Data exploration

```r
library(knitr)
library(here)
```

```
## Warning: Paket 'here' wurde unter R Version 4.4.3 erstellt

## here() starts at C:/Users/TristanLeiter/Documents/Privat/ILAB/CreditRisk_ML
```

```r
Path <- file.path(here::here(""))
Directory <- file.path(Path, "Documentation_Data_Exploration_Preparation.Rnw")
Charts_Directory <- file.path(Path, "03_Charts")
Charts_Data_Exploration_Directory <- file.path(Charts_Directory, "Data Exploration")

# knit2pdf(Directory)
```

1.1 Dependent variable

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.

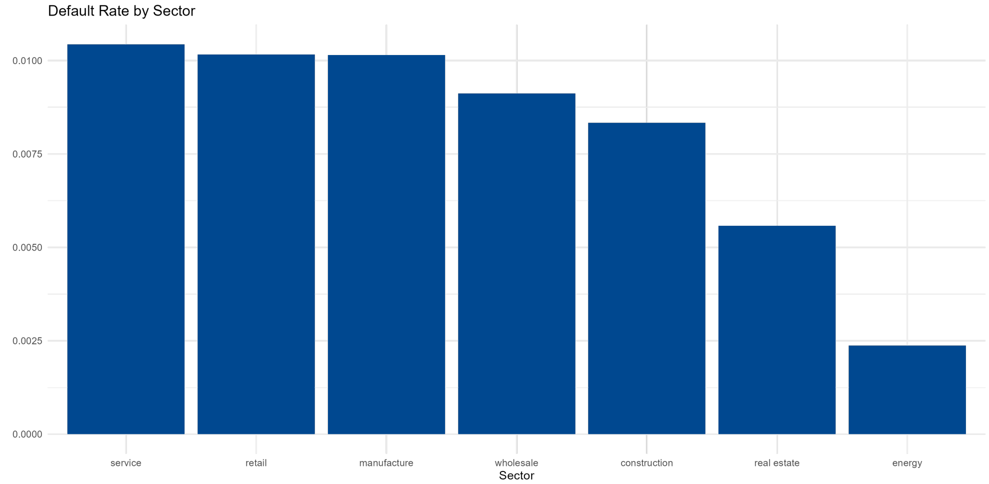
1.1.1 Data imbalance

To address potential sampling bias from the severe data imbalance, we employ **stratified sampling**. This technique ensures that the original class distribution of the target variable is preserved in both the training and test sets.

The primary advantages of this approach are:

- **Guaranteed Representation:** Stratification forces the data splits to maintain the original ratio of defaults to non-defaults. Given that 0.0865% of the original dataset consists of defaults, both the training and test sets will contain approximately this same percentage.
- **Reliable Evaluation:** When the test set is truly representative, the performance metrics we calculate (such as accuracy, precision, recall, and F1-score) are meaningful. Evaluating a model on a test set with a skewed default rate would give a misleading and overly optimistic (or pessimistic) performance score.
- **Improved Model Generalization:** Training a model on a set that accurately reflects the real-world data distribution helps it learn the distinct patterns of both the majority (non-default) and minority (default) classes. This leads to a more robust and generalizable model.

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



1.1.2 Implication

To address potential sampling bias from the severe data imbalance, we employ **stratified sampling**. This technique ensures that the original class distribution of the target variable is preserved in both the training and test sets.

The primary advantages of this approach are:

- **Guaranteed Representation:** Stratification forces the data splits to maintain the original ratio of defaults to non-defaults. Given that 0.0865% of the original dataset consists of defaults, both the training and test sets will contain approximately this same percentage.
- **Reliable Evaluation:** When the test set is truly representative, the performance metrics we calculate (such as accuracy, precision, recall, and F1-score) are meaningful. Evaluating a model on a test set with a skewed default rate would give a misleading and overly optimistic (or pessimistic) performance score.
- **Improved Model Generalization:** Training a model on a set that accurately reflects the real-world data distribution helps it learn the distinct patterns of both the majority (non-default) and minority (default) classes. This leads to a more robust and generalizable model.

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

1.1.3 Theory

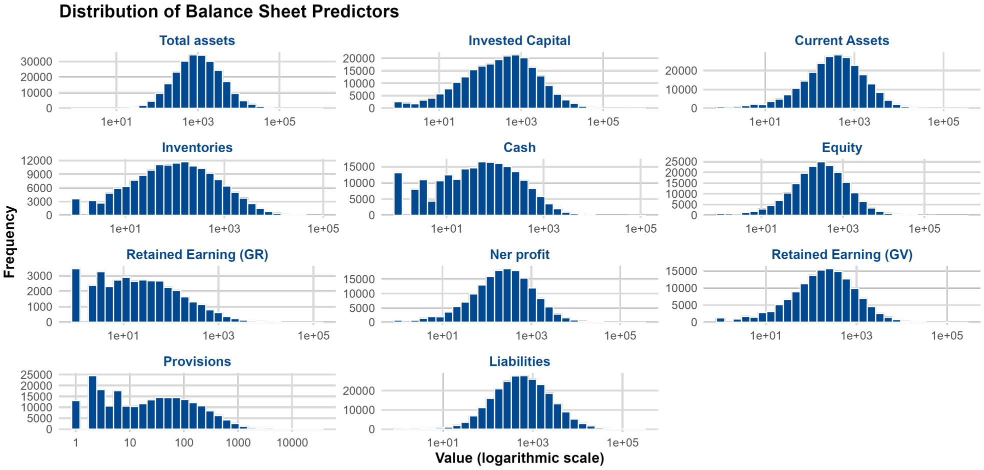
The dataset is partitioned into a training set and a testing set using the `rsample` package, which is part of the tidymodels framework. This process ensures that the resulting datasets are representative of the original data's class distribution, which is a critical step in handling data imbalance.

The used `initial_split` function does not simply take a random 80% of the data. When the `strata = y` argument is used, it follows a more intelligent **stratified sampling** algorithm:

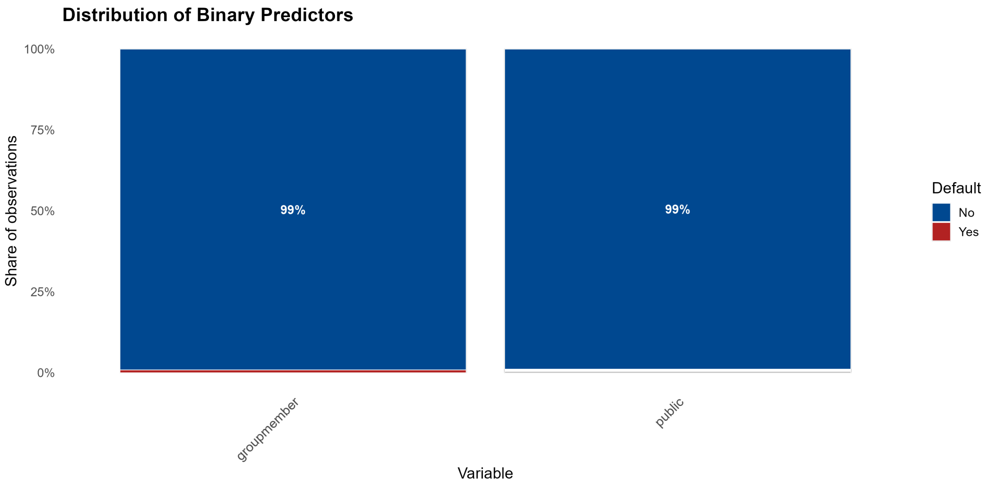
1. **Group Data by Stratum:** The function first identifies all unique values in the specified `strata` column (`y`). In this case, it creates two distinct groups: one for all the defaulting firms (`y = 1`) and another for all the non-defaulting firms (`y = 0`).
2. **Sample Proportionally within Each Group:** The function then takes an 80% sample from the defaulting firms group and a separate 80% sample from the non-defaulting firms group.
3. **Create the Training Set:** These two separately sampled subsets (80% of defaults and 80% of non-defaults) are combined to form the final training set (`Train`).
4. **Create the Test Set:** The remaining 20% of the defaulting firms and the remaining 20% of the non-defaulting firms are combined to form the final testing set (`Test`).

1.2 Distribution and bivariate Data Analysis

1.2.1 Distributions

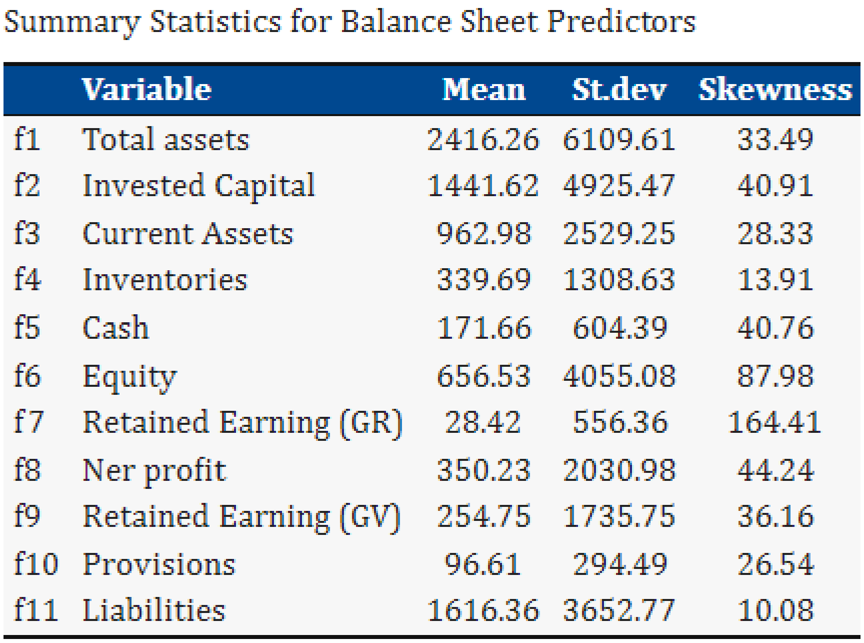


The histograms on the logarithmic scale show that all Balance Sheet Predictors are heavily right-tailed and spread across several orders of magnitude. Without the log scale, the values would be too uneven to visualize properly. After applying the log scale, the distribution of most variables reminds a log-normal distribution. Most observations are relatively small, while a small number of very large values stretch the distributions far to the right.



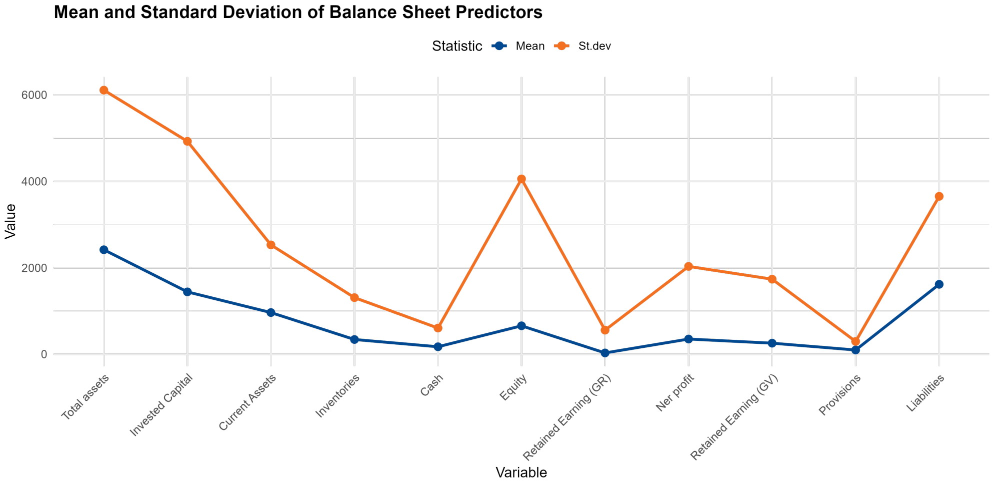
From the distribution plot of binary variables, we see that both group member and public are extremely imbalanced.

2. Summary statistics

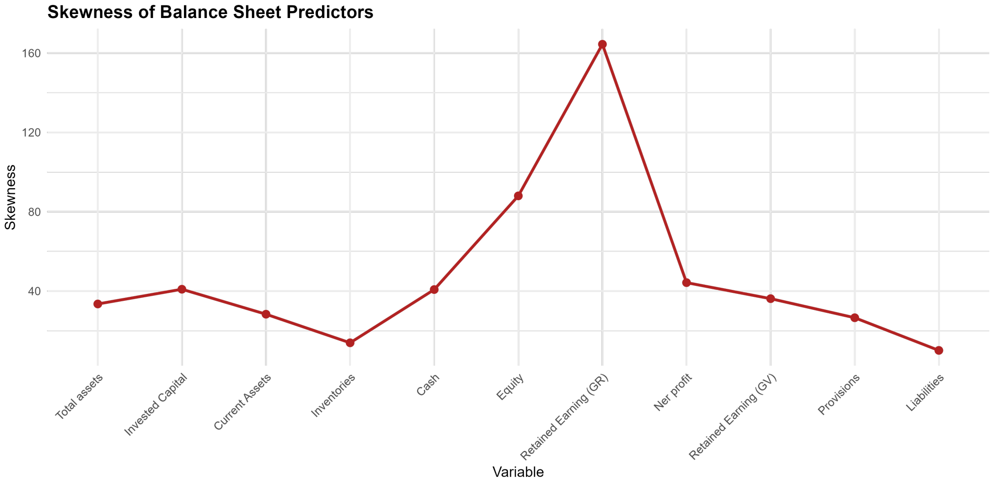


Summary Statistics for Balance Sheet Predictors

	Variable	Mean	St.dev	Skewness
f1	Total assets	2416.26	6109.61	33.49
f2	Invested Capital	1441.62	4925.47	40.91
f3	Current Assets	962.98	2529.25	28.33
f4	Inventories	339.69	1308.63	13.91
f5	Cash	171.66	604.39	40.76
f6	Equity	656.53	4055.08	87.98
f7	Retained Earning (GR)	28.42	556.36	164.41
f8	Ner profit	350.23	2030.98	44.24
f9	Retained Earning (GV)	254.75	1735.75	36.16
f10	Provisions	96.61	294.49	26.54
f11	Liabilities	1616.36	3652.77	10.08



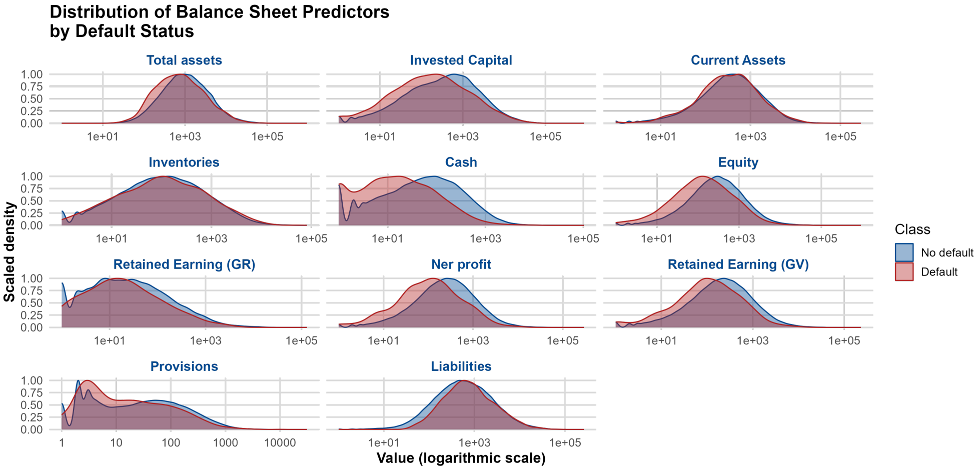
Mean and Standard Deviation Since the standard deviation is larger than the mean for every Predictor from the Balance Sheet. there is high variability within each variable. Variables: total assets (f1), invested capital (f2), equity (f6), net profit (f8), retained earning (f9) and Liabilities (f11) demonstrate especially large spreads, pointing to extreme values in the upper tail. Variables, such as cash (f5), retained earning (GR) (f7), and provisions (f10) show considerably lower difference between mean and standard deviation.The data is not centered around a typical value and includes many extreme observations that increase the overall spread.



Skewness

All variables show positive skewness, meaning they have long right tails. Equity (f6) and retained earnings (GV) (f7) stand out with very high skewness values, indicating that a few very large observations dominate the shape of the distribution. Inventories (f4) and liabilities (f11) are the least skewed but still clearly asymmetric. This consistent right-skewness across all predictors matches what we see in the plots: most values are concentrated near the lower end, with a small number of extreme values pushing the distributions far to the right.

1.2.2 Data Dependency



The plot shows the distribution comparison for Balance Sheet Predictors. We applied scaled density to properly visualize the data, without scaling the distribution of the default class would be mostly invisible due to the class imbalance.

Obvious separation:

1. Invested capital (f2). Defaulted companies tend to have lower invested capital on average compared to non-default companies. Default distribution has a slightly heavier right tail, meaning a few defaulting customers have unusually high values of invested capital.

2. Cash (f5), Net profit (f8), Retained Earning (GV) (f9). Defaulted companies tend to have lower cash, net profit and retained earning on average compared to non-default companies.

1.3 Feature selection

Before building the predictive model, it is essential to assess the predictive power of each independent variable. This helps in eliminating irrelevant features, understanding variable relationships, and focusing on the most promising candidates. We use the Information Value (IV), significance tests and evaluation of multicollinearity for this purpose.

1.3.1 Informational Value

The function follows a systematic statistical procedure to assign a single score—the Information Value—to each variable, quantifying its ability to separate the “good” outcomes (non-defaults, y=0) from the “bad” ones (defaults, y=1). Here is the step-by-step process:

1. **Binning (Grouping):** The function first discretizes each continuous independent variable into a set of bins or groups (e.g., by quantiles). For categorical variables, each category is treated as a separate group.
2. **Counting Goods and Bads:** For each bin of a variable (e.g., for the "Age: 20-30" bin), the algorithm counts:
   - The number of non-defaults.
   - The number of defaults.

3. **Calculating Proportions:** It then calculates the proportion of the total goods and total bads that fall into that specific bin:
   - **% Goods** = (# Goods in the bin) / (Total # Goods in the entire dataset)
   - **% Bads** = (# Bads in the bin) / (Total # Bads in the entire dataset)
4. **Calculating Weight of Evidence (WoE):** The WoE for each bin measures how much the evidence in that bin supports one outcome over the other. The formula is:

$$\text{WoE} = \ln\left(\frac{\%\text{Goods}}{\%\text{Bads}}\right)$$

   A large positive WoE means the bin is strongly associated with non-defaults, while a large negative WoE means it is strongly associated with defaults.

5. **Calculating Information Value (IV):** Finally, the total IV for the entire variable is calculated by summing a weighted value across all its bins. The formula is:

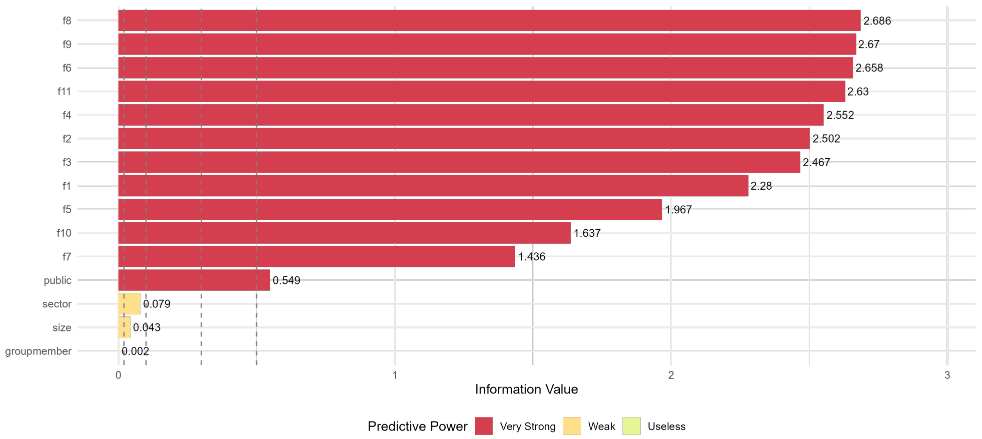
$$\text{IV} = \sum(\%\text{Goods} - \%\text{Bads}) \times \text{WoE}$$

   This single number represents the total predictive power of the variable.

Why This is Useful Before Modeling

Exploring the IV summary is not just a formality; it is a critical step in the exploratory data analysis (EDA) phase for several reasons:

- **Efficient Feature Selection:** IV provides a simple, powerful metric to rank all variables by their predictive strength.
  - **IV < 0.02:** Useless predictor.
  - **0.02 to 0.1:** Weak predictor.
  - **0.1 to 0.3:** Medium predictor.
  - **0.3 to 0.5:** Strong predictor.
  - **IV > 0.5:** Suspiciously high; may indicate data leakage or a variable that is too good to be true.
- **Understanding Variable Relationships:** While the total IV gives a summary, the detailed WoE for each bin of a variable reveals the nature of its relationship with the outcome. For example, by looking at the WoE for different age groups, you can see if the default risk increases, decreases, or follows a non-linear pattern as age changes.
- **Data Quality and Sanity Checks:** The IV calculation can expose data issues. A variable with an unexpectedly high IV might be a "leaky" variable (e.g., a variable that is only known *after* the default has occurred, pointing to a problematic quasi-seperation issue). Similarly, strange WoE patterns can highlight potential data entry errors or anomalies that need investigation.



1.3.2 Significance test

Numerical variables

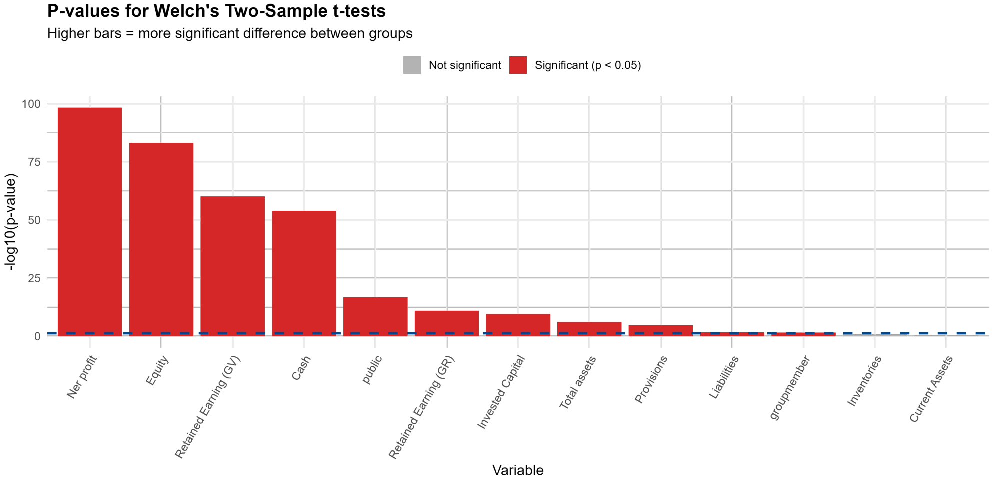
By calculating two group means and doing Welch's Two-Sample Test, we check weather each variables differs systematically between the two groups y=0 and y=1. For example if the group mean differ substantially, it indicated that x_j is associated with the response variable y and it could potentially be a useful predictor in a logistic regression, tree or any classification model.

To determine the difference in the group mean, we conduct two-sample t-test

H0: $E[x_j|y=0] = E[x_j|y=1]$

H1: $E[x_j|y=0] \neq E[x_j|y=1]$

We create a table and order the features by the lowest p-value. It means that its group means differ the most relative to the within-group variability.



The plots evaluated how well each predictor distinguish between the two groups. We use -log(10) in the charts for better visualization, taller bars indicate stronger difference between the group means.Low p-values indicate predictors that separate the classes and are useful in a model; high p-values indicate weak predictors that do not contribute meaningful information. Low p-value means that the variable contains a useful signal (information that meaningfully differs between groups) and can potentially improve the model's ability to predict the target.

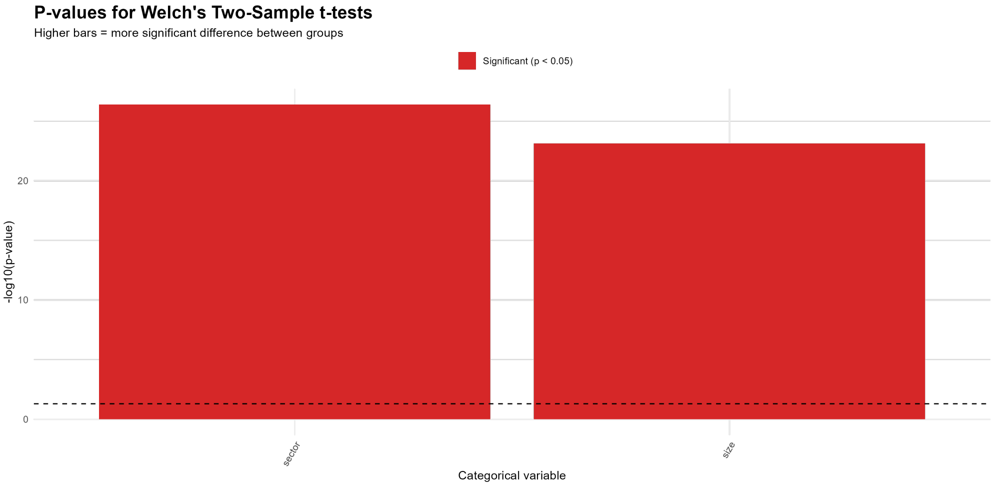
Variables with substantially different group means are Net profit, Equity, Retained Earnings (GV) asd Cash.

Categorical Variables

The analysis helps us understand whether the categorical variable is associated with default.For each categorical predictor, construct a contingency table against the binary outcome (default vs non-default) and performed a chi-squared test of independence. Variables with p-values below 0.05 were considered significantly associated with default, indicating that the distribution of default events differs across their categories.

To check whether category distributions differ between groups, we conduct a Chi-squared test

From the results (p-values being < 0.5) we see that there's strong association of categorical variables and the outcome y. Distribution differs significantly between the two groups.



The distribution of classes differs significantly across sectors and across size categories, which means that both variables have a useful signal for distinguishing between default and non-default observations.

1.3.3.b VIF

1.Regress each predictor X_j on all the other predictors:

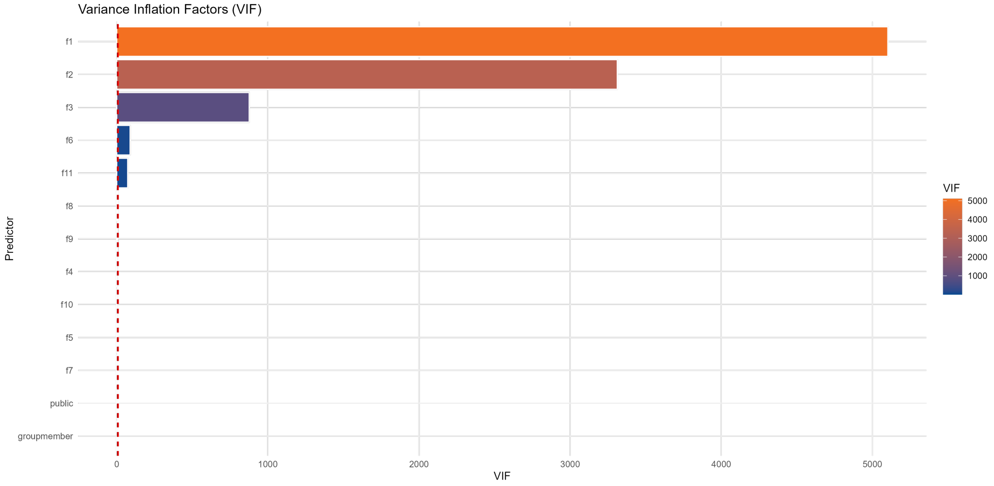
$$X_j = \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon.$$

2. Obtain the R_j^2 of this regression:
   - If $R_j^2 = 0$: the predictor is independent (no multicollinearity).
   - If $R_j^2 = 1$: the predictor is perfectly explained by the other predictors (full multicollinearity).
3. Compute the variance inflation factor (VIF):

$$\mathrm{VIF}_j = \frac{1}{1 - R_j^2}.$$

- Variance Inflation Factor (VIF) quantifies how much the variance of a regression coefficient is increased due to multicollinearity.

- A VIF of 1 indicates no correlation with other predictors, while higher values indicate that the predictor can be linearly explained by other predictors.
- VIFs above 5 suggest moderate multicollinearity, and values above 10 indicate severe redundancy and unstable parameter estimates.



Total assets, invested capital,current assets and cash show the highest VIF, which is obvious beacause total assets is a linear combination of all others.

1.3.4 Multicollinearity

After evaluating the individual predictive power of each variable using Information Value, the next critical step is to examine the relationships between the predictor variables themselves. **Multicollinearity** occurs when two or more independent variables are highly correlated with each other.

While multicollinearity does not necessarily reduce the overall predictive accuracy of a model, it poses a significant problem for interpretation. When variables are highly correlated, it becomes difficult for the model to distinguish their individual impacts on the target variable. This can lead to:

- Unstable coefficient estimates that can change dramatically with small changes in the data.
- High standard errors for the coefficients, making them seem statistically insignificant even when they are not.
- Difficulty in interpreting the model's logic, as the effect of one variable is confounded by the effect of another.

From the correlation matrix, several pairs of variables exhibit strong linear relationships (correlation > 0.70), which is a clear indicator of multicollinearity.

The analysis reveals two primary groups of highly inter-correlated variables:

1. **Group 1: `f1`, `f2`, `f6`, and `f11`**
   - The correlation between `f2` and `f1` is extremely high at **0.92**.
   - `f6` is also highly correlated with both `f1` (0.79) and `f2` (0.74).
   - `f11` is highly correlated with `f1` (0.74) and `f2` (0.66).

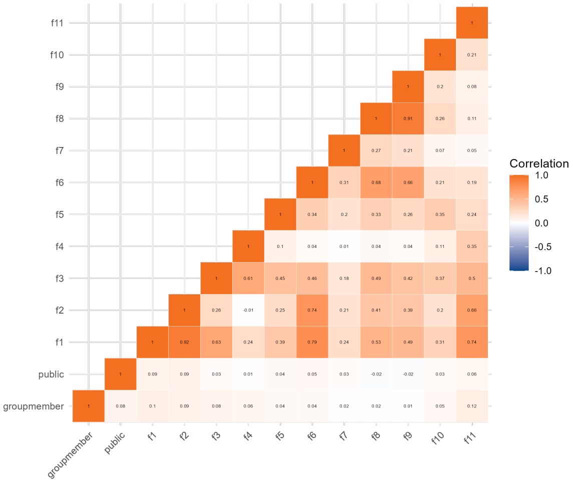
These variables likely measure a similar underlying economic or financial concept. Including all of them in a model would be redundant and problematic.

2. **Group 2: `f8` and `f9`**

   - The correlation between `f8` and `f9` is exceptionally high at **0.91**.

   These two variables are nearly interchangeable from a linear perspective.

Our strategy will be to select only one representative variable from each of these groups to proceed with modeling. The best candidate for selection is typically the variable with the highest Information Value (IV), as it holds the most individual predictive power.

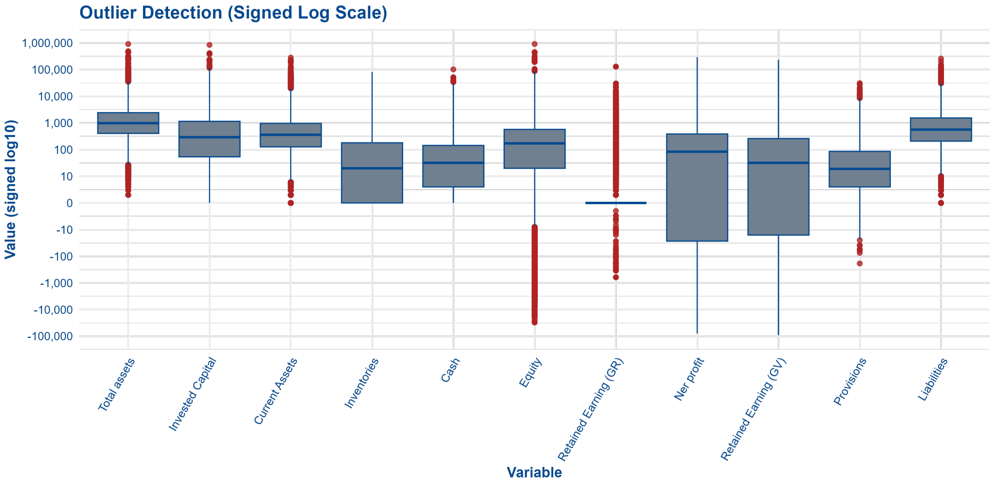


1.3.5 Feature selection

IV and multicollinearity:

1.5 Outliers

1.5.1 Outliers detection



Since the data set has negative values and zeros, we used signed log transformation to visualize the outliers.

- signedlog$(x) = sign(x) \cdot \log_{10}(1 + |x|)$.

The transformation doesn't change the rank or ordering of the data, it only rescales it. Signed log keeps negative values negative, but compresses their magnitude in the way as positive values. **Boxplots interpretation:**

All predictors from the Balance Sheet are highly skewed and heavy-tailed distributed. Equity (f6), Retained Earnings GR (f7) have dense clusters of extreme values. Presence of numerous outliers suggests considerable variability in the underlying Balance Sheet Data.

1.5.2 Robust scaling

One of the ways to handle the outliers is to use **Robust scaling method**.

1. Robust Scaler transforms centers the variable and IQR (interquartile range) scales the spread.

$$x_{scaled} = \frac{x - \text{median}(x)}{\text{IQR}(x)}$$

   . ' This way, it reduces the influence of extreme values without removing them, preserves the underlying structure. The method is suitable for using in tree-based models or boosting.

1.5.3 Weight of Evidence

WOE is a binning-based transformation primarily used in binary classification.

WOE is calculated as:

$$\text{WOE} = \ln\left(\frac{\%\text{Good}}{\%\text{Bad}}\right)$$

.

A large positive WOE means that the bin is strongly associated with non-dafeults, while a large negative value demonstrates the opposite. After calculating WOE, the original value is replaced by WOE of the bin it belongs to. The method is suitable for using in glm models and iff the outcome is binary.

1.6 Data Splitting

In our methodology, we partition the data into an 70% training set and a 30% final hold-out test set. All model development, including hyperparameter tuning and feature selection, is performed on the 70% training set using a technique called **k-fold cross-validation**. This approach is methodologically superior to a simpler three-way split (e.g., 50% train, 25% validation, 25% test) for several critical reasons.

1.6.1 Superiority Over a 50/25/25 Split

1. **Maximizes Data for Model Training:** The most significant drawback of a 50/25/25 split is that the model is only ever trained on 50% of the available data. In contrast, with an 70/30 split and k-fold cross-validation, the model is repeatedly trained on a much larger portion of the data (e.g., in 5-fold CV, it trains on 70% of the 70% training set, which is 64% of the total data, and this is done 5 times). More training data allows the model to learn more complex patterns and generalize better, reducing the risk of underfitting.

2. **Robust and Stable Performance Estimation:** In a 50/25/25 split, model performance is evaluated and tuned based on a single, static 25% validation set. The results on this single set can be highly variable depending on the "luck of the draw" of that particular split. A different random split could lead to different hyperparameter choices and a misleading performance estimate.

   Cross-validation solves this problem. By creating multiple (k) validation sets from the training data and averaging the performance metrics across all k folds, it provides a much more stable and reliable estimate of the model's true performance. It mitigates the risk of overfitting to a single validation set, leading to a more robust final model.

3. **More Efficient Use of Data:** With cross-validation, every single observation in the 70% training set is used as part of a validation set exactly once. This ensures that the entire training dataset is leveraged for both training and validation, making the process highly data-efficient. In a 50/25/25 split, the 25% validation set is never used for training, and the 50% training set is never used for validation, effectively "wasting" data from the perspective of each task.

1.6.2 Summary

The following table provides a direct comparison of the two approaches.

In conclusion, while a simple three-way split is faster, the 70/30 split combined with cross-validation is the modern gold standard. It produces more robust, reliable, and better-performing models by using the available data more effectively, which is a trade-off well worth the additional computational cost.