OeNB Industry Lab - Documentation

Tristan Leiter

November 7, 2025

Contents

Chapter 1

Einleitung

1.1 Einleitung

Chapter 2

Exploratory Data Analysis

2.1 Overview

2.1.1 Description of the dataset

2.1.2 Description of the dataset

2.2 Data Splitting with Imbalanced Data

2.2.1 Class Imbalance

In this credit risk dataset, the target variable, y, is inherently imbalanced. The number of non-defaults (0) significantly outnumbers the number of defaults (1). This is a common and expected characteristic of credit risk data.

This imbalance poses a significant challenge for model development. If we were to use a simple random split to create our training, validation, and test sets, we would face a high risk of creating unrepresentative samples. For example, a small test set could, purely by chance, end up with a much higher or lower percentage of defaults than the original dataset—or, in the worst case, zero defaults.

2.2.2 The Solution: Stratified Sampling

To prevent this, we employ stratified sampling. This is a technique that ensures the original class distribution of the target variable is preserved in each of the new data splits.

Here is the reasoning for its use:

Guarantees Representation: Stratification forces the splits to maintain the original ratio of defaults to non-defaults. If 0.0865% of the original dataset are defaults, the training set, validation set, and test set will all contain approximately 0.0865% defaults.

Enables Reliable Evaluation: When the test set is representative, the performance metrics we calculate (like accuracy, precision, recall, and F1-score) are meaningful. Evaluating a model on a test set with a skewed default rate would give us a misleading and over-optimistic (or pessimistic) score.

Promotes Model Generalization: By training the model on a set that accurately reflects the real-world data distribution, we help it learn the patterns of both the majority (non-default) and minority (default) classes, leading to a more robust and generalizable model.

In summary, using stratified sampling on the y variable is a critical step to ensure our model is trained and evaluated on a reliable, representative foundation.

Chapter 3

Loss-function

3.1 Area under the curve (AuC)

The **Area Under the Curve (AUC)** is a single, aggregate metric that evaluates the performance of a binary classification model across all possible classification thresholds.

It is the area under the **ROC (Receiver Operating Characteristic) curve** , which plots the model's **Sensitivity** (True Positive Rate) against its **Specificity** (True Negative Rate) at every conceivable threshold.

- An **AUC of 1.0** represents a perfect model that can distinguish between positive and negative classes with 100% accuracy.
- An **AUC of 0.5** represents a model with no discriminatory power, equivalent to a random guess (as shown by the diagonal line in the plot).
- A good model will have an AUC well above 0.5.

Relevance to Credit Risk: The AUC is extremely valuable because it measures the model's ability to *rank* clients by risk. It answers the question: "If I pick a random defaulting firm and a random non-defaulting firm, what is the probability that my model gives a higher risk score to the defaulting firm?"

A high AUC (e.g., > 0.75) means the model is effective at separating the "bad" clients from the "good" ones. This is crucial before deciding on a specific *business-level threshold* (like "what probability score triggers a loan rejection?"). It tells us the model itself is fundamentally sound.

3.2 Recall

Recall, also known as **Sensitivity** or the **True Positive Rate (TPR)**, is a metric that measures a model's ability to identify all relevant instances of a class.

It is calculated using the following formula, based on the outputs of a **confusion matrix** at a specific threshold:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

- **True Positives (TP):** The number of firms that *actually defaulted* and were *correctly flagged* as defaults by the model.
- **False Negatives (FN):** The number of firms that *actually defaulted* but were *incorrectly flagged* as non-defaults. This is the most costly error in credit risk.

Relevance to Credit Risk: Recall answers the single most important question for a risk-averse institution: **"Of all the firms that actually defaulted, what percentage did we successfully catch?"**

In credit risk, the cost of a **False Negative** (missing a default) is extremely high, as it results in a direct financial loss. The cost of a **False Positive** (flagging a healthy firm as risky) is much lower—it might lead to a manual review or a rejected loan application (an opportunity cost).

Therefore, banks often prioritize high Recall. They are willing to accept a higher number of false alarms to ensure they minimize the number of defaults that slip through undetected. Setting a threshold to achieve a specific Recall (e.g., 95%) is a common, risk-averse strategy.

Chapter 4

Generalized linear models (GLM)

4.1 Overview

Chapter 5

Regularized GLMs

5.1 Description of the algorithm

The regularized estimator is obtained by solving the following optimization problem:

$$\hat{\beta}^{pen} = \arg\min_{\beta} \left\{ -\frac{1}{N} l(\beta, \phi | \mathbf{y}, \mathbf{X}) + \lambda P(\beta) \right\}$$

This objective function consists of two parts. In our logistic regression model, we predict the probability of default for each firm. We assume that the outcome (Y) is a Bernoulli random variable (a special case of the Binomial distribution). Then we model the probability of this specific firm defaulting given their features. This is reflected in the loss function:

1. **The Loss Function:** $-\frac{\mathbf{1}}{\mathbf{N}}\mathbf{l}(\beta, \phi | \mathbf{y}, \mathbf{X})$

   This term measures how well the model fits the data.

   - $l(\beta, \phi | \mathbf{y}, \mathbf{X})$ is the **log-likelihood function**. Since we assume a Bernoulli/Binomial outcome, this is the log-likelihood of the binomial distribution.
   - We want to find parameters (β) that *maximize* the likelihood of observing our data.
   - Maximizing the log-likelihood is equivalent to *minimizing* the **negative log-likelihood** (which is why the negative sign is there).
   - The $\frac{1}{N}$ term scales it by the number of observations N to get the **average negative log-likelihood**. This is also known as the **log-loss** or **cross-entropy loss**.

2. **The Penalty Term:** $\lambda\mathbf{P}(\beta)$

   This term, where $\lambda \geq 0$, penalizes the size of the coefficients to prevent overfitting. The `alpha` parameter controls the type of penalty:

   - `alpha = 1` corresponds to the **Lasso** estimator (L1 penalty).
   - `alpha = 0` corresponds to the **Ridge** estimator (L2 penalty).

The optimal regularization parameter, λ, is found by k-fold cross-validation. For example, for $k = 10$, the algorithm splits the training data into 10 equal-sized folds. It then iterates 10 times:

- In each iteration, it trains the model on 9 folds and evaluates it on the 1 held-out fold (e.g., train on folds 2-10, test on fold 1).
- This entire process is repeated for a sequence of different λ values.

Finally, the algorithm calculates the average error (e.g., AUC or deviance) across all 10 folds for each λ value. The λ that produces the lowest average error is chosen as the optimal value.

In the `glmnet` package, this is implemented by:

```
cv_model_logit <- cv.glmnet(x_train,
                            y_train,
                            family = "binomial",
                            alpha = 1)
```

The optimal λ value is stored in `cv_model_logit$lambda.min`, which is then used by the `predict()` function.

5.1.1 Description of the algorithm

Appendix A

Overview