Business-Guided Regression Designs

Embedded 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 various modeling areas and ensuring robust statistical modeling are key objectives for businesses and modelers.
- Good practice shows that inputs from business experts plays a key role in model development. Modelers consider and incorporate these inputs into the overall model design.
- This presentation focuses on the embedded 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`.

Embedded 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 embedding is similar to other blockwise approaches and can be attributed 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 embedded 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.

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 that modelers should incorporate into model development is prioritizing risk factors from the first group, making them a mandatory driver for the model run on the next block of risk factors.
- 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

```
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.

```r
#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
```

```r
#block 1 model predictions
db.woe$block_1 <- unname(predict(object = b1.r))
head(db.woe$block_1)
```

```
## [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 ~ block_1 + Account_Balance + Payment_Status + Purpose +
                           Credit_Amount + Savings +  Employment + Available_Asset"
b2.r <- glm(formula = frm.b2,
            family = "binomial",
            data = db.woe)
round(x = summary(b2.r)$coefficients,
      digits = 4)
```

```
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.1409     0.1278 -1.1020   0.2705
## block_1           0.8308     0.1240  6.7002   0.0000
## Account_Balance  -0.7892     0.1043 -7.5657   0.0000
## Payment_Status   -0.7193     0.1544 -4.6598   0.0000
## Purpose          -0.9934     0.2047 -4.8532   0.0000
## Credit_Amount    -0.5773     0.2341 -2.4662   0.0137
## Savings          -0.7580     0.1979 -3.8303   0.0001
## Employment       -0.6073     0.2754 -2.2046   0.0275
## Available_Asset  -0.5371     0.2544 -2.1114   0.0347
```

Simulation Results - `Python` Code

```python
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
db_woe["block_1"] = b1_r.predict(which = "linear")
db_woe["block_1"].iloc[0:5]
```

```
## 0    0.189083
## 1   -1.865748
## 2   -0.627532
## 3   -1.764573
## 4   -1.517735
## Name: block_1, dtype: float64
```

Simulation Results - Python Code cont.

```
#block 2 model
frm_b2 = "Creditability ~ block_1 + 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()).fit()
round(b2_r.summary2().tables[1], 4)
```

```
##                    Coef.  Std.Err.       z   P>|z|  [0.025  0.975]
## Intercept        -0.1409    0.1278 -1.1020  0.2705 -0.3914  0.1097
## block_1           0.8308    0.1240  6.7002  0.0000  0.5877  1.0738
## Account_Balance -0.7892    0.1043 -7.5657  0.0000 -0.9936 -0.5847
## Payment_Status  -0.7193    0.1544 -4.6598  0.0000 -1.0219 -0.4168
## Purpose         -0.9934    0.2047 -4.8532  0.0000 -1.3946 -0.5922
## Credit_Amount   -0.5773    0.2341 -2.4662  0.0137 -1.0362 -0.1185
## Savings         -0.7580    0.1979 -3.8303  0.0001 -1.1459 -0.3702
## Employment      -0.6073    0.2754 -2.2046  0.0275 -1.1471 -0.0674
## Available_Asset -0.5371    0.2544 -2.1114  0.0347 -1.0356 -0.0385
```