Benchmarking Low Default Portfolios to Third Party Ratings

Distance-Based Deviation Testing

Andrija Djurovic

www.linkedin.com/in/andrija-djurovic

Low Default Portfolios Benchmarks

- Validating Low Default Portfolios (LDP) presents significant challenges to practitioners, as most statistical tests applicable to high default portfolios do not have equivalents for low default portfolios.
- Therefore, practitioners often rely on a benchmarking approach to validate certain types of LDP where external ratings are available.
- Benchmarking helps assess how well the rating system distinguishes between low- and high-risk counterparties and whether the average risk assessment significantly differs from an independent observer's.
- It also evaluates whether deviations in ratings are unreasonably large.
- The drawbacks of benchmarking include that external ratings are typically available for only part of the portfolio, which may not represent the entire portfolio. Unlike default prediction, deviations from external ratings are not necessarily incorrect. Therefore, considering factors such as differences in rating and default definitions, careful analysis and detailed interpretation of benchmarking results are essential.
- A fundamental assumption of benchmarking is confidence in the correctness of external ratings. Accordingly, selecting the relevant type of external rating requires great care and an understanding of potential discrepancies.
- This presentation focuses on distance-based deviation testing.
- More on LDP benchmarking can be found in the following reference: Franz, C., & Lawrenz, C. (2007). Benchmarking low-default portfolios to third-party ratings. Journal of Credit Risk, 3(2), 87–100.

Distance-Based Deviation Testing

A commonly used measure alongside the tendency measure is average deviation.

Deviation evaluates the magnitude of differences between internal and external ratings, providing insight into rating volatility and calibration quality.

The most straightforward measure of calibration quality is average deviation, calculated as:

$$D = \frac{1}{N} \sum_{i=1}^{N} (y_i - x_i)$$

where:

- y_i and x_i are the internal and external ratings for obligor i, respectively;
- N is total number of obligros.

The absolute deviation calculation can often supplement the simple average deviation.

Practitioners can use bootstrapping to obtain the confidence interval for deviation. Unlike the basic bootstrapping method, deriving the confidence interval for deviation typically involves certain assumptions. One key assumption is that external ratings are correct, allowing specific observed distributions to remain fixed. Another practical approach that can be applied for the same purpose is based on convolutions.

Simulation Study

- The following slides present `R` and `Python` code for calculating average deviation and its confidence interval.
- The dataset is available here.
- The simulated dataset consists of internal and external ratings, the only inputs needed for this calculation.
- The presented example uses convolutions to calculate the confidence interval for average deviation. In this specific case, the assumption is that the observed distribution for each unique deviation remains constant.
- Practitioners are encouraged to modify the presented framework, adjust it to their needs, and compare the results under different assumptions.

R Code

```r
#utils function import (d.ci)
source("https://raw.githubusercontent.com/andrija-djurovic/adsfcr/refs/heads/main/ldp/utils_benchmarking_ldp.R")

#data import
url <- "https://raw.githubusercontent.com/andrija-djurovic/adsfcr/refs/heads/main/ldp/ldp_benchmarking.csv"
db <- read.csv(file = url,
               header = TRUE)

#convert ratings to factors
db$external <- factor(x = db$external ,
                      levels = sort(unique(db$external)),
                      order = TRUE)
db$internal <- factor(x = db$internal,
                      levels = sort(unique(db$external)),
                      order = TRUE)

#calculate deviations
deviation <- as.numeric(db$internal) - as.numeric(db$external)

#frequency table
table(deviation)
```

```
## deviation
##  -4  -3  -2  -1   0   1   2   3   4   5   6
##  18  55 105 171 173 173 144 116  32  10   3
```

```r
#average deviation confidence interval
d.ci(deviation = deviation,
     cl = 0.95)
```

```
##     est lower upper
## 1 0.387 0.265 0.509
```

Python Code

```
import pandas as pd
import numpy as np
from scipy import stats
import requests

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

#data import
url = "https://raw.githubusercontent.com/andrija-djurovic/adsfcr/refs/heads/main/ldp/ldp_benchmarking.csv"
db = pd.read_csv(filepath_or_buffer = url)

#convert ratings to factors
external_levels = np.sort(db["external"].unique())
db["external"] = pd.Categorical(db["external"],
                                categories = external_levels,
                                ordered = True)
db["internal"] = pd.Categorical(db["internal"],
                                categories = external_levels,
                                ordered = True)
```

Python Code cont.

```
#calculate deviations
deviation = (db["internal"].cat.codes + 1) - (db["external"].cat.codes + 1)

#frequency table
deviation.value_counts().sort_index()
```

```
## -4     18
## -3     55
## -2    105
## -1    171
##  0    173
##  1    173
##  2    144
##  3    116
##  4     32
##  5     10
##  6      3
## Name: count, dtype: int64
```

```
#average deviation confidence interval
d_ci(deviation = np.array(deviation),
     cl = 0.95)
```

```
##      est  lower  upper
## 0  0.387  0.265  0.509
```

Simulation Results - Summary