Group 13 Final Project

Load libraries

```r
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 error
```

```r
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
```

```r
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()
```

```r
library(vip)
```

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

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

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

```r
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
```

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

Set seed

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

Load data

```r
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

```r
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

```r
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

```r
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

```r
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

```r
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)
```

```r
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

```r
# 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
```

```r
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
```

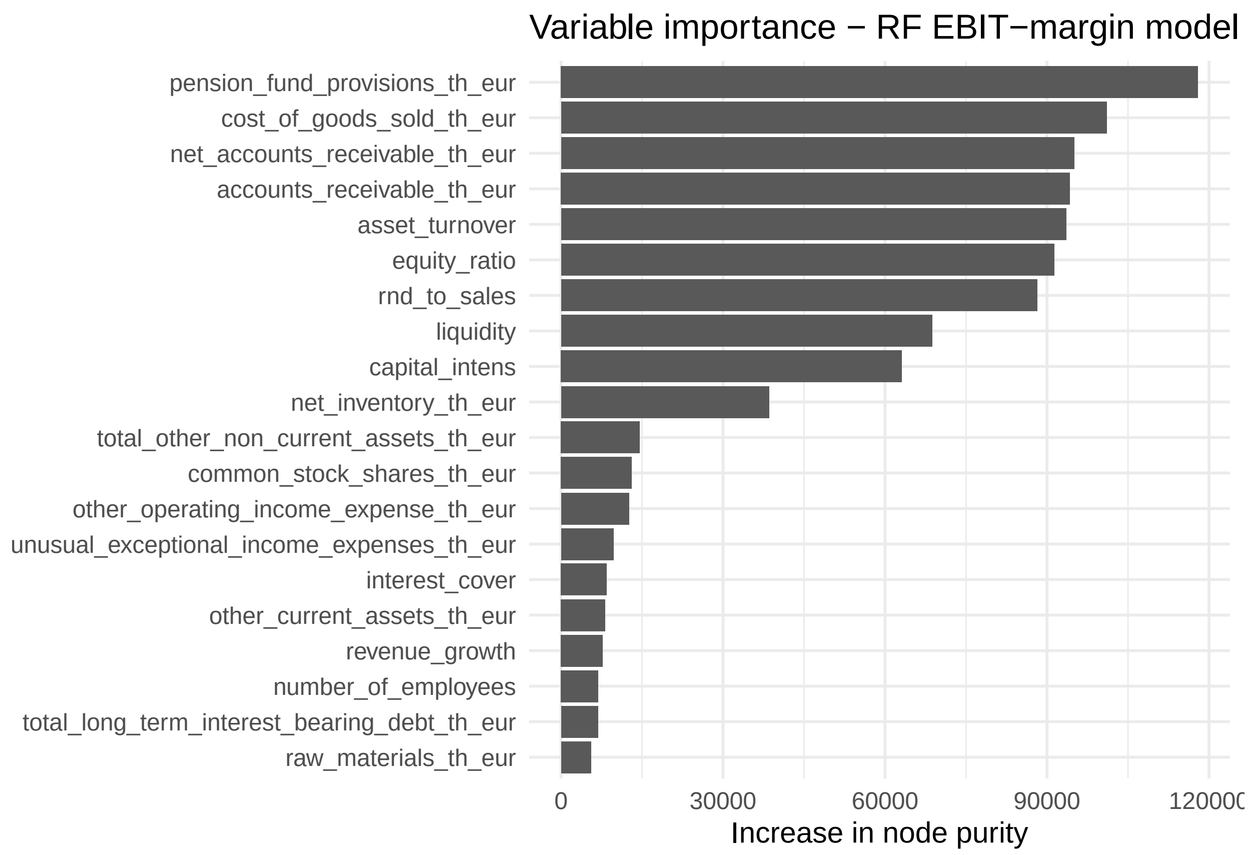
Fit on full training set & evaluate

```r
# 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
```

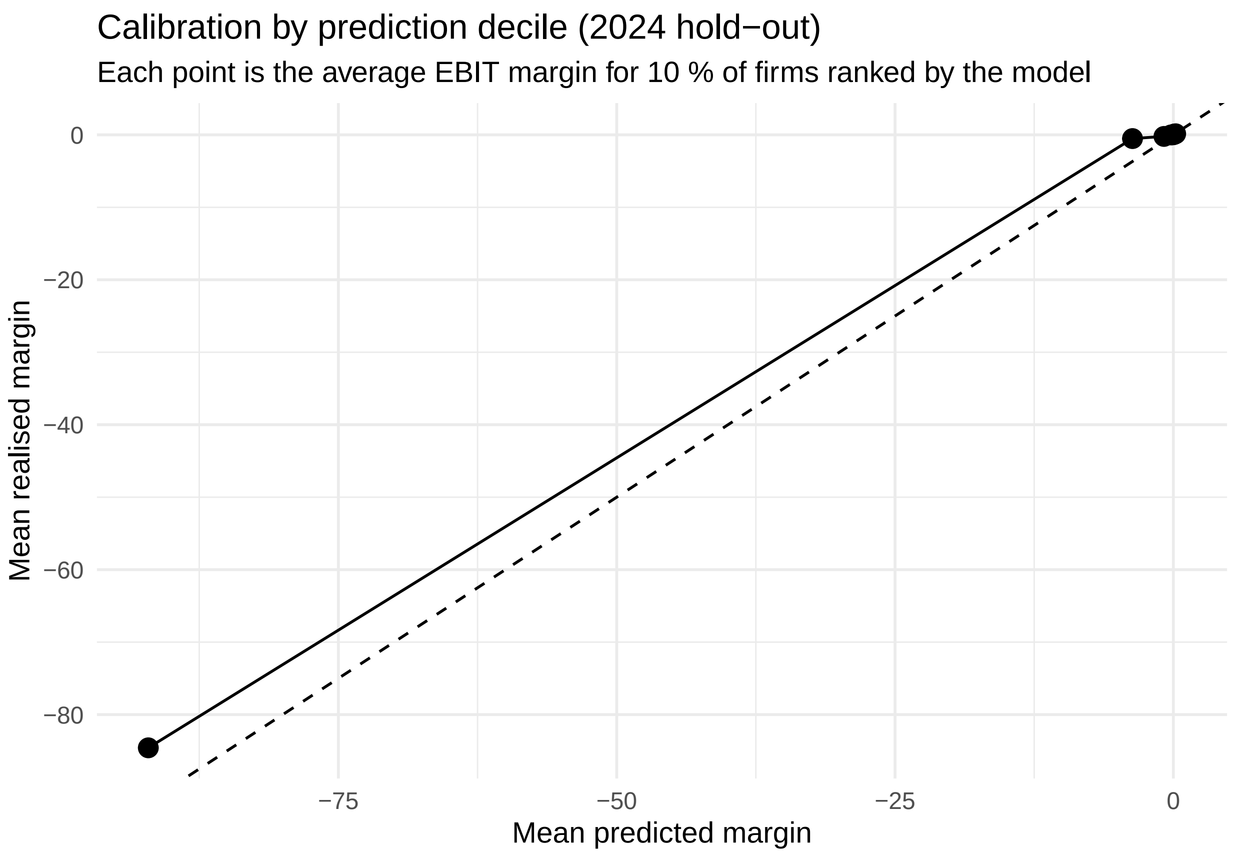
```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      50.5
## 2 rsq     standard       0.394
## 3 mae     standard      11.7
```

Visualizations

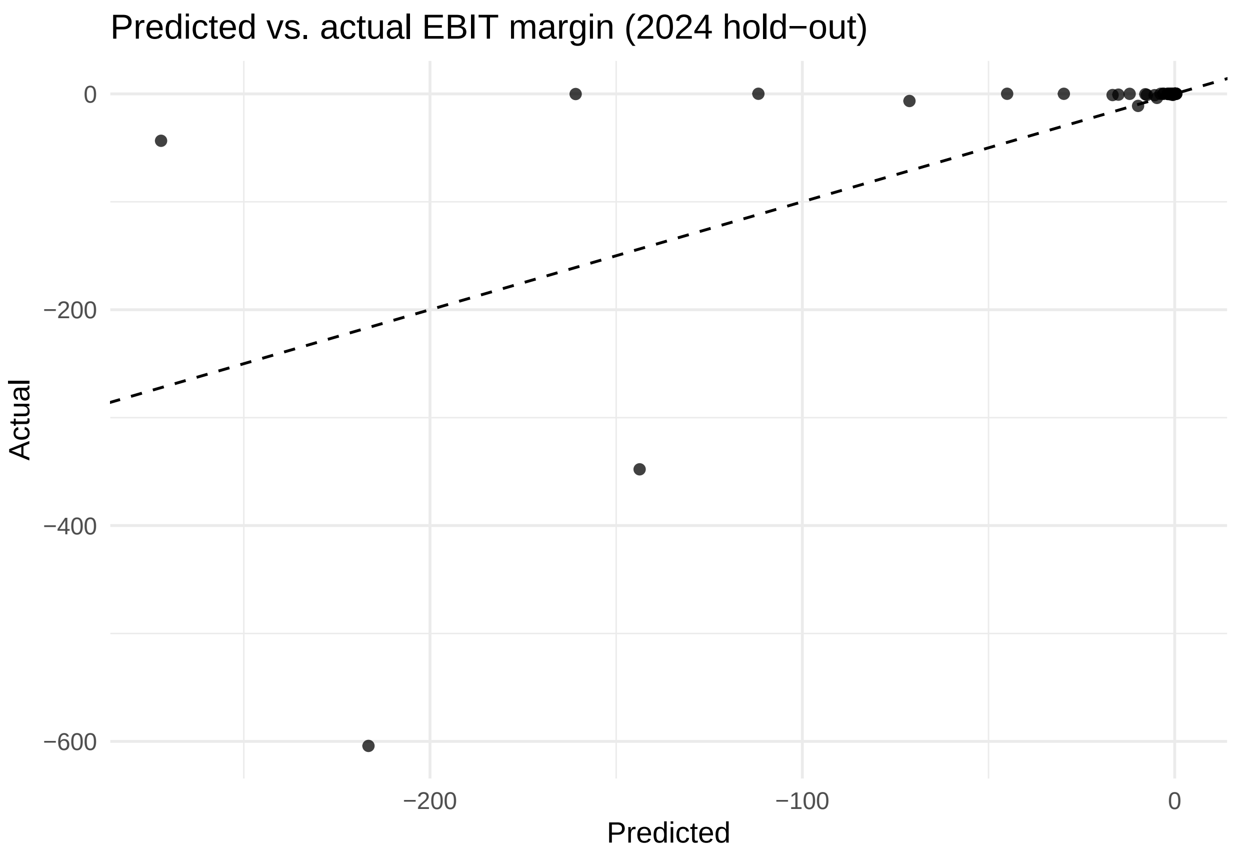
```r
# Variable-importance ranking for the winning random forest
rf_fit <- extract_fit_parsnip(final_fit)$fit
vip_plot <- vip(rf_fit,
                num_features = 20,
                geom = "col") +
  labs(title = "Variable importance - RF EBIT-margin model",
       x = NULL, y = "Increase in node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



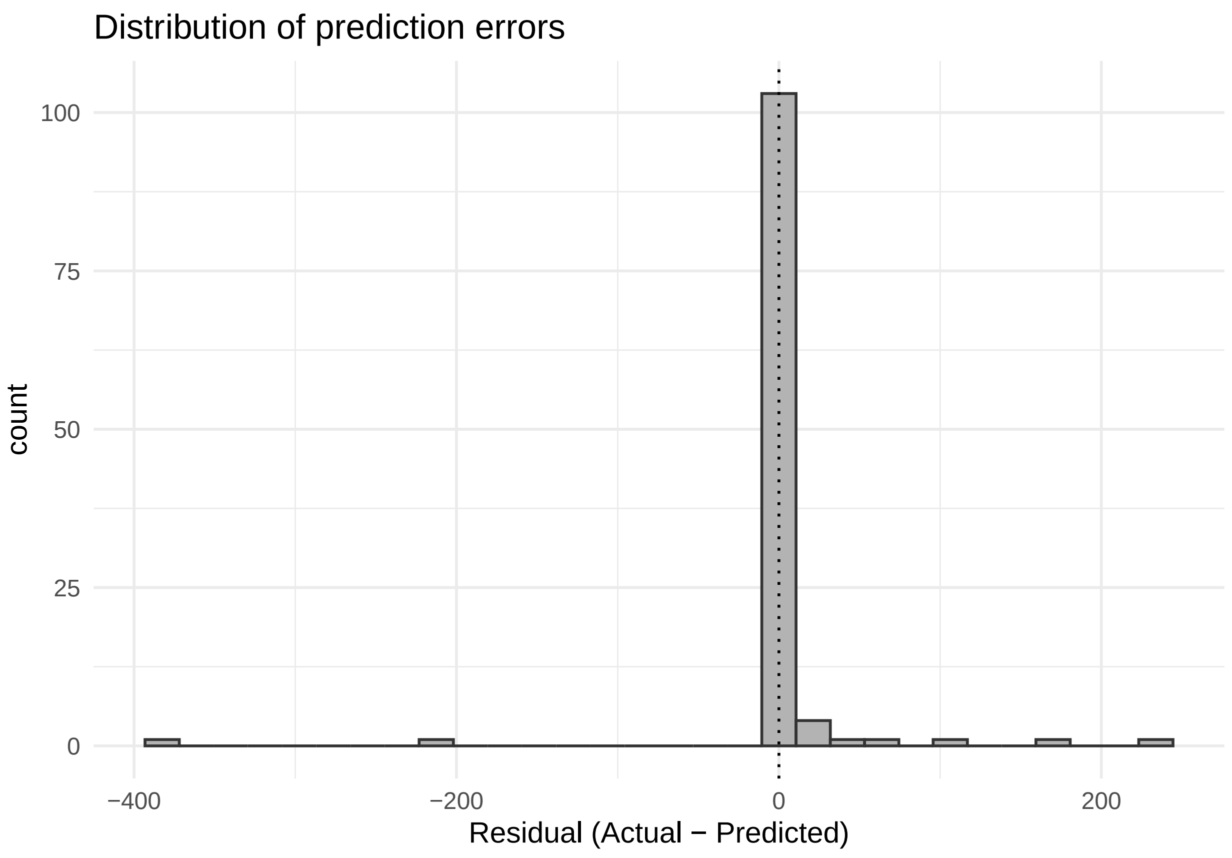
```r
# Calibration curve
pred_df <- test_df |>
  select(company_name_latin_alphabet, year, ebit_margin) |>
  bind_cols(predict(final_fit, new_data = test_df)) |>
  mutate(residual = ebit_margin - .pred) |>
  filter(is.finite(ebit_margin), is.finite(.pred))

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
```



```r
# 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
```



```r
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)