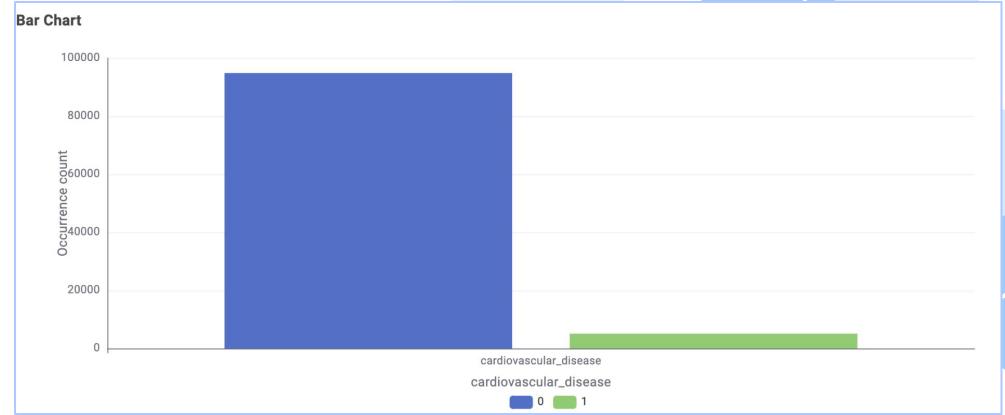
Target feature (predictor variable)

Nature: Categorical, binary

Description: Indicates whether the individual has been diagnosed with cardiovascular disease.

Categories: 0 (no cardiovascular disease) or 1 (cardiovascular disease present)

Insights: The variable is heavily imbalanced, with only ~5.1% of individuals reporting cardiovascular disease. Cardiovascular disease is a major predictor of healthcare utilization, long-term cost, and likelihood of claims, its presence typically indicates high-risk individuals. No missing or infinite values.



Mental_health

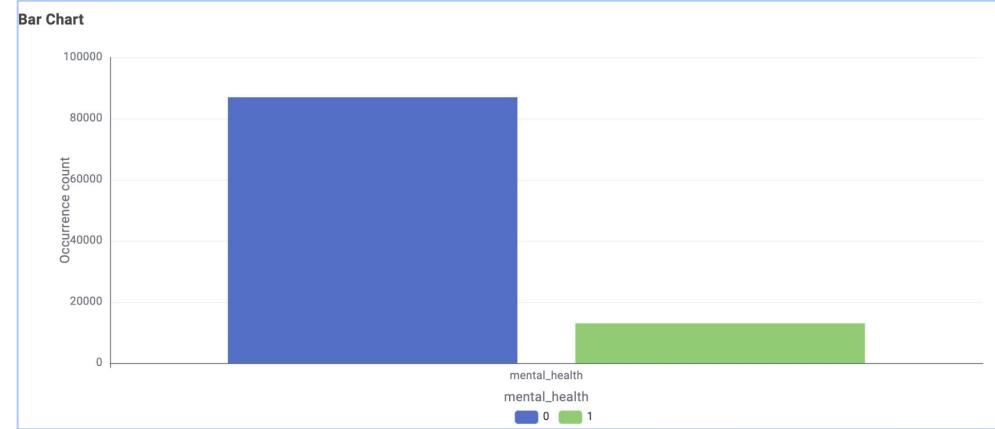
Target feature (predictor variable)

Nature: Categorical, binary

Description: Indicates whether the individual has a diagnosed mental health condition.

Categories: 0 (no diagnosed mental health condition) or 1 (mental health condition present)

Insights: The variable is imbalanced, with approximately 13% of individuals reporting a mental health condition and 87% without. Mental health conditions are strongly associated with increased medical visits, medication usage, and potential long-term costs, making this a potentially influential predictor in claim modeling.



Visits_last_year

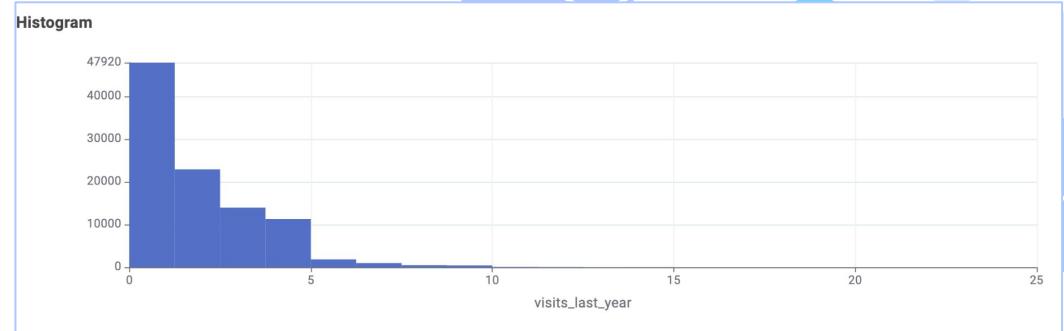
Target feature (predictor variable)

Nature: Numeric, discrete

Description: Number of healthcare provider visits the individual made during the last calendar year.

Ranges: from 0 to 25

Insights: The distribution is heavily right-skewed, with most individuals having 0 to 3 visits in the past year. A few individuals have very high utilization (up to 25 visits). Mean ≈ 1.93 visits indicates that the typical patient uses healthcare services infrequently. Around 19,865 people reported zero visits, while the majority of the remainder cluster in the 1–5 range. No missing or infinite values.



Days_hospitalized_last_3yrs

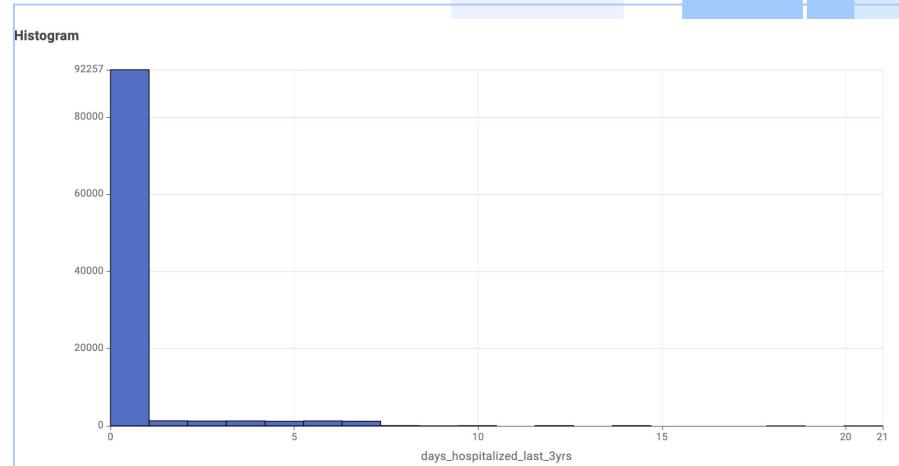
Target feature (predictor variable)

Nature: Numeric, discrete

Description: Total number of days the individual spent in the hospital within the past three years.

Ranges: from 0 to 21 days

Insights: The distribution is extremely right-skewed, with the overwhelming majority of individuals reporting zero hospital days in the past three years. A small subset of individuals had between 1 and 8 days, and very few had more than 10 days, with the maximum being 21. Mean = 0.37 days, reflecting very low typical hospitalization utilization. No missing or infinite values.



COPAY

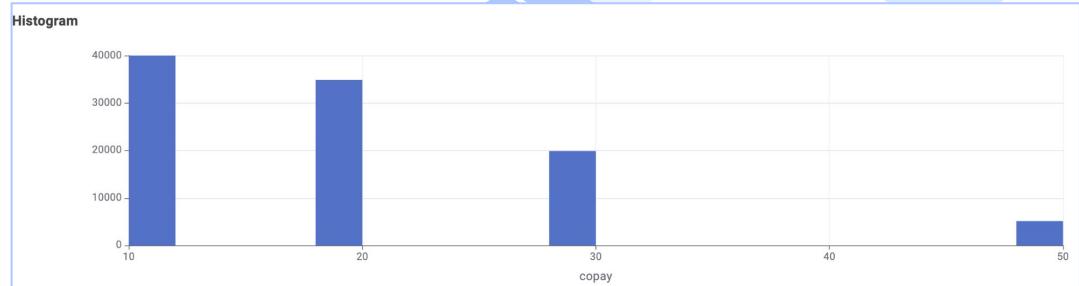
Target feature (predictor variable)

Nature: Numeric, discrete

Description: Copayment amount per healthcare service

Ranges: from 10 to 50 USD

Insights: The distribution is right-skewed, with most individuals paying lower copay amounts (10–22 USD) and fewer individuals at the higher end (34–50 USD). The mean copay is around 19.5 USD, reflecting a moderate average cost-sharing level. No missing or invalid values are present.



Other variables

network_tier

Coverage tier of the insurance network; Silver is the most common tier.

diastolic_bp

Diastolic blood pressure; values cluster around the "normal" range.

deductible

Annual deductible in USD, with many policies at lower deductible levels.

policy_changes_last_2yrs

Policy modifications in the last two years, which are rare.

household_size

Number of people in the household; small households dominate the dataset.

medication_count

Number of active medications; many policyholders take no or few.

copay

Copayment per service in USD; most copays are modest.

provider_quality

Quality rating of the provider; ratings cluster around moderate to high.

dependents

Number of financial dependents; most individuals have few or none.

hospitalizations_last_3yrs:

Hospitalizations over the past three years, which are very rare events.

policy_term_years

Length of the policy in years, balanced around mid-length contracts.

risk_score

Normalized health risk score, centered slightly above 0.5.

Other variables

annual_medical_cost

Total annual medical cost; distribution is right-skewed with few very high expenditures.

claims_count

Number of claims filed in a year; most individuals file zero or one.

copd

Indicator for Chronic Obstructive Pulmonary Disease; prevalence is relatively rare (~4%).

liver_disease

Indicator for liver disease, which has a similarly low prevalence.

annual_premium

Annual insurance premium, right-skewed, higher for higher-risk plans.

avg_claim_amount

Average amount per claim; typically modest, but a few large claims increase the average

cancer_history

Indicator for past cancer diagnosis; a very small group (~2%) has this history.

proc_imaging_count

Number of imaging procedures in the year; most members undergo none.

monthly_premium

Monthly premium; mirrors the annual premium distribution.

asthma

Indicator for asthma; about 6% of the sample has the condition.

kidney_disease

Indicator for chronic kidney disease; it is rare in the dataset (~1.5%).

proc_surgery_count

Number of surgeries in the year, which are infrequent events.

Other variables

proc_physio_count

Number of physiotherapy sessions; a minority engages in repeated sessions.

is_high_risk

High-risk flag; around 37% of the population is labeled high-risk.

urban_rural

Type of area where the individual lives; most insureds live in Urban areas.

employment_status

Current employment situation; over half of the sample is Employed.

proc_consult_count

Number of specialist consultations; many have no specialist visits.

had_major_procedure

Indicator for a major procedure; roughly 17% experienced a major event.

education

Highest level of education attained, concentrated in moderate-to-high levels.

alcohol_freq

Frequency of alcohol consumption; occasional drinkers are the largest group.

proc_lab_count

Number of laboratory tests, with moderate testing overall.

region

Geographic region of residence; the South is the most represented region.

marital_status

Marital status; married individuals form the largest group.

plan_type

Type of health insurance plan; PPO and HMO plans dominate the portfolio.

03

Data preparation

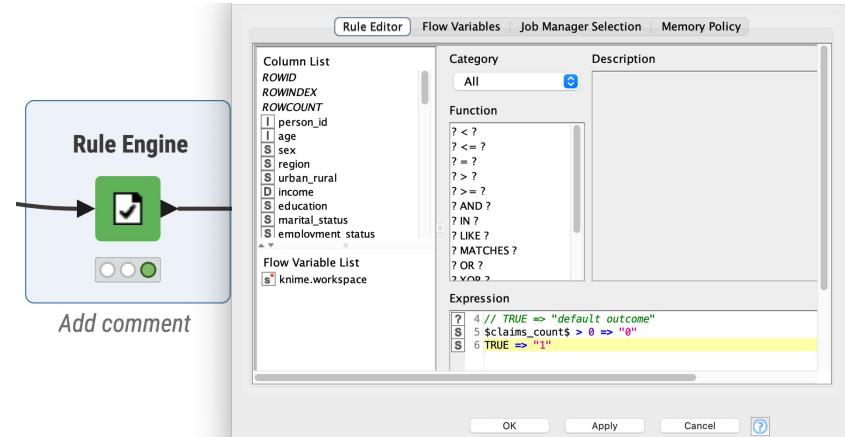


FEATURE ENGINEERING

To prepare our data for modeling, we began with a key feature-engineering step: creating one of our target variables using a **Rule Engine** node.

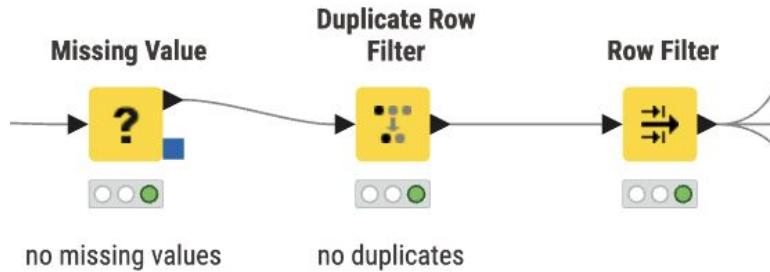
Specifically, we generated a binary indicator that flags whether a policyholder filed at least one claim.

This new variable is essential for training our predictive models, as it allows us to distinguish between claimants and non-claimants in a supervised learning setting.



INITIAL DATA CHECKS

After defining the new target variable, we carried out a Data Quality Assurance step to make sure the dataset was accurate and reliable.



We used diagnostic tools to check whether the data needed cleaning, and we found that the dataset was already in very good condition. A **Missing Value** node confirmed that there were no empty fields in

any variable, and a **Duplicate Row Filter** showed that each policyholder appeared only once in the data.

We then checked the main numerical variables to make sure there were no strange or impossible values that could distort the analysis.

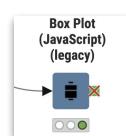
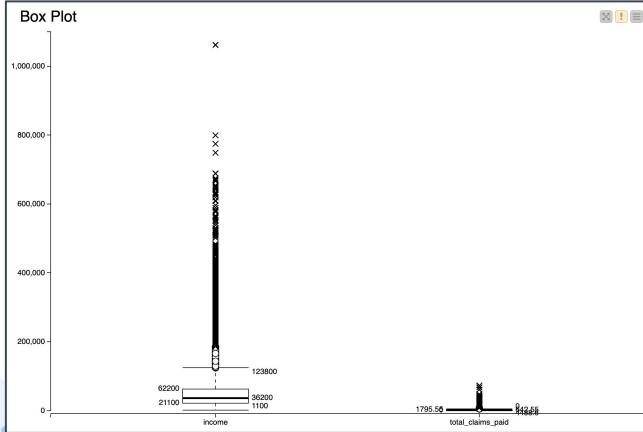
This included verifying that measures such as age, BMI, and blood pressure stayed within realistic and medically reasonable ranges. All observations passed these checks, so no rows needed to be removed.

Thanks to this validation step, we ended up with a dataset that is clean, consistent, and ready for analysis. This gives us a solid and reliable base for the next steps, including correlation analysis, feature selection, and predictive modeling.

OUTLIERS

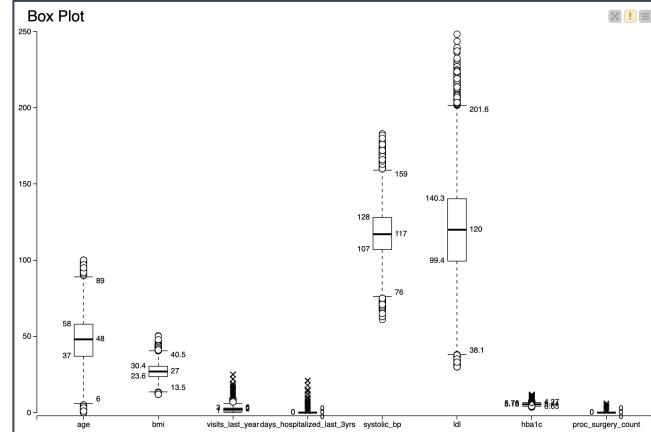
We performed outlier detection only on the most relevant continuous variables, as many numerical features in the dataset are binary or have very limited ranges, making boxplots uninformative for them.

Using the **Box Plot** node, we generated boxplots and grouped variables by scale to keep the visualizations interpretable. *Income* and *total_claims_paid* were plotted separately due to their much larger ranges and naturally heavy right-skewness.



The high-value outliers seen in these variables are expected in insurance data and represent real high-cost cases, so we retained them.

For the remaining continuous variables (age, BMI, blood pressure, LDL, HbA1c, and healthcare utilization counts), the **Box Plot** node showed moderate spread with only a small number of natural outliers. Since these did not indicate data quality issues, we kept all observations to preserve the full variability necessary for accurate modeling.



OUTLIERS

We used the **Numeric Outliers** node to quantify outliers across selected numerical variables using the standard Tukey criterion ($1.5 \times \text{IQR}$). The results confirm the patterns observed in the boxplots: variables such as *income* and *total_claims_paid* exhibit a substantial number of high-value outliers due to their naturally heavy-tailed distributions.

Utilization-related variables (e.g., *visits_last_year*, *days_hospitalized_last_3yrs*) also show higher outlier counts, reflecting rare but legitimate high-use cases.

Clinical measures, including *BMI*, *systolic_bp*, and *LDL*, show relatively few outliers, indicating stable distributions. Binary variables were included but correctly reported zero outliers.

Since all detected outliers correspond to realistic and meaningful observations in a medical insurance context, we retained them to preserve predictive value.

The screenshot shows a data mining interface with a sidebar labeled 'Numeric Outliers' containing a yellow icon of a box plot with three colored dots below it. A callout arrow points from this icon to the table below. The main area displays a table with the following columns: #, RowID, Outlier column, Member count, Outlier count, Lower bound, and Upper bound. The table rows represent the analysis for various variables:

#	RowID	Outlier column	Member count	Outlier count	Lower bound	Upper bound
	Row1	7 selected	Member count	Outlier count	Lower bound	Upper bound
2	Row1	income	100000	6236	-40,550	123,850
4	Row3	bmi	100000	676	13.4	40.6
5	Row4	visits_last_year	100000	2083	-2	6
6	Row5	days_hospitalized_last_3yrs	100000	8969	0	0
8	Row7	systolic_bp	100000	613	75.5	159.5
9	Row8	ldl	100000	668	38.05	201.65
17	Row16	total_claims_paid	100000	6963	-2,693.351	4,488.919

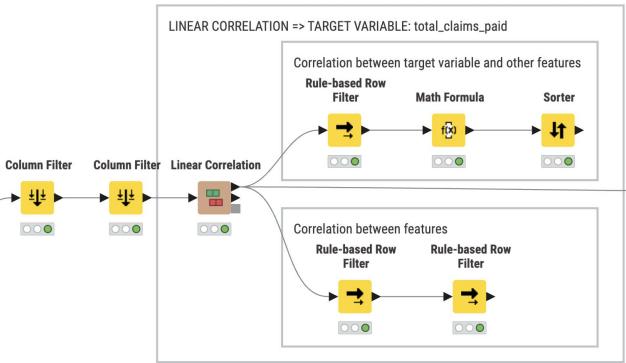
BIVARIATE ANALYSIS

LINEAR CORRELATION

To understand which variables are most closely associated with our target, *total_claims_paid*, we first computed **Pearson** correlations between the target and all numeric features. After filtering out categorical and ID variables, we generated the correlation matrix.

From there, we focused specifically on the correlations involving the target. A **Rule-based Row Filter** allowed us to keep only the rows where one of the columns is *total_claims_paid*, isolating its direct relationships with each predictor.

We then computed the absolute value of the coefficients and sorted them, obtaining a clear and ordered ranking of the features most strongly associated with the target.



Expression

```
1 // enter ordered set of rules, e.g.:
2 // $double column name$ > 5.0 => FALSE
3 // $string column name$ LIKE "*blue*/*" => FALSE
4 // TRUE => TRUE
5 $first column name$ = "total_claims_paid" => TRUE
6 $Second column name$ = "total_claims_paid" => TRUE
7 TRUE => FALSE
```

Include TRUE matches Exclude TRUE matches

Expression

```
1 abs($Correlation value$)
```

RESULTS & DATA LEAKAGE

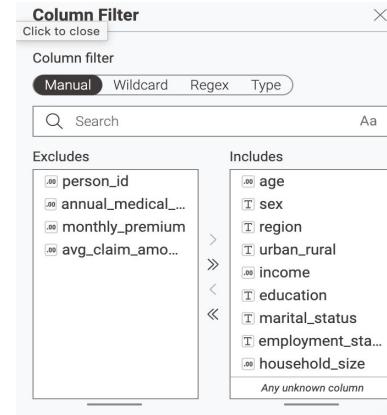
#	RowID	First column name T String	Second column name T String	Correlation value ∞ Number (Float)
1	Row63	annual_medical_cost	total_claims_paid	0.739
2	Row71	avg_claim_amount	total_claims_paid	0.737
3	Row67	monthly_premium	total_claims_paid	0.714
4	Row65	annual_premium	total_claims_paid	0.714
5	Row69	claims_count	total_claims_paid	0.511
6	Row21	visits_last_year	total_claims_paid	0.442
7	Row73	total_claims_paid	chronic_count	0.366
8	Row60	risk_score	total_claims_paid	0.306
9	Row74	total_claims_paid	is_high_risk	0.256
10	Row73	total_claims_paid	hypertension	0.186

This introduces a major issue: **data leakage**. Some variables contain information that would not be available at prediction time or are mathematically linked to the target.

Keeping them would allow the model to infer the final claim amount, inflating performance and preventing the model from generalising to new data.

Looking at the correlation table, we see that several variables have very strong linear associations with *total_claims_paid*. In particular, annual medical cost, average claim amount, and both premium values exceed a correlation of 0.70.

Such high levels indicate that these features are not independent predictors but instead reflect the target outcome itself.

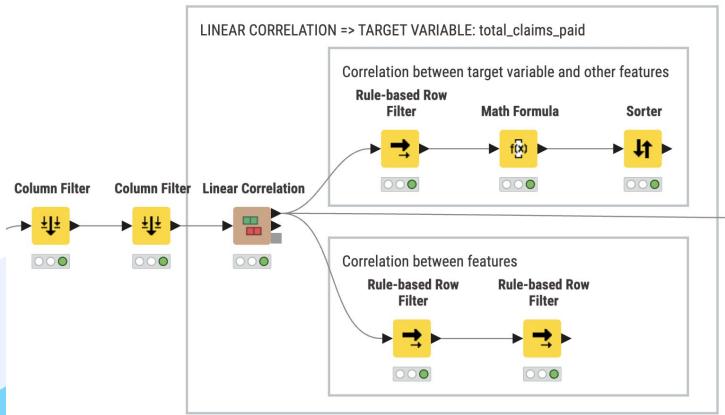


To avoid this, we removed all leakage-prone features, those that mirror medical spending, are derived from the target, or embed post-claim information, as well as identifiers like *person_id*, which add no predictive value.

This ensures the model is trained only on information realistically available before claims are processed.

LINEAR CORRELATION

To detect multicollinearity, we examined the pairwise correlations between all numerical features using the Linear Correlation node. The resulting table contains correlations for every possible feature pair, so we applied two **Rule-based Row Filters** to extract only the information we needed.



The first filter removed any rows where either column corresponded to *total_claims_paid*, ensuring that only feature-feature correlations remained.

The second filter kept only those pairs with a strong absolute correlation ($|r| > 0.7$), allowing us to quickly identify redundant predictors that contribute to multicollinearity.

Expression

```
1 // enter ordered set of rules, e.g.:  
2 // $double column name$ > 5.0 => FALSE  
3 // $string column name$ LIKE "*blue*" => FALSE  
4 // TRUE => TRUE  
5 $First column name$ = "total_claims_paid" => FALSE  
6 $Second column name$ = "total_claims_paid" => FALSE  
7 TRUE => TRUE
```

Expression

```
1 // enter ordered set of rules, e.g.:  
2 // $double column name$ > 5.0 => FALSE  
3 // $string column name$ LIKE "*blue*" => FALSE  
4 // TRUE => TRUE  
5 $Correlation value$ > 0.7 => TRUE  
6 TRUE => FALSE
```

RESULTS & MULTICOLLINEARITY

The screenshot shows a 'Column Filter' dialog box. At the top, there are tabs for 'Manual', 'Wildcard', 'Regex', and 'Type'. Below is a search bar with placeholder 'Search' and a case sensitivity switch 'Aa'. The 'Excludes' section contains a list of variables: dependents, hospitalizations..., diastolic_bp, hba1c, annual_premium, proc_surgery_co..., and is_high_risk. The 'Includes' section lists variables: age, sex, region, urban_rural, income, education, marital_status, employment_sta..., household_size, and Any unknown column.

To reduce multicollinearity, we removed variables that showed very **high** correlations with other predictors. Such features add redundant information and can distort model coefficients and interpretation. Eliminating them ensures that the remaining predictors contribute unique and independent signals.

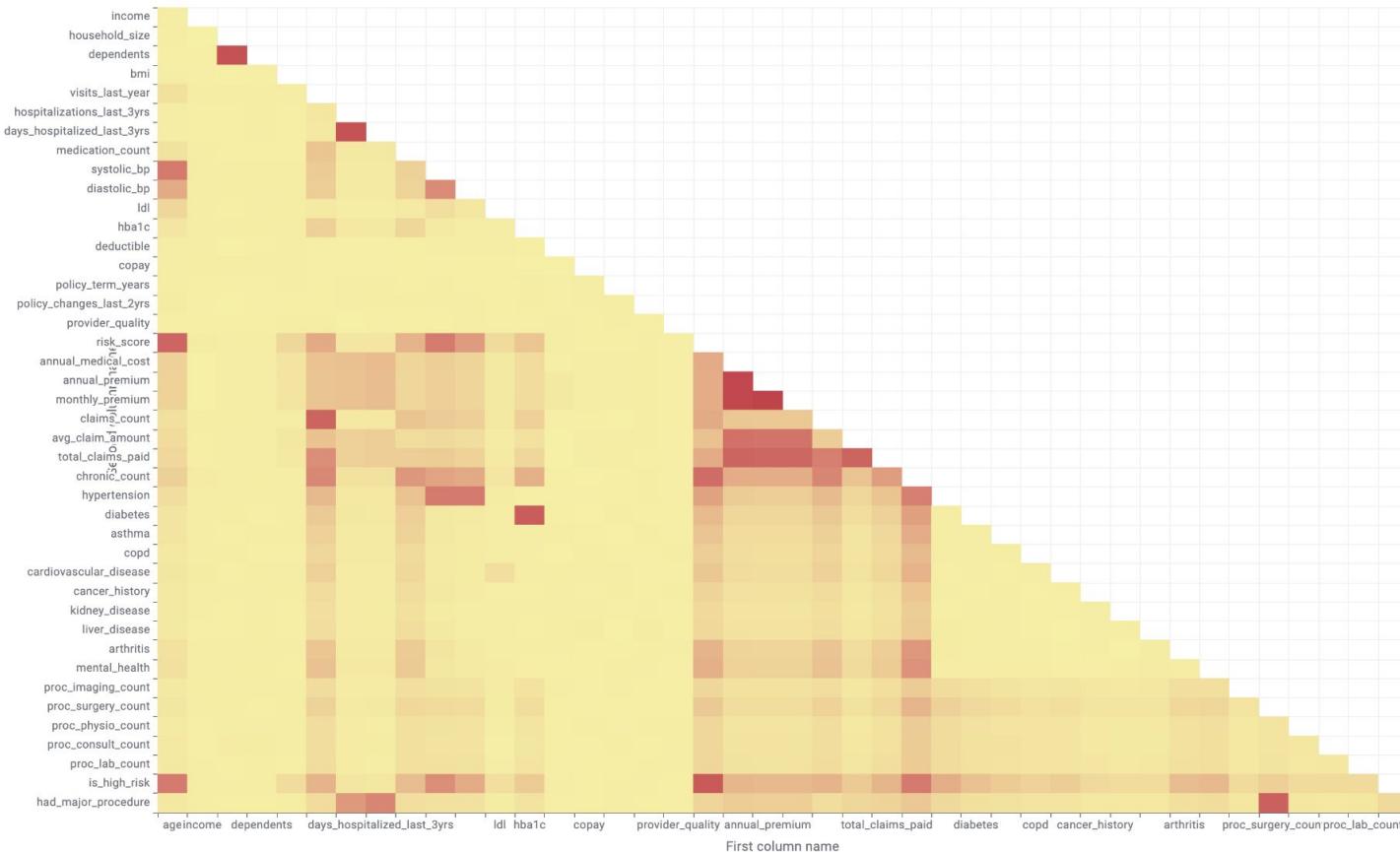
#	RowID	First column name T String	Second column name T String	Correlation value .00 Number (Float)
1	Row17	age	risk_score	0.721
2	Row83	household_size	dependents	0.886
3	Row21	visits_last_year	claims_count	0.74
4	Row23	hospitalizations_last_3yrs	days_hospitalized_last_3yrs	0.886
5	Row45	hba1c	diabetes	0.791
6	Row62	risk_score	is_high_risk	0.821
7	Row62	annual_medical_cost	annual_premium	0.965
8	Row62	annual_medical_cost	monthly_premium	0.965
9	Row65	annual_premium	monthly_premium	1
10	Row89	proc_surgery_count	had_major_procedure	0.757

diastolic_bp was excluded because it offered minimal predictive value and largely duplicated the information already captured by systolic_bp, adding noise rather than insight.

As previously shown, we also examined each feature's correlation with the target variable. Among those with high multicollinearity, features showing very low correlation with the target contributed limited explanatory power and were therefore **removed** to streamline the dataset. By retaining only variables with meaningful relationships to the outcome, we improved model efficiency while ensuring that the remaining predictors provide stronger and more relevant signals for modeling.

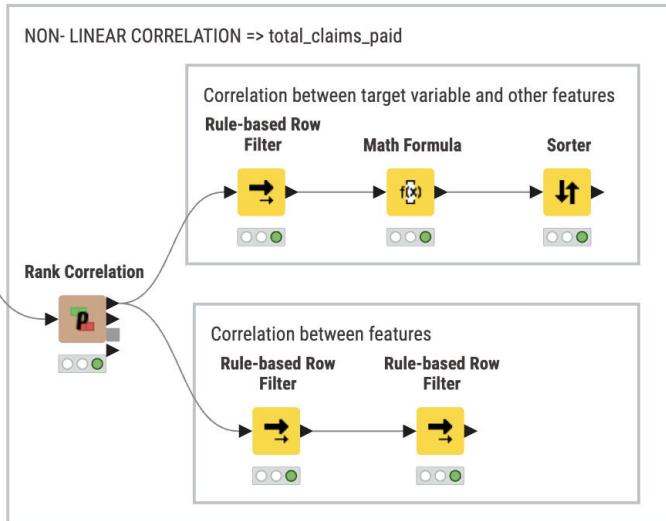
We generated a correlation heatmap to visually summarize how the numerical features relate to one another.

Heatmap



NON-LINEAR CORRELATION

We repeated the analysis using **Spearman rank correlation** to capture monotonic but non-linear relationships with *total_claims_paid*. After isolating the target rows and ranking the absolute coefficients, we found that non-linear patterns remain limited.



The strongest associations come from:

#	RowID	First column name [T] String	Second column name [T] String	Correlation value .00 Number (Float)
1	Row73	avg_claim_amount	total_claims_paid	0.93
2	Row77	total_claims_paid	filed_claim	-0.863
3	Row71	claims_count	total_claims_paid	0.842
4	Row22	visits_last_year	total_claims_paid	0.625
5	Row65	annual_medical_cost	total_claims_paid	0.484
6	Row69	monthly_premium	total_claims_paid	0.472
7	Row67	annual_premium	total_claims_paid	0.472
8	Row75	total_claims_paid	chronic_count	0.409
9	Row62	risk_score	total_claims_paid	0.316
10	Row77	total_claims_paid	is_high_risk	0.267

Once leakage variables are removed, most remaining features show **weak non-linear influence**, confirming that claim amounts are not strongly driven by monotonic patterns in the cleaned dataset.

DATA CLEANING

DATA LEAKAGE

To further prevent data leakage, we removed several variables that could reveal information about the target. Although **visits_last_year** initially appeared to be a harmless historical feature, further inspection showed that it was highly correlated with claim behavior and indirectly leaked target information, so it was excluded from the models.

We also removed **claims_count** for both regression and classification, since it directly determines whether a claim was filed. In the regression task, **filed_claim** was removed because it is derived from claims_count and would leak target structure. For the classification task, **total_claims_paid** was removed for the same reason, as any positive payment implies a claim was filed.

All leakage-prone variables were filtered out prior to modeling (except for PCA since it is unsupervised) to ensure that predictions rely only on information available at the time of decision-making and to maintain the validity of our evaluation.

Excludes

- visits_last_year
- claims_count
- filed_claim

Regression models

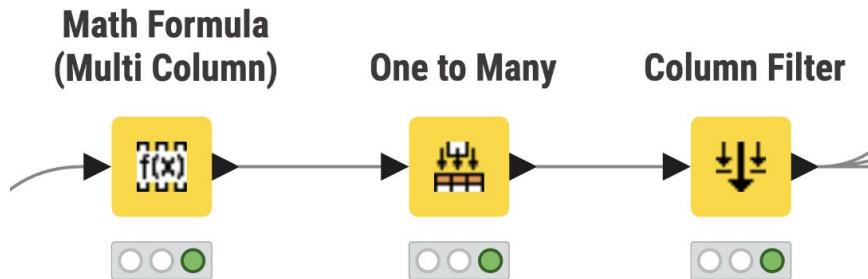
Excludes

- visits_last_year
- claims_count
- total_claims_paid

Classification models

DATA MANIPULATION

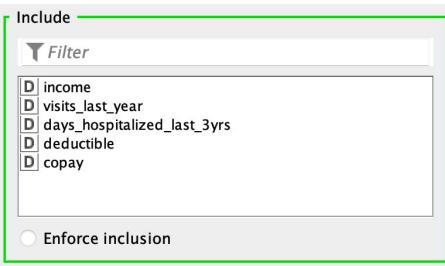
(Preparing data for PCA, Linear regression, GLM & Logistic Regression)



FEATURE ENGINEERING

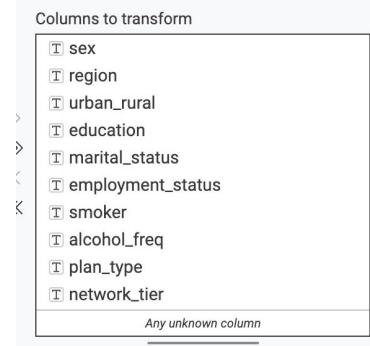
Our initial data manipulation involved using the **Math Formula** node to apply a Logarithmic Transformation to high-variance, continuous inputs like **Income** and **Days Hospitalized**.

This was necessary because their heavily skewed distributions and extreme outliers could destabilize model learning, which directly improves the accuracy and reliability of their splits and final predictions.



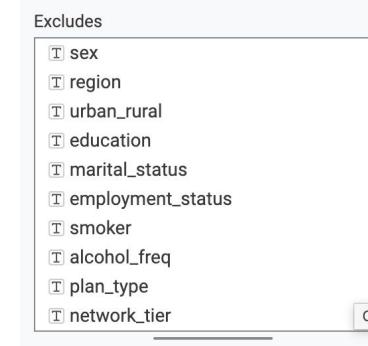
Next, we converted all nominal features into a numeric format essential for the ensemble models using the **One to Many** node to perform **One-Hot Encoding**.

This process converts attributes into distinct binary indicator columns, ensuring that every specific category is explicitly and equally evaluated by the algorithms.



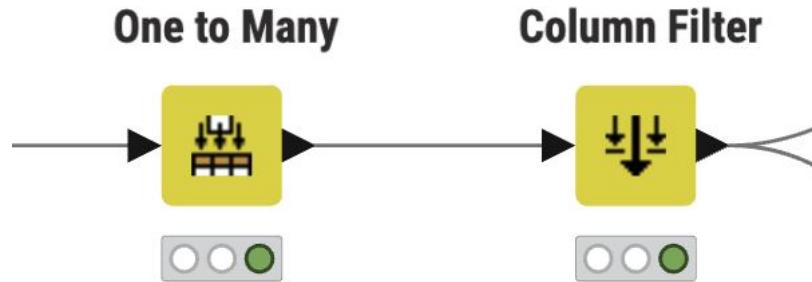
The final step in data manipulation was streamlining the inputs using the **Column Filter** node.

This stage systematically removed redundant columns, specifically, the un-transformed features that were now represented by our new log-transformed and one-hot encoded variables.



DATA MANIPULATION

(Preparing data for Random Forest and Gradient Boosting)



FEATURE ENGINEERING

To prepare data for the Random Forest and Gradient Boosting models, we applied the same one-hot encoding step using the **One to Many** node. Ensemble methods in KNIME also require fully numeric inputs, so categorical variables must be expanded into binary indicator columns to allow the models to correctly interpret each category.

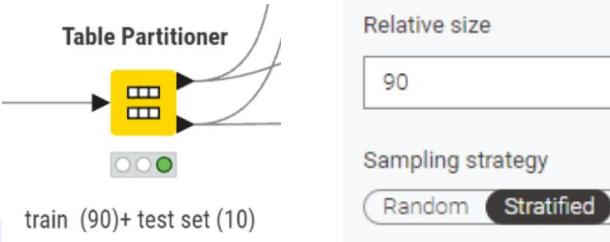
Columns to transform
<input type="checkbox"/> sex
<input type="checkbox"/> region
<input type="checkbox"/> urban_rural
<input type="checkbox"/> education
<input type="checkbox"/> marital_status
<input type="checkbox"/> employment_status
<input type="checkbox"/> smoker
<input type="checkbox"/> alcohol_freq
<input type="checkbox"/> plan_type
<input type="checkbox"/> network_tier
<i>Any unknown column</i>

After encoding, we used a **Column Filter** to remove the original categorical features, keeping the dataset clean, compact, and fully numeric. Since tree-based models do not require normalization or log transformations, this streamlined preprocessing was sufficient to provide them with a reliable and consistent set of input variables.

Excludes
<input type="checkbox"/> sex
<input type="checkbox"/> region
<input type="checkbox"/> urban_rural
<input type="checkbox"/> education
<input type="checkbox"/> marital_status
<input type="checkbox"/> employment_status
<input type="checkbox"/> smoker
<input type="checkbox"/> alcohol_freq
<input type="checkbox"/> plan_type
<input type="checkbox"/> network_tier

Data Split and important comments on modelling choice

We split the data into the train and test sets using **Table Partitioner** node with the settings of relative size 90% and 10%. This means that we will train on 90,000 of observations and test our final model performance on the 10,000 observations. This ensures vast training data capabilities, while leaving enough observations for the final evaluation on unseen data. Moreover, we stratified the data split on `filed_claim` target variable for the classification models in order to get the class proportion in all of our splits.

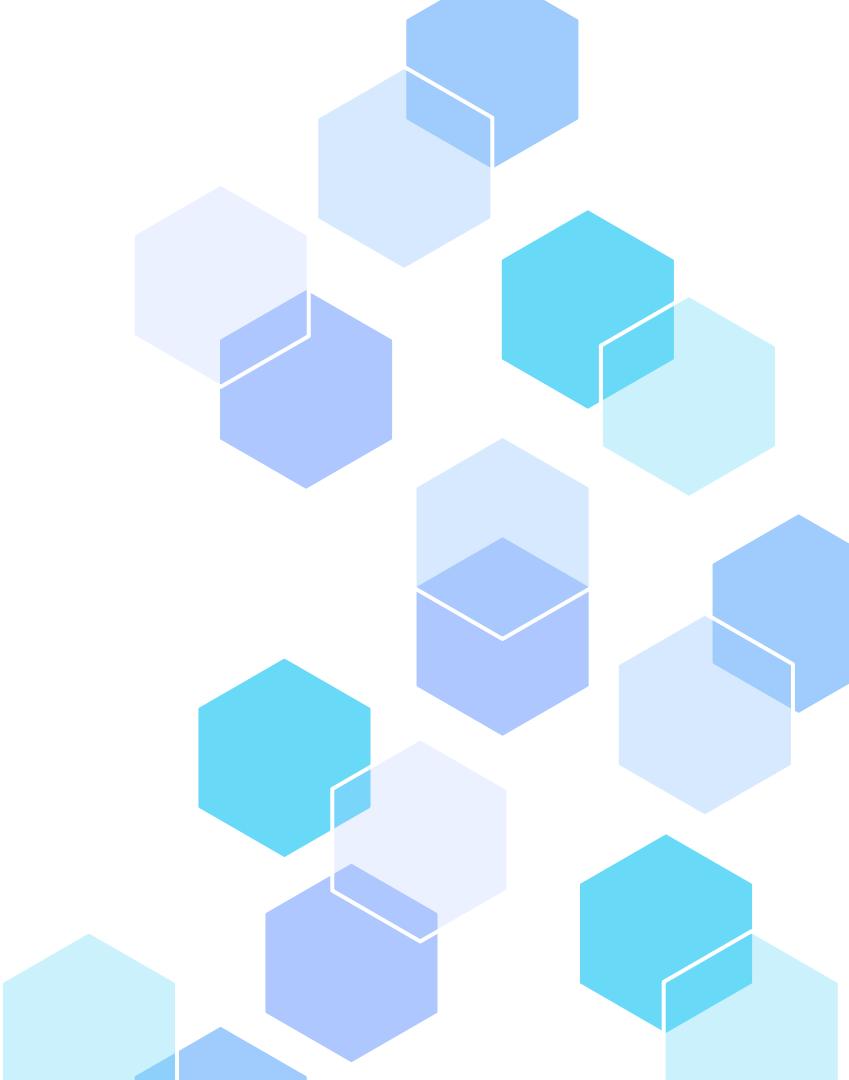


Choice of implementation.
For the regression task on `Total_claims_paid` we used the H2O extension in KNIME. This implementation supports the **Tweedie distribution family** as an objective function, which is well-suited to insurance claim costs that are continuous, highly skewed and contain many zeros. Using H2O therefore allowed us to model the heavy-tailed claim distribution more appropriately.

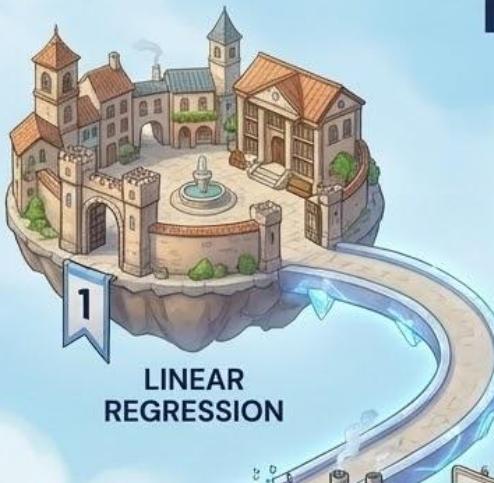
For the classification task on `filed_claim` we relied on the **standard KNIME learners** (logistic regression and gradient boosting). Here the target is binary and a simple binomial/logistic loss is sufficient, so the standard KNIME nodes already provide all required functionality while keeping the workflow simpler and easier to reproduce without additional extensions.

04

Modeling



MODELS OVERVIEW



LINEAR
REGRESSION



LOGISTIC
REGRESSION



GENERALIZED
LINEAR MODEL



PRINCIPAL COMPONENT
ANALYSIS

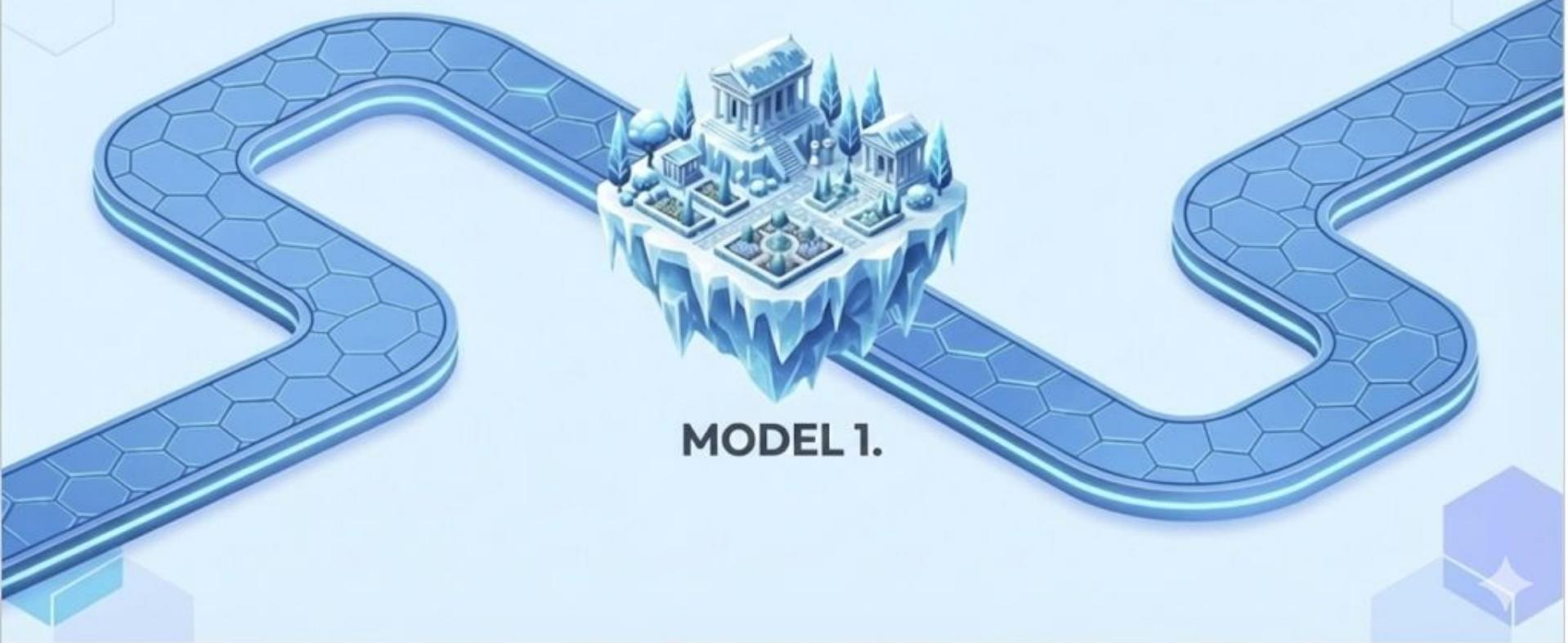


RANDOM FOREST
REGRESSION +
CLASSIFICATION



GRADIENT BOOSTING
REGRESSION +
CLASSIFICATION

MODELS OVERVIEW: LINEAR REGRESSION



Linear Regression

REGRESSION

GOAL

- Predict **how** much a policyholder will claim in total.

TARGET VARIABLE

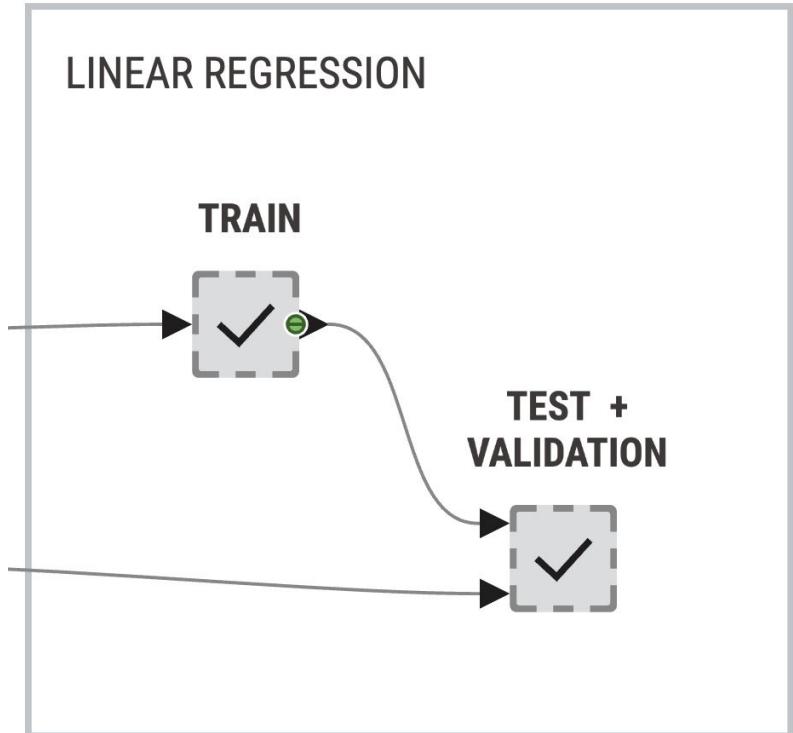
- Total_claims_paid**
- Continuous. Highly skewed. Extreme outliers.

WHY LINEAR REGRESSION?

- A good baseline model for our insurance claim amounts prediction task

WHAT THE MODEL PRODUCES?

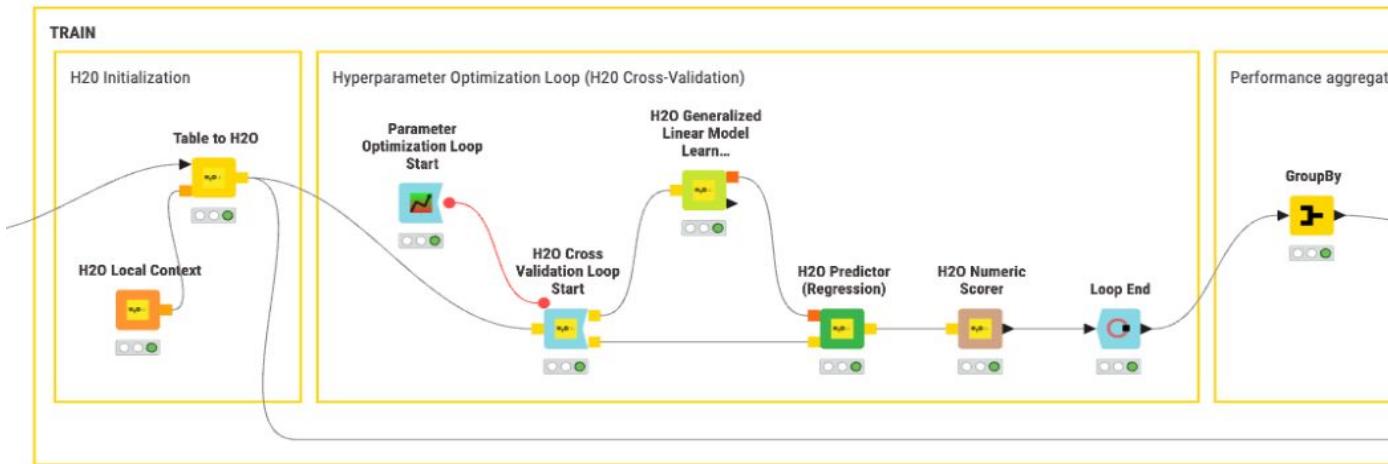
- A numerical estimate of yearly claim cost (USD).
- Performance evaluated with RMSE, R².



Linear Regression Train...

DATA

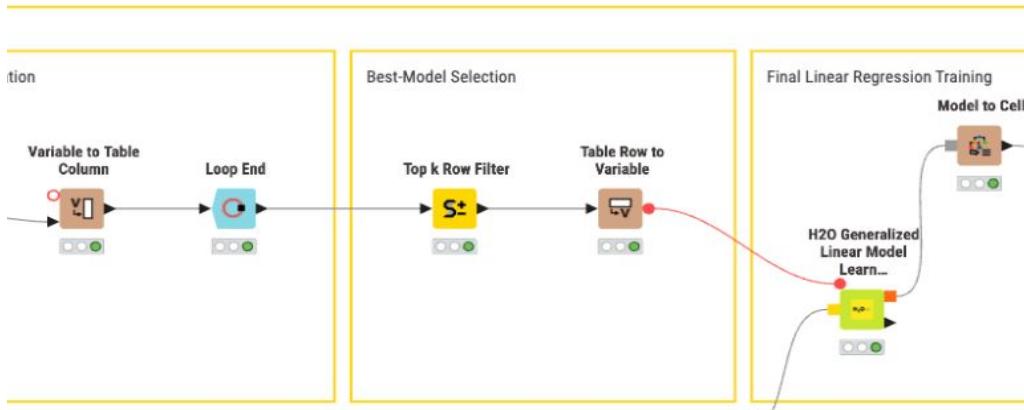
To use the H2O extension, we first converted the training table into an H2O frame with the **Table to H2O** node and added the **H2O Context** node to initialize the H2O session.



K-FOLD CROSS VALIDATION

- To optimize the model hyperparameters, we built a 3-fold cross-validation loop using the **H2O Cross Validation Loop** node, which provides train/validation splits to the Learner and Predictor nodes.
- At each iteration, the data is split into three equal-sized folds (around 26,667 observations in each fold, given the training set of 80,000 observations), the model is trained on two folds with a given set of parameters and evaluated on the remaining fold.
- Performance on the validation fold is measured with the **H2O Numeric Scorer**, comparing predictions to the total_claims_paid column.
- The validation fold is then rotated so that each of the three folds is used once for validation, ensuring that every observation is used for both training and validation.
- Then we get the mean prediction of all of the results on validation folds testing, getting a better picture of model performance.

...Linear Regression Train



PERFORMANCE AGGREGATION

- After testing one set of parameters the CV loop proceeds to the new set, until it tries all sets of hyperparameters.
- At the end of the CV loop execution the workflow will have all of the averaged results of the model performance on all different sets of hyperparameters.

BEST-MODEL SELECTION AND FINAL MODEL TRAINING

- Then we select the hyperparameters which yield the best performance-wise model (criteria for selection are RMSE and R^2) and retrain our linear regression model on the whole train dataset using the best hyperparameters.
- We get the Linear Regression model out of the H2O's GLM learner node with the settings of Gaussian Family and Identity link
- Also, numeric variables were standardized automatically inside the GLM learner node, which improves numerical stability, makes gradient-based optimization easier, and lets regularization treat all coefficients fairly.

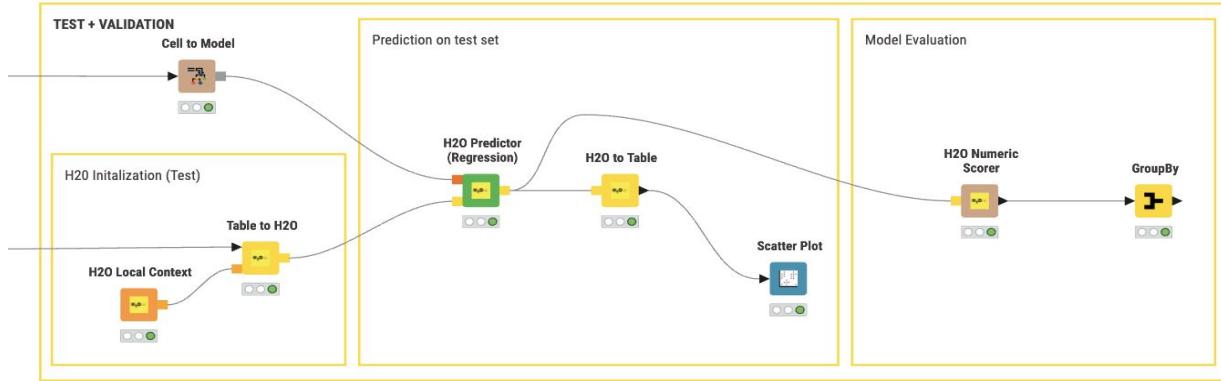
Linear Regression Hyperparameters

Mean(Mean Squared Error)	Mean(R2)	Mean(RMSE)	Mean(RMSLE)	Mean(Residual Deviances)	Alpha
4,487,141.052	0.163	2,118.116	NaN	4,487,141.052	1
4,491,872.654	0.163	2,119.233	NaN	4,491,872.654	0.75
4,496,439.963	0.162	2,120.302	NaN	4,496,439.963	0.5
4,496,449.989	0.162	2,120.305	NaN	4,496,449.989	0
4,506,688.428	0.16	2,122.709	4.246	4,506,688.428	0.25

- A parameter optimization loop tested **alpha values from 0 to 1** (0, 0.25, 0.50, 0.75, 1.0) to evaluate different levels of L1/L2 regularization.
- **Lasso ($\alpha = 1.0$)** produced the **best performance**, with the lowest RMSE (2118) and the highest R² (~0.164).
- L1 regularization improved generalization by **shrinking irrelevant coefficients**, suggesting the dataset benefits from feature selection.
- Models with **mid-range alpha values (0.25 and 0.50)** performed poorly, with **R² = 0.16-0.162** and higher RMSE (2120).
- These results indicate that **elastic-net combinations do not improve predictive accuracy** for this dataset.

- **Ridge regression ($\alpha = 0$)** also underperformed, showing that evenly shrinking all coefficients does not capture the underlying claim patterns.
- Across all configurations, overall model performance remained limited, with R² never exceeding ~0.163, confirming the **complex, nonlinear nature** of medical claim amounts.
- The optimization results highlight that while **Lasso is the best linear option**, linear regression as a whole is not sufficient for modeling heavy-tailed and highly variable claim data.

Linear Regression Test



LINEAR REGRESSION TESTING AND EVALUATION

- After training the linear regression model on the training dataset, the **Linear Regression Testing Node** was applied to evaluate model performance.
- This testing process was carried out using the **H2O Predictor** in combination with the **H2O Numeric Scorer**.
- The evaluation was performed on a **separate, unseen test set**, ensuring an unbiased assessment.
- This step measures the model's **out-of-sample predictive ability**, showing how well it generalizes to new policyholders.
- The main purpose is to determine whether the model can **accurately predict medical claim amounts** for individuals not included in the training phase.

MODELS OVERVIEW: LOGISTIC REGRESSION



Logistic Regression

CLASSIFICATION

Goal

- Predict **whether** a policyholder will file at least one claim.

Target Variable

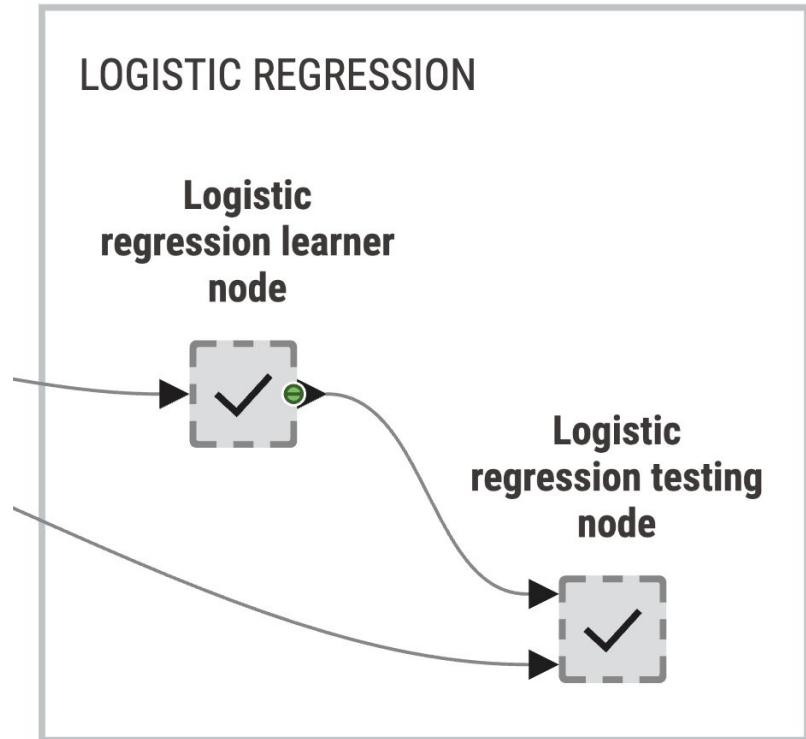
- `filed_claim`
- Binary. Imbalanced (62% yes, 38% no).

Why Logistic Regression

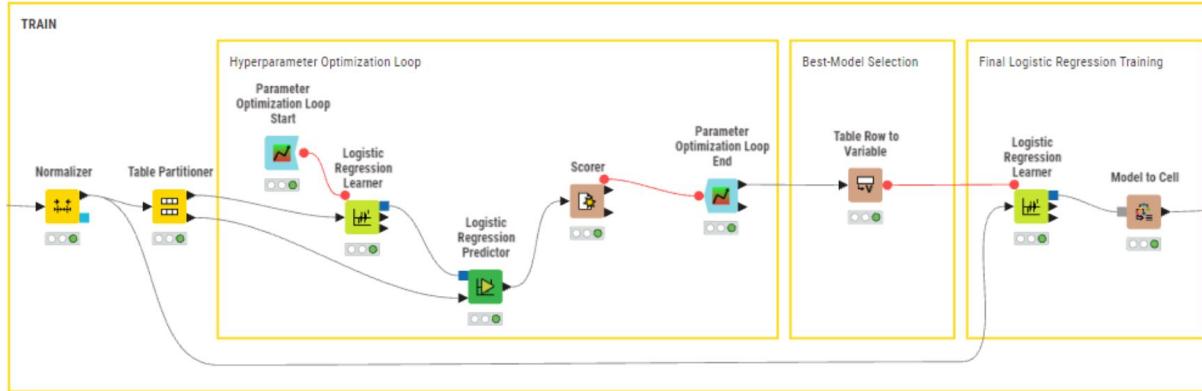
- Provides a solid baseline for our predictions.

What the Model Produces

- A probability and a binary decision (0/1).
- Performance evaluated with Accuracy, F1, AUC.



Logistic regression train



DATA SPLIT INTO TRAIN AND VALIDATION SETS

- First we normalized all of the numerical features, which puts all predictors on a comparable scale, improves stability, and ensures regularization penalizes all coefficients fairly.
- Then we split the data into train (80000 observations) and validation (10000 observations) sets using table partition node, stratifying the split on filed_claims variable.

HYPERPARAMETER OPTIMIZATION

- In order to **find the best hyperparameters** for our model we built loop, which trained logistic regression on train split with different set of hyperparameters and tested the results on the validation set.

FINAL MODEL TRAINING

- We then trained the **logistic regression model** with filed_claim variable as a target on whole train data with gradient descent using a fixed learning rate of 0.1 for up to 1000 epochs, with early stopping when the change in the objective fell below 10^{-5} .

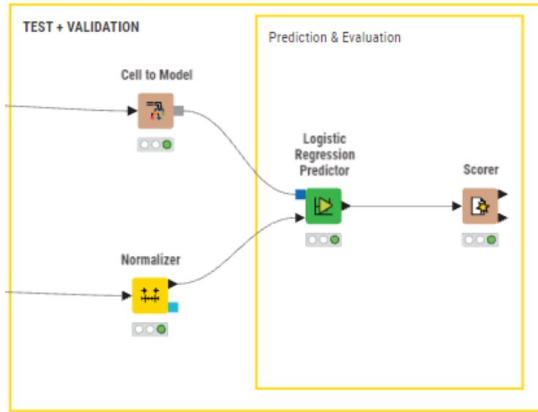
Logistic regression Hyperparameters

RowID	New parameter <small>.00 Number (Float)</small>	Objective value <small>.00 Number (Float)</small>
Row9	0.46	6,428
Row8	0.41	6,427
Row2	0.11	6,426
Row13	0.66	6,426
Row14	0.71	6,426

CHOICE OF THE HYPERPARAMETERS:

- For logistic regression we optimized **priorVariance**, which controls regularization in the model.
- For hyperparameter selection metric we used the **number of correct predictions** in cross-validation, which is equivalent to maximizing accuracy in our setting. Since the class distribution is only moderately imbalanced (67% vs 33%), **accuracy-based metrics** are appropriate and directly reflect overall predictive performance.
- The **best priorVariance value** is established at 0.46, which means moderate penalty on large coefficients.

Logistic regression test



PREDICTION

&

EVALUATION

- After the final training on 90000 observations we pass the final model to the Logistic regression testing node
- We then pass the test data with 10000 observations stratified from the whole dataset on our target variable
- We normalize the test data as we did for the train
- Then pass our final model and our normalized test data to the Logistic Regression Predictor node and get the final evaluation results

MODELS OVERVIEW: GENERALIZED LINEAR MODEL



GLM with Tweedie family

REGRESSION

GOAL

- Predict **how** much a policyholder will claim in total.

TARGET VARIABLE

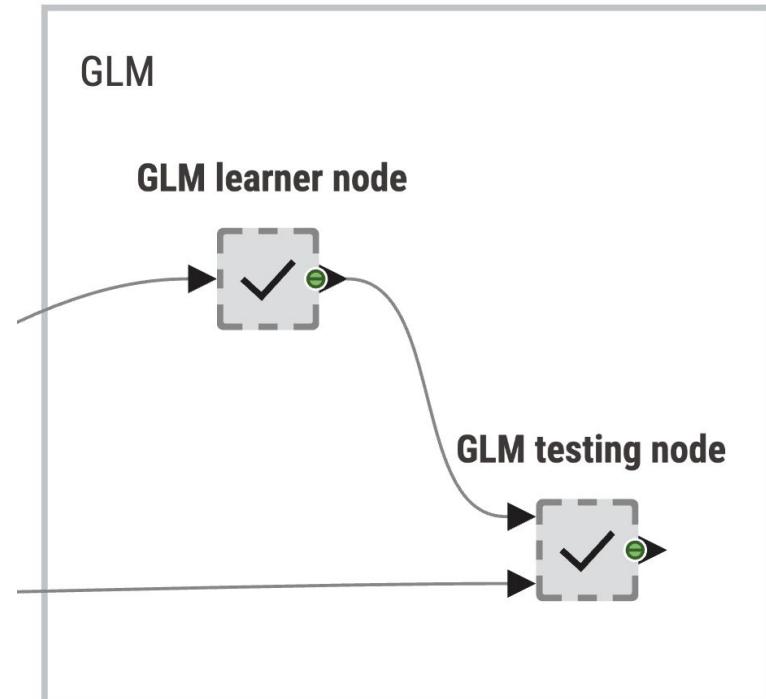
- Total_claims_paid**
- Continuous. Highly skewed. Extreme outliers.

WHY LINEAR REGRESSION

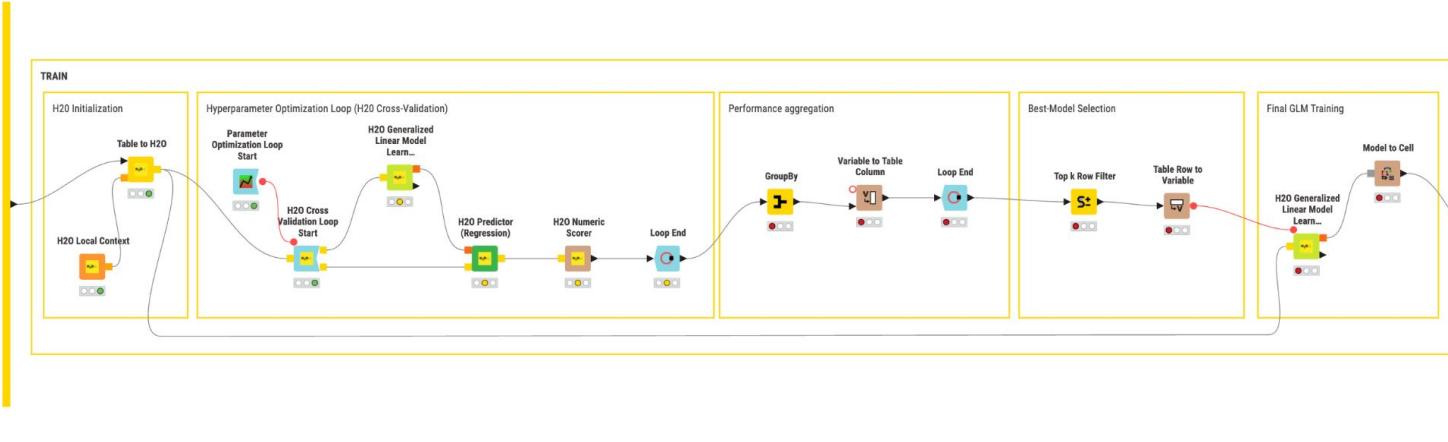
- well suited for insurance claim costs because it can model **many zero claims plus positive, highly skewed amounts** in a single framework (combined frequency–severity)
- the model interpretable and actuarially standard, allowing us to estimate clear effects of each predictor on expected claim cost

WHAT THE MODEL PRODUCES

- A numerical estimate of yearly claim cost (USD).
- Performance evaluated with RMSE, R².



GLM with Tweedie family Train



CROSS-VALIDATION PIPELINE

- The workflow for this model data preparation and hyperparameter cross-validation directly follows the same workflow as in the linear regression workflow.
- We build the 3-fold CV loop, using data converted to H2O format
- Then we aggregate the averaged results of the hyperparameter grid search
- And we get the best hyperparameters, which are then passed to the final GLM model training using whole train data.

FINAL MODEL CONFIGURATION

- Model type and link:** We fit a generalized linear model with **Tweedie family**, using the Tweedie link function and including an intercept term, so the model directly estimates the expected claim cost as a function of the predictors.
- Optimization setup:** The solver is left on Auto, and we allow up to 100 iterations for convergence

GLM with Tweedie family Hyperparameters

Mean(R2) Number (Float)	Mean(RMSE) Number (Float)	Mean(RMSLE) Number (Float)	Mean(Residual Devian... Number (Float)	Alpha Number (Float)	Variance power Number (Float)
0.151	2,133.98	4.277	4,555,279.115	0	1.1
0.151	2,133.98	4.277	4,555,279.115	0.25	1.1
0.151	2,133.98	4.277	4,555,279.115	0.5	1.1
0.151	2,133.98	4.277	4,555,279.115	0.75	1.1
0.15	2,134.96	4.285	4,558,420.281	1	1.1

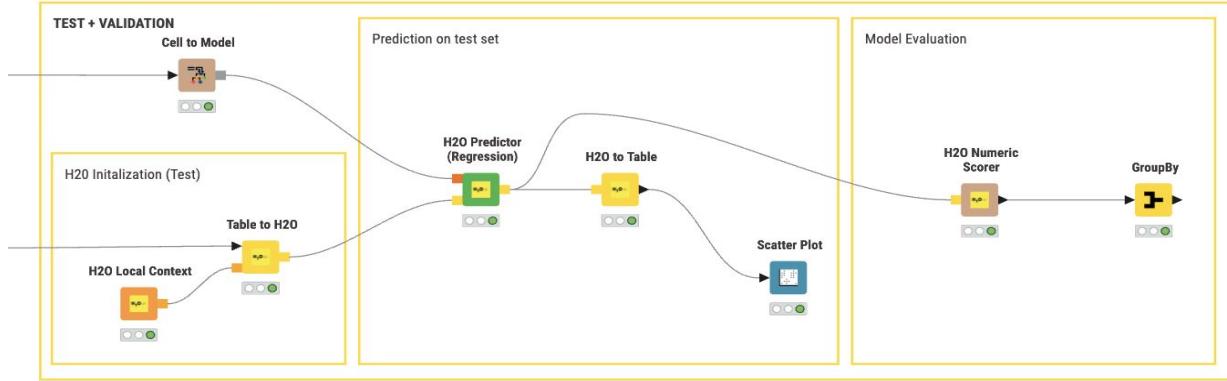
CHOICE OF THE PARAMETERS

- **Variance power** controls the relationship between the mean and the variance in the Tweedie distribution; values between 1 and 2 are suitable for insurance data because they model a mixture of many zeros and right-skewed positive claim amounts.
- **Alpha** controls the type of regularization in the GLM: $\alpha = 0$ corresponds to pure ridge (L2) shrinkage, $\alpha = 1$ to lasso (L1), and intermediate values give an elastic-net mix that can both shrink and sparsify coefficients.

HYPERPARAMETER OPTIMIZATION

- We performed a grid search over variance power from 1.1 to 1.9 (step 0.1) and alpha from 0 to 1 (step 0.25), evaluating each configuration with cross-validated error metrics (RMSE, R^2) on the validation folds
- Across this grid the performance was very stable, with almost identical errors for all α values at variance power 1.1 and only a slight deterioration for the most extreme setting ($\alpha = 1$), indicating that the Tweedie GLM is quite robust to the exact regularization mix.
- Given this flat response surface, we selected a simple configuration with variance power ≈ 1.1 and a low/medium α , which matches the best validation RMSE while keeping the model well-regularized and easy to interpret.

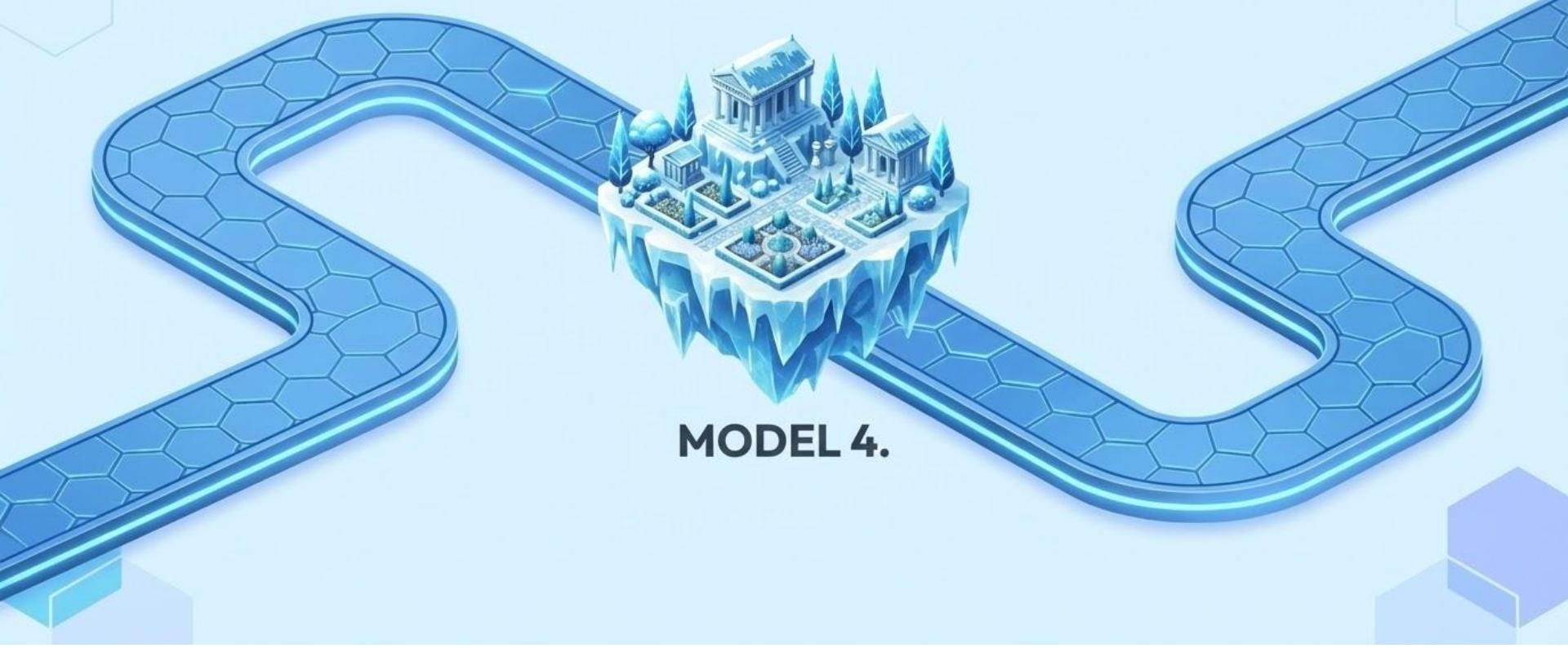
GLM with Tweedie family Test



GLM TESTING AND EVALUATION

- The testing pipeline is identical to the Linear Regression testing workflow
- We pass the trained model with optimized hyperparameters to the H2O predictor node and predict using the test data in H2O format
- Then we generate numeric scores and scatter plot

MODELS OVERVIEW: PRINCIPAL COMPONENT ANALYSIS



K-Means Clustering with PCA

After evaluating the performance of the supervised regression models, the next step in our analysis focuses on unsupervised learning methods. While regression predicts how much a policyholder may claim, K-Means clustering helps us understand how different types of policyholders naturally group together. It reveals hidden patterns in the dataset that regression cannot capture. This allows us to identify distinct customer segments based on health, lifestyle and utilization characteristics.

ROLE OF PCA (PRINCIPAL COMPONENT ANALYSIS)

- Reduces high-dimensional, highly correlated insurance features into a few **interpretable components**.
- Helps visualize complex data and improves clustering performance.
- Ensures clusters are formed based on the **strongest underlying patterns**.

ROLE OF K-MEANS CLUSTERING

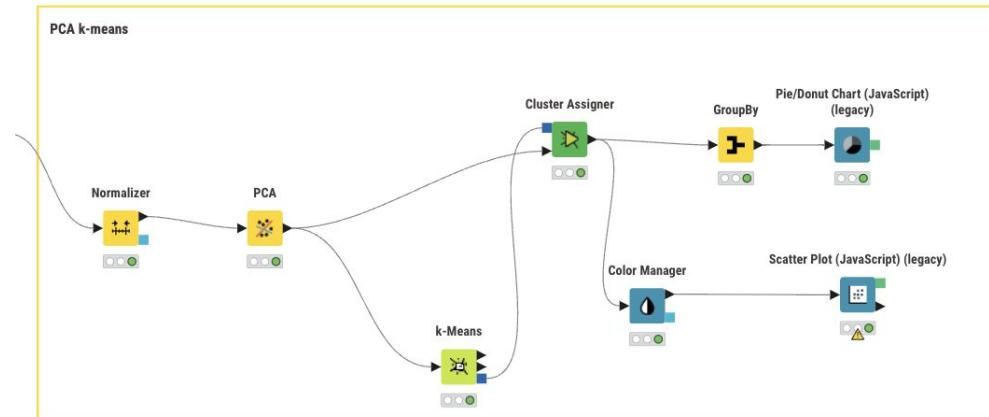
- Groups policyholders into **homogeneous segments** based on their PCA scores.
- Highlights differences in risk, utilization, and behavior across groups.
- Allows for targeted managerial actions such as risk-adjusted pricing and preventive care programs.

K-Means Clustering Workflow

First, we standardized the data using a **Z-Score Normalizer** in order to ensure that all numeric variables contributed equally to distance calculations and preventing features with large scales from dominating the analysis.

Next, **Principal Component Analysis** (PCA) was applied to reduce the high-dimensional and highly correlated feature space into a smaller set of principal components, which capture the majority of variance related to lifestyle risk, demographics, healthcare utilization etc.

These components were then passed into the **K-Means Learner**, where the algorithm was configured with **k = 2** to identify two natural clusters within the policyholder population. Lastly, we assigned all data points to their corresponding cluster and added some visualizations.



K-Clustering Scatter Plot

PCA & K-MEANS FOR STRUCTURE

We used PCA and K-Means to uncover natural structure in the policyholder data. PCA reduces the high-dimensional dataset into meaningful components. This removes noise for clean clustering.

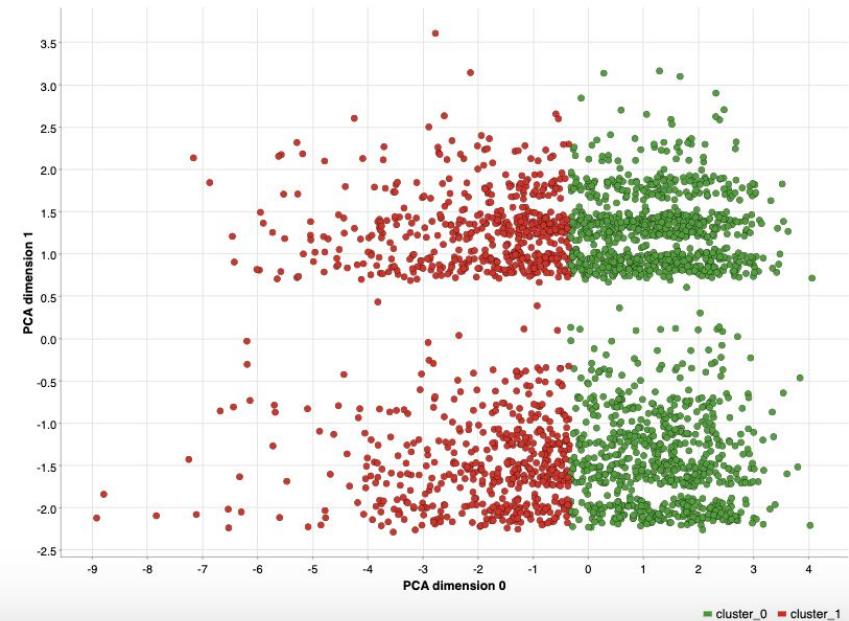
CLUSTER INTERPRETATION

K-Means assigns all policyholders to clusters based on their PCA components.

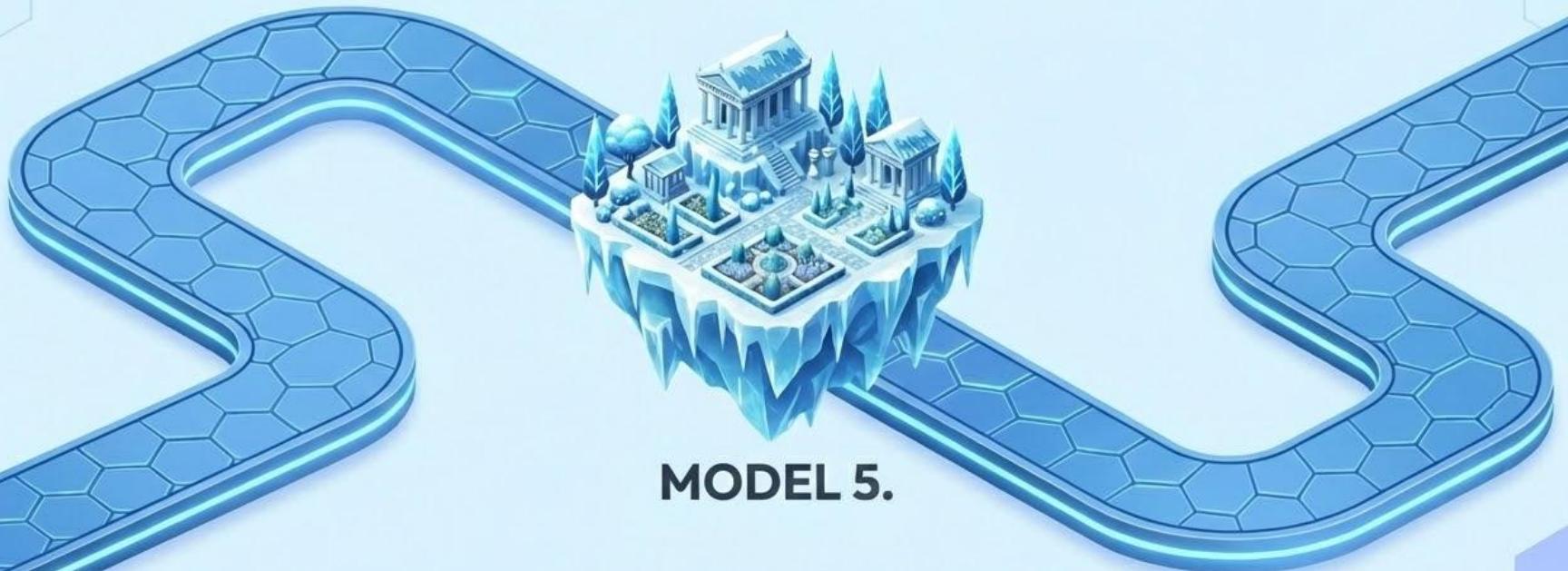
The scatter plot shows two clear groups:

- Cluster 0 (green) is a lower-risk segment.
- Cluster 1 (red) is a higher-risk group.

We decided to set $k=2$ to get meaningful subgroups. We found that higher values for k are difficult to interpret.



MODELS OVERVIEW: RANDOM FOREST CLASSIFICATION & REGRESSION



MODEL 5.

Random Forest Models

REGRESSION

GOAL

- Predict **how** much a policyholder will claim in total.

TARGET VARIABLE

- Total_claims_paid**
- Continuous. Highly skewed. Extreme outliers.

WHY RANDOM FOREST REGRESSION

- Handles nonlinear relations and heavy-tailed cost data.
- Captures interactions missed by linear models.

WHAT THE MODEL PRODUCES

- A numerical estimate of yearly claim cost (EUR).
- Performance evaluated with RMSE, MAE, R².

CLASSIFICATION

GOAL

- Predict **whether** a policyholder will file at least one claim.

TARGET VARIABLE

- filed_claim**
- Binary. Imbalanced (62% yes, 38% no).

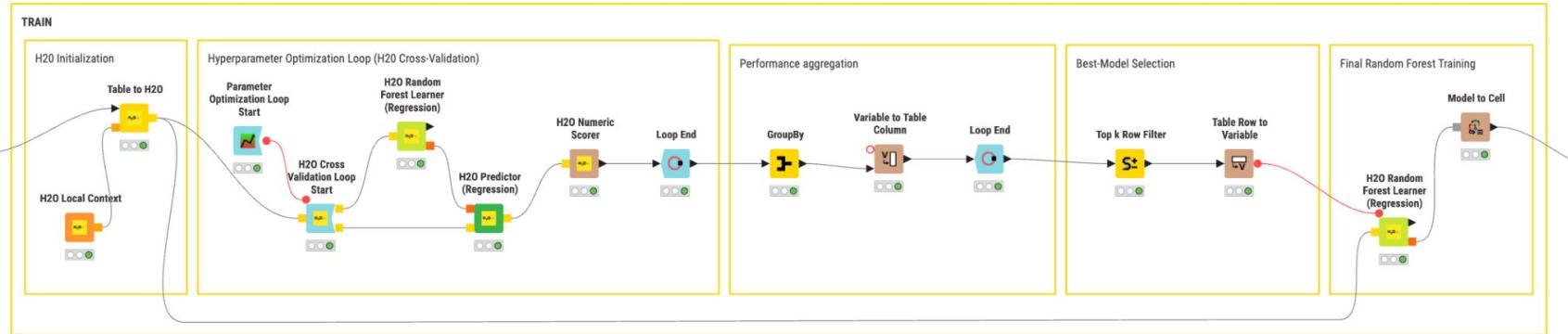
WHY RANDOM FOREST REGRESSION

- Effective for categorical outcomes.
- Naturally handles imbalance through bootstrapping.

WHAT THE MODEL PRODUCES

- A probability and a binary decision (0/1).
- Performance evaluated with Accuracy, F1, AUC.

Random Forest Regression Train



CROSS-VALIDATION PIPELINE

- The workflow for data preparation and hyperparameter cross-validation follows the same logic as in the linear regression setup, but with the Random Forest regression as the base learner.
- We build a 3-fold CV loop on data converted to H2O format; the loop trains the Random Forest on two folds and evaluates it on the remaining fold.
- The averaged results from the grid search over the RF hyperparameters are then aggregated, and the best-performing configuration is passed to the final model training on the full training data.

FINAL MODEL CONFIGURATION

- Model type:** We fit an **H2O Random Forest regressor** with `total_claims_paid` as the target and all available predictors included as input features.
- Ensemble structure:** The forest consists of **100 trees**, with a fixed static random seed so that the results are fully reproducible.
- Bagging & histograms:** Each tree is trained on a bootstrap-style subsample of about 63.2% of the rows (`row_samplerate = 0.632`), and numeric and categorical features are binned into histograms (20 bins for numeric and 1024 for categorical/root bins), which speeds up split search while preserving enough detail in the predictors.

Random Forest Cross-Fold Hyperparameters

Mean(Me... Number (...))	Mean(R2) Number (...))	Mean(RM... Number (...))	Mean(RM... Number (...))	Mean(Re... Number (...))	Column_s... Number (...))	Min_obs... Number (...)	Max_depth Number (...)
4,488,565.616	0.163	2,118.556	4.246	4,488,565.616	0.8	16	10
4,490,547.053	0.163	2,119.03	4.246	4,490,547.053	0.8	11	10
4,490,854.176	0.163	2,119.104	4.24	4,490,854.176	1	6	10
4,492,803.029	0.162	2,119.555	4.24	4,492,803.029	1	16	10
4,496,549.966	0.162	2,120.442	4.24	4,496,549.966	1	11	10

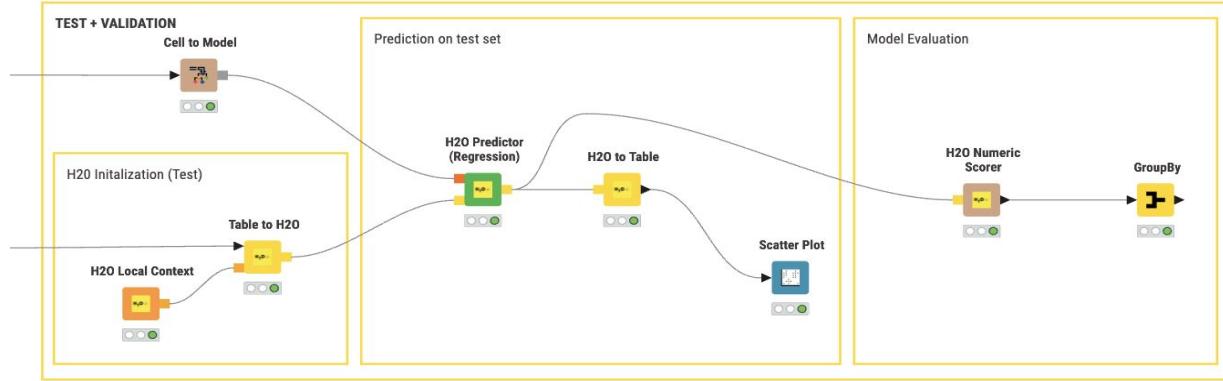
CHOICE OF THE PARAMETERS

- **Max depth** controls how deep each tree in the Random Forest can grow. Deeper trees can model more complex interactions between predictors but are more prone to overfitting
- **Min observations** specifies the smallest number of observations that must fall into a node before it can be split. Small values allow very fine partitions of the feature space, larger values enforce smoother trees and reduce variance.
- **Column sample rate** determines what fraction of predictors is randomly considered at each split. Sampling a subset of columns decorrelates the trees and can improve generalization, while using all columns makes each tree slightly stronger but increases the risk of overfitting.

HYPERPARAMETER OPTIMIZATION

- We ran a grid search over max depth from 5 to 30 (step 5), min observations from 1 to 20 (step 5), and column sample rate from 0.6 to 1.0 (step 0.2), evaluating each configuration with cross-validated regression metrics (RMSE, R²) on the validation folds.
- The results show that performance is quite stable across this grid: several combinations achieve almost identical RMSE and R², with only minor differences between settings.
- The best scores are obtained for a moderately deep forest (**max depth ≈ 10**), relatively large **minimum node size (around 16 observations)**, and a **column sample rate of 0.8**, so we chose this configuration as our final Random Forest regressor.

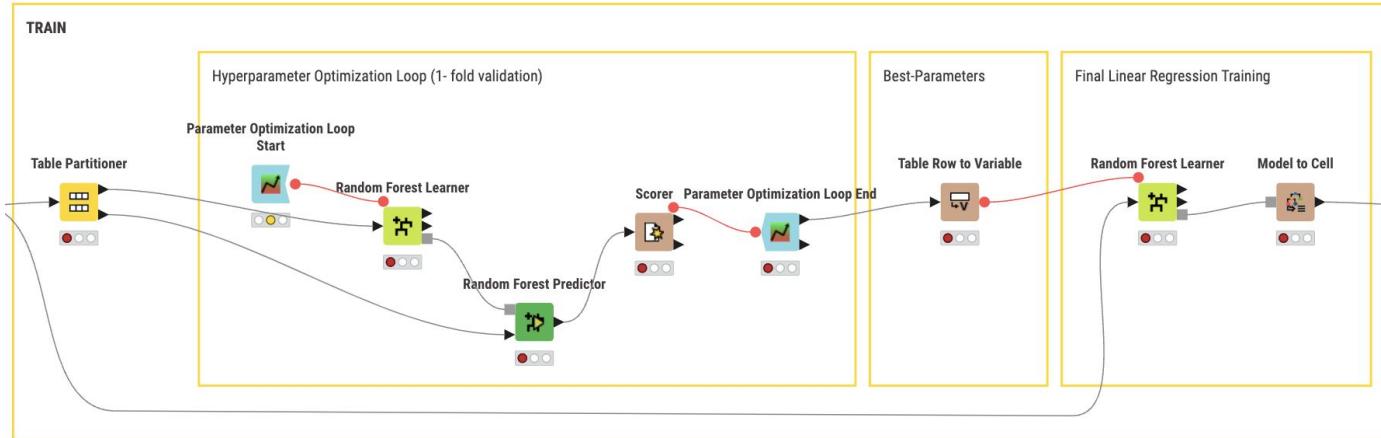
Random Forest Regression Test



RANDOM FOREST TESTING AND EVALUATION

- The Random Forest test pipeline replicates the Linear Regression evaluation flow. The trained RF model is loaded into the H2O predictor and applied to the held-out test set.
- Predictions are converted back into table format and evaluated through H2O Numeric Scorer to compute all regression metrics.
- The workflow outputs both aggregated scores and the scatter plot for visual inspection of prediction performance.

Random Forest Classification Train



DATA AND SPLIT

- Using the similar pipeline of the train and validation set preparation we create two stratified splits of the train (80000 observations) and validation (10000 observations) splits

HYPERPARAMETER OPTIMIZATION

- Then optimized the hyperparameters for more robust and accurate model predictions on the future test set and in the production

FINAL MODEL TRAINING AND CONFIGURATION

- Tree structure:** Random Forest employs 100 decision trees
- Split criterion:** Trees use the Information Gain Ratio for splitting
- Ensemble & randomness:** All predictors are included, and a fixed random seed ensures full reproducibility of the model and results.

Random Forest Classification Hyperparameters

Tree depth	Minimum node size	Objective value ↓
Number (Integer)	Number (Integer)	Number (Float)
15	2	6,504
15	7	6,504
15	12	6,504
15	17	6,504
15	22	6,504

CHOICE OF THE PARAMETERS

- **Tree depth** controls how many levels each tree in the random forest can grow. Deeper trees can model more complex interactions but are more prone to overfitting, while shallower trees act as a stronger regularizer
- **Minimum node size** specifies the minimum number of observations that must fall into a node before it can be split. Small node sizes allow very fine partitions of the feature space, larger node sizes enforce smoother trees and reduce variance.

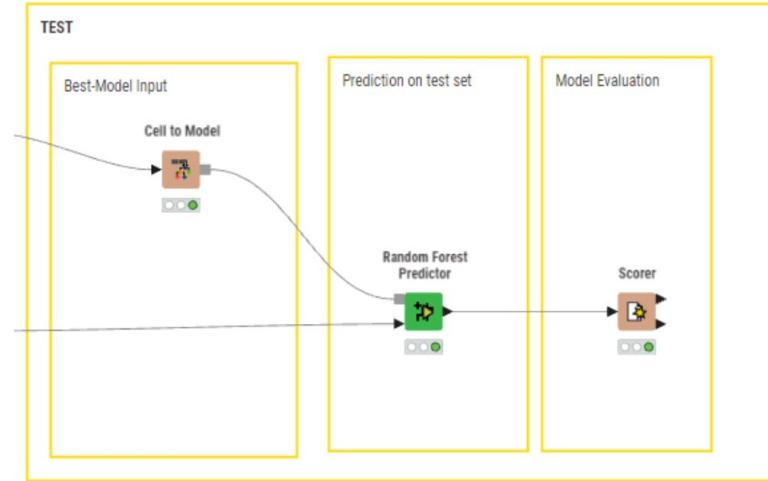
HYPERPARAMETER OPTIMIZATION

- We performed a grid search over **tree depth from 5 to 25** (step 5) and **minimum node size from 2 to 30** (step 5), using the **number of correct predictions on the validation set** as the objective
- Across this grid, the objective value (6,504 correct predictions) was essentially constant, indicating that within these ranges the random forest performance is quite insensitive to the exact node-size settings
- Given this flat response surface, we selected a **relatively simple configuration** (tree depth of 15 with a reasonably large minimum node size), which matches the best validation performance while keeping the model more compact, faster to train, and less prone to overfitting

Random Forest Classification Test

RANDOM FOREST PREDICTION & EVALUATION

- We build a testing pipeline analogous to the one used for logistic regression, but using the **Random Forest classification model** as the final estimator.
- The trained Random Forest model is loaded from the model cell and passed to the **Random Forest Predictor**, which generates class probabilities and predicted labels.
- The held-out **test set** is fed into this predictor node, and the resulting predictions are evaluated with the **Scorer** node to obtain Accuracy, F1, AUC, and the confusion matrix. **No normalization is applied** here: tree-based models split on feature order rather than on absolute distances, so rescaling the inputs does not change the splits, making normalization unnecessary for Random Forest.



MODELS OVERVIEW: GRADIENT BOOSTING CLASSIFICATION & REGRESSION



MODEL 6.

Gradient Boosting Models

Regression

GOAL

- Predict **how** much a policyholder will claim in total.

TARGET VARIABLE

- Total_claims_paid**
- Continuous. Highly skewed. Extreme outliers.

WHY RANDOM FOREST REGRESSION

- Captures complex nonlinear relationships and interactions in claim costs by sequentially adding trees that correct previous errors.
- Handles skewed, heavy-tailed **Total_claims_paid** reasonably well and can be tuned to be robust to outliers.

WHAT THE MODEL PRODUCES

- A numerical estimate of yearly claim cost (EUR).
- Performance evaluated with RMSE, MAE, R².

Classification

GOAL

- Predict **whether** a policyholder will file at least one claim.

TARGET VARIABLE

- filed_claim**
- Binary. Imbalanced (62% yes, 38% no).

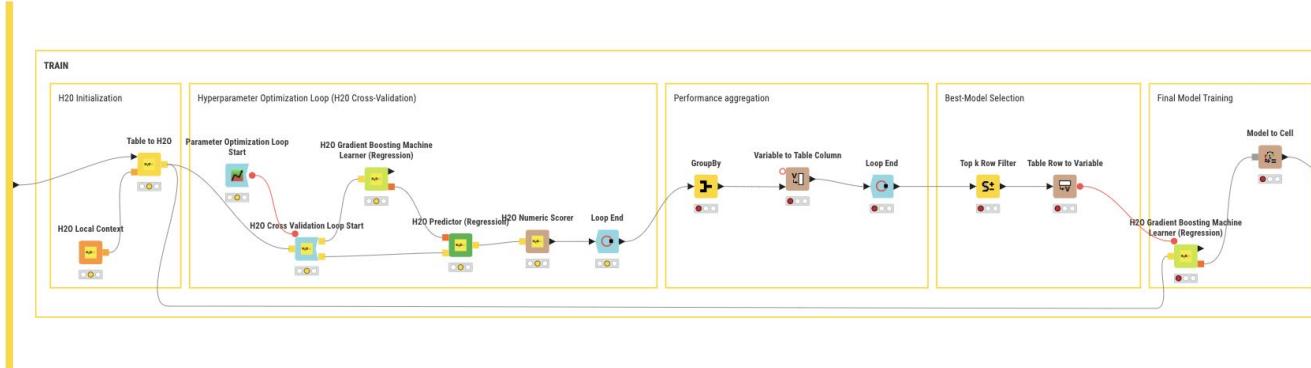
WHY RANDOM FOREST REGRESSION

- Learns flexible nonlinear decision boundaries for separating claimers vs non-claimers, often outperforming simple linear models.
- Iteratively focuses on previously misclassified observations, which can improve performance on the minority “no-claim” class in the dataset.

WHAT THE MODEL PRODUCES

- A probability and a binary decision (0/1).
- Performance evaluated with Accuracy, F1, AUC.

Gradient Boosting Regression Train



CROSS-VALIDATION PIPELINE

The workflow for data preparation and hyperparameter cross-validation follows the same logic as in the regression models setups, but with the Gradient Boosting regression as the base learner.

FINAL MODEL CONFIGURATION

- **Model type and target:** We fit an H2O **Gradient Boosting Machine regressor** with `total_claims_paid` as the target and all available predictors included in the model.
- **Ensemble structure:** The model builds an ensemble of **500 decision trees**, each tree requiring at least 10 (weighted) observations in a node before it can be split, which prevents overly tiny, unstable leaves.

- **Sampling:** For each tree we use **row and column sampling rates of 0.8**, so every tree sees a different subsample of rows and features, adding randomness and regularization; splits are found using histogram binning (20 bins for numeric features, 1024 for categorical/root bins), and the loss is based on the **Tweedie distribution**, which is suitable for zero-inflated, heavy-tailed claim amounts (however, we also tried Gaussian distribution, but the results were the same)
- **Early stopping & randomness:** Early stopping is enabled with an automatic stopping metric, a tolerance of 0.001, and a moving-average window of 3, using 10% of the data as an internal validation set; a **fixed random seed** is set so that training and results are fully reproducible.

Gradient Boosting Regression Hyperparameters

Mean(Mean Sq... Number (Float))	Mean(R2) Number (Float)	Mean(RMSE) Number (Float)	Mean(RMSLE) Number (Float)	Mean(Residual ... Number (Float))	Tweedie power Number (Float)	Learning_rate Number (Float)	Max_depth Number (Integer)
4,463,856.781	0.168	2,112.736	4.245	4,463,856.781	1.5	0.05	2
4,466,024.024	0.167	2,113.253	4.242	4,466,024.024	1.5	0.1	2
4,466,602.082	0.167	2,113.377	4.24	4,466,602.082	1.9	0.1	2
4,468,633.925	0.167	2,113.872	4.242	4,468,633.925	1.5	0.15	2
4,468,734.975	0.167	2,113.906	4.247	4,468,734.975	1.1	0.15	2

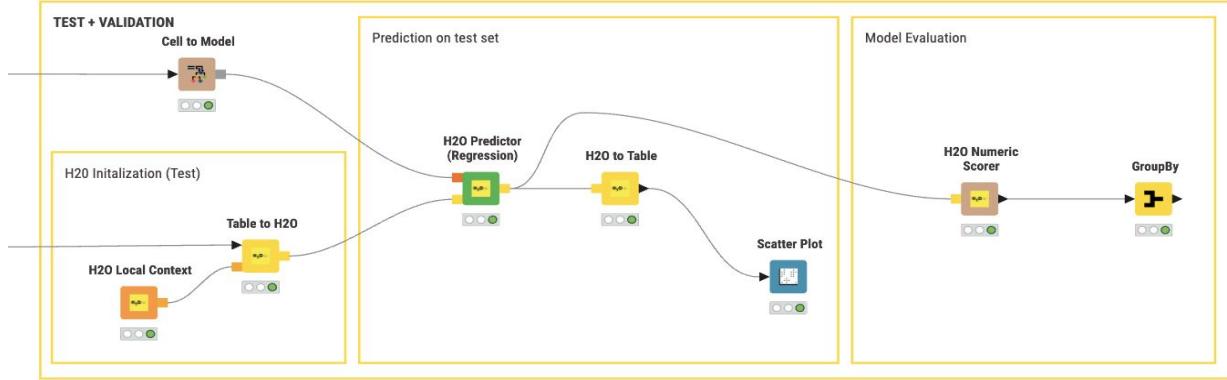
CHOICE OF THE PARAMETERS

- **Max depth** controls how deep each tree in the gradient boosting ensemble can grow; shallow trees (small depth) act as simple “weak learners”, reducing overfitting and letting the boosting process build complexity gradually.
- **Learning rate** scales the contribution of each new tree; smaller learning rates make the model learn more slowly and robustly (often needing more trees), while larger rates speed up learning but can overshoot and overfit.
- **Tweedie power** specifies the variance structure of the Tweedie loss; values between 1 and 2 let the model capture the typical insurance pattern of many zero claims plus right-skewed positive amounts.

HYPERPARAMETER OPTIMIZATION

- We performed a grid search over max depth from 2 to 10 (step 2), learning rate from 0.05 to 0.20 (step 0.05), and Tweedie power from 1.1 to 1.9 (step 0.4), evaluating each setting with cross-validated regression metrics (RMSE, R²) on the validation folds.
- Across this grid, performance was very similar, with only small differences in RMSE; the best result was obtained with shallow trees (depth 2), a relatively low learning rate (0.05), and Tweedie power around 1.5
- Given this flat response surface, we chose this configuration as our final GBM model, since it offers the lowest validation error while keeping the model simple and stable

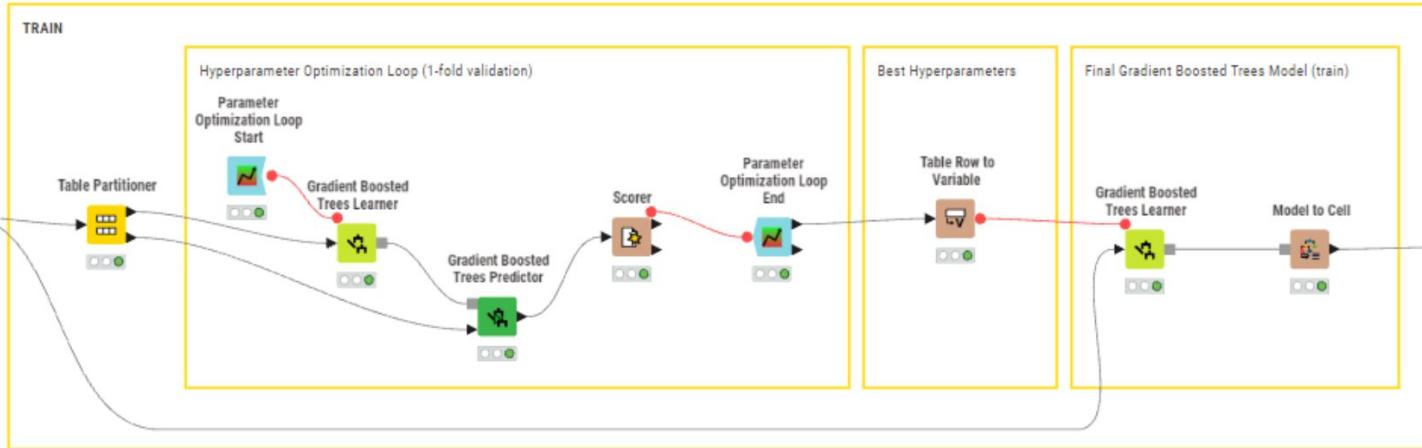
Gradient Boosting Test



GRADIENT BOOSTING TESTING AND EVALUATION

- The testing pipeline is identical to the Linear Regression testing workflow
- We pass the trained model with optimized hyperparameters to the H2O predictor node and predict using the test data in H2O format
- Then we generate numeric scores and scatter plot

Gradient Boosting Classification



DATA AND SPLIT

- We used the same logic of creation of the stratified train/validation split feature set as in the logistic-regression and random forest pipeline, with `filed_claim` as the target.

HYPERPARAMETER OPTIMIZATION

- Then the pipeline utilizes the same hyperparameter optimization idea, where we want to optimize the model's parameters for better model performance.

FINAL MODEL TRAINING AND CONFIGURATION

- **Tree structure:** Gradient Boosting builds an ensemble of up to 200 shallow decision trees with maximum depth being determined by the hyperparameter optimization loop
- **Splits & missing values:** Trees use mid-point splits for numeric attributes, binary splits for categorical attributes, and handle missing values internally with the XGBoost strategy.
- **Sampling & randomness:** All rows and all columns are used for every tree (no subsampling), and a fixed random seed ensures fully reproducible training.

Gradient Boosting Classification Hyperparameters

Tree depth <small>Number (Integer)</small>	learning rate <small>Number (Float)</small>	Objective value ↓ <small>Number (Float)</small>
2	0.05	6,497
2	0.1	6,482
4	0.05	6,475
4	0.15	6,474
2	0.25	6,473

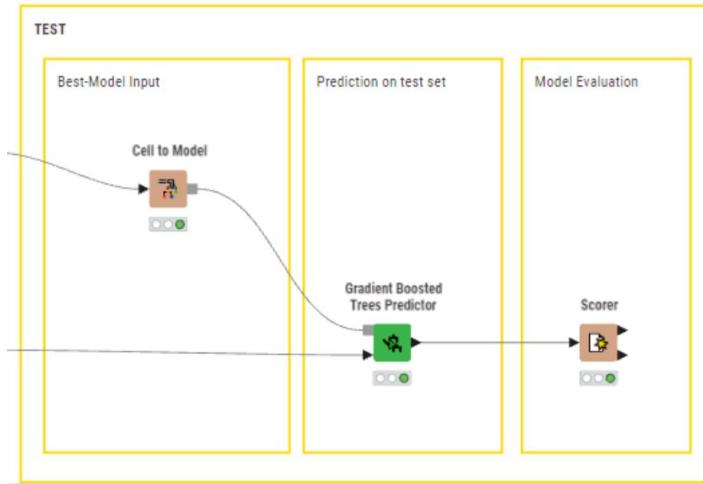
CHOICE OF THE PARAMETERS

- **Tree depth** is how many levels of splits each tree in the gradient boosting model can have, deeper trees capture more complex patterns but are more prone to overfitting.
- The **learning rate** is a scaling factor on each new tree's contribution, smaller learning rates make the model learn more slowly but usually more robustly, often requiring more trees.
- They're the two hyperparameters that **most strongly control model complexity and learning speed**, so they usually give most of the performance gain, while the search space as efficient as possible

HYPERPARAMETER OPTIMIZATION

- We tried the variables combination of **tree depth** starting from 2 to 8 with step 2, since gradient boosting models utilize shallow trees and learning rate from 0.05 to 0.25 with step 0.05 in order to test high and low learning rate values
- According to the results, lower learning rates generally lead to a better objective value.
- Deeper trees (depth 4) don't clearly outperform shallower ones and the best result is with depth 2 and learning rate 0.05 (objective 6,497).
- So, a shallow model with a relatively high learning rate seems to work best

Gradient boosting classification test



PREDICTION

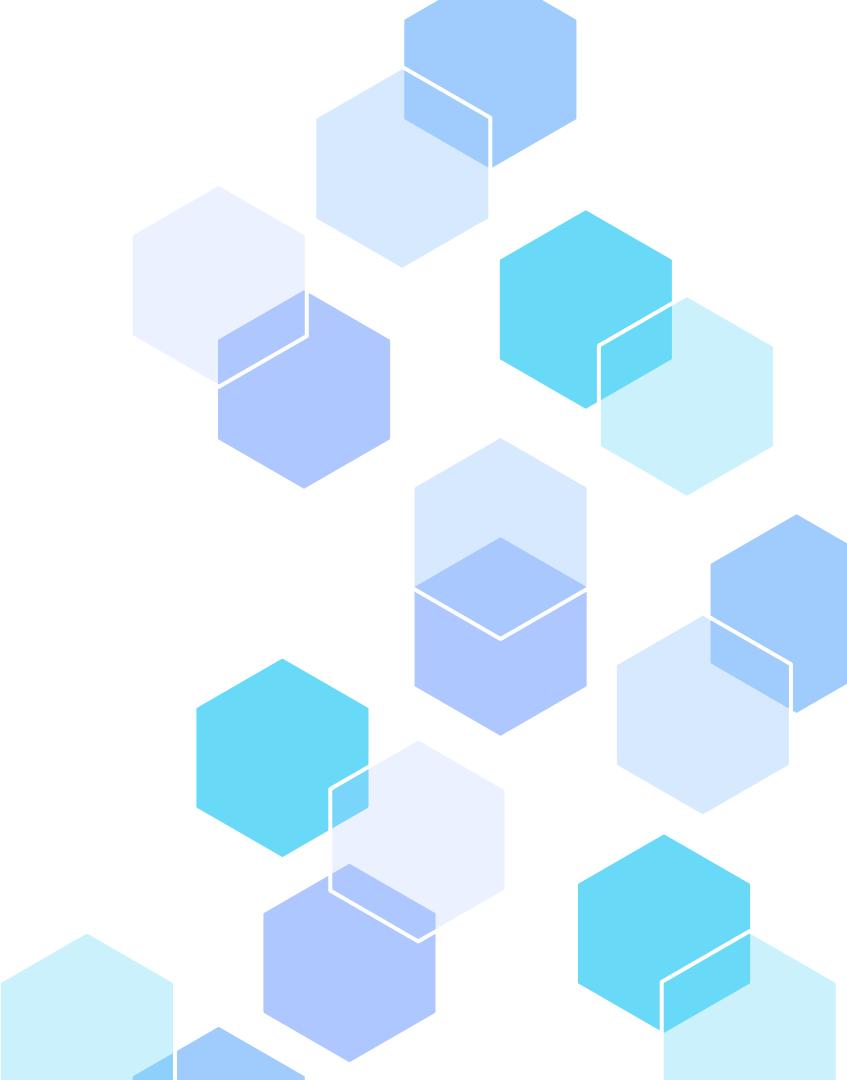
&

EVALUATION

- Following the same logic for the model testing as we did for Logistic Regression we implement the testing pipeline for Gradient Boosting model
- The model is passed to the Gradient Boosted Trees Predictor
- We also pass the test set of our data
- The final performance of the model is then captured

05

Results



Linear Regression

Test Evaluation & Results

MSE & Residual Variance:

4,128,556.158

- Both measures capture the total unexplained variance in the predictions.
- The value is high because claim amounts range from 0 up to very large outliers, so errors on rare extreme claims dominate the variance.

RMSE: 2,031.885

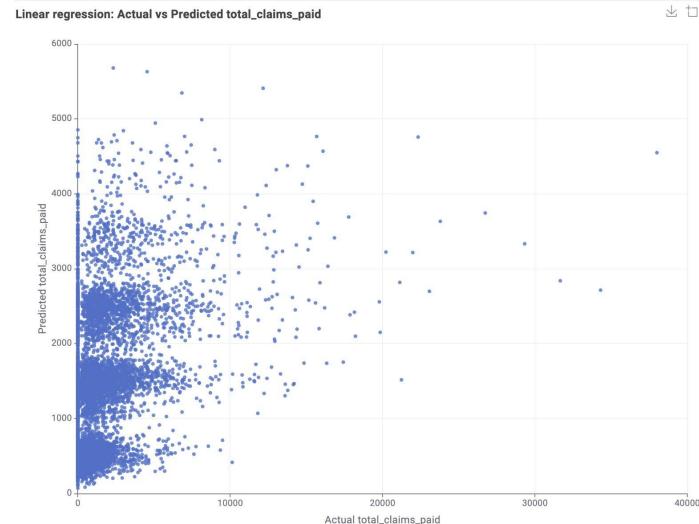
- On average, the model's prediction error is about **\$2,032** per policyholder.
- This again reflects the difficulty of predicting rare, very high-cost claims in a highly skewed distribution.

R²: 0.154

- The model explains only about 15% of the variability in claim amounts.
- This indicates limited predictive power: a simple linear relationship between predictors and total claims is not rich enough to capture the complex, erratic pattern of insurance costs.

RMSLE: 4.213

- The high log-scale error shows that the model makes **large multiplicative errors**, especially for higher claims
- It confirms that the linear model tends to **underestimate the largest claims** and struggles with the extreme right tail.



- The model predicts low to medium claim amounts reasonably consistently, as seen from the dense cloud of points at the lower ranges.
- For high-cost claims, predictions plateau around a few thousand euros and do not increase with the actual values, so large claims are systematically underpredicted.

Linear Regression Insights

The testing node results confirm several important insights:

Linear regression is limited in predictive performance, largely because the relationship between predictors (age, BMI, smoking status, etc.) and claim amounts is nonlinear and influenced by extreme outliers.

High-cost policyholders are poorly predicted, suggesting that real-world claim behavior exhibits threshold effects, conditional risk patterns, and interactions that linear regression cannot capture.

The model performs similarly across training and testing, indicating good generalization but overall insufficient accuracy.

Despite these limitations, the linear regression model still provides value as a **baseline benchmark** and as an **interpretable tool** for understanding general cost drivers.

Logistic Regression Test Evaluation & Results

ACCURACY & OVERALL BALANCE:

ACCURACY: ~65% (6,491 / 10,000)

- Better than the majority baseline (~62% “no-claim”), but still only moderate discriminatory power.
- The model is **clearly better at recognizing non-claimants than claimants**, reflecting the slight class imbalance.
- Logistic regression provides a **reasonable baseline classifier**, but it **under-detects future claimants**, which could be problematic if missing risky customers is costly.

CLASS-WISE PERFORMANCE

- No-claim class (0) – Precision ≈ 0.72, Recall ≈ 0.72, F1 ≈ 0.72
- The model correctly identifies most non-claimants and makes relatively few false “no-claim” predictions.
- Claim class (1) – Precision ≈ 0.53, Recall ≈ 0.53, F1 ≈ 0.53
- Around half of true claimants are missed, and about half of predicted claimants are actually non-claimants.

Confusion Matrix

		Predicted Class →	
		0	1
Actual Class ↑	0	4434	1800
	1	1662	2104

Logistic Regression Insights

Logistic regression has only moderate classification power, mainly because the relationship between the features and the “filed claim” outcome is nonlinear and the classes are somewhat imbalanced (more non-claimants than claimants). The model clearly performs **better on non-claimants** (≈ 0.72 precision/recall) than on claimants (≈ 0.53 precision/recall), so many future claimants are still missed.

Claimants are systematically under-detected, indicating that a single linear decision boundary is too simple to capture complex risk patterns such as interactions between health, lifestyle, and coverage variables. In a real underwriting setting this would be problematic, because missed claimants can be costly.

Performance is similar on training and test data, suggesting that the model generalizes reasonably well but is limited by its own capacity rather than overfitting.

Despite these limitations, logistic regression remains a **useful baseline classifier** and a highly **interpretable model**: its coefficients provide clear, directional insights into how each predictor is associated with the probability of filing a claim, even if more flexible models are needed for stronger predictive performance.

Tweedie GLM Test Evaluation & Results

MSE & Residual Deviance:

4,198,078.075

- Both metrics capture the **total unexplained variance** under the Tweedie loss.
- The value is high because claim amounts span a wide 0–70k+ range and contain many extreme outliers.

RMSE: 2,048.921

- The average prediction error is about **\$2,049** per policyholder.
- Similar to the other models, this reflects the difficulty of accurately predicting rare, very high-cost claims.

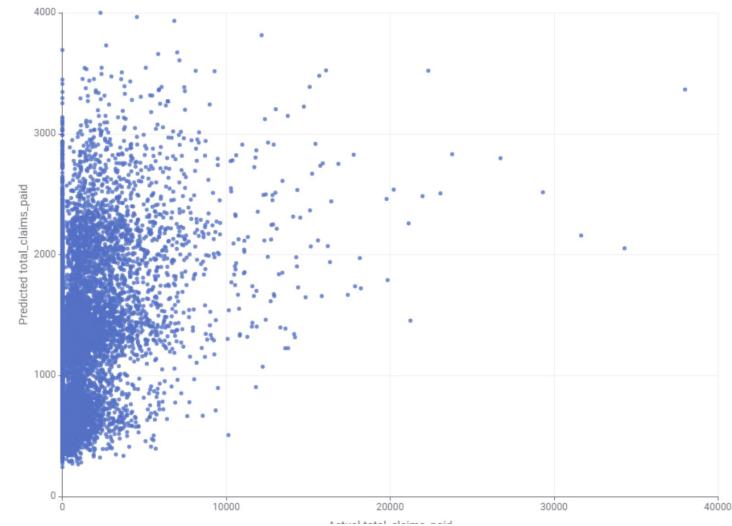
R²: 0.14

- The model explains around **14% of the variability** in total claims.
- This indicates **limited predictive power**: even with a Tweedie distribution tailored to insurance data, much of the claim variability remains unexplained.

RMSLE: 4.255

- The relatively high log-scale error shows that the model still makes large multiplicative errors, especially for larger claims.
- It confirms that high claims tend to be underestimated on the log scale.

GLM Tweedie objective: Actual vs Predicted total_claims_paid



- The Tweedie GLM predicts **low to moderate claim amounts fairly consistently**, with predictions concentrated in a band roughly between \$500 and \$2,500.
- For **higher actual claims**, predicted values rise more slowly and cluster below \$3–3.5k, so the largest claims are **systematically underpredicted**.

Tweedie GLM Insights

The testing results for the Tweedie GLM confirm that **predictive performance remains limited**, even with a distribution tailored to insurance data.

The model captures the general level of low and medium claims, but the relationship between predictors and total_claims_paid is still highly nonlinear and driven by rare, extreme outliers, which the Tweedie GLM struggles to fit precisely.

High-cost policyholders continue to be underpredicted, as seen from predicted values saturating around a few thousand euros while actual claims can reach much higher amounts.

The model behaves similarly on training and testing data, suggesting reasonable generalization but insufficient accuracy for precise pricing of individual policies.

Compared with the linear regression baseline, the Tweedie GLM offers a **more appropriate distributional assumption and similar error levels**, so it is best viewed as a **slightly refined but still baseline-level model** that remains highly interpretable for understanding overall claim cost drivers rather than as a high-accuracy prediction tool.

Random Forest Regression

Test Evaluation & Results

MSE & Residual Variance:

4,112,449.748

- Both metrics measure total unexplained variance in predictions.
- High due to extreme claim outliers and the wide 0–72k range.

RMSE: 2,027.918

- Average prediction error is around **\$2,028**.
- Reflects difficulty with rare high-cost claims.

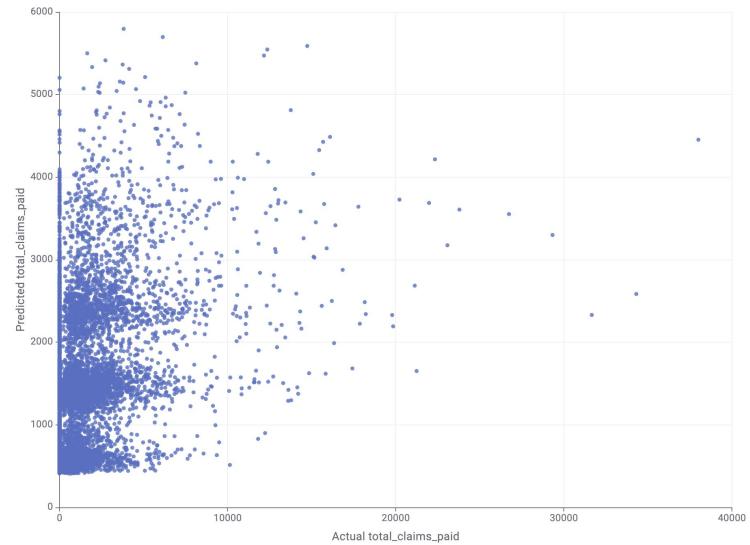
R²: 0.158

- The model explains about 16% of claim variability.
- Indicates limited predictive power because claim amounts are highly erratic and nonlinear.

RMSLE: 4.239

- High log-scale error. Confirms the model consistently underestimates the largest claims.

Random Forest Regression: Actual vs Predicted total_claims_paid



- **The model predicts low to moderate claim amounts reasonably well**, shown by the dense cluster of points around the lower ranges. Predictions stay within a relatively narrow band even as actual values vary.
- **High-cost claims are consistently underpredicted**, visible in the wide horizontal spread of points where actual values rise above €10k but predicted values plateau around €4–5k. This reflects the model's difficulty with rare, extreme claim amounts.

Random Forest Regression Insights

The testing results show that **Random Forest regression improves slightly over linear models but still has limited predictive power.**

By aggregating many trees, it can capture some nonlinear relationships and interactions between policyholder characteristics and claim amounts, which helps for low to medium claims.

However, the model still struggles with the highly skewed, heavy-tailed distribution of costs: **very high-cost policyholders are systematically underpredicted**, with predictions saturating around €4–5k even when actual claims are much larger.

The model behaves similarly on training and test data, suggesting that it generalizes reasonably well and is not severely overfitting, but the overall accuracy ($R^2 \approx 0.16$, $RMSE \approx €2,028$) remains modest.

Compared with linear regression, Random Forest offers **better flexibility and slightly lower errors**, making it a stronger benchmark for claim severity.

Nevertheless, its performance on extreme claims is still inadequate, so it is best viewed as a **more powerful but still imperfect baseline**, to be complemented by other models or business rules when pricing very high-risk policies.

Random Forest Classification

Test Evaluation & Results

Confusion-matrix metrics (per class)

- **No-claim (0):** Recall ≈ 0.65 , Precision ≈ 0.76 , F1 ≈ 0.70 – most non-claimants are correctly identified and predictions of “no claim” are relatively clean.
- **Claim (1):** Recall ≈ 0.66 , Precision ≈ 0.53 , F1 ≈ 0.59 – around one-third of claimants are still missed, and many predicted claimants turn out to be non-claimants.
- Overall accuracy is about **65%**, only modestly better than the 62% majority baseline.
- The Random Forest **improves class separation compared with a random classifier** and slightly over logistic regression, but performance is still moderate.
- The model is **more confident and accurate for non-claimants** than for claimants, which is a concern if the business cost of missing high-risk customers (false negatives) is high.

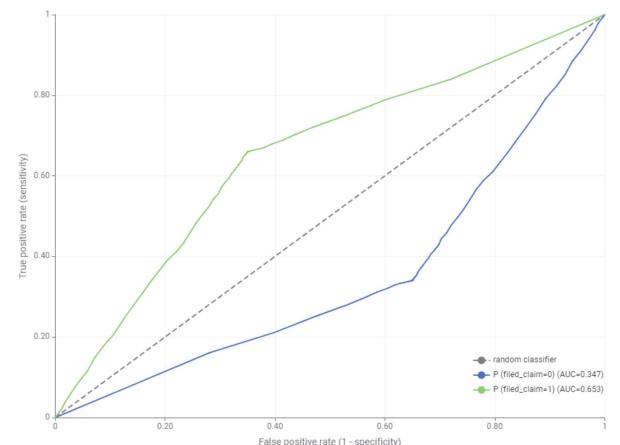
ROC & AUC

- The ROC curve for the claim class (*filed_claim* = 1) has **AUC ≈ 0.65** , clearly above the random classifier line but far from perfect, indicating **moderate ability to rank claimants above non-claimants**.
- The ROC for the non-claim class (*filed_claim* = 0, AUC ≈ 0.35) is the mirror image of this and mainly reflects the same information.

Confusion Matrix

		Predicted Class →	
		0	1
Actual Class ↓	0	4055	2179
	1	1284	2482

RF classification ROC curve



Gradient Boosting Regression Test Evaluation & Results

MSE & Residual Deviance:

4,094,609.731

- Both metrics measure the total unexplained variance in predictions.
- Still high due to extreme claim outliers and the wide 0–72k range, but slightly lower than Random Forest, indicating a small improvement.

RMSE: 2,023.514

- The average prediction error is about \$2,024 per policyholder
- Again reflects difficulty with rare very high-cost claims, with a marginally smaller error than the RF model.

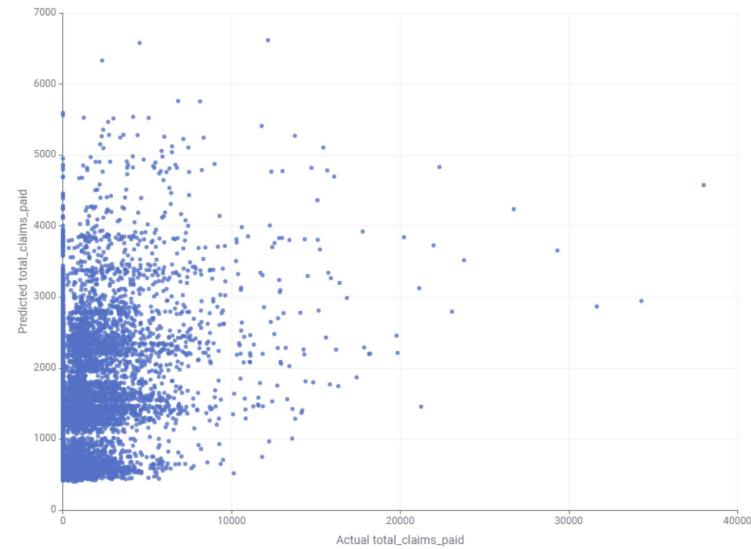
R²: 0.161

- The model explains about **16% of the variability** in claim amounts.
- This is only a small gain over Random Forest, showing that even with boosted trees, most claim variability remains unexplained.

RMSLE: 4.231

- High log-scale error, confirming that the model **underestimates the largest claims**, though slightly better than RF on the log scale.

GB Regression: Actual vs Predicted total_claims_paid



- The GB model predicts **low to medium claims reasonably well**
- **High-cost claims remain underpredicted**: above ~\$10k, predicted values mostly plateau around \$4–6k. gradient boosting lifts this ceiling slightly compared with RF but still struggles with rare, extreme claims.

Gradient Boosting Regression Insights

The testing results show that **gradient boosting is the strongest regression model in our study**. By sequentially combining many shallow trees, it captures more nonlinear effects and interactions than linear regression, Tweedie GLM, or Random Forest, leading to the lowest MSE/RMSE and the highest R².

It predicts **low to medium claim amounts relatively well**, but, like the other models, still **underestimates very high-cost policyholders**, reflecting the extreme skewness and noise in claim amounts.

The model performs similarly on training and test data, indicating good generalization and no severe overfitting, even with many trees.

Despite its remaining limitations on extreme claims, gradient boosting provides **the best overall predictive accuracy** in our analysis and serves as our **preferred model for claim severity**, while the simpler linear and Tweedie models remain useful as interpretable baselines.

Gradient Boosting Classification Test

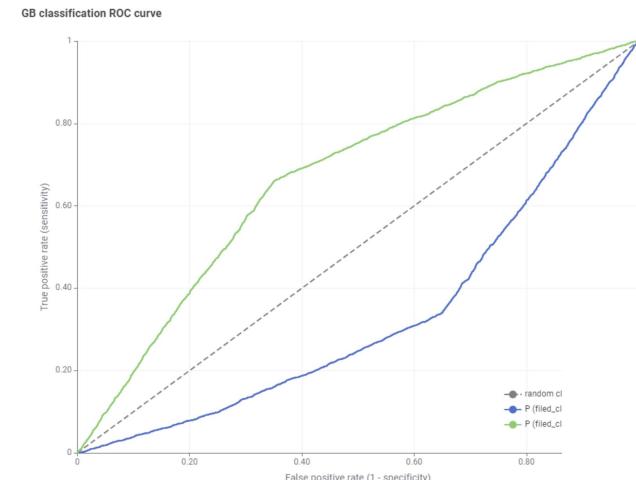
Evaluation & Results

Confusion-matrix metrics (per class)

- **No-claim (0):** Recall ≈ 0.65 , Precision ≈ 0.76 , F1 ≈ 0.70 – most non-claimants are correctly identified and “no claim” predictions are fairly reliable.
- **Claim (1):** Recall ≈ 0.66 , Precision ≈ 0.53 , F1 ≈ 0.59 – roughly one-third of claimants are still missed, and many predicted claimants are actually non-claimants.
- Overall accuracy is again around **65%**, only modestly better than the 62% majority baseline.
- Overall, **Gradient Boosting offers the best discrimination among our classification models**, with a small AUC and F1 improvement over Random Forest and logistic regression.
- However, it still performs noticeably better for **non-claimants than claimants**, so the risk of missing future claimants (false negatives) remains an important limitation.
- The ROC curve for the **claim class** (*filed_claim* = 1) has **AUC ≈ 0.67** , clearly above the random classifier line and slightly higher than for the Random Forest, indicating **moderate ability to rank claimants above non-claimants**.
- The ROC curve for the **no-claim class** (*filed_claim* = 0, AUC ≈ 0.33) is the mirror image and carries the same information.

ROC & AUC

		Predicted Class →	
		0	1
Actual Class ↑ ↓	0	4060	2174
	1	1296	2470



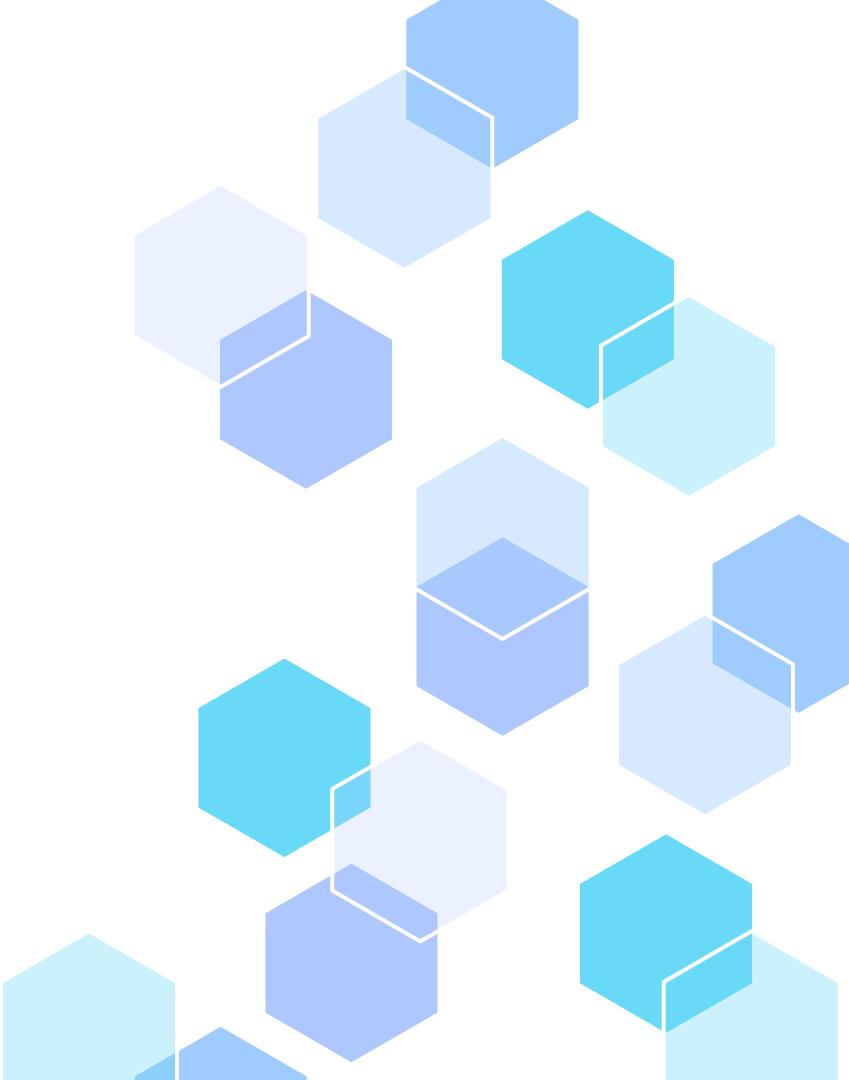
Gradient Boosting Classification Insights

The test results show that **gradient boosting is the strongest classifier in our analysis**. By combining many shallow trees, it captures more nonlinear risk patterns than logistic regression and slightly outperforms the Random Forest in both AUC (≈ 0.67 for claimants) and F1-score. The model distinguishes reasonably well between claimants and non-claimants, especially at low and medium risk levels.

However, performance is still only **moderate**: accuracy is about 65% and around one-third of true claimants are still missed, meaning false negatives remain a concern if missing risky customers is costly. Overall, gradient boosting is our **preferred classification model** for predicting whether a policyholder will file a claim, while simpler models remain useful for interpretability and as baselines.

06

Managerial Implications



Managerial Implications of Gradient Boosting Regression

GB Provides the Most Accurate Predictions for Claim Severity

- Gradient Boosting achieved the **lowest MSE and RMSE** across all models tested. This shows that GB is the most accurate numerical prediction of claim amounts.
- This improved accuracy allows managers to **price policies more precisely**. This can reduce the risk of systematic underpricing.
- More stable predictions on low- to medium-severity claims help actuarial teams build **more reliable reserve estimates** and improve forecasting accuracy.

Better Capture of Nonlinear Relationships

- Gradient boosting effectively models complex interactions (e.g., chronic conditions \times age \times hospitalization history), outperforming linear and Tweedie models.
- This ability to capture nonlinear structure enables more nuanced **risk segmentation**, allowing insurers to differentiate high-risk and medium-risk policyholders more effectively.
- Managers can incorporate these refined risk scores into underwriting guidelines or automated underwriting tools to support **more consistent and data-driven decision making**.

GB Enables More Effective Intervention and Wellness Strategies

- Since the model predicts low- to medium-cost claims reasonably well, insurers can identify clients trending toward higher risk **before** they reach catastrophic cost levels.
- Managers can use these predictions to design **preventive outreach programs**, such as:
 - Chronic condition management
 - Lifestyle health coaching
 - Targeted reminders for screenings or checkups
- These interventions reduce claims over time and improve customer health outcomes.

Better Resource Allocation in Claims and Fraud Management

- More accurate prediction of expected costs helps claims departments plan resources more efficiently.
- Although high-cost claims remain difficult to predict, improved accuracy in other ranges allows managers to:
 - Prioritize high-risk cases for early investigation
 - Allocate specialized case managers to complex claims
 - Flag anomalies that deviate from model predictions, potentially aiding **fraud detection**

Managerial Implications of Gradient Boosting Classification

Gradient Boosting Provides the Best Overall Classification Performance

- Gradient Boosting outperforms logistic regression and Random Forest in **AUC (~0.67 for claimants)** and **F1-score**, which makes it the strongest classifier in the study.
- This performance indicates a **moderate ability to rank policyholders by likelihood of filing a claim**. This is particularly valuable for underwriting and risk scoring.
- Managers can rely on this model as the *primary predictive tool* for identifying potentially high-risk customers.

Reliable Identification of Non-Claimants Helps Reduce Overpricing

- The model performs well for the **no-claim (0)** class:
 - Recall ≈ 0.65
 - Precision ≈ 0.76
 - F1 ≈ 0.70
- This means the model correctly identifies most low-risk customers. This prevents paying unnecessary premium increases.
- From a commercial perspective, accurate identification of non-claimants helps maintain **competitive pricing** and reduces the chance of losing low-risk policyholders to competitors.

Use Gradient Boosting as the Preferred Classification Model, with Human Oversight

- Despite its limitations, Gradient Boosting remains the **most effective model** tested for predicting claim occurrence.
- It should be integrated into underwriting workflows as a **risk-scoring supplement**, not as a standalone decision tool.
- Human actuarial judgment, regulatory constraints, and business rules must still guide decisions involving:
 - Pricing
 - Policy acceptance/declination
 - Risk mitigation strategies

Strategic Use Cases for Managers

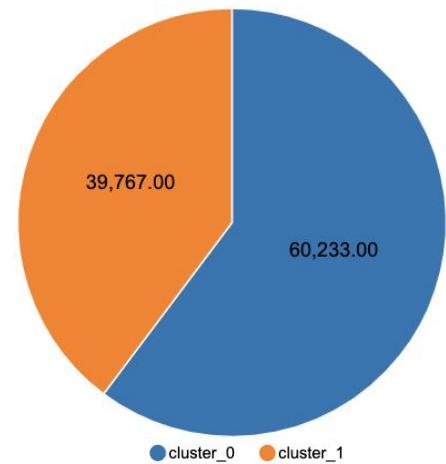
- **Risk-Based Pricing:** Use predicted probabilities to adjust premiums more accurately.
- **Preventive Outreach:** Identify medium-risk customers for wellness programs or early intervention.
- **Profit Protection:** Focus on reducing false negatives by tightening the threshold for classifying someone as “likely to claim.”
- **Resource Allocation:** Allocate case managers to high-probability claimants to reduce downstream costs.

Managerial Implications of PCA + k-means

The k-means algorithm allows us to divide the dataset into low-risk and high-risk individuals. **Cluster 0** represents the majority of policyholders (60233), which typically have lower BMI, age and other risk scores. **Cluster 1** is smaller (39767) but significantly higher-risk as the members of cluster 1 are older, more likely to smoke and have a higher BMI. This group contributes disproportionately to total medical costs.

From a **managerial standpoint**, this allows us to implement the following actions:

- **Prioritize preventive health programs** for high-risk members (Cluster 1) through targeted outreach and chronic-condition management.
- **Implement risk-adjusted pricing** that aligns premiums, deductibles, and copays with each cluster's risk profile.
- **Design differentiated insurance plans** for high-risk vs. low-risk groups to optimize coverage and cost control.
- **Offer wellness incentives and loyalty benefits** to low-risk members (Cluster 0) to support retention and promote healthy behavior.
- **Allocate medical and administrative resources efficiently** by focusing care management and telehealth on high-utilization individuals.
- **Monitor cluster shifts over time** to detect rising risk early and intervene before major claim costs occur.
- **Strengthen fraud detection** by identifying individuals whose behavior sharply deviates from their cluster norms.



Feature Importance Analysis

The feature importance results indicate that **chronic conditions** are by far the strongest driver of claim costs, accounting for **66.4%** of the model's explanatory power.

- Managers should prioritize chronic disease management programs, preventive care strategies, and targeted health interventions for policyholders with chronic illnesses.

Risk score is the second most influential variable (**21.5% importance**), confirming that internal or external risk indices are effective predictors for expected claim severity.

- Insurers should invest in improving their risk scoring methodologies and integrate them more deeply into pricing and underwriting processes.

Days hospitalized is also a meaningful predictor (**7.9% importance**), indicating that prior hospital admissions are strong indicators of future medical expenditures.

- Managers can use hospitalization history to refine risk segmentation, design case-management programs, and identify high-risk individuals for early intervention.

RowID	Relative Importance [# Number (Float)]	Scaled Importance [# Number (Float)]	Percentage [# Number (Float)]
chronic	313,568,608	1	0.664
risk_sc	101,366,296	0.323	0.215
days_h	37,192,580	0.119	0.079
Curren	8,850,049	0.028	0.019
Never	6,864,906	0.022	0.015
bmi	1,439,832.375	0.005	0.003
mental	538,982.875	0.002	0.001
income	445,161.125	0.001	0.001
hypert	324,429	0.001	0.001
proc_ir	295,635.969	0.001	0.001
provide	293,863.719	0.001	0.001
age	243,215.859	0.001	0.001

Variables such as **current smoker**, **never smoked**, and **BMI** contribute smaller but meaningful effects on claim costs.

- This highlights the value of lifestyle-based pricing strategies and wellness incentives (e.g., smoking cessation programs, weight-management initiatives).

The combined interpretation of variable importance suggests that **health status** and **prior utilization patterns** dominate cost prediction, whereas demographic variables have comparatively limited influence.

- Managers should allocate resources toward *medical risk management* rather than broad demographic underwriting adjustments.

Prioritizing Reduction of False Negatives

In insurance classification, it is crucial to minimize the amount of **false negatives**. A false negative occurs when the model predicts that a customer will **not** file a claim, but the customer **actually does**. Random Forest performs slightly better than Gradient Boosting at identifying actual claimants (i.e., reducing false negatives).

From a managerial standpoint, false negatives are far more costly than false positives because:

- They lead to **underestimating the policyholder's risk**.
- This results in **underpricing**, which directly increases **loss ratios**.
- Repeated underpricing across many customers can significantly **hurt profitability**.

In contrast, a **false positive** (predicting a claim when none occurs) results in **overpricing**, which is financially safer. Although overpricing might reduce customer satisfaction or retention among some segments, it does not create **direct financial losses** for the insurer.

For these reasons, insurance classification models should be optimized not just for overall accuracy but for **risk-averse decision thresholds** that favor identifying potential claimants, even if this increases the number of false positives. Because of that, managers should work closely with data science teams to:

- Adjust classification probability thresholds
- Prioritize metrics such as **recall** or **F2 score**, which heavily penalize missed risks
- And ensure the model aligns with the company's risk appetite and pricing strategy



THANK YOU!