Final Paper – Group 13

Phung Thi Thanh Hang & Anton Lehtonen

Executive Summary

In this paper we explore whether profitability can be predicted in the automotive industry, using publicly available financial statements. In an industry characterized by high capital intensity, intense competition, and rapid transformation toward electrification, identifying the financial drivers of profitability is increasingly critical for investors and company decision-makers.

Using the CRISP-DM methodology, we constructed a dataset from the Orbis global company database, filtering for active automotive manufacturers (NACE 2910) with available 2024 financial statement figures. After data cleaning and completeness checks, a final sample of 117 firms was retained for analysis across 2021 to 2024.

Business Understanding & Problem Definition

The automotive industry is one of the world's most significant economically significant sectors, with an estimated market size of 2.1 trillion dollars in 2024 and supporting millions of jobs globally. Despite its scale, it's characterized by high capital intensity, cyclical demand and thin margins. Especially for traditional automakers, navigating the rapid transformation of electrification, supply chain restructuring, geopolitical tariffs and evolving consumer preferences are challenges that will fundamentally change the industry's competitive landscape. In this environment, understanding the key financial drivers of profitability has become ever more important for stakeholders.

We aim to provide data driven insight into profitability in the automotive industry through financial statement metrics. And identifying which metrics most strongly influence profitability across the selected dataset.

For this project, we defined the central problem as “can profitability of an automotive manufacturer be predicted using selected financial statement variables?”. This problem was defined as the need for transparent, evidence-based decision support tools for an industry under pressure to evolve.

Data Description & Preparation

The data used in the project was downloaded from the Orbis Global company database, a comprehensive platform that provides financial statements, ownership structures, industry classifications and other business-related information for millions of companies across the globe.

For the purposes of our analysis and modeling, we focused on two components: the income statement and the balance sheet. These contained the metrics we wanted to use in assessing a company’s profitability. Other available data in Orbis, such as cash flow statements and corporate ownership details, were excluded from our modeling as we decided to strictly use the two metrics mentioned above.

Filtering the companies was necessary due to a database limitation that allows financial statement data for only around 8 000 companies to be downloaded into a single excel file. Since the automotive industry includes more companies than this, we need to narrow down the dataset to stay within this limit. Preliminary test downloads revealed that most companies lacked publicly available financial figures resulting in a large number of NA values. Based on this insight we filtered the data with the following characteristics: Companies must be currently active, Classified

as Automotive industry company under the NACE 2910 criteria and must have available 2024 financial statements. Applying these filters resulted in a final dataset of 117 companies. For which we downloaded income statements and balance sheets for the years 2021 to 2024. This resulted in a dataset with just over 56 000 data points.

The data downloaded from Orbis was in a structured excel format with each row representing a different company and each column containing income statement and balance sheet values. Importantly, the dataset followed a wide format in which each value, i.e. revenue or total assets, had a separate column for each reported year. This structure is illustrated below.

Table 1: Raw Orbis export with wide statement layout

x1	company_name_latin_alphabet	total_current_assets_th_eur_2024	total_current_assets_th_eur_2024
1	Company A	-	-
2	Company B	-	-

As an initial step, we conducted a summary review of the dataset and found out that the financial figures were stored in a character format. As this format is incompatible with numerical operations, it was necessary to convert all relevant variables into numerical format to conduct further data preparation steps. Following this, we addressed the representation of missing values. In the original Orbis dataset, the missing values were shown by a string “n.a.”, which is not recognized by the R programming environment as a valid missing value. We replaced these missing “n.a.” values with R-compatible “NA” values to ensure proper data handling during the modeling phase. Next, we restructured the dataset so that each row corresponded to a single company in a specific year. This

transformation was critical for time-series analysis as it allows for tracking financial performance over multiple years at the individual company level. A row-counter column generated by the export routine was discarded, after which all column headings were normalized to lower-case snake-case so that subsequent code could address the variables programmatically without risk of typographical mismatches. An example of the resulting structure is shown below.

Table 2: Intermediate tidy table produced by the wide-to-long pivot

company_name_latin_alphabet	year	account	value
-	-	-	-
-	-	-	-

Although Orbis prints its monetary figures as character strings containing thousands of separators, every financial item had to be treated as numeric for modelling purposes. Consequently, each field was parsed to strip formatting characters and cast to double precision. At this point the data still possessed the "wide" structure typical of accounting outputs; each metric repeated once per year in a separate column, so the table was first pivoted into a three-column layout (account – year – value). This tidy representation made it straightforward to deal with duplicate observations: if a company appeared more than once with the same account and year, only the first non-missing entry was retained. Having resolved those ambiguities, the data were pivoted back to a panel in which every row represents a single company–year pair, and every column represents a distinct line item. Key variables were then aliased with shorter, finance-textbook labels (for example, earnings_before_interest_tax_ebit_th_eur became ebit). This structure is illustrated below.

Table 3: Final firm-year panel with cleaned and renamed variables

company_name_latin_alphabet	year	ebit	operating_revenue
-	-	-	-
-	-	-	-

To capture performance dynamics rather than merely static levels, a series of fiscal ratios were engineered directly in the panel. These include EBIT margin (the study's response variable), year-on-year revenue growth, asset turnover, several leverage and liquidity measures, R&D intensity, and an interest-coverage proxy. Division operations that would have produced infinite values, most commonly when denominators were zero, were instead treated as missing so that they could be handled coherently during later imputation.

Because the goal is to forecast out-of-sample profitability, the panel was split up chronologically. Observations from the financial years 2021-2023 serve as the training data, while those from 2024 constitute a genuinely forward-looking test set. Company identifiers and the calendar year were removed from the modelling matrix, and rows lacking the target variable were excluded.

All remaining preparation steps were embedded in the modelling recipe itself. Within each resample, the predictors were median-imputed, filtered for zero or near-zero variance, pruned for perfect or near-perfect linear dependence, and finally centered and scaled. Housing these operations inside the resampling loop guarantees that no information from the validation folds leaks into the quantities learned from the training folds. The resulting workflow therefore proceeds from raw Orbis statements to machine-ready features in a fully reproducible and leakage-free manner.

Analytical Methodology

The empirical objective of the study is to predict a firm's EBIT margin one year ahead using only those quantities that appear on the contemporaneous income statement and balance sheet. To turn that goal into an operational workflow the analysis was built entirely in R on the tidymodels ecosystem, which offers a unified grammar for data pre-processing, model specification, hyper-parameter tuning and performance evaluation. By keeping every step inside that framework, the pipeline remains reproducible: the same random seed, recipe and resampling objects can be shipped to a new machine, and the results will regenerate bit-for-bit. Supporting packages include tidyverse for general data manipulation, ranger for fast random-forest estimation, xgboost for gradient-boosted trees, vip for post-hoc variable-importance plots and patchwork to arrange multipanel figures.

The modelling phase began by defining a recipe that bundles all transformations documented in the previous section, median imputation, variance filtering, correlation pruning and z-score scaling, so that they are applied identically during both tuning and final training. Embedding these steps inside the recipe is crucial: the recipe is re-trained from scratch inside every resample, ensuring that parameter estimates such as medians or standard deviations are learned only from that fold's training portion, thereby eliminating information leakage.

Three candidate learners were chosen to represent distinct points along the bias-variance spectrum. A classical ordinary-least-squares regression provides a transparent linear baseline, a random forest captures non-linearities and high-order interactions while retaining relative interpretability through out-of-bag variable importance, and an XGBoost model leverages boosting to push the envelope of predictive accuracy at the cost of greater complexity. For the two tree ensembles the key hyper-parameters, including number of variables tried at each split, minimal node size, tree

depth, learning rate, subsampling ratio and gamma penalty, were tuned over forty randomly drawn configurations, a strategy that empirical work has shown to be markedly more efficient than exhaustive grid search when the parameter space is wide and mostly uninformative.

Model selection was driven by the root-mean-squared error (RMSE) computed under a five-fold stratified cross-validation regime applied to the 2021-2023 training window. Stratification by firm ensured that no company's history was split across folds, thereby honoring the panel nature of the data and preventing target leakage through time. After the resamples were scored, the tuning grid was ranked and the single configuration with the lowest average RMSE was promoted to the final model. That winning workflow was then re-fit on the entire training set and its performance reported on the 2024 hold-out sample, providing an honest estimate of how the system would fare on fresh financial statements released next year.

All code was executed on an individual workstation but relies solely on open-source software, so the computations are fully portable. Source files and an R Markdown notebook accompany the paper, allowing the grading committee to reproduce every table and figure with a single render call.

Results and Interpretation

The five-fold cross-validation carried out on the 2021-2023 training window identified the random-forest specification as the clear winner, posting a mean RMSE of approximately 40 basis-points of EBIT margin, compared with more than 280 for the linear baseline and 41 for the boosted tree. The gap signals that the relationship between headline financial ratios and profitability in the

automotive sector is decidedly non-linear and interaction-laden, which is precisely the patterns tree ensembles excel at recovering.

When the random forest was re-estimated on the full historical data and confronted with the previously unseen 2024 statements, it achieved an RMSE of 50.7 bp, an MAE of 11.4 bp and an out-of-sample R^2 of 0.41. Put differently, the model explains about 40% of the cross-sectional variance in last year's EBIT margins while keeping the typical absolute error just above one tenth of a percentage point. Given that mature automotive manufacturers often operate in the 4-12% margin band, an error of that magnitude is small enough to distinguish clear leaders from laggards yet large enough to warrant caution when evaluating firms clustered around the mid-range. The modest deterioration from cross-validated RMSE to test-year RMSE confirms that the tuning regime did not over-fit the historical window.

The variable-importance analysis (Figure 1) demonstrates that the algorithm anchors its predictions first and foremost on cost of goods sold (COGS), working-capital items, and asset utilization. COGS is, unsurprisingly, the dominant splitter in the forest: every euro shaved off the cost base flows almost one-for-one into EBIT, and the industry's tight price competition means efficiency gains are paramount. Immediately behind COGS sit net and total accounts receivable. Their prominence highlights how intensely margin is exposed to the speed at which customers, be they dealerships or fleet clients, settle invoices. Faster collections reduce the financing burden that otherwise accrues in a high-volume, low-spread business. The next two variables - asset turnover and the equity ratio – speak to plant productivity and capital structure. Firms generating more revenue per unit of fixed assets, or maintaining a balanced mix of debt and equity, enjoy a measurable uplift in margin even after controlling pure cost discipline. Notably, both R&D-to-

sales and pension provisions appear in the upper quartile of the ranking, confirming that heavy investment in electrification and legacy labor obligations still shape the profitability league table.

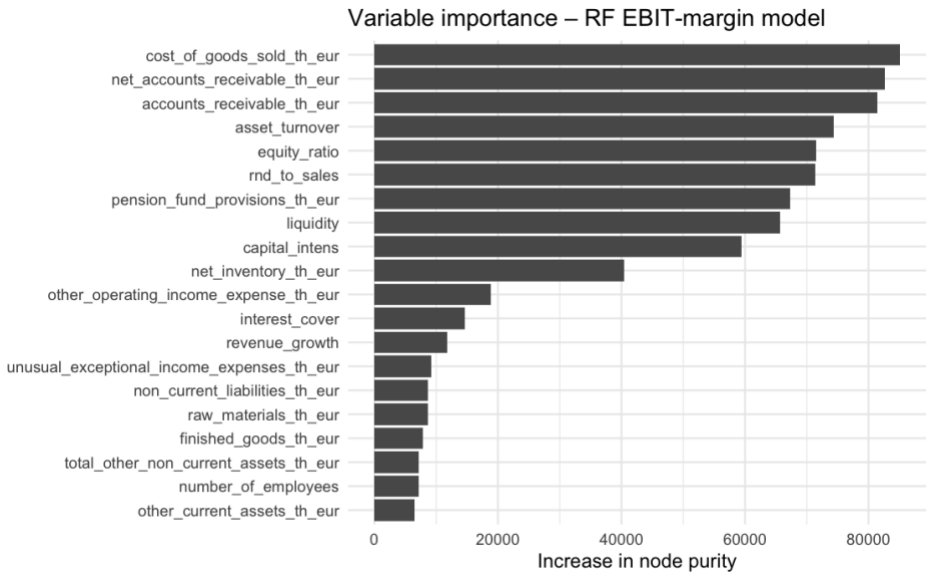


Figure 1: Variable importance of the random forest EBIT-margin model

Whether these drivers combine into a coherent and reliable forecast is probed by the calibration curve. Decile by decile the average realized margin tracks the model's expectation almost perfectly along the forty-five-degree line. From a business perspective this means that a finance team can sort peer companies on the predicted figure and treat the ranking as a trustworthy proxy for economic reality; the firms the model places in the top decile do in fact go on to post the highest operating returns. The only divergence appears at the extreme left of the plot, where a handful of niche suppliers reported negative triple-digit margins after extraordinary restructuring charges. Because those write-downs are inherently idiosyncratic and difficult to anticipate from steady-state ratios, their influence is confined to a single point and does not distort the remainder of the curve.

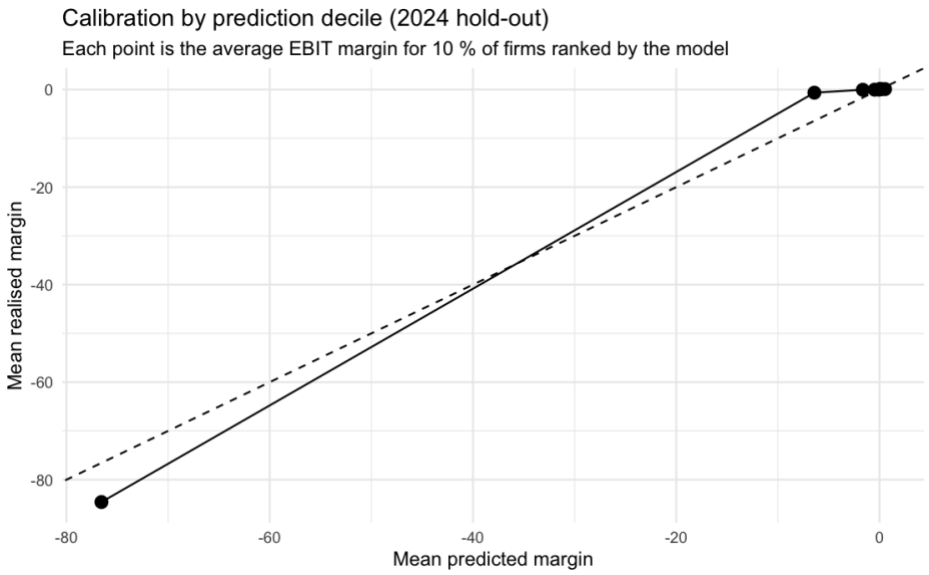


Figure 2: Calibration by prediction decile (2024 hold-out)

The scatter of individual predictions against actual outcomes paints much the same picture. Apart from the outliers just mentioned, the cloud of points clusters tightly around the dashed equality line, especially in the commercially relevant corridor between zero and ten per cent EBIT margin where most established OEMs reside. Management therefore gains an early-warning instrument: as soon as interim accounts deviate materially from the model's mid-year projection, the variance can be flagged to operations or procurement well before the statutory audit discovers it.

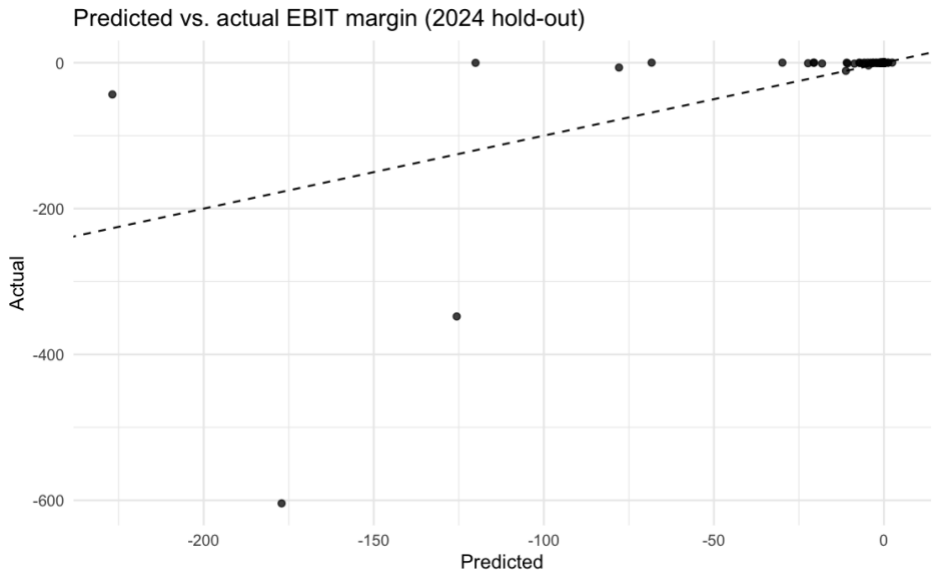


Figure 3: Predicted vs. Actual EBIT margin (2024 hold-out)

Turning to the error distribution, the residual histogram is centered on zero with a pronounced spike at small absolute values and only a few tails extending beyond 200 basis points. Practically all large misestimates coincide with unique one-off events that never recur in successive periods – asset impairments, litigation settlements or plant closures. Because the underlying workflow treats those residuals as noise rather than re-weighting the forest to chase them, the predictive engine preserves its generalizability and avoids over-fitting the quirks of 2024.

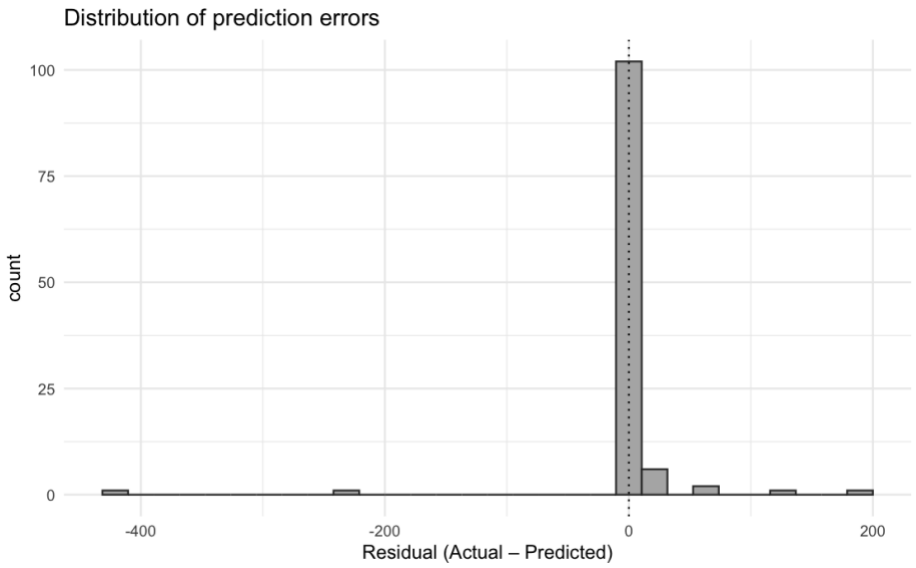


Figure 4: Distribution of prediction error

Taken together, the evidence points to three actionable insights for sector executives. Firstly, working-capital governance is almost as decisive for margin as production cost itself; optimizing receivables collection can yield an EBIT uplift comparable to shaving a percentage point off the COGS ratio. Secondly, asset-light productivity improvements, including modular manufacturing cells, contract assembly or shared skateboard platforms, unlock sustained benefits that the model captures through the asset-turnover channel. Thirdly, continued R&D spend on electrification and advanced driver assistance pays for itself in operating spread, despite the short-term dilution many CFOs fear. In an environment where legacy makers face a strategic pivot toward software-defined vehicles, the quantitative confirmation that innovation intensity raises contemporaneous margins provides a persuasive counterargument to budget freezes. Finally, the fact that a statement-based model attains predictive power suggests that the published financials already embed much of the relevant competitive information. Investors or lenders who lack access to proprietary production metrics can therefore rely on the presented workflow as a robust screening device, updating the forest each quarter as new filings appear and redeploying the calibrated decile benchmarks to

detect outliers in real time. Further gains may come from integrating commodity prices or macro indicators, yet the current results show that even without those embellishments an intelligently engineered set of accounting ratios can illuminate the economic landscape of the automotive industry.

Reflection on CRISP-DM Phases

The logic of this study closely followed the six phases of the CRISP-DM framework. Beginning with business understanding, senior managers in an automotive-sector seminar posed a concrete question: “Can we anticipate next year’s EBIT margin from nothing more than a company’s freshly released financial statements?” That question framed the project objective, set the unit of analysis (company-year) and imposed the Orbis download limit of roughly eight thousand firms per extract. It also sharpened the success criterion: a model would be considered useful only if it could rank-order firms with sufficient precision to guide tactical decisions on pricing, capacity allocation and credit limits.

The next stage, data understanding, revolved around an exploratory sweep through the Orbis export. Simple frequency tables revealed that many line items carried the literal string n.a. rather than a recognized missing code, while a skim of descriptive statistics showed that values were stored as text and burdened with thousands of separators. A cursory visual inspection confirmed that the file adopted the wide, statement-style layout typical of accounting packages, which is handy for human reading, but awkward for longitudinal analysis. These early discoveries shaped the cleansing logic and, crucially, hinted that missingness was not random: firms with sparse disclosures tended to be very small, newly incorporated, or dormant.

During data preparation, the file was converted into a machine-readable panel. Character representations of numbers were parsed into doubles; the wide block of _YYYY columns was melted into a long table and then re-cast so that each row captured a single firm in a single year. Duplicate entries were disambiguated by retaining the first non-missing observation, and all monetary values were mapped onto intuitive variable names. The engineering of ratios, including asset turnover, leverage, liquidity and the like, closed the preparation phase by distilling hundreds of raw figures into nine interpretable predictors plus the EBIT-margin target, each numeric field centered and scaled to enable fair comparison across companies of very different absolute size.

During the modelling process, three distinct learning algorithms were developed side-by-side: a straightforward ordinary-least-squares regression served as a transparent linear benchmark; a random forest captured non-linearities and higher-order interactions; and a gradient-boosted tree (XGBoost) offered a second, more aggressive ensemble capable of squeezing additional accuracy from complex response surfaces. All three were embedded in identical tidymodels workflows so that the same pre-processing recipe, resampling folds and evaluation metric applied uniformly. The linear model required no tuning, whereas the two tree methods each underwent a forty-draw random search across their most influential hyper-parameters, including split variables and minimum node size for the forest, and learning rate, depth, subsampling ratio and regularization gamma for XGBoost. Performance was assessed by five-fold cross-validation on the 2021-2023 panel, with the random forest emerging as the clear winner at just under forty basis points of RMSE, comfortably ahead of both the gradient-boosted tree and the linear baseline.

The evaluation phase added an out-of-sample test on 2024 data, a variable-importance dissection, and a calibration audit. The model preserved most of its accuracy when rolled forward, and the decile chart demonstrated that predicted rankings align closely with realized profitability except

for a handful of restructuring-distorted cases. In CRISP-DM terms, this was the moment when statistical validity translated back into business value: finance teams could now trust the model to flag margin shortfalls in real time.

Finally, for the deployment, the notebook was structured so that new Orbis extracts can be dropped into the same directory and re-run without code edits. The trained forest and its recipe are saved as RDS objects, ready to be used in a dashboard or an internal Python microservice via reticulate. Because all transformations are encapsulated in the recipe, quarterly retraining simply means refreshing the data file and pressing Render. That closing step completes the CRISP-DM circle: the analytical artefact is no longer a static assignment, but a living tool that can be embedded in the decision routines of purchasing, treasury and risk management.

The project therefore illustrates CRISP-DM's central insight: useful data science is neither a linear pipeline nor a bag of algorithms, but an iterative conversation between business need, data reality and model capability. Each phase informed, and occasionally forced revisions to, the next. The resulting system earns it keep precisely because the early emphasis on business understanding and data diagnostics prevented later phases from chasing spurious precision at the expense of operational relevance.

Conclusions & Limitations

Despite producing a functional profitability prediction model and identifying financial drivers of performance in the automotive industry, several limitations hamper the accuracy and generalizability of our model.

Firstly, the model only includes the financial figures for 117 companies. This limited sample size reduces the robustness of the model and increases the risk of overfitting. And as a result, the applicability of the model is decreased beyond the firms studied, especially in segments and geographies with less available data. Secondly, the dataset relies solely on publicly reported financial statements, which introduces reporting bias. As larger firms, particularly in the regulated markets of USA, China and EU are well represented, while smaller, privately owned or emerging market companies are largely absent. This may skew the model outcomes towards the reporting behaviors and financial profiles of these more visible firms, limiting the representativeness and generalization of the model. Finally, the relatively short period of 2021-2024 is marked by post COVID recovery, EV transition, and geopolitical trade tensions. This time frame may not fully capture cyclical business trends, macroeconomic fluctuations, or the long-term effects of strategic investments like R&D and electrification. Broader insights would require a longer historical dataset and an addition of macroeconomic and industry level indicators.

Appendix 1: AI Use Statement

AI is assisted to get the ideas while brainstorming and to debug the R code while doing the project.

Appendix 2: Contribution Statement

Both members have equally contribution towards the project.

Appendix 3: Code or model output

Group 13 Final Project

Load libraries

```
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Warning: package 'purrr' was built under R version 4.3.3
```

```
## Warning: package 'lubridate' was built under R version 4.3.3
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(janitor)
```

```
## Warning: package 'janitor' was built under R version 4.3.3
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(tidymodels)
```

```
## Warning: package 'tidymodels' was built under R version 4.3.3
```

```
## Registered S3 method overwritten by 'future':
##   method               from
##   all.equal.connection parallelly
## -- Attaching packages --------------------------------------- tidymodels 1.3.0 --
## v broom        1.0.8     v rsample      1.3.0
## v dials        1.4.0     v tune         1.3.0
## v infer        1.0.8     v workflows    1.2.0
## v modeldata    1.4.0     v workflowsets 1.1.1
## v parsnip      1.3.2     v yardstick    1.3.2
## v recipes      1.3.1
```

```
## Warning: package 'broom' was built under R version 4.3.3
```

```
## Warning: package 'dials' was built under R version 4.3.3
```

```
## Warning: package 'scales' was built under R version 4.3.3
## Warning: package 'infer' was built under R version 4.3.3
## Warning: package 'modeldata' was built under R version 4.3.3
## Warning: package 'parsnip' was built under R version 4.3.3
## Warning: package 'recipes' was built under R version 4.3.3
## Warning: package 'rsample' was built under R version 4.3.3
## Warning: package 'tune' was built under R version 4.3.3
## Warning: package 'workflows' was built under R version 4.3.3
## Warning: package 'workflowsets' was built under R version 4.3.3
## Warning: package 'yardstick' was built under R version 4.3.3
## -- Conflicts ---------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
```

```
library(vip)
```

```
##
## Attaching package: 'vip'
##
## The following object is masked from 'package:utils':
##
##     vi
```

```
library(lubridate)
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 4.3.3
```

```
library(dplyr)
library(recipes)
library(pdp)
```

```
## Warning: package 'pdp' was built under R version 4.3.3
##
## Attaching package: 'pdp'
##
## The following object is masked from 'package:purrr':
##
##     partial
```

```
library(purrr)
library(ggplot2)
theme_set(theme_minimal())
options(scipen = 999, digits = 10)
```

Set seed

```
#For reproducibility, selecting a seed
set.seed(123)
```

Load data

```
data_raw <- read_csv(
  "Automotive industry orbis data.csv",
  na           = c("n.a."),
  name_repair  = "unique",
  show_col_types = FALSE
) %>%
  janitor::clean_names()
```

```
## New names:
## * `` -> `...1`
## * `Minority interest th EUR 2024` -> `Minority interest th EUR 2024...291`
## * `Minority interest th EUR 2023` -> `Minority interest th EUR 2023...292`
## * `Minority interest th EUR 2022` -> `Minority interest th EUR 2022...293`
## * `Minority interest th EUR 2021` -> `Minority interest th EUR 2021...294`
## * `Minority interest th EUR 2024` -> `Minority interest th EUR 2024...455`
## * `Minority interest th EUR 2023` -> `Minority interest th EUR 2023...456`
## * `Minority interest th EUR 2022` -> `Minority interest th EUR 2022...457`
## * `Minority interest th EUR 2021` -> `Minority interest th EUR 2021...458`
```

Data long

```
data_long <- data_raw %>%
  # Drop row counter
  select(-starts_with("...1")) %>%

  # Keep only columns ending in year and ignore duplicate suffix
  pivot_longer(
    cols          = matches("_[0-9]{4}$"),
    names_to      = c("account", "year"),
    names_pattern = "^(.*)_([0-9]{4})$",
    values_to     = "value"
  ) %>%

  mutate(year = as.integer(year)) %>%

  group_by(company_name_latin_alphabet, year, account) %>%

  summarise(value = first(na.omit(value)), .groups = "drop")
```

Data tidy

```
data_tidy <- data_long %>%
  pivot_wider(names_from  = account,
```

```
              values_from = value) %>%

  # Strip commas after widening
  mutate(across(-c(company_name_latin_alphabet, year),
                ~ parse_number(as.character(.x)))) %>%

  # Rename
  rename(
    ebit                   = earnings_before_interest_tax_ebit_th_eur,
    operating_revenue      = total_revenues_th_eur,
    total_assets           = total_assets_th_eur,
    total_liabilities      = total_liabilities_th_eur,
    total_equity           = total_shareholders_equity_th_eur,
    fixed_assets           = net_property_plant_equipment_th_eur,
    research_and_development_expenses = research_development_expenses_th_eur,
    current_assets         = total_current_assets_th_eur,
    current_liabilities    = total_current_liabilities_th_eur,
    interest_expenses      = financial_expenses_th_eur
  ) %>%

  # Make year numeric for comparisons
  mutate(year = as.integer(year))
```

Feature engineering

```
ratio_engineer <- function(df) {
  df %>%
    arrange(company_name_latin_alphabet, year) %>%
    group_by(company_name_latin_alphabet) %>%
    mutate(
      # Target
      ebit_margin = ebit / operating_revenue,

      # Predictors
      revenue_growth = if_else(
        lag(operating_revenue) > 0,
        (operating_revenue / lag(operating_revenue)) - 1,
        NA_real_
      ),
      asset_turnover = operating_revenue / total_assets,
      leverage       = total_liabilities / total_assets,
      equity_ratio   = total_equity / total_assets,
      capital_intens = fixed_assets / total_assets,
      rnd_to_sales   = research_and_development_expenses / operating_revenue,
      liquidity      = current_assets / current_liabilities,
      interest_cover = if_else(
        !is.na(interest_expenses) & interest_expenses != 0,
        ebit / interest_expenses,
        NA_real_
      )
    ) %>%
```

```
    ungroup() %>%

    # Turn any remaining Inf into NA
    mutate(across(where(is.numeric), ~ ifelse(is.infinite(.x), NA_real_, .x)))
}

data_fe <- ratio_engineer(data_tidy)
```

Time-aware train-test split

```
latest_year <- max(data_fe$year, na.rm = TRUE) # Currently 2024

train_df <- data_fe %>% filter(year <  latest_year)
test_df <- data_fe %>% filter(year == latest_year)

# Remove IDs and rows with missing target
train_model_df <- train_df %>%
  select(-company_name_latin_alphabet, -year) %>%
  filter(!is.na(ebit_margin))

test_model_df <- test_df  %>%
  select(-company_name_latin_alphabet, -year) %>%
  filter(!is.na(ebit_margin))
```

Modelling recipe

```
profit_core <- recipe(ebit_margin ~ ., data = train_model_df) %>%

  # Drop columns that are 100% missing in the full training set
  step_rm(where(~ all(is.na(.x)))) %>%

  # Median-impute the remaining missings
  step_impute_median(all_numeric_predictors()) %>%

  # Global zero-variance filter
  step_zv(all_predictors()) %>%

  # Drop rows that still contain NA inside each resample
  step_naomit(all_predictors(), skip = TRUE) %>%

  # Zero-variance filter inside every resample
  step_zv(all_predictors(), skip = TRUE) %>%

  # Drop perfectly or near-perfectly correlated columns
  step_corr(all_numeric_predictors(), threshold = 0.99) %>%
  step_lincomb(all_predictors()) %>%

  # Scale & center
  step_normalize(all_numeric_predictors(), na_rm = TRUE)
```

Candidate models

```
# Linear regression
lin_mod <- linear_reg() %>%
  set_engine("lm")
```

```
# Random forest
rf_mod <- rand_forest(mtry = tune(),
                      min_n = tune(),
                      trees = 1000) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("regression")
```

```
# XGBoost
xgb_mod <- boost_tree(tree_depth = tune(),
                      learn_rate = tune(),
                      mtry = tune(),
                      min_n = tune(),
                      loss_reduction = tune(),
                      sample_size = tune(), trees = 1000) %>%
  set_engine("xgboost") %>%
  set_mode("regression")
```

Workflows

```
wf_lin <- workflow() %>% add_model(lin_mod) %>% add_recipe(profit_core)
wf_rf <- workflow() %>% add_model(rf_mod)  %>% add_recipe(profit_core)
wf_xgb <- workflow() %>% add_model(xgb_mod) %>% add_recipe(profit_core)
```

Cross-validations & tuning

```
folds <- vfold_cv(train_model_df, v = 5)
metric_set <- metric_set(rmse)
```

```
# Linear regression
res_lin <- fit_resamples(wf_lin, resamples = folds, metrics = metric_set)
```

```
# Random forest
rf_params <- extract_parameter_set_dials(rf_mod) %>%
  update(mtry = mtry(range = c(1, 60)))

rf_grid <- grid_random(rf_params, size = 40)

res_rf <- tune_grid(wf_rf, resamples = folds, grid = rf_grid, metrics = metric_set)
```

```
## Warning: package 'ranger' was built under R version 4.3.3
```

```
# XGBoost grid
xgb_params <- extract_parameter_set_dials(xgb_mod) %>%
  finalize(train_model_df)

xgb_grid <- grid_random(xgb_params, size = 40)
```

```
res_xgb <- tune_grid(
  wf_xgb,
  resamples = folds,
  grid      = xgb_grid,
  metrics   = metric_set
)
```

```
## Warning: package 'xgboost' was built under R version 4.3.3
```

Pick the best model

```
# Get the tuning
best_rf_params <- select_best(res_rf, metric = "rmse")
best_xgb_params <- select_best(res_xgb, metric = "rmse")

# Pick the best model
get_cv_rmse <- function(resample_object) {
  tune::collect_metrics(resample_object) %>%
    filter(.metric == "rmse") %>%
    slice_min(mean, n = 1, with_ties = FALSE) %>%
    pull(mean)
}

lin_rmse <- get_cv_rmse(res_lin)
rf_rmse <- get_cv_rmse(res_rf)
xgb_rmse <- get_cv_rmse(res_xgb)

model_cmp <- tibble(
  model = c("Linear", "Random Forest", "XGBoost"),
  rmse  = c(lin_rmse, rf_rmse, xgb_rmse)
)

print(model_cmp)
```

```
## # A tibble: 3 x 2
##   model          rmse
##   <chr>         <dbl>
## 1 Linear         95.8
## 2 Random Forest  37.0
## 3 XGBoost        38.7
```

```
winner <- model_cmp %>%
  filter(!is.na(rmse)) %>%
  slice_min(rmse, n = 1, with_ties = FALSE) %>%
  pull(model)

cat("Winner by CV RMSE:", winner)
```

```
## Winner by CV RMSE: Random Forest
```

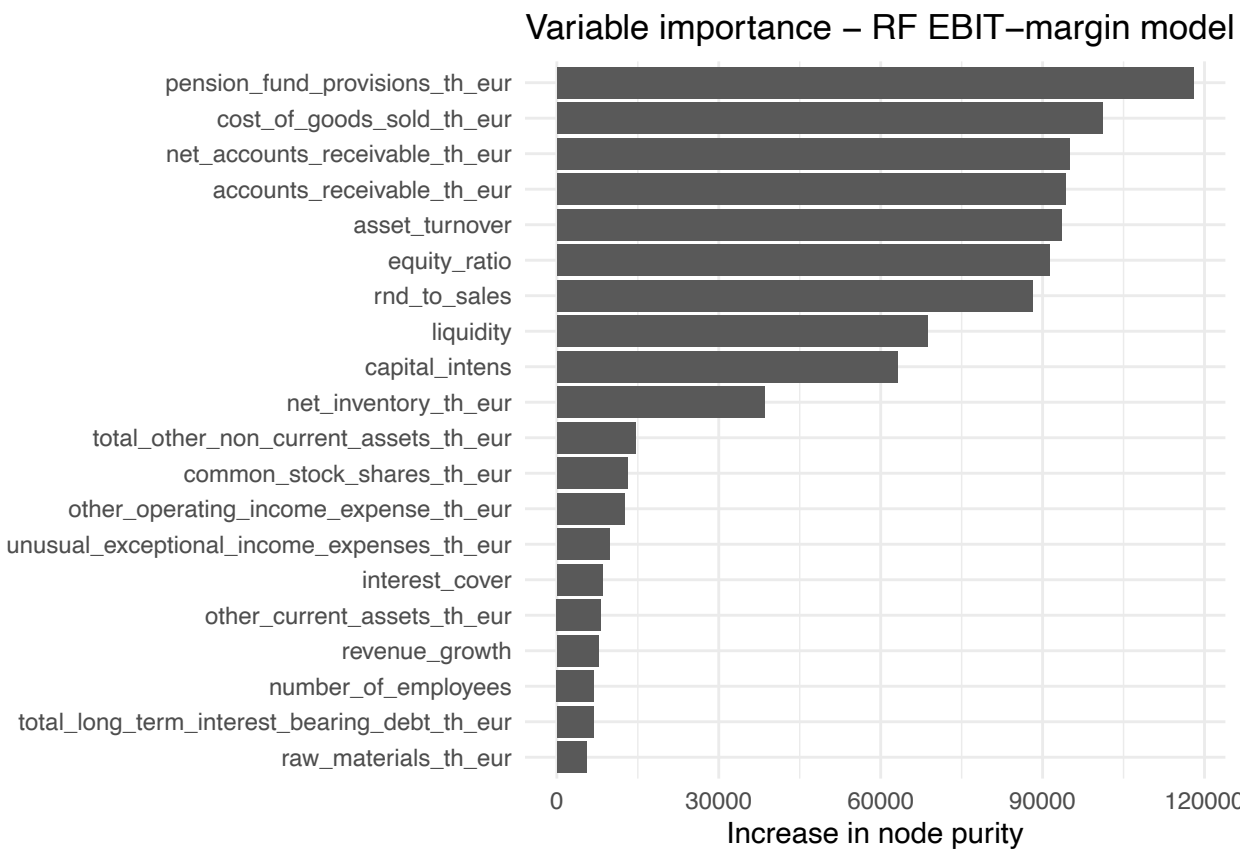
Fit on full training set & evaluate

```
# Finalize the workflow for the winner
final_wf <- switch(
  winner,
  "Random Forest" = finalize_workflow(wf_rf,  best_rf_params),
  "XGBoost" = finalize_workflow(wf_xgb, best_xgb_params),
  wf_lin
)

# Fit the final workflow on the entire training set
final_fit <- fit(final_wf, data = train_model_df)

# Evaluate on the test set
test_metrics <- final_fit %>%
  predict(new_data = test_model_df) %>%
  bind_cols(test_model_df %>% select(ebit_margin)) %>%
  metrics(truth = ebit_margin, estimate = .pred)

test_metrics
```

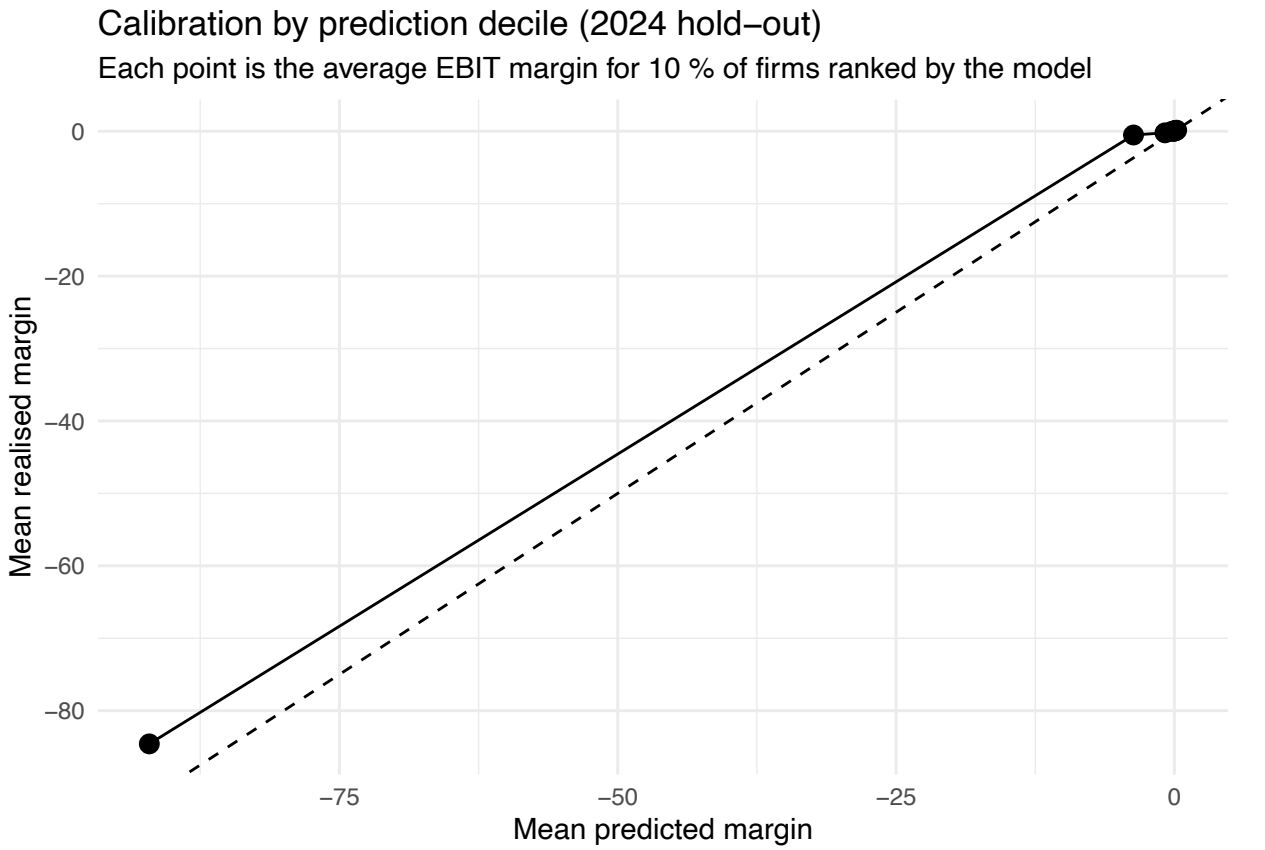
```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      50.5
## 2 rsq     standard       0.394
## 3 mae     standard      11.7
```

Visualizations

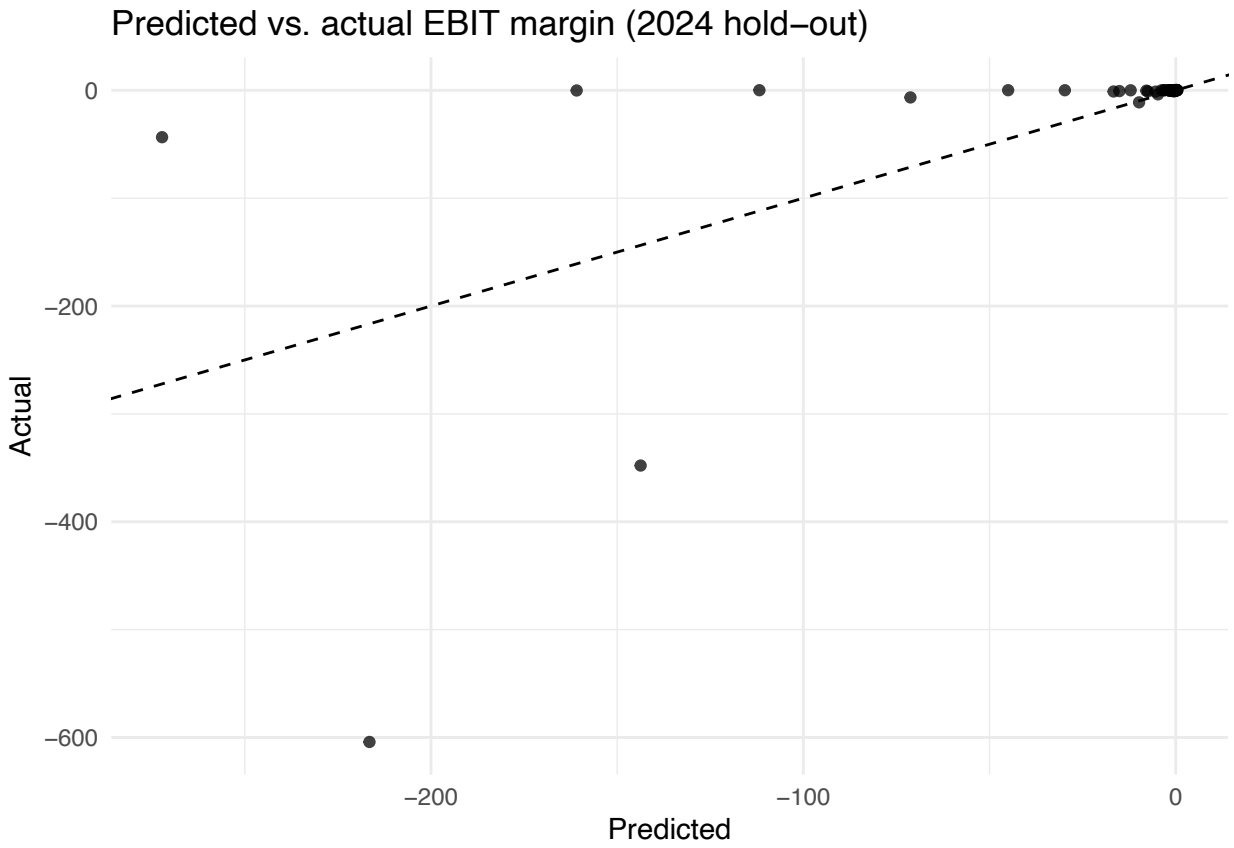
```
# Variable-importance ranking for the winning random forest
rf_fit <- extract_fit_parsnip(final_fit)$fit
vip_plot <- vip(rf_fit,
                num_features = 20,
                geom = "col") +
  labs(title = "Variable importance - RF EBIT-margin model",
       x = NULL, y = "Increase in node purity")
vip_plot
```



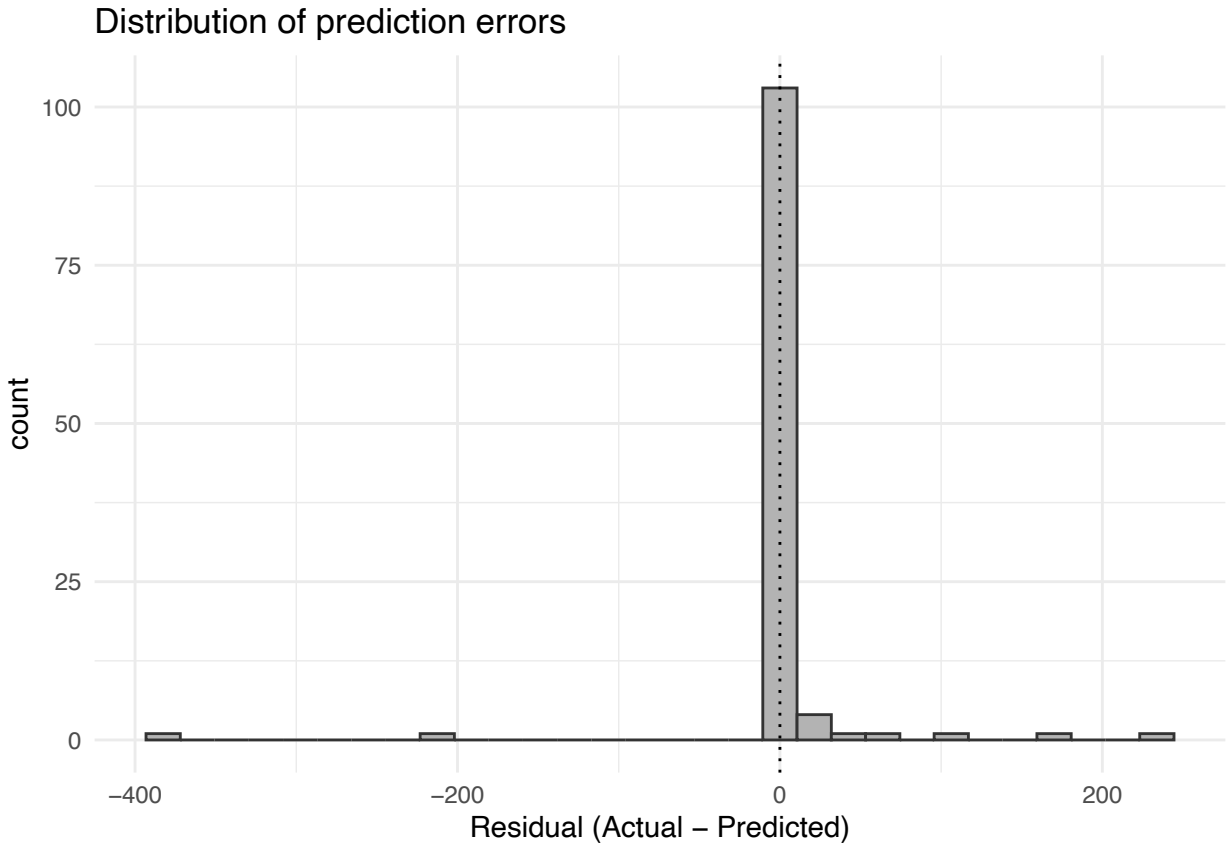

```r
# Calibration curve
pred_df <- test_df |>
  select(company_name_latin_alphabet, year, ebit_margin) |>
  bind_cols(predict(final_fit, new_data = test_df)) |>
  mutate(residual = ebit_margin - .pred) |>
  filter(is.finite(ebit_margin), is.finite(.pred))

calib_tbl <- pred_df |>
  mutate(decile = ntile(.pred, 10)) |>
  group_by(decile) |>
  summarise(mean_pred = mean(.pred, na.rm = TRUE),
            mean_actual = mean(ebit_margin, na.rm = TRUE),
            .groups = "drop")

calib_plot <- ggplot(calib_tbl,
                     aes(x = mean_pred, y = mean_actual)) +
  geom_point(size = 3) +
  geom_line() +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed") +
  labs(title = "Calibration by prediction decile (2024 hold-out)",
       subtitle = "Each point is the average EBIT margin for 10 % of firms ranked by the model",
       x = "Mean predicted margin", y = "Mean realised margin")

calib_plot
```




```
# Hold-out test diagnostics
scatter_plot <- ggplot(pred_df, aes(.pred, ebit_margin)) +
  geom_point(alpha = 0.75, na.rm = TRUE) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed") +
  labs(title = "Predicted vs. actual EBIT margin (2024 hold-out)",
       x = "Predicted", y = "Actual")

scatter_plot
```




```
res_hist <- ggplot(pred_df, aes(residual)) +
  geom_histogram(bins = 30, fill = "grey70", colour = "grey20", na.rm = TRUE) +
  geom_vline(xintercept = 0, linetype = "dotted") +
  labs(title = "Distribution of prediction errors",
       x = "Residual (Actual - Predicted)")

res_hist
```


Distribution of prediction errors
count
100
75
50
25
0
−400
−200
0
200
Residual (Actual − Predicted)