The Instability of WoE Encoding in PD Modeling

Andrija Djurovic
Dr. Alan Forrest

Standard Approaches to PD Modeling

- Probability of Default (PD) rating modeling is widely considered a cornerstone of credit risk.
- The most commonly used method for modeling PD is binomial logistic regression with categorized risk factors.
- Within this framework, the standard approaches for encoding categorical risk factors are dummy encoding and Weights of Evidence (WoE) encoding.
- Practitioners often debate using one encoding method over another, while some advocate combining both within the final model.
- This presentation investigates how the choice of encoding method affects the replicability of a perfect model. In other words, if a model predicts the target phenomenon perfectly, can we replicate the exact estimates on the predicted target?
- By examining the replicability of a perfect model using two encoding methods, this presentation further explores the potential instability of the WoE encoding method.

Dummy Encoding - Does the Model Replicate?

The following steps outline replicating the model estimates on the observed and predicted target. As can be seen, the dummy encoding replicates the exact estimates.

The simulation dataset can be found here.

1. Estimate the model of the form `Creditability ~ Maturity + Amount + Age`:

```
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -1.55       0.38   -4.08     0.00
## Maturity02 [8,12)        0.67       0.44    1.51     0.13
## Maturity03 [12,16)       0.97       0.38    2.53     0.01
## Maturity04 [16,36)       1.30       0.37    3.48     0.00
## Maturity05 [36,45)       1.57       0.43    3.68     0.00
## Maturity06 [45,Inf)      2.06       0.46    4.50     0.00
## Amount02 [3914,6758)     0.19       0.21    0.90     0.37
## Amount03 [6758,Inf)      0.58       0.25    2.29     0.02
## Age02 [26,35)           -0.54       0.19   -2.79     0.01
## Age03 [35,Inf)          -0.90       0.19   -4.68     0.00
```

2. Using the model from step 1, generate within-sample predictions `pred`.

3. Estimate the model of the form `pred ~ Maturity + Amount + Age` and compare the estimates with those from step 1:

```
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -1.55       0.38   -4.08     0.00
## Maturity02 [8,12)        0.67       0.44    1.51     0.13
## Maturity03 [12,16)       0.97       0.38    2.53     0.01
## Maturity04 [16,36)       1.30       0.37    3.48     0.00
## Maturity05 [36,45)       1.57       0.43    3.68     0.00
## Maturity06 [45,Inf)      2.06       0.46    4.50     0.00
## Amount02 [3914,6758)     0.19       0.21    0.90     0.37
## Amount03 [6758,Inf)      0.58       0.25    2.29     0.02
## Age02 [26,35)           -0.54       0.19   -2.79     0.01
## Age03 [35,Inf)          -0.90       0.19   -4.68     0.00
```

WoE Encoding - Does the Model Replicate?

Using the same dataset as in the previous slide, the following steps outline replicating the model estimates on the observed and predicted target using WoE encoding. Unlike dummy encoding, this process involves recalculating the risk factor WoE on the predicted sample. As can be seen, WoE encoding does not replicate the estimates of the perfect model.

1. Replace each modality of the risk factors with the observed WoE values.
2. Estimate the model of the form `Creditability ~ Maturity + Amount + Age`:

```
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -0.85       0.07  -11.69     0.00
## Maturity       -0.83       0.16   -5.12     0.00
## Amount         -0.49       0.22   -2.20     0.03
## Age            -1.05       0.23   -4.67     0.00
```

3. Using the model from step 2, generate within-sample predictions `pred`.
4. Recalculate the WoE values for the risk factor `Maturity` and compare them with the values from step 1:

WoE Values on the Observed Target:

```
##          rf          bin   woe
## 8  Maturity 01 (-Inf,8)  1.31
## 2  Maturity   02 [8,12)  0.58
## 3  Maturity  03 [12,16)  0.27
## 1  Maturity  04 [16,36) -0.11
## 19 Maturity  05 [36,45) -0.52
## 14 Maturity 06 [45,Inf) -1.13
```

WoE Values on the Predicted Target:

```
##          rf          bin   woe
## 1  Maturity 01 (-Inf,8)  1.23
## 2  Maturity   02 [8,12)  0.55
## 3  Maturity  03 [12,16)  0.29
## 4  Maturity  04 [16,36) -0.09
## 5  Maturity  05 [36,45) -0.56
## 6  Maturity 06 [45,Inf) -1.17
```

Since the WoE values calculated on the observed and predicted targets differ, this implies that the perfect model does not replicate.

WoE Encoding Instability

The following slides present two simulations examining the instability of WoE encoding.

Simulation 1:

This simulation visualizes the WoE difference per model recycle iteration and the number of risk factors in the model. In other words, it calculates the difference in WoE values for the same modalities of a risk factor between the initial WoE value and the one recalculated based on the predicted target (perfect model), given the number of risk factors in the final model. The number of risk factors in the final model starts at two and increases to 13, following the order of columns in the simulation dataset. The target variable used is `Creditability`.

The simulation dataset is available here.

For the sake of simplicity, only the WoE differences for the `Account_Balance` risk factor are presented. Practitioners are also encouraged to test and examine this phenomenon for other risk factors.

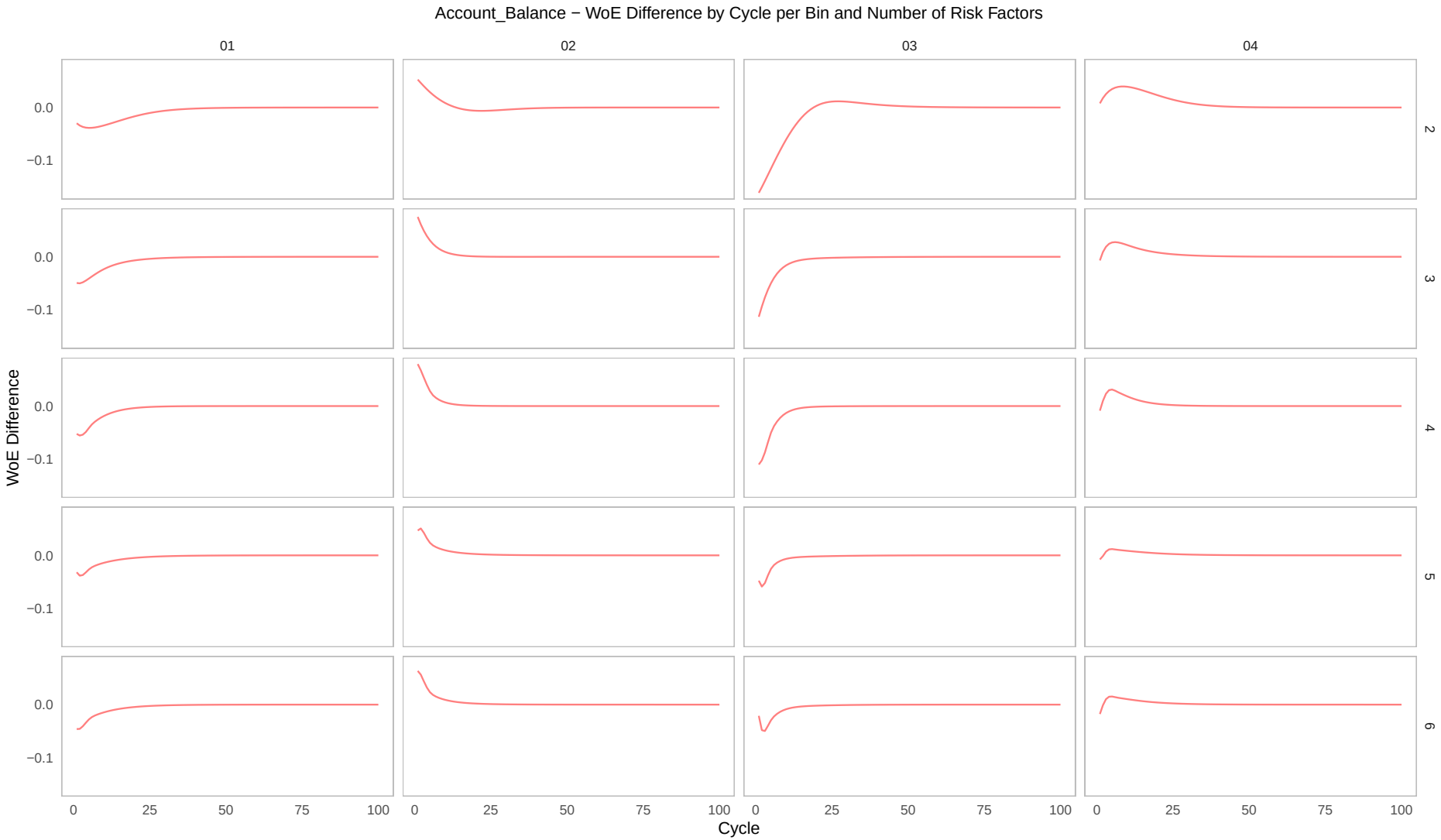
Simulation 2:

As practitioners sometimes use the absolute WoE gap as an input for binning numeric risk factors, this simulation examines breaches of a specific threshold in a perfect model. Assuming a final model of the form `Creditability ~ Maturity + Amount + Age`, the risk factor WoE gap between adjacent bins is set to be at least 0.10.
A subsample of 100 observations is drawn 5000 times from the final model's dataset. WoE values are calculated for each subsample based on the model's predictions for each risk factor modality.
Finally, for each risk factor, the differences between adjacent WoE bins are computed and categorized as either below 0.10 or equal to/above 0.10. Simulations identifying a WoE gap below the 0.10 threshold would indicate potential issues with the binning process despite the model perfectly predicting the target.

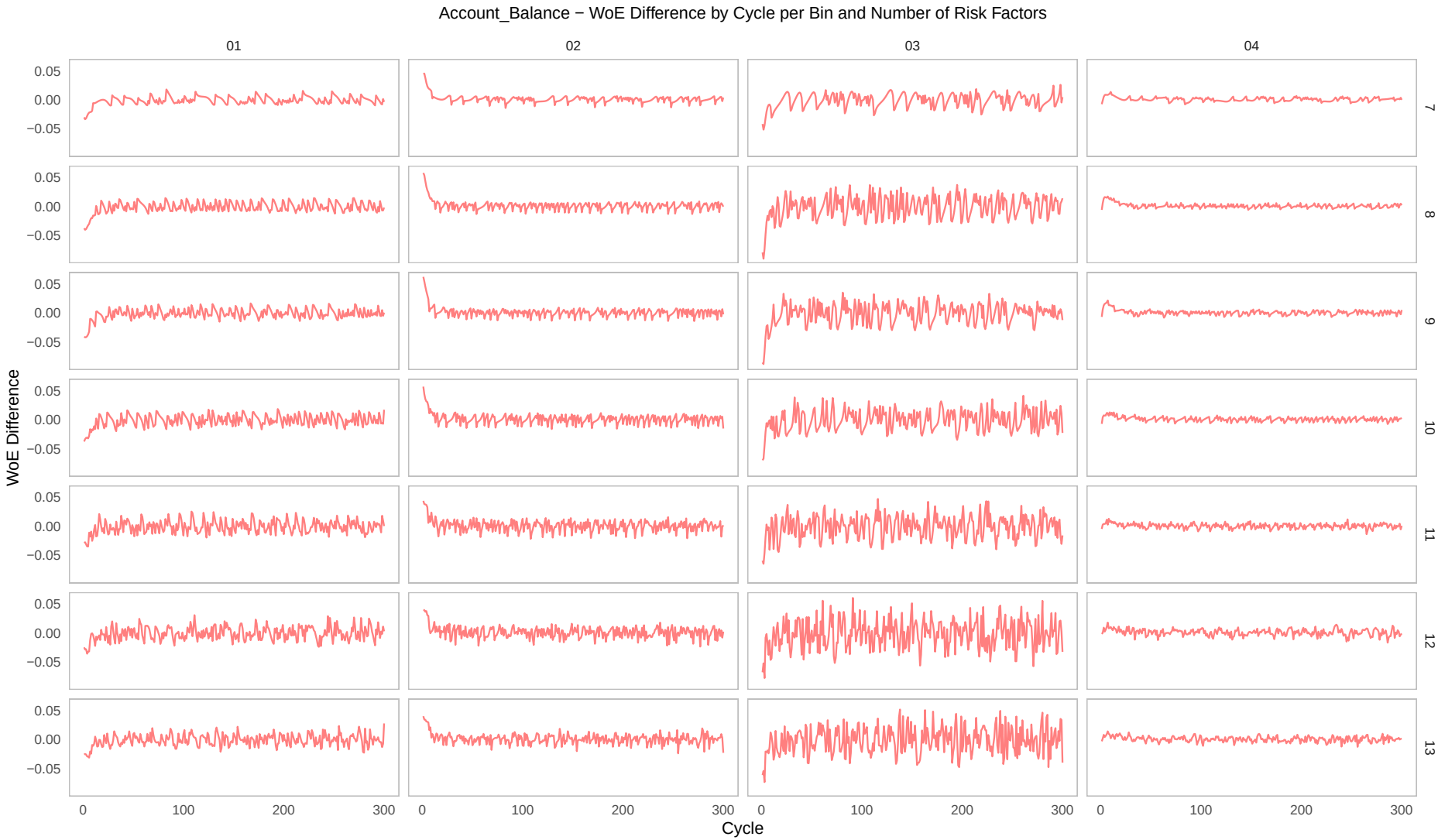
The simulation dataset is available here.

How many breaches of thresholds can be attributed to WoE instability in the perfect model?

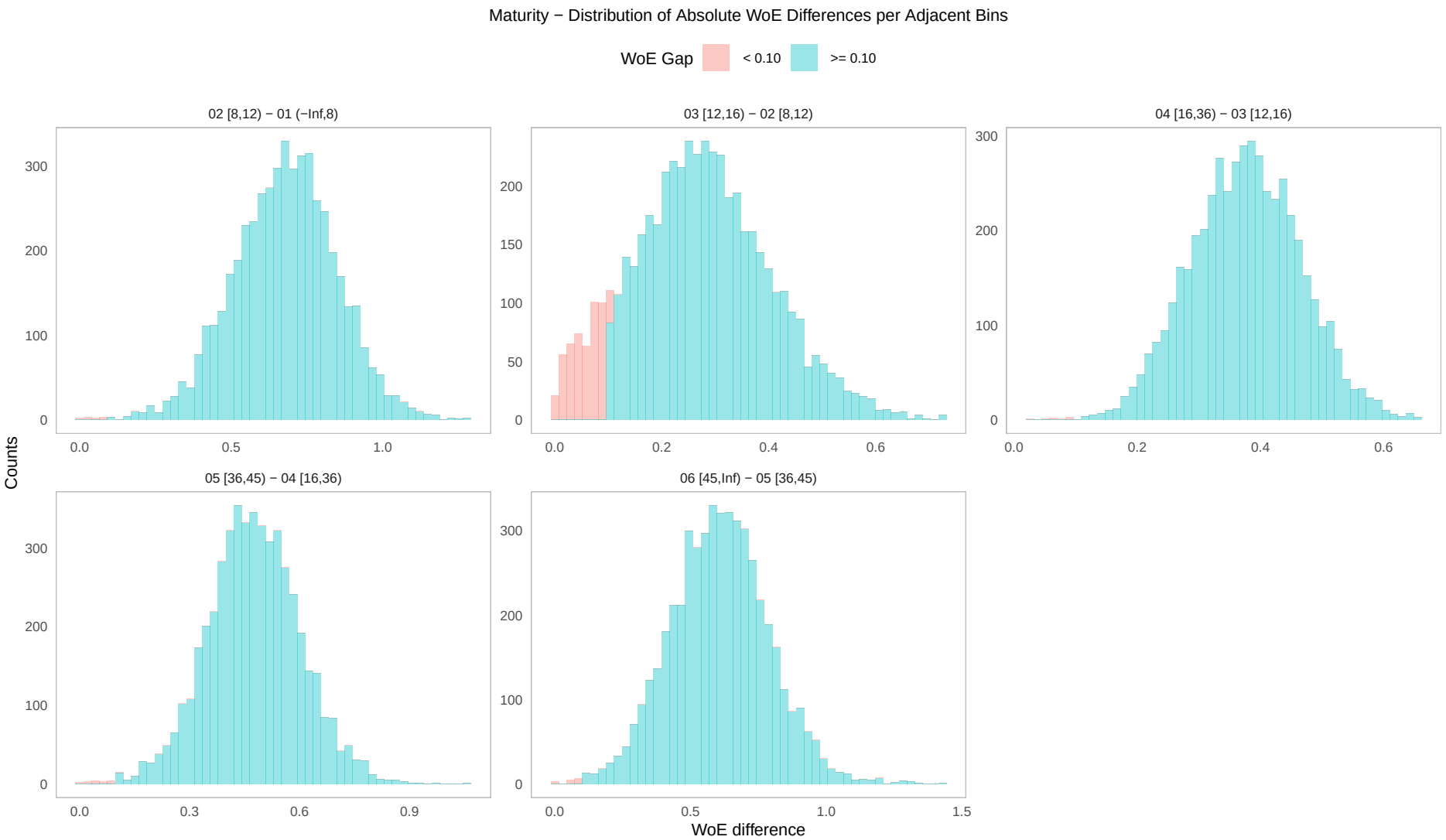
WoE Encoding Instability - Simulation 1 Results



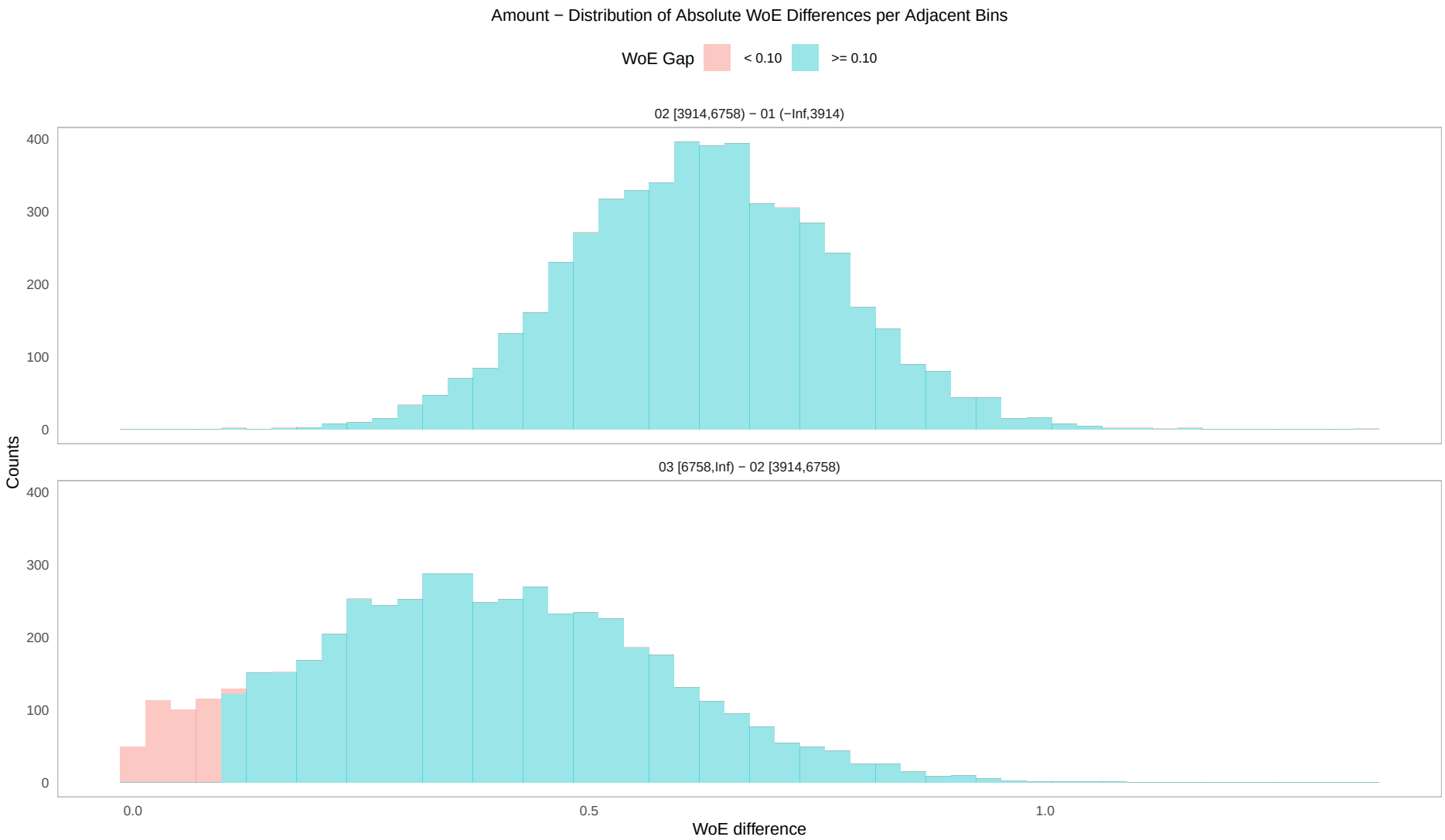
WoE Encoding Instability - Simulation 1 Results cont.



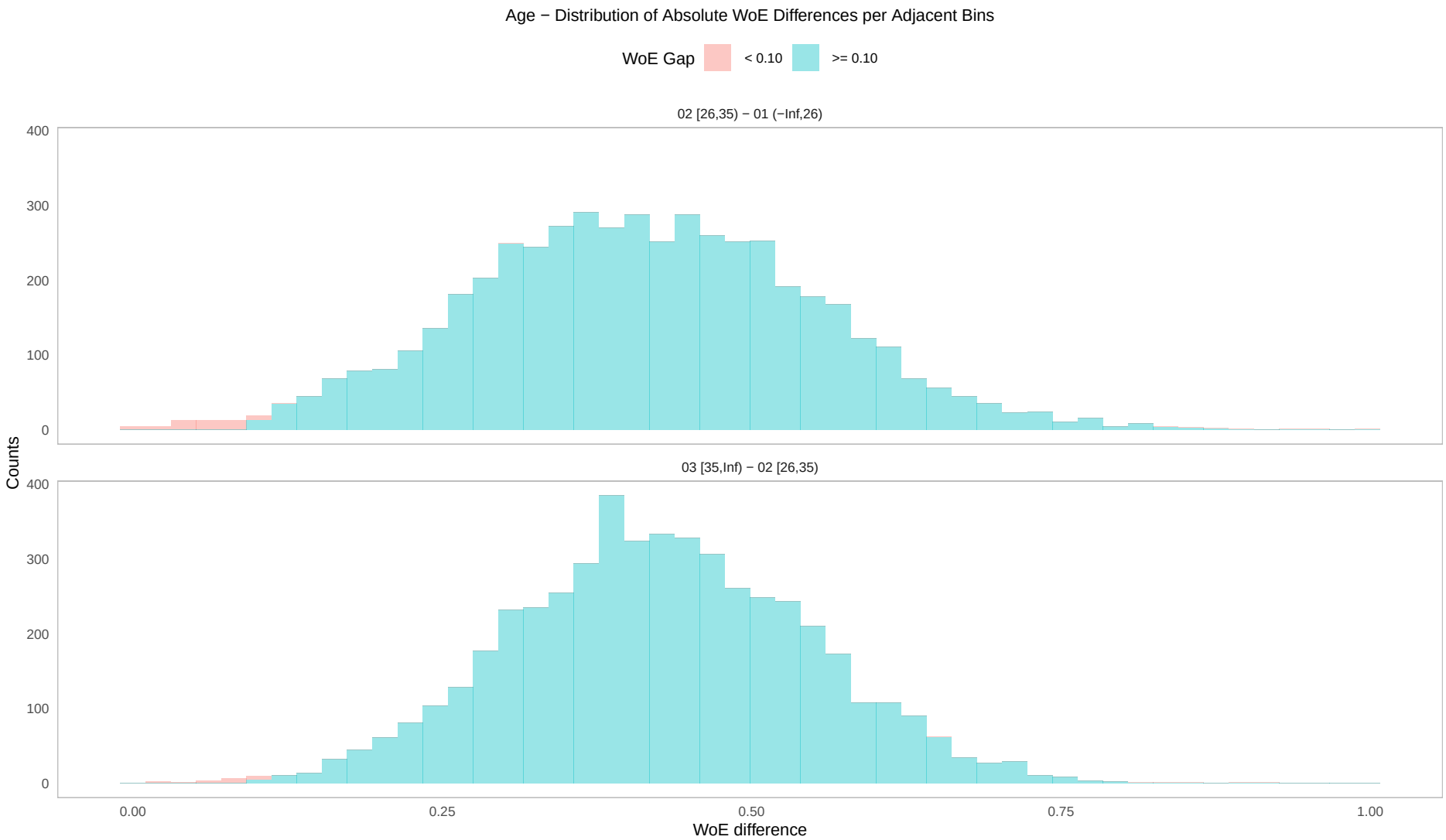
WoE Encoding Instability - Simulation 2 Results



WoE Encoding Instability - Simulation 2 Results cont.



WoE Encoding Instability - Simulation 2 Results cont.



Discussion Points

- Should model validators be concerned about WoE encoding instability? How should this concern be addressed?
- What performance outcomes would justify keeping the model unchanged?
- In some cases it seems that a perfect performance is not good enough?