Model Calibration

William Chiu

2022-10-06

Summary

A company may not have sufficient data to train a logistic regression model if the sample size is small or the number of events is rare. In the context of banking, loan defaults are generally rare (less than 10%). For a bank with strong underwriting standards, loans defaults could be extremely rare (less than 1%).

Through two simulated data sets, I show a model calibration approach:

1. Train a model using a large-size industry data set with a rare event rate
2. Calibrate the model to a small-size internal data set with a small number of events
3. Test the calibrated model on an unseen medium-size internal data set with a similar event rate as step 2

The calibration approach reduces the prediction bias, that arises from differences between the industry and internal data sets. The terms "internal data set" and "company data set" are used interchangeably.

Simulate industry data set (n = 1 MM)

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caTools)
library(MLmetrics)
```

```
##
## Attaching package: 'MLmetrics'
##
## The following object is masked from 'package:base':
##
##     Recall
```

```r
set.seed(1)

credit_score <- rlnorm(n = 1e6, meanlog = 3, sdlog = 1)

linear_risk <- 12 + -2*credit_score + rnorm(1e6)

default_event <- rbinom(n = 1e6, size=1, prob=boot::inv.logit(linear_risk))

industry_df <- data.frame(default_event=default_event,
                          credit_score=credit_score)

summary(industry_df)
```

```
##  default_event     credit_score
##  Min.   :0.0000   Min.   :   0.1523
##  1st Qu.:0.0000   1st Qu.:  10.2353
##  Median :0.0000   Median :  20.0956
##  Mean   :0.1141   Mean   :  33.1132
##  3rd Qu.:0.0000   3rd Qu.:  39.4578
##  Max.   :1.0000   Max.   :2102.6288
```

The industry data set has an overall default rate of 0.114116. The middle 50% of the credit scores are between 10.2352678, 39.4577617.

Simulate internal data set (n = 10,000)

```r
set.seed(1)

credit_score <- rlnorm(n = 1e4, meanlog = 3.5, sdlog = 1)

linear_risk <- 1.5 + -0.9*credit_score + rnorm(1e4)

default_event <- rbinom(n = 1e4, size=1, prob=boot::inv.logit(linear_risk))

company_df <- data.frame(default_event=default_event,
                          credit_score=credit_score)

summary(company_df)
```

```
##  default_event     credit_score
##  Min.   :0.0000   Min.   :   0.8426
##  1st Qu.:0.0000   1st Qu.:  16.8880
##  Median :0.0000   Median :  32.5921
##  Mean   :0.0102   Mean   :  54.5722
##  3rd Qu.:0.0000   3rd Qu.:  65.2131
##  Max.   :1.0000   Max.   :1495.5909
```

The internal data set has an overall default rate of 0.0102. The middle 50% of the credit scores are between 16.8880395, 65.2131069.

The internal data set has a much lower default rate due to much more favorable credit score distribution.

At first glance, the two data sets appear very different from each other based on summary statistics. However, if you examine the data generating processes, you would notice that both processes share a common risk driver – credit score. As credit scores increase, the linear risk falls in both data sets.

However, in the internal data set, linear risk is less sensitive to credit scores than the industry data set. The intercepts are also different between the two data generating processes. As a result of the differences in processes, training a model on the industry data and applying the uncalibrated model on the internal data set would lead to biased predictions.

In reality, the data generating processes are unknown. If you choose to use model calibration, you're making an assumption that the drivers for the industry response and for the internal response are nearly identical.

Train an industry model (uncalibrated)

```
industry_model <- glm(default_event ~ credit_score, data=industry_df,
                      family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(industry_model)
```

```
##
## Call:
## glm(formula = default_event ~ credit_score, family = binomial,
##     data = industry_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0693  -0.0011   0.0000   0.0000   4.1473
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  10.459892   0.045391   230.4   <2e-16 ***
## credit_score -1.745809   0.007415  -235.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 710070  on 999999  degrees of freedom
## Residual deviance: 117682  on 999998  degrees of freedom
## AIC: 117686
##
## Number of Fisher Scoring iterations: 12
```

Sample the company data set

To avoid confusion, I will refer to the internal data set as the "company data set". If you are using industry data, then your company data set is probably small (and/or small number of events). If the company data set was large and the events were plentiful, we could have skipped the industry model altogether and fit a logistic regression directly on the company data set.

The sampling below reflects a realistic sample size with a small number of default events.

```
train_id <- sample.split(company_df$default_event, SplitRatio=0.1)

company_df[train_id, ] %>%
  group_by(default_event) %>%
  summarize(count=n())
```

```
## # A tibble: 2 x 2
##   default_event count
##           <int> <int>
## 1             0   990
## 2             1    10
```

For the purpose of calibration, the company data set has only 10 defaults.

Train a calibrated model

First, append the linear risk from the uncalibrated model to the company training set. Notice that `type="link"`.

```
train_company_df <- company_df[train_id, ]

train_company_df$industry_linear_risk <- predict(industry_model,
                                                 newdata=train_company_df, type="link")

summary(train_company_df)
```

```
##  default_event   credit_score     industry_linear_risk
##  Min.   :0.00   Min.   :  1.339   Min.   :-1146.022
##  1st Qu.:0.00   1st Qu.: 15.934   1st Qu.:  -99.360
##  Median :0.00   Median : 31.040   Median :  -43.730
##  Mean   :0.01   Mean   : 52.710   Mean   :  -81.562
##  3rd Qu.:0.00   3rd Qu.: 62.905   3rd Qu.:  -17.358
##  Max.   :1.00   Max.   :662.433   Max.   :    8.122
```

Second, train a calibrated model. Notice, that I am using the company data set and not the industry data set in the calibration step.

```
calibrated_model <- glm(default_event ~ industry_linear_risk,
                        data=train_company_df, family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(calibrated_model)
```

```
##
## Call:
## glm(formula = default_event ~ industry_linear_risk, family = binomial,
##     data = train_company_df)
##
## Deviance Residuals:
```

```
##      Min        1Q    Median        3Q       Max
## -1.38370  -0.00152   0.00000   0.00000   2.11114
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -3.9001     0.7750  -5.032 4.84e-07 ***
## industry_linear_risk   0.5393     0.1503   3.588 0.000333 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 112.003  on 999  degrees of freedom
## Residual deviance:  47.176  on 998  degrees of freedom
## AIC: 51.176
##
## Number of Fisher Scoring iterations: 14
```

Test the calibrated model

First, get the test data set.

```
test_company_df <- company_df[-train_id,]
```

Second, append the industry linear risk to each observation in the test data.

```
test_company_df$industry_linear_risk <- predict(industry_model,
                                                newdata = test_company_df, type="link")
```

Third, execute the calibrated model on the test data. Notice that I'm using `type="response"` to get the probability of default.

```
test_company_df$pred_calibrated_default <- predict(calibrated_model,
                                                   newdata=test_company_df, type="response")
```

For comparison purposes, let us get the probability of default from the uncalibrated industry model as well.

```
test_company_df$pred_industry_default <- boot::inv.logit(test_company_df$industry_linear_risk)

summary(test_company_df)
```

```
##  default_event     credit_score       industry_linear_risk
##  Min.   :0.0000   Min.   :   0.8426   Min.   :-2600.556
##  1st Qu.:0.0000   1st Qu.:  16.8873   1st Qu.: -103.391
##  Median :0.0000   Median :  32.6007   Median :  -46.455
##  Mean   :0.0102   Mean   :  54.5759   Mean   :  -84.819
##  3rd Qu.:0.0000   3rd Qu.:  65.2139   3rd Qu.:  -19.022
##  Max.   :1.0000   Max.   :1495.5909   Max.   :    8.989
##  pred_calibrated_default pred_industry_default
##  Min.   :0.0000000       Min.   :0.0000
##  1st Qu.:0.0000000       1st Qu.:0.0000
```

```
##  Median :0.0000000       Median :0.0000
##  Mean   :0.0076384       Mean   :0.0484
##  3rd Qu.:0.0000007       3rd Qu.:0.0000
##  Max.   :0.7206920       Max.   :0.9999
```

Since the calibrated linear risk is a linear transformation of the uncalibrated linear risk, we expect the AUC to be similar before and after calibration.

```
with(test_company_df, colAUC(pred_calibrated_default, default_event))
```

```
##              [,1]
## 0 vs. 1 0.9876859
```

```
with(test_company_df, colAUC(pred_industry_default, default_event))
```

```
##              [,1]
## 0 vs. 1 0.9876859
```

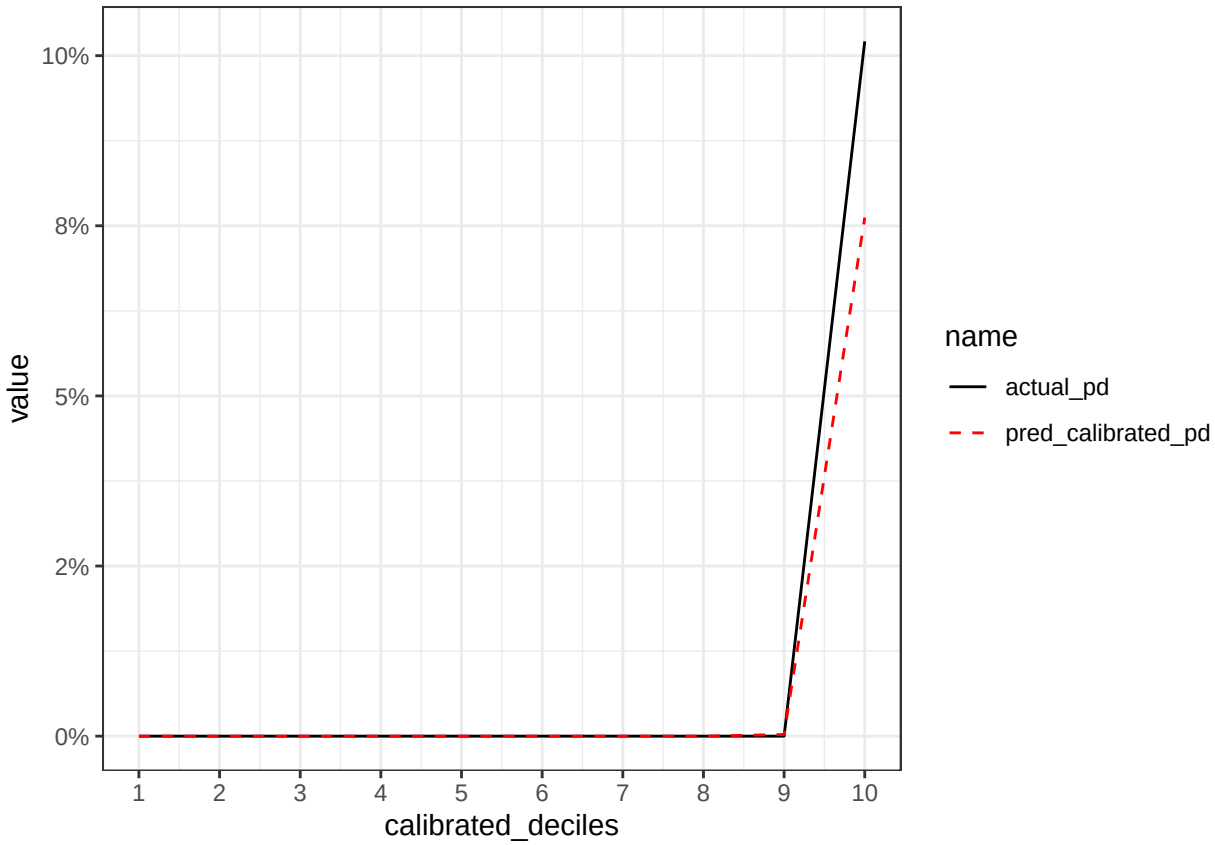
A better comparison is to bin the predicted default rates into deciles and compare actual and predicted default rates in each bin.

First, create deciles for each model.

```
test_df_with_deciles <- test_company_df %>%
  mutate(calibrated_deciles = ntile(pred_calibrated_default, 10),
         industry_deciles = ntile(pred_industry_default, 10))
```

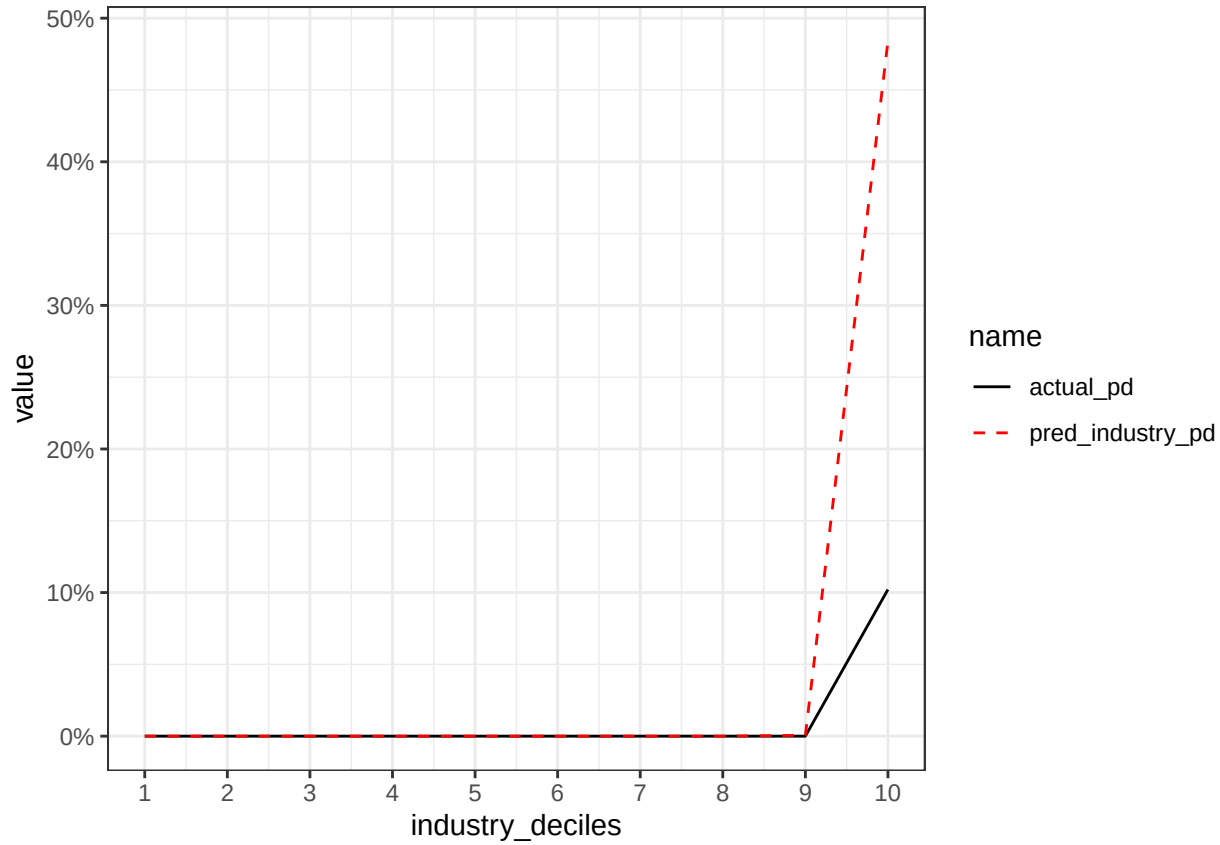
Second, compare actual and predicted for the calibrated model.

```
test_df_with_deciles %>%
  group_by(calibrated_deciles) %>%
  summarize(actual_pd = mean(default_event),
            pred_calibrated_pd = mean(pred_calibrated_default)) %>%
  pivot_longer(-calibrated_deciles) %>%
  ggplot(., aes(x=calibrated_deciles, y=value, color=name, linetype=name)) +
  geom_line() +
  scale_linetype_manual(values=c(1, 2)) +
  scale_color_manual(values=c('black','red')) +
  scale_x_continuous(breaks = 1:10) +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1)) +
  theme_bw()
```



Third, compare actual and predicted for the industry model.

```
test_df_with_deciles %>%
  group_by(industry_deciles) %>%
  summarize(actual_pd = mean(default_event),
            pred_industry_pd = mean(pred_industry_default)) %>%
  pivot_longer(-industry_deciles) %>%
  ggplot(., aes(x=industry_deciles, y=value, color=name, linetype=name)) +
  geom_line() +
  scale_linetype_manual(values=c(1, 2)) +
  scale_color_manual(values=c('black','red')) +
  scale_x_continuous(breaks = 1:10) +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1)) +
  theme_bw()
```



An alternative to plotting is to compute the log loss (lower is better).

Log Loss for industry model.

```
industry_log_loss <- LogLoss(test_company_df$pred_industry_default,
                             test_company_df$default_event)

industry_log_loss
```

```
## [1] 0.123674
```

Log Loss for calibrated model.

```
calibrated_log_loss <- LogLoss(test_company_df$pred_calibrated_default,
                               test_company_df$default_event)

calibrated_log_loss
```

```
## [1] 0.02484121
```

Log Loss for a null model (always predict using the mean default rate from the training set).

```
null_log_loss <- LogLoss(rep(mean(train_company_df$default_event),
                  nrow(test_company_df)), test_company_df$default_event)

null_log_loss
```

```
## [1] 0.05692525
```

Log Loss for a naive model (always predict 0 probability of default)

```
naive_log_loss <- LogLoss(rep(0, nrow(test_company_df)),
                          test_company_df$default_event)

naive_log_loss
```

```
## [1] 0.3523308
```

The log loss from the industry model is high. Among the null model, naive model, industry model, and calibrated model, the log loss is the lowest for the calibrated model.

Summary

Through two simulated data sets, I show that an industry model could be calibrated to a small company data set where the number events is extremely rare. The example was focused on a binary response (default vs. not default). Calibration could be easily extended to continuous responses by swapping out `glm` with `lm`.