Business-Guided Regression Designs

Staged Blocks

Andrija Djurovic

www.linkedin.com/in/andrija-djurovic

Business-Guided Model Designs

- Developing credit risk models is not just a regulatory exercise; first and foremost, they should serve business needs.
- Recognizing this, businesses and modelers should collaborate closely when designing models.
- Often, model quality is not solely determined by strong statistical performance but by how well it supports business needs while maintaining sufficient statistical performance.
- Designing a model is not just about collaboration between businesses and modelers in selecting risk factors for the final model. It is usually a highly iterative process that incorporates multiple dimensions.
- A recent trend in credit risk model development shows that practitioners often opt for a blockwise or modular approach rather than developing a model using a fully consolidated dataset.
- Providing inputs for different modeling areas on one side and ensuring robust statistical modeling on the other are not the only objectives for businesses and modelers, respectively.
- Good practice shows that other critical inputs from business experts play a key role in model development. Modelers consider and incorporate these inputs into the overall model design.
- This presentation focuses on the staged blocks approach, which is briefly described in the following slides. The rest of the presentation provides a simplified simulation study demonstrating the implementation of this design in `R` and `Python`.

Staged Blocks Approach

- Practitioners usually prefer a blockwise (modular) approach over running a single optimization algorithm on a consolidated modeling dataset.
- This modular strategy employs multiple submodels or blocks based on business inputs, data coverage, and industry insights. These submodels are then combined to form a unified model score.
- One blockwise approach is the staged approach, where predictions from the previous block serve as offsets for models in subsequent blocks.
- The rationale for staging can be related to:
  - data sources, particularly their cost, quality, reliability, and availability;
  - risk factor prioritization;
  - extensive knowledge of end-users about predictive risk factors.
- All the above points essentially relate to prioritizing risk factors from different perspectives.
- Implementing staged blocks and other blockwise designs presents various challenges in practice. One of the biggest is properly handling the different samples used within the block model development.
- Given these challenges, modelers are strongly encouraged to consider the statistical consequences of using different samples and to collaborate closely with the business to better integrate process-related considerations.
- Further details on the block-wise model design approaches can be found in Anderson, R.A. (2021), Credit Intelligence & Modelling: Many Paths through the Forest of Credit Rating and Scoring.

Simulation Setup

- Assume a modeling dataset with 12 risk factors and binary target.
- Further, assume that these risk factors are classified into two groups based on the opinion of business experts.
- An additional business input modelers should incorporate into model development is that risk factors from the first group should be prioritized, with only the remaining variance modeled using risk factors from the second group.
- Finally, assume that the modelers chose Weights of Evidence (WoE) encoding for categorical risk factors.
- The dataset used for this simulation can be accessed at the following link.

Dataset Overview

```
## 'data.frame':    1000 obs. of  13 variables:
##  $ Creditability        : num  0 0 0 0 0 ...
##  $ Account_Balance      : chr  "1" "1" ...
##  $ Duration             : chr  "03 [16,36)" "02 [8,16)" ...
##  $ Payment_Status       : chr  "4" "4" ...
##  $ Purpose              : chr  "2" "0" ...
##  $ Credit_Amount        : chr  "01 (-Inf,3914)" "01 (-Inf,3914)" ...
##  $ Savings              : chr  "1" "1" ...
##  $ Employment           : chr  "2" "3" ...
##  $ Installment          : chr  "4" "2" ...
##  $ Gender_Marital_Status: chr  "2" "3" ...
##  $ Guarantors           : chr  "1" "1" ...
##  $ Available_Asset      : chr  "2" "1" ...
##  $ Age                  : chr  "01 (-Inf,26)" "03 [35,Inf)" ...
```

Blocks Design Overview

```
##                       rf block
## 1               Duration     1
## 2           Installment      1
## 3  Gender_Marital_Status     1
## 4             Guarantors     1
## 5                    Age     1
## 6        Account_Balance     2
## 7         Payment_Status     2
## 8                Purpose     2
## 9          Credit_Amount     2
## 10               Savings     2
## 11            Employment     2
## 12       Available_Asset     2
```

Simulation Results - R Code

```r
library(openxlsx)
library(dplyr)

#data import
fp <- "https://andrija-djurovic.github.io/adsfcr/bgrd/db_00_bgrd.xlsx"
db <- read.xlsx(xlsxFile = fp,
                sheet = 1)

#utils function import (woe.calc)
source("https://raw.githubusercontent.com/andrija-djurovic/adsfcr/refs/heads/main/bgrd/utils.R")

#woe encoding
db.woe <- db
rf <- names(db)[-1]
for   (i in 1:length(rf)) {
      rf.i <- rf[i]
      woe.res.i <- woe.calc(db = db,
                            x = rf.i,
                            y = "Creditability")
      db.woe[, rf.i] <- woe.res.i[[2]]
      }
#sample of encoded dataset
head(db.woe[, c(1:3, 10:12)])
```

```
##   Creditability Account_Balance   Duration Gender_Marital_Status   Guarantors Available_Asset
## 1             0      -0.8180987 -0.1086883            -0.2353408 0.0005250722     -0.02857337
## 2             0      -0.8180987  0.3466246             0.1655476 0.0005250722      0.46103496
## 3             0      -0.4013918  0.3466246            -0.2353408 0.0005250722      0.46103496
## 4             0      -0.8180987  0.3466246             0.1655476 0.0005250722      0.46103496
## 5             0      -0.8180987  0.3466246             0.1655476 0.0005250722     -0.02857337
## 6             0      -0.8180987  0.3466246             0.1655476 0.0005250722      0.46103496
```

Simulation Results - R Code cont.

```
#block 1 model
frm.b1 <- "Creditability ~ Duration + Installment + Gender_Marital_Status +
                           Guarantors + Age"
b1.r <- glm(formula = frm.b1,
            family = "binomial",
            data = db.woe)
round(x = summary(b1.r)$coefficients,
      digits = 4)
```

```
##                        Estimate Std. Error  z value Pr(>|z|)
## (Intercept)             -0.8560     0.0733 -11.6733   0.0000
## Duration                -1.0292     0.1474  -6.9821   0.0000
## Installment             -1.1127     0.4566  -2.4369   0.0148
## Gender_Marital_Status   -1.1202     0.3503  -3.1977   0.0014
## Guarantors              -0.9451     0.4108  -2.3007   0.0214
## Age                     -0.9362     0.2301  -4.0684   0.0000
```

```
#block 1 model predictions
b1.p <- unname(predict(object = b1.r))
head(b1.p)
```

```
## [1]  0.1890835 -1.8657478 -0.6275319 -1.7645730 -1.5177348 -1.9723970
```

Simulation Results - R Code cont.

```r
#block 2 model
frm.b2 <- "Creditability ~ Account_Balance + Payment_Status + Purpose + Credit_Amount +
                          Savings +  Employment + Available_Asset"
b2.r <- glm(formula = frm.b2,
            family = "binomial",
            data = db.woe,
            offset = b1.p)
round(x = summary(b2.r)$coefficients,
      digits = 4)
```

```
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.0102     0.0840 -0.1210   0.9037
## Account_Balance  -0.7931     0.1054 -7.5263   0.0000
## Payment_Status   -0.7184     0.1564 -4.5942   0.0000
## Purpose          -1.0230     0.2065 -4.9542   0.0000
## Credit_Amount    -0.5171     0.2323 -2.2258   0.0260
## Savings          -0.7689     0.1998 -3.8473   0.0001
## Employment       -0.5847     0.2784 -2.1001   0.0357
## Available_Asset  -0.5302     0.2580 -2.0550   0.0399
```

Simulation Results - Python Code

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
import requests

#data import
fp = "https://andrija-djurovic.github.io/adsfcr/bgrd/db_00_bgrd.xlsx"
db = pd.read_excel(io = fp,
                   sheet_name = 0,
                   dtype = "category")
db["Creditability"] = pd.to_numeric(db["Creditability"])
blocks = pd.read_excel(io = fp,
                       sheet_name = 1)

#utils function import (woe_calc)
url = "https://raw.githubusercontent.com/andrija-djurovic/adsfcr/refs/heads/main/bgrd/utils.py"
r = requests.get(url)
exec(r.text)

#woe encoding
db_woe = db.copy()
rf = db.columns[1:].tolist()
for i in range(len(rf)):
    rf_i = rf[i]
    woe_res_i = woe_calc(db = db,
                         x = rf_i,
                         y = "Creditability")
    db_woe[rf_i] = woe_res_i["x_trans"]
```

Simulation Results - Python Code cont.

```
#block 1 model
frm_b1 = "Creditability ~ Duration + Installment + Gender_Marital_Status + \
                          Guarantors + Age"
b1_r = smf.glm(formula = frm_b1,
               family = sm.families.Binomial(),
               data = db_woe).fit()
round(b1_r.summary2().tables[1], 4)
```

```
##                          Coef.  Std.Err.        z   P>|z|  [0.025  0.975]
## Intercept              -0.8560    0.0733 -11.6733  0.0000 -0.9997 -0.7123
## Duration               -1.0292    0.1474  -6.9821  0.0000 -1.3181 -0.7403
## Installment            -1.1127    0.4566  -2.4369  0.0148 -2.0076 -0.2178
## Gender_Marital_Status -1.1202    0.3503  -3.1977  0.0014 -1.8067 -0.4336
## Guarantors             -0.9451    0.4108  -2.3007  0.0214 -1.7503 -0.1400
## Age                    -0.9362    0.2301  -4.0684  0.0000 -1.3872 -0.4852
```

```
#block 1 model predictions
b1_p = b1_r.predict(which = "linear")
b1_p[0:5]
```

```
## array([ 0.18908348, -1.86574779, -0.62753191, -1.764573  , -1.51773477])
```

Simulation Results - Python Code cont.

```
#block 2 model
frm_b2 = "Creditability ~ Account_Balance + Payment_Status + Purpose + Credit_Amount + \
                          Savings + Employment + Available_Asset"
b2_r = smf.glm(formula = frm_b2,
               data = db_woe,
               family = sm.families.Binomial(),
               offset = b1_p).fit()
round(b2_r.summary2().tables[1], 4)
```

```
##                   Coef.  Std.Err.       z   P>|z|  [0.025  0.975]
## Intercept       -0.0102    0.0840 -0.1210  0.9037 -0.1747  0.1544
## Account_Balance -0.7931    0.1054 -7.5263  0.0000 -0.9997 -0.5866
## Payment_Status  -0.7184    0.1564 -4.5942  0.0000 -1.0248 -0.4119
## Purpose         -1.0230    0.2065 -4.9542  0.0000 -1.4277 -0.6183
## Credit_Amount   -0.5171    0.2323 -2.2258  0.0260 -0.9724 -0.0618
## Savings         -0.7689    0.1998 -3.8473  0.0001 -1.1605 -0.3772
## Employment      -0.5847    0.2784 -2.1001  0.0357 -1.1303 -0.0390
## Available_Asset -0.5302    0.2580 -2.0550  0.0399 -1.0358 -0.0245
```