Distributed Computation of the AUROC-GLM Confidence Intervals Using DataSHIELD

67th Biometric Colloquium

Daniel Schalk[1], Stefan Buchka[2], Ulrich Mansmann[2], Verena Hoffmann[2]

March 14-17, 2021

[1]Department of Statistics (LMU Munich)
[2]Institute for Medical Information Processing, Biometry, and Epidemiology (IBE, LMU Munich)

DIFUTURE
Data Integration for Future Medicine

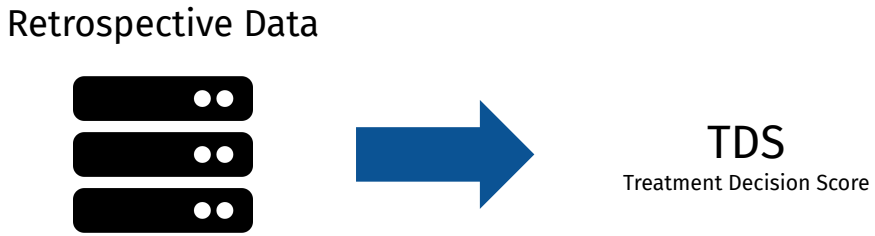
MEDIZIN INFORMATIK INITIATIVE

Aim of this talk

The goal of this talk is to give you insights about our progress on evaluating the predictive ability of a model on distributed data:

1. Approximate the ROC curve and AUC using a parametrized approach, the ROC-GLM [3].
   $\rightarrow$ Novelty: Extention to distributed data
2. Conduct tests by calculating a confidence interval (CI) as proposed by DeLong [1] on the distributed data.
   $\rightarrow$ Novelty: Extention to distributed data
3. Short demonstration of a DataSHIELD implementation of the methods..

ProVal-MS (DRKS: 00014034)

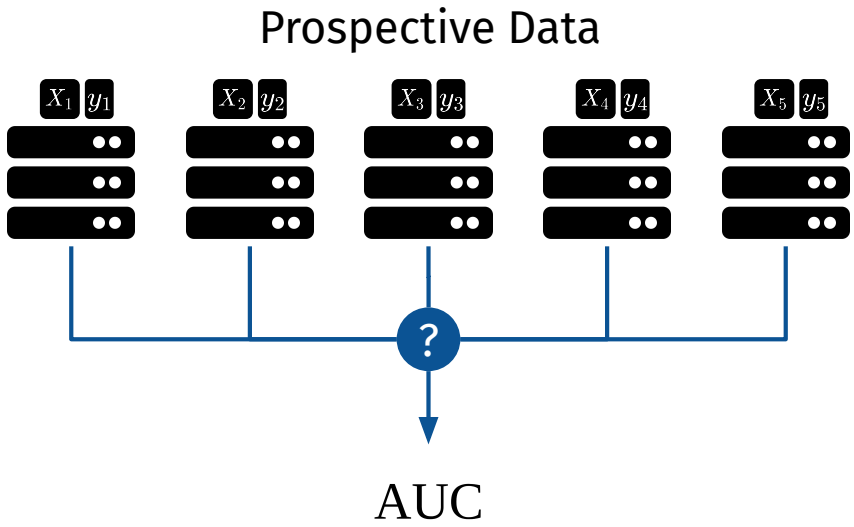


- Development of a **Treatment Decision Score** (**TDS**) on retrospective data.
- Goal: Validation of the TDS in prospective data collected in ProVal-MS with an AUC of at least 0.7.

"Compared to a patient with no success in therapy, the probability of having a higher TDS is at least 70 % for a randomly selected patient who shows successful therapy."

Decentralized data

- Prospective data are distributed over five servers.
- For the study, we want to evaluate if we can replace the centralized with a decentralized ROC analysis and thus AUC calculation.
- But: To calculate the empirical AUC we need to share parts of the data.
- Problem: How to calculate the empirical AUC without sharing the data?



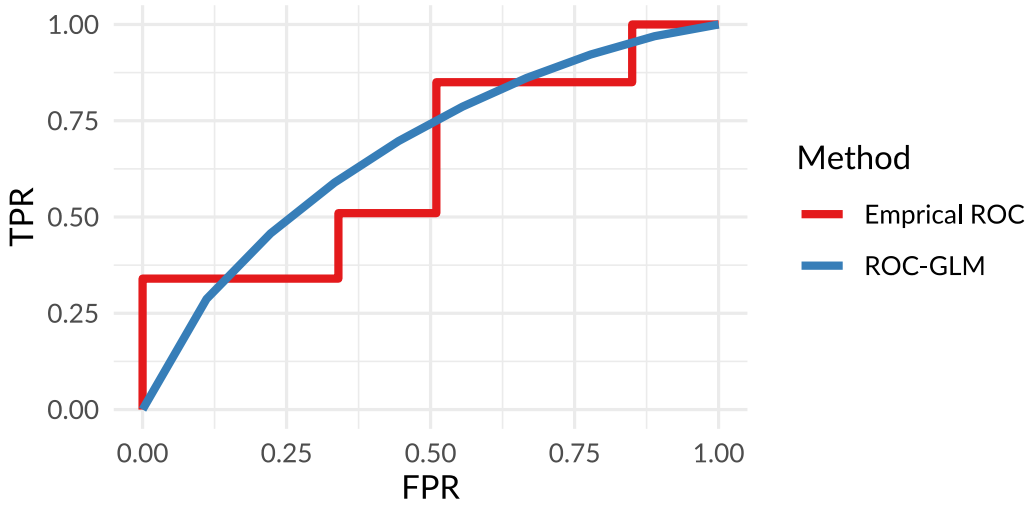
Theoretical background - ROC-GLM

The ROC-GLM i

- Parametric approximation of the empirical ROC curve.
- The ROC-GLM has the following (binormal) form based on the parameter vector β:

$$\text{tpr}_\beta(\text{fpr}) = \Phi\left(\beta_1 + \beta_2 \Phi^{-1}(\text{fpr})\right) \tag{1}$$

For example:



$$\widehat{\beta} = \begin{pmatrix} 0.6556 \\ 0.9973 \end{pmatrix}$$

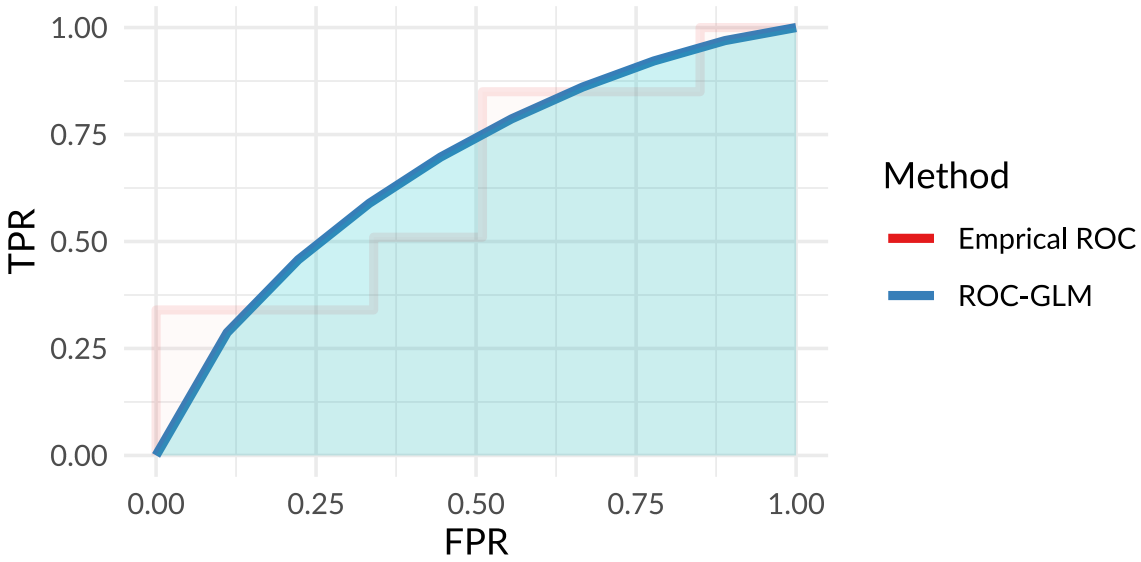
Φ and Φ^{-1} denotes the probability and quantile function of the standard normal distribution.

The ROC-GLM ii

- The approximated AUC from the ROC-GLM can be obtained by calculating the following integral:

$$\text{AUC}_{\text{app}} = \int_0^1 \text{tpr}_\beta(\text{fpr})\, d\,\text{fpr} = \int_0^1 \Phi\left(\beta_1 + \beta_2 \Phi^{-1}(\text{fpr})\right)\, d\,\text{fpr} \quad (2)$$

For example:



$$AUC_{\text{app}} = 0.6788$$

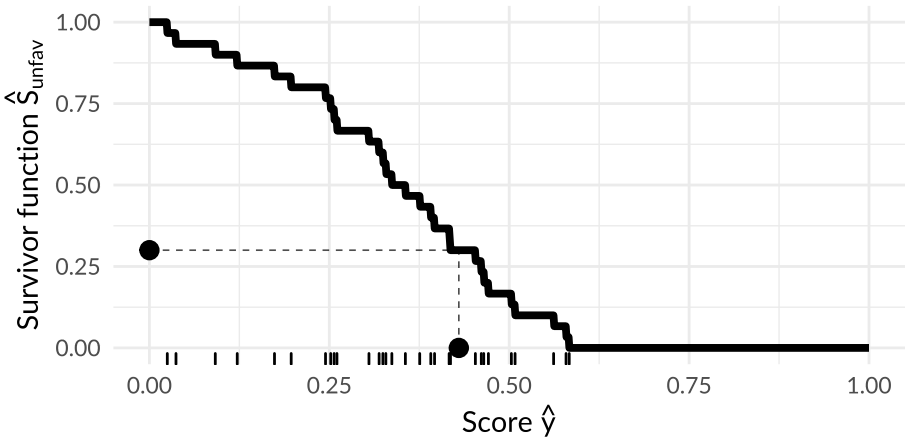
The ROC-GLM - Step 1

1. Given the prediction model $\hat{f}$ and data $\{(y_1, x_1), \ldots, (y_n, x_n)\}$ we calculate prediction scores $\hat{y}_i = \hat{f}(x_i)$. Note:
   - The target variable y is binary with *favourable* and *unfavourable* (disease progression) outcomes.
   - The prediction scores are continuous.
   - Scores, functions, and in general any value corresponding to an favourable/unfavourable outcome are denoted with *fav* or *unfav* subscript, e.g. scores for favourable $\hat{y}_{\text{fav}}$ and unfavourable outcomes $\hat{y}_{\text{unfav}}$.

The ROC-GLM - Step 2

2. Calculate empirical survivor function $\hat{S}_{\text{unfav}} = 1 - F_n$ of scores corresponding to unfavourable observation:

$$\hat{S}_{\text{unfav}}(x) = n_{\text{unfav}}^{-1} \sum_{i \in \mathcal{I}_{\text{unfav}}} \mathbb{1}(\hat{y}_i > x) \tag{3}$$

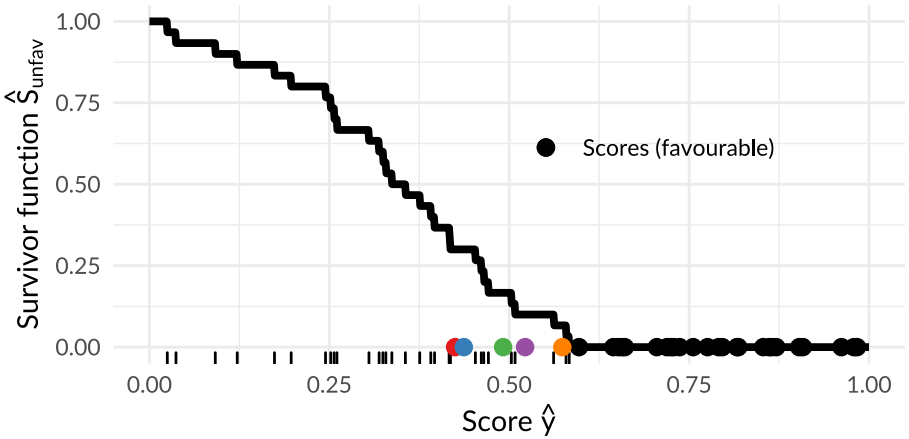


Example:
$\hat{S}_{\text{unfav}}(0.43) = 0.3$ means that 30 % of the scores of unfavourable outcomes are greater than 0.43.

The ROC-GLM - Step 3

3. Calculate **placement values** by plugging in the scores of favourable observations into the survivor function:

$$pv_i = \hat{S}_{\text{unfav}}(\hat{y}_i), \forall i \in \mathcal{I}_{\text{fav}} \tag{4}$$

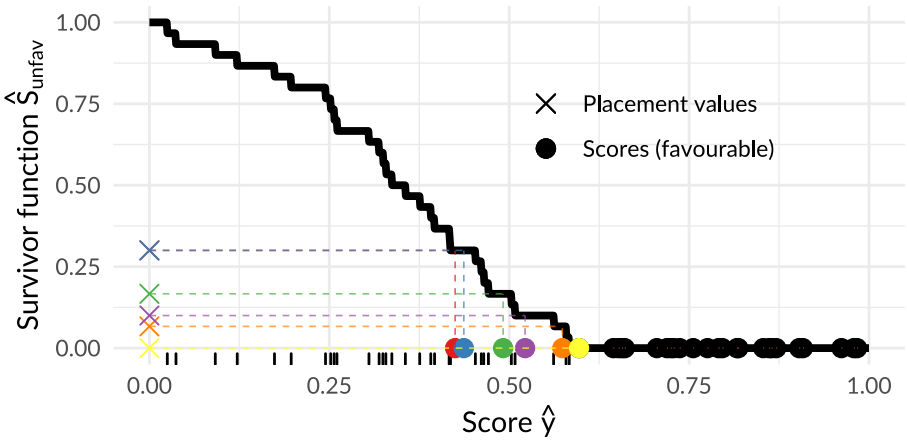


- Scores of favourable outcomes within the range of the scores of unfavourable outcomes (colored points) are “problematic” because the order is mixed
  $\Rightarrow$ Risk for missclassification.

The ROC-GLM - Step 3

3. Calculate **placement values** by plugging in the scores of favourable observations into the survivor function:

$$pv_i = \hat{S}_{\text{unfav}}(\hat{y}_i), \forall i \in \mathcal{I}_{\text{fav}} \tag{5}$$



- Scores of favourable outcomes within the range of the scores of unfavourable outcomes (colored points) are "problematic" because the order is mixed
  $\Rightarrow$ Risk for missclassification.
- The smaller this variation of the placement values is, the bigger the AUC.

The ROC-GLM - Step 4

4. Calculate an intermediate matrix U and X^* for the ROC-GLM:

$$U_{ij} = \mathbb{1}(pv_i < t_j),\ i = 1, \ldots, n_{\text{pos}},\ j = 1, \ldots, m \tag{6}$$

$$X^* = \left.\begin{pmatrix} X \\ \vdots \\ X \end{pmatrix}\right\} m\text{-times};\ X = \begin{pmatrix} 1 & \Phi^{-1}(t_1) \\ \vdots & \vdots \\ 1 & \Phi^{-1}(t_m) \end{pmatrix} \tag{7}$$

- The thresholds $t_1, \ldots, t_m$ are used to measure variability of the placement values. They are set to possible false positive rates or an arbitrary series between 0 and 1, e.g. equidistant grid.
- The matrix X corresponds to the binormal form $\Phi(\beta_1 + \beta_2 \Phi^{-1}(fpr)) = \Phi(X\beta)$.
- **Note:** It is possible to include data characteristics into the deisgn matrix X, such as age, to model individual ROC curves.

The ROC-GLM - Step 5

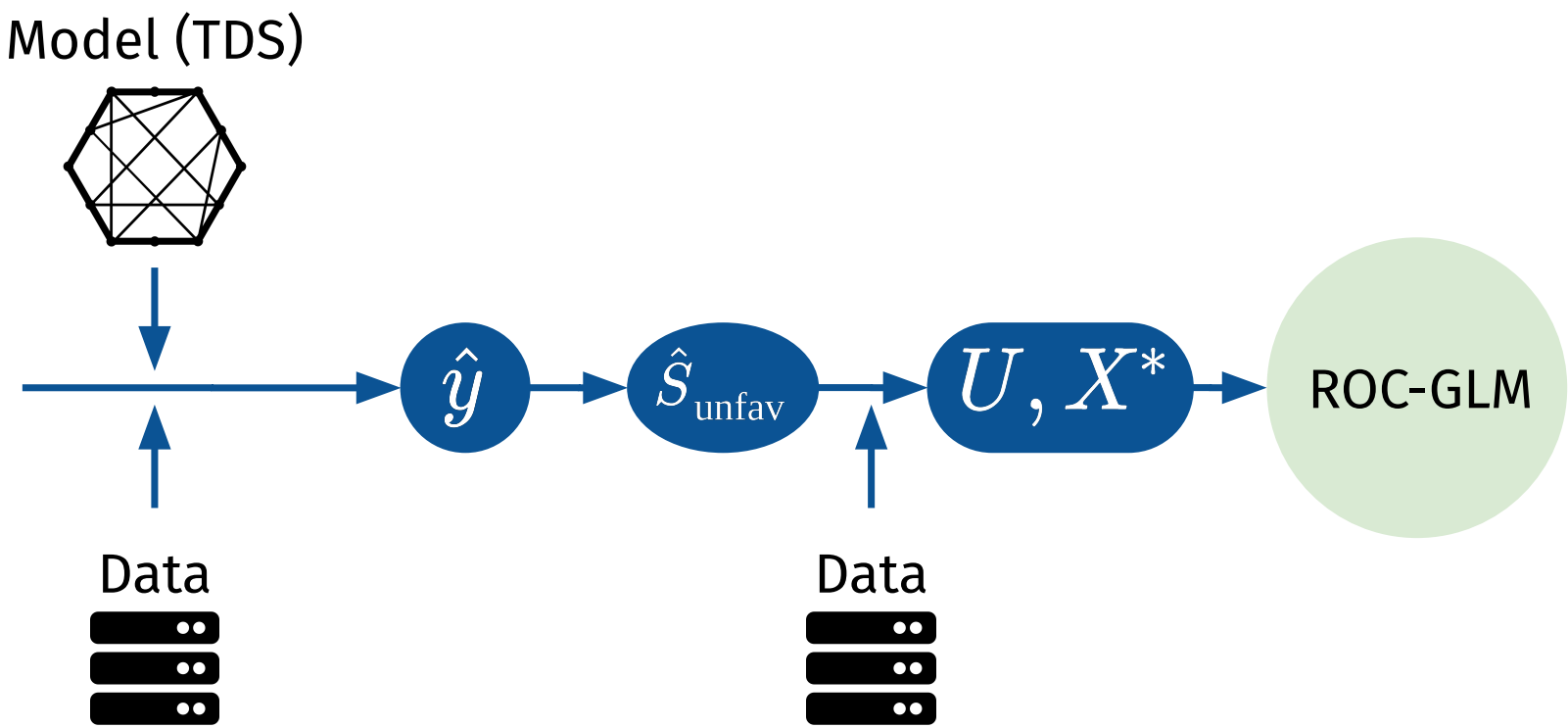
5. The ROC-GLM is a probit regression, where U acts as response, X^* as model matrix, and response function $h = \Phi$:

$$\begin{pmatrix} U_1^T \\ U_2^T \\ \vdots \\ U_{m-1}^T \\ U_m^T \end{pmatrix} \sim h \begin{pmatrix} X \\ X \\ \vdots \\ X \\ X \end{pmatrix} = h(X^*) \tag{8}$$

Remark: Rows are now transformed to the response vector.

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ & \cdots & & & & \cdots & \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_{m-1} \\ U_m \end{pmatrix}$$

The ROC-GLM - Illustration



Extension to distributed data

Two components are crucial for the ROC-GLM:

1. The survivor function $\hat{S}_{unfav}$:
   - 1.1 Scores of unfavourable outcomes are shared by the servers with extra noise added (differential privacy).
   - 1.2 Merge these scores and calculate a pooled survivor function.
   - 1.3 Share pooled survivor function with all servers.

Extension to distributed data

Two components are crucial for the ROC-GLM:

1. The survivor function $\hat{S}_{\text{unfav}}$:
   - 1.1 Scores of unfavourable outcomes are shared by the servers with extra noise added (differential privacy).
   - 1.2 Merge these scores and calculate a pooled survivor function.
   - 1.3 Share pooled survivor function with all servers.
2. The probit regression:
   - Objective is to maximize the log-likelihood $\ell_\beta(y, X)$ for β.
   - The optimization technique used in GLMs is the Fisher scoring with score vector s and Fisher information I:

$$\hat{\beta} \leftarrow \hat{\beta} + I(\hat{\beta})^{-1} s(\beta) \tag{9}$$

   - The key to calculate the probit regression (and GLMs in general) distributively is to aggregate the Fisher information I and score vector s over all servers.

Distributed GLM

input : K client data $(y_1, X_1), \ldots, (y_K, X_K)$
output: Estimated parameter vector $\widehat{\beta}$

Initialize: m_{stop} (e.g. $m_{stop} = 25$), ε (e.g. $\varepsilon = 10^{-8}$), and $\widehat{\beta}^{[0]}$ (e.g. $\widehat{\beta}^{[0]} = \vec{0}$)

while $dev_{stop}^{[m]} \leq \varepsilon$ ***and*** $m \leq m_{stop}$ **do**

  for $k \leftarrow 1, \ldots, K$ **do**

    Calculate and share $l_k(\widehat{\beta}^{[m-1]})$ and $s_k(\widehat{\beta}^{[m-1]})$

  end

  Sum up client parts:

  - $l(\widehat{\beta}^{[m-1]}) \leftarrow \sum_{k=1}^{K} l_k(\widehat{\beta}^{[m-1]})$
  - $s(\widehat{\beta}^{[m-1]}) \leftarrow \sum_{k=1}^{K} s_k(\widehat{\beta}^{[m-1]})$

  Conduct Fisher scoring step: $\widehat{\beta}^{[m]} \leftarrow \widehat{\beta}^{[m-1]} + l(\widehat{\beta}^{[m-1]})^{-1} s(\widehat{\beta}^{[m-1]})$

  Iterate: $m \leftarrow m + 1$

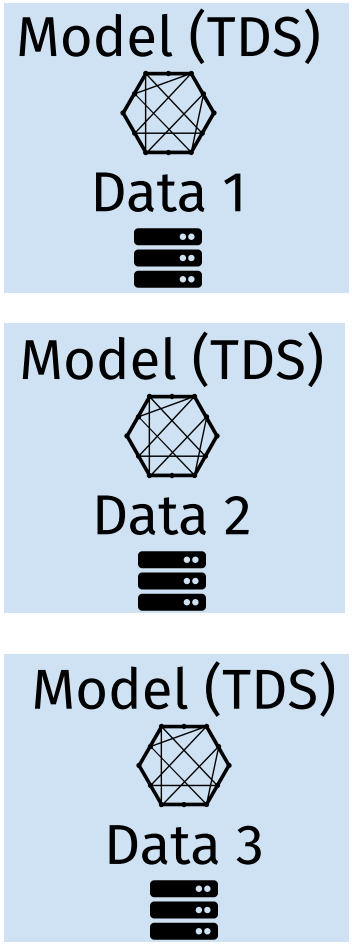
end

return : $\widehat{\beta} \leftarrow \widehat{\beta}^{[m]}$

Algorithm 1: Distributed probit regression. Further information can be found in the backup slides.

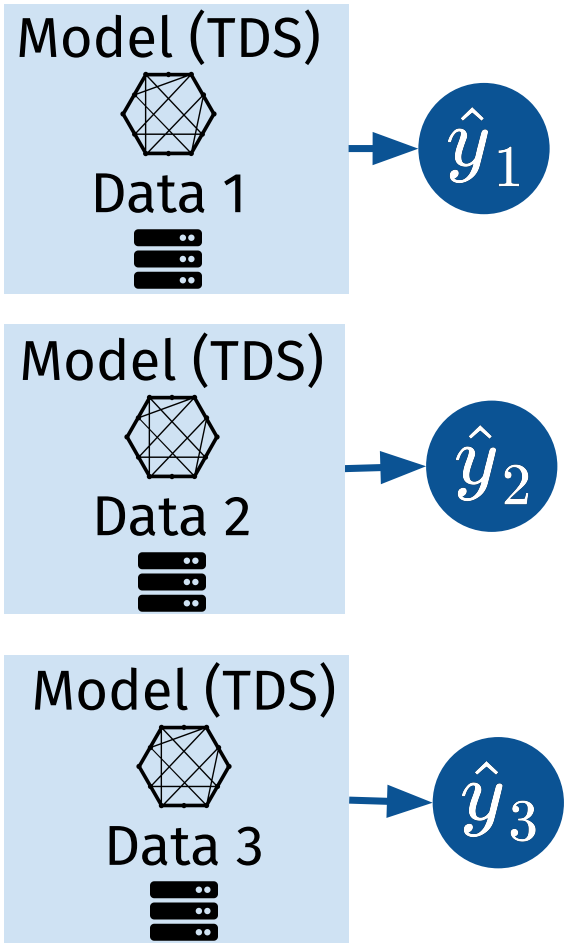
Distributed ROC-GLM - Step 1

Given data and model:



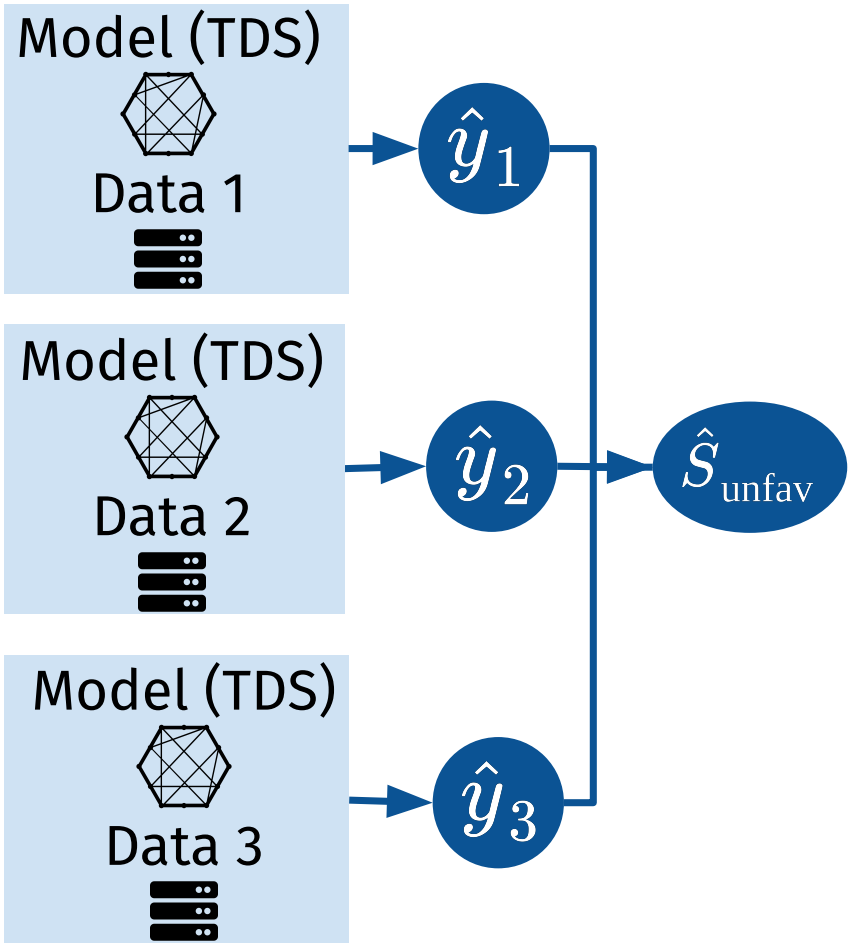
Distributed ROC-GLM - Step 2

The servers are sharing the scores of unfavourable outcomes with additional noise added to them:



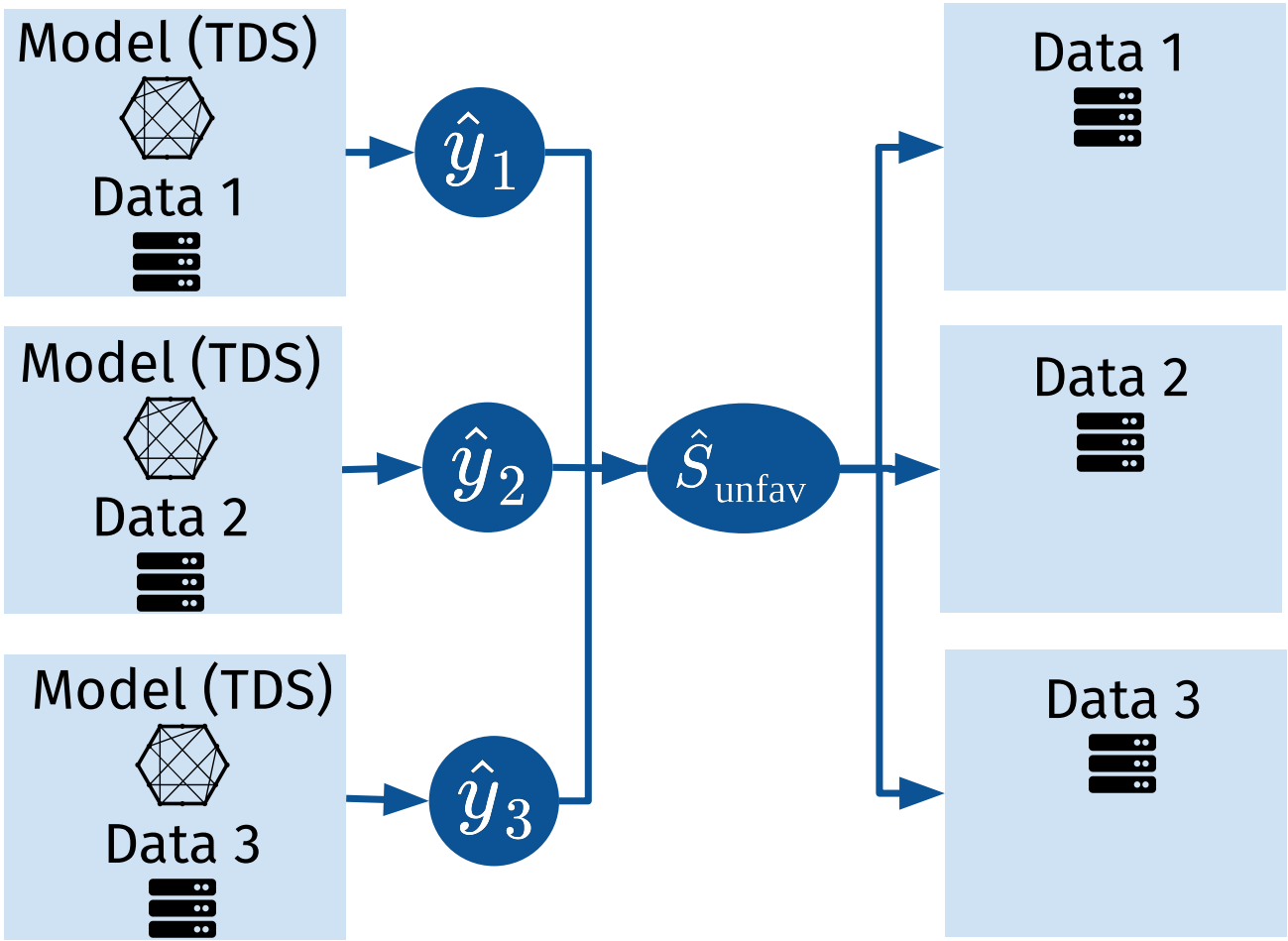
Distributed ROC-GLM - Step 3

The host calculates the survivor function $\hat{S}_{unfav}$ on the pooled scores as described for the ROC-GLM:



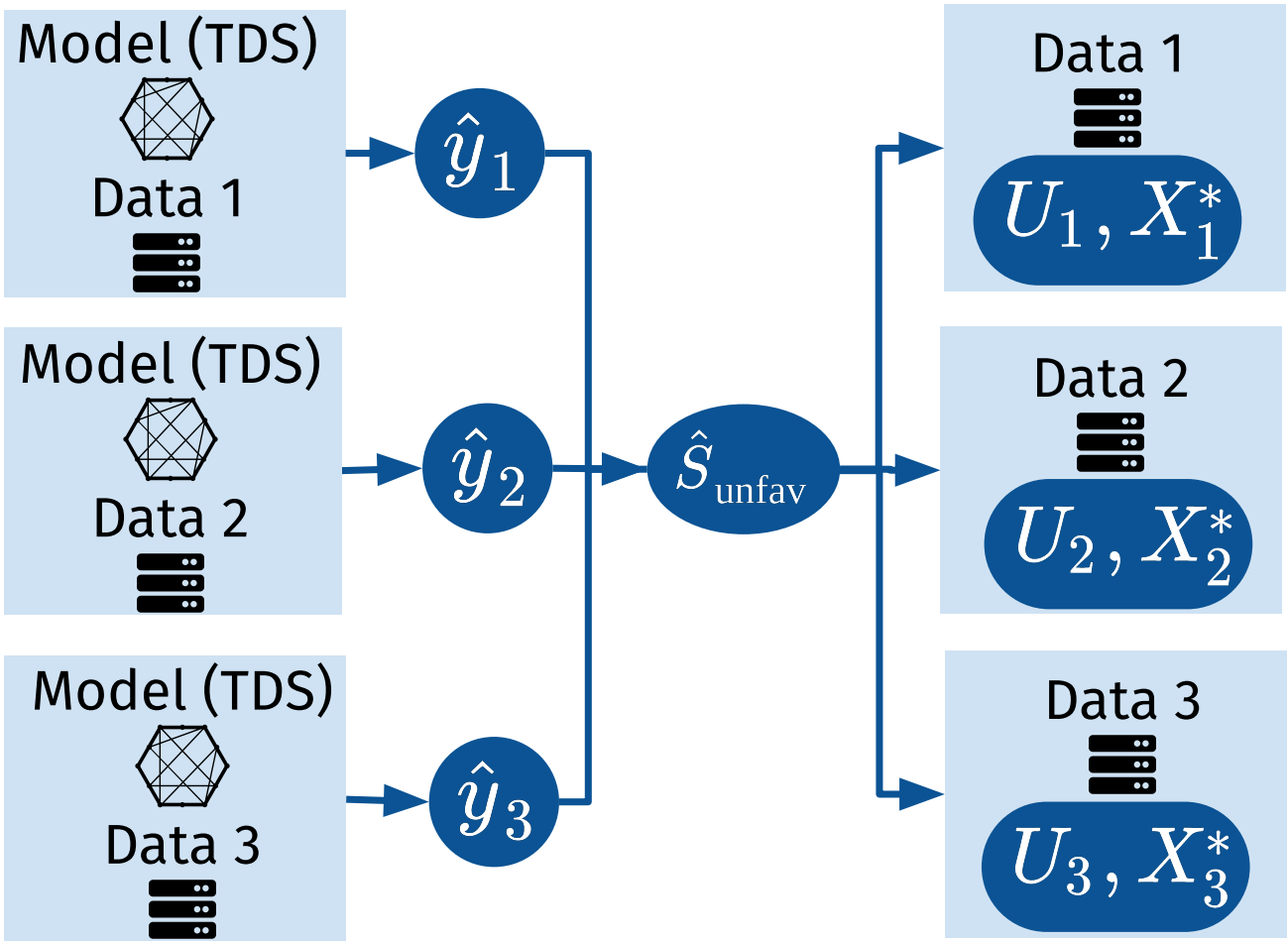
Distributed ROC-GLM - Step 4

The survivor function is shared to all servers:



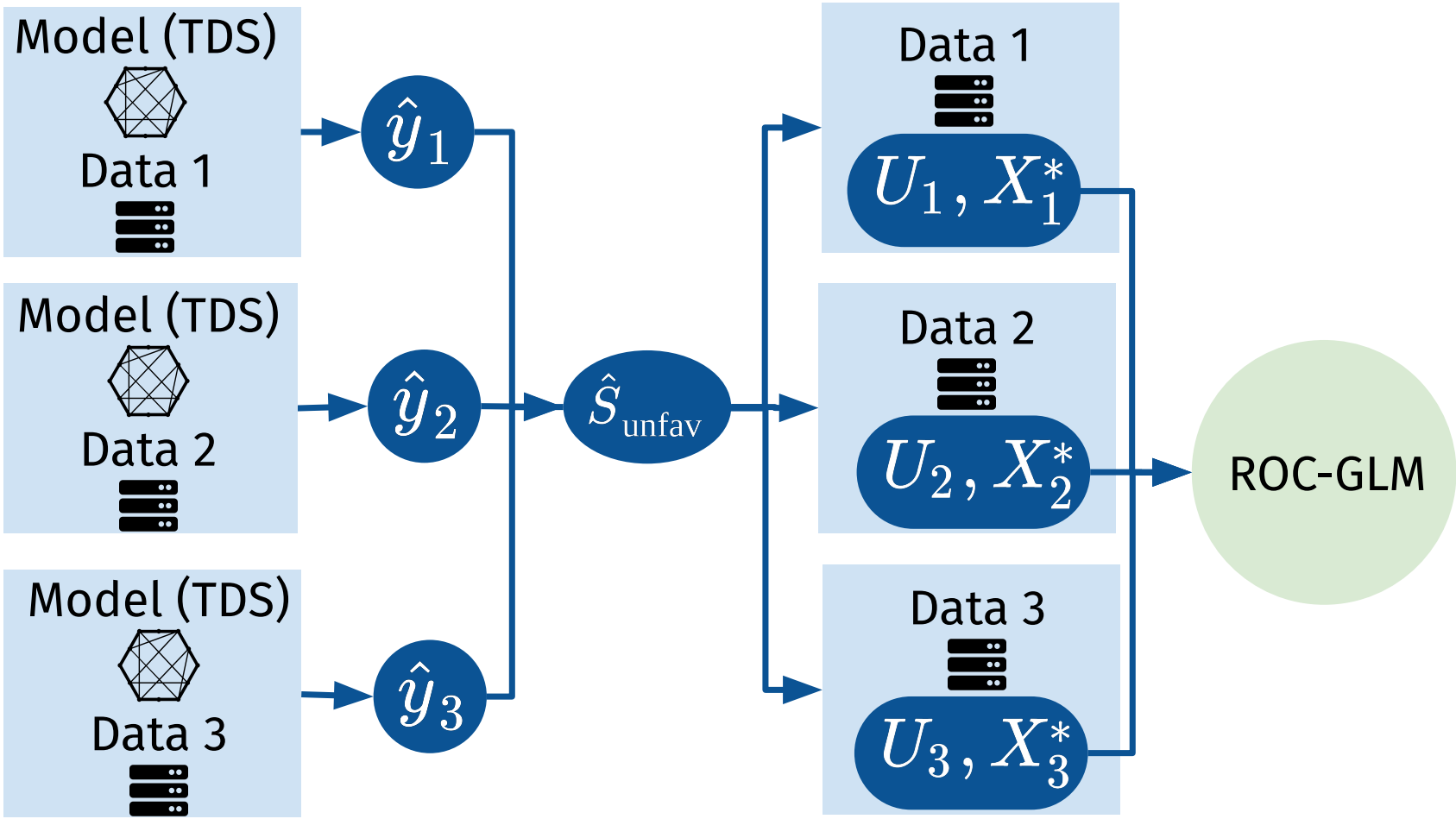
Distributed ROC-GLM - Step 5

The servers are now able to calculate the intermediate matrices U and X^* needed for the ROC-GLM:



Distributed ROC-GLM - Step 6

The ROC-GLM can be calculated as distributed probit regression:

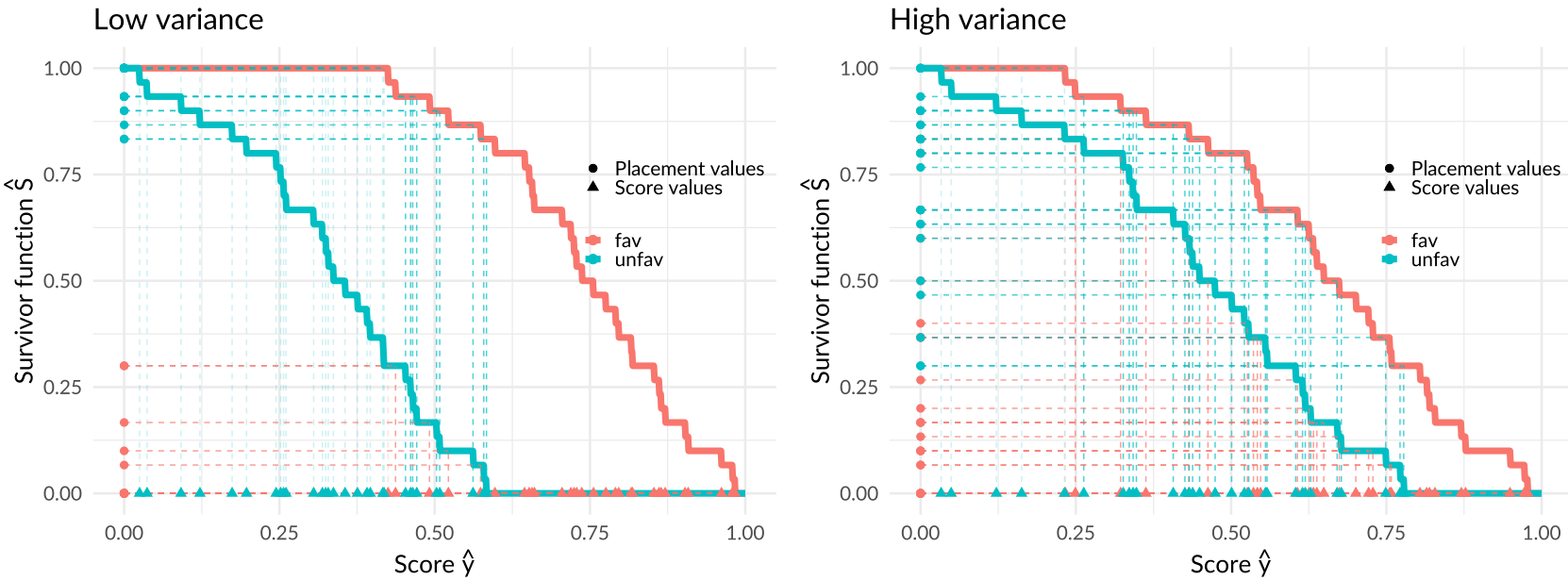


Theoretical background - Confidence intervals

DeLong confidence interval i

- CIs of the AUC after DeLong [1] are based on the variability within the placement values ($\widehat{\text{var}}$: sample variance):

$$\widehat{\text{var}}(\text{AUC}) = \frac{\widehat{\text{var}}(\hat{S}_{\text{fav}}(\hat{y}_{\text{unfav}}))}{n_{\text{unfav}}} + \frac{\widehat{\text{var}}(\hat{S}_{\text{unfav}}(\hat{y}_{\text{fav}}))}{n_{\text{fav}}} \tag{10}$$



DeLong confidence interval ii

- To not fall outside of the AUC domain of $[0, 1]$, the CI is calculated for the logit transformation

$$\text{logit}\left(\text{AUC}\right) = \log(\text{AUC}/(1 - \text{AUC})). \tag{11}$$

- The confidence interval then is defined by

$$\text{ci}_\alpha = \text{logit}\left(\text{AUC}\right) \pm \Phi^{-1}\left(1 - \frac{\alpha}{2}\right) \frac{\sqrt{\widehat{\text{var}}\left(\text{AUC}\right)}}{\text{AUC}\left(1 - \text{AUC}\right)}. \tag{12}$$

- The boundaries of ci_α then can be transformed back to the $[0, 1]$ domain of the AUC.

Distributed DeLong confidence interval

- Similar to log-likelihood, the variance can be split into the individual sums calculated on each of the K servers:

$$\widehat{\text{var}}(\hat{S}_{\text{unfav}}(\hat{y}_{\text{fav}})) = \sum_{k=1}^{K} \widehat{\text{var}}(\hat{S}_{\text{unfav}}(\hat{y}_{\text{fav},k})) \tag{13}$$

$$\widehat{\text{var}}(\hat{S}_{\text{fav}}(\hat{y}_{\text{unfav}})) = \sum_{k=1}^{K} \widehat{\text{var}}(\hat{S}_{\text{fav}}(\hat{y}_{\text{unfav},k})) \tag{14}$$

  $\hat{y}_{\text{unfav},k}$ indicates the negative scores of the k-th server.
- Each server can share $\widehat{\text{var}}(\hat{S}_{\text{unfav}}(\hat{y}_{\text{fav},k})$ and $\widehat{\text{var}}(\hat{S}_{\text{fav}}(\hat{y}_{\text{unfav},k}))$ without privacy breaches (highly aggregated).
- The received variance values are then aggregated by formula (14) and (13). Then, formula (12) can be applied to get the CI.

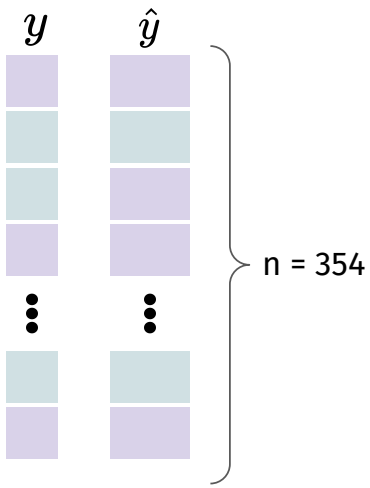
Experimental results

Simulation setup

1. Simulate prediction scores $Y_i \sim U[0, 1]$ and true class $D_i = \mathbb{1}(Y_i \geq 0.5)$ for $i = 1, \ldots, n$ with n randomly picked from $\{100, \ldots, 2500\}$.
2. Draw a random index set $\mathcal{I}$ of size $\lfloor \gamma n \rfloor$ with $\gamma \sim U[0.2, 0.8]$.
3. Randomly flip true labels $D_i \sim \text{Ber}(0.5), \ \forall i \in \mathcal{I}$.
4. Calculate empirical AUC and AUC from the ROC-GLM. Note that the AUC values in the distributed setup are exactly the same as the ones from the ROC-GLM.
5. For both AUC methods, calculate the CI after DeLong and bootstrap CIs for control.

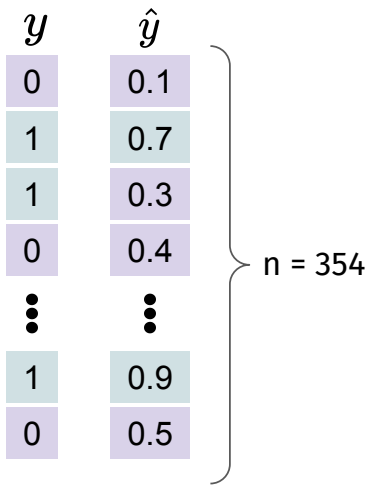
Steps 1 - 5 are repeated 10 000 times.

Simulation setup (illustration)



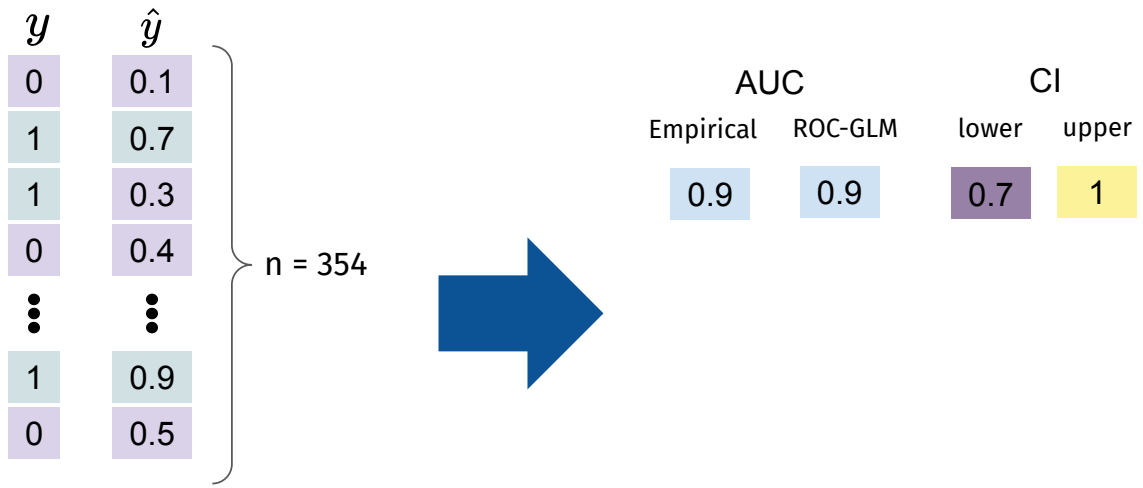
$y, \hat{y}$ and n randomly generated

Simulation setup (illustration)



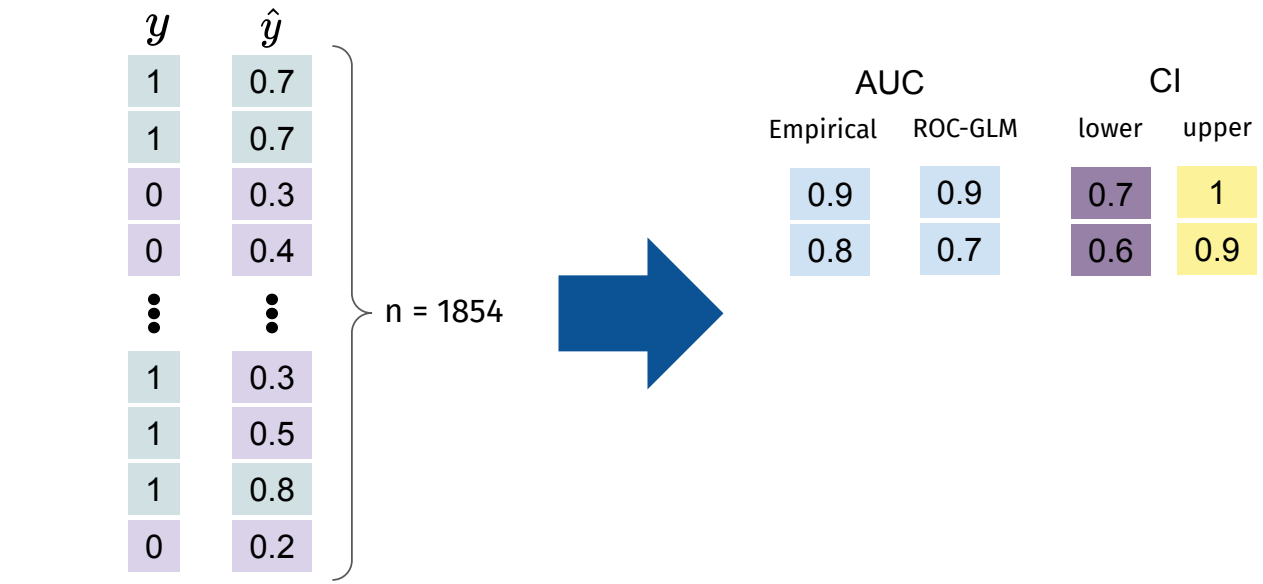
$y, \hat{y}$ and n randomly generated

Simulation setup (illustration)



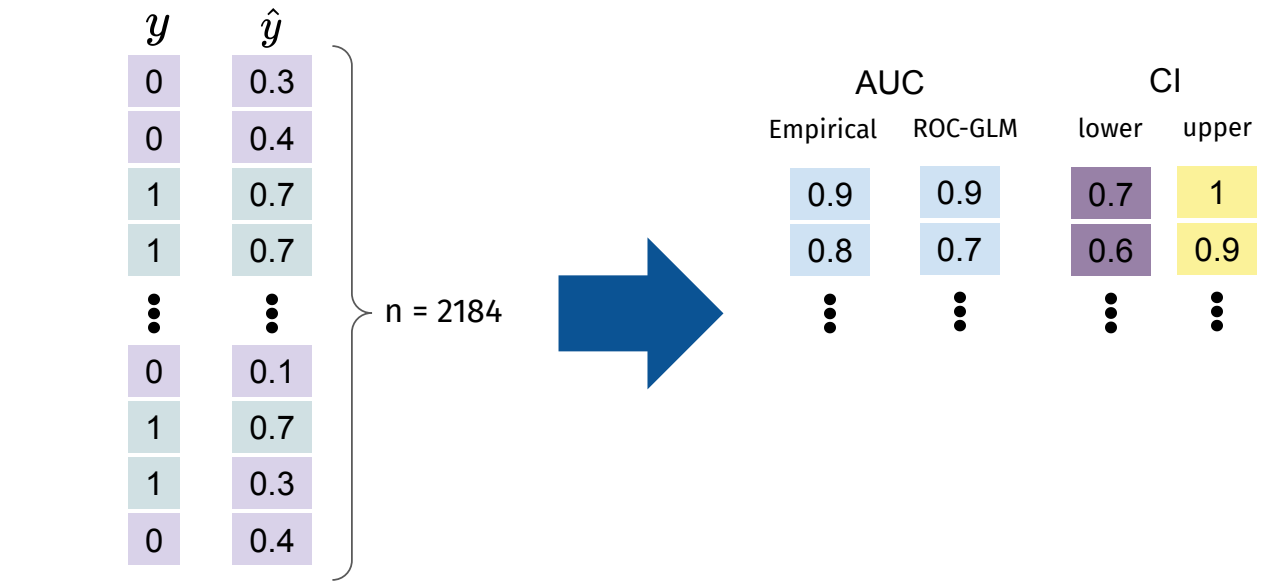
$y, \hat{y}$ and n randomly generated

Simulation setup (illustration)



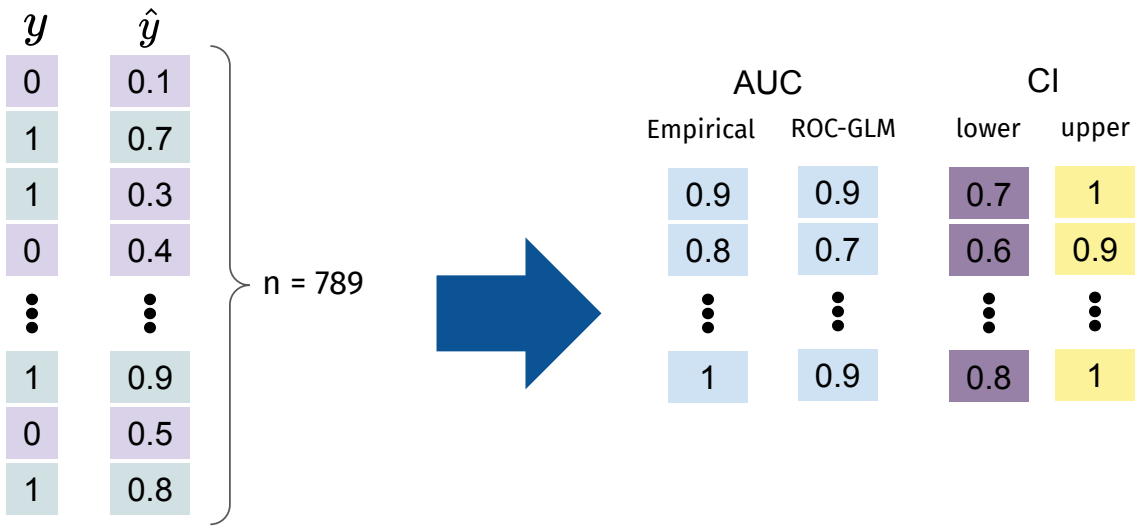
$y, \hat{y}$ and n randomly generated

Simulation setup (illustration)



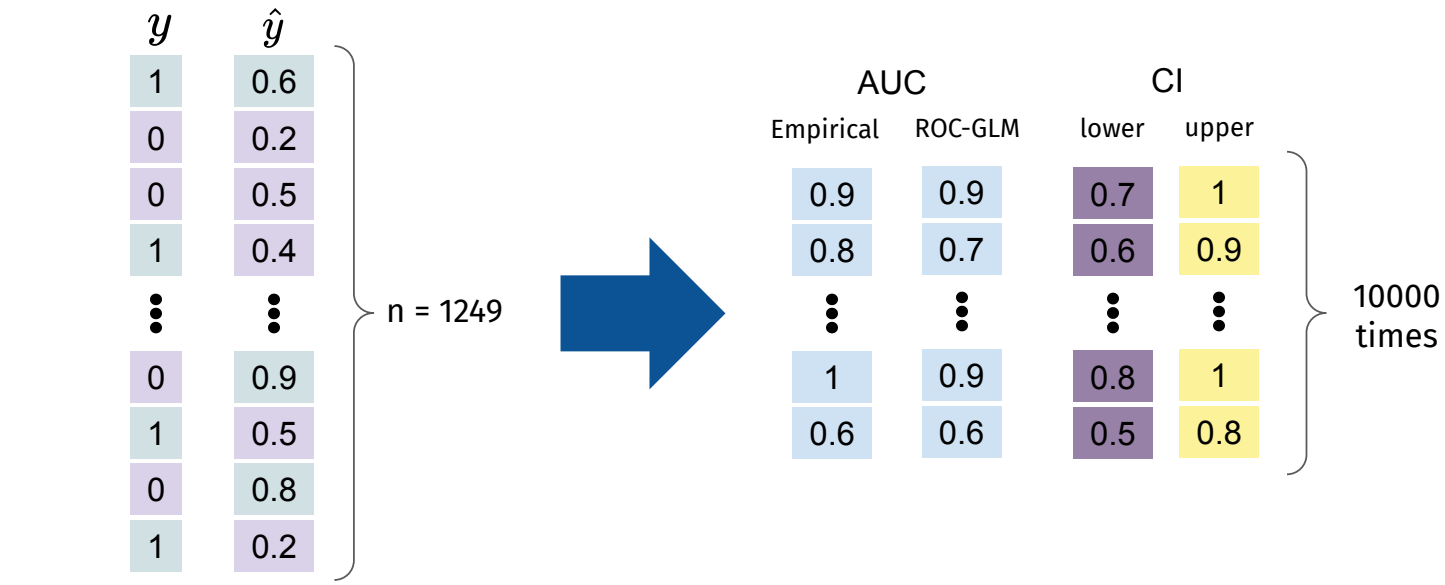
$y, \hat{y}$ and n randomly generated

Simulation setup (illustration)



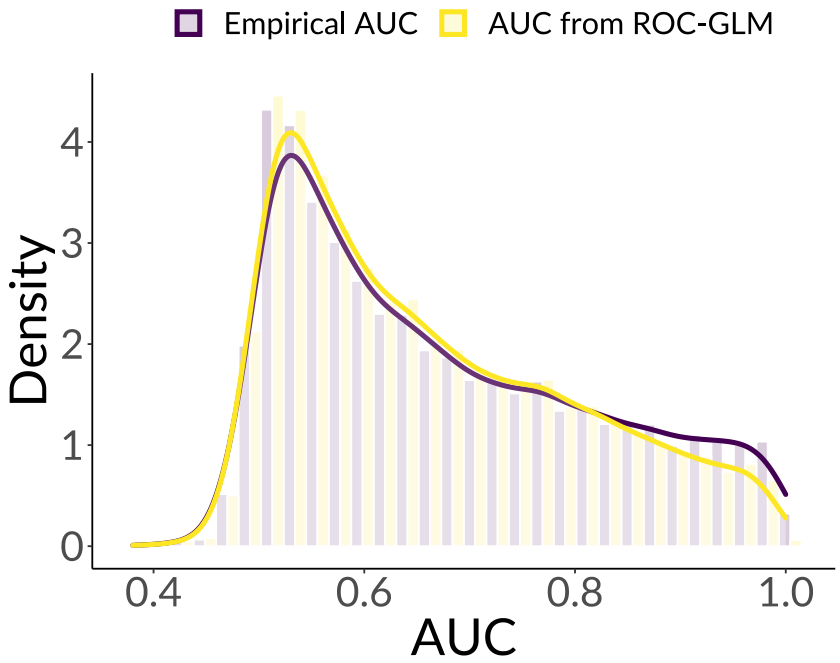
$y, \hat{y}$ and n randomly generated

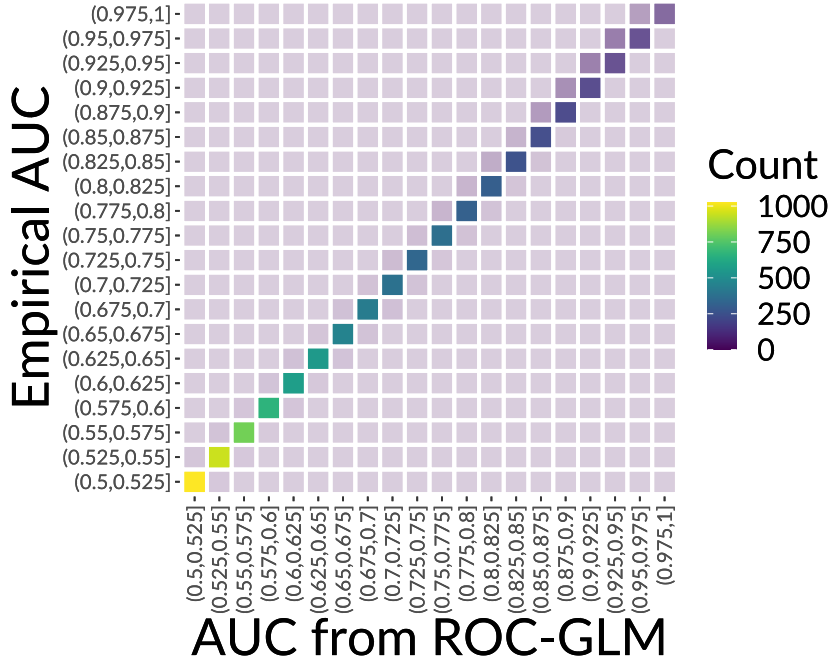
Simulation setup (illustration)



$y, \hat{y}$ and n randomly generated

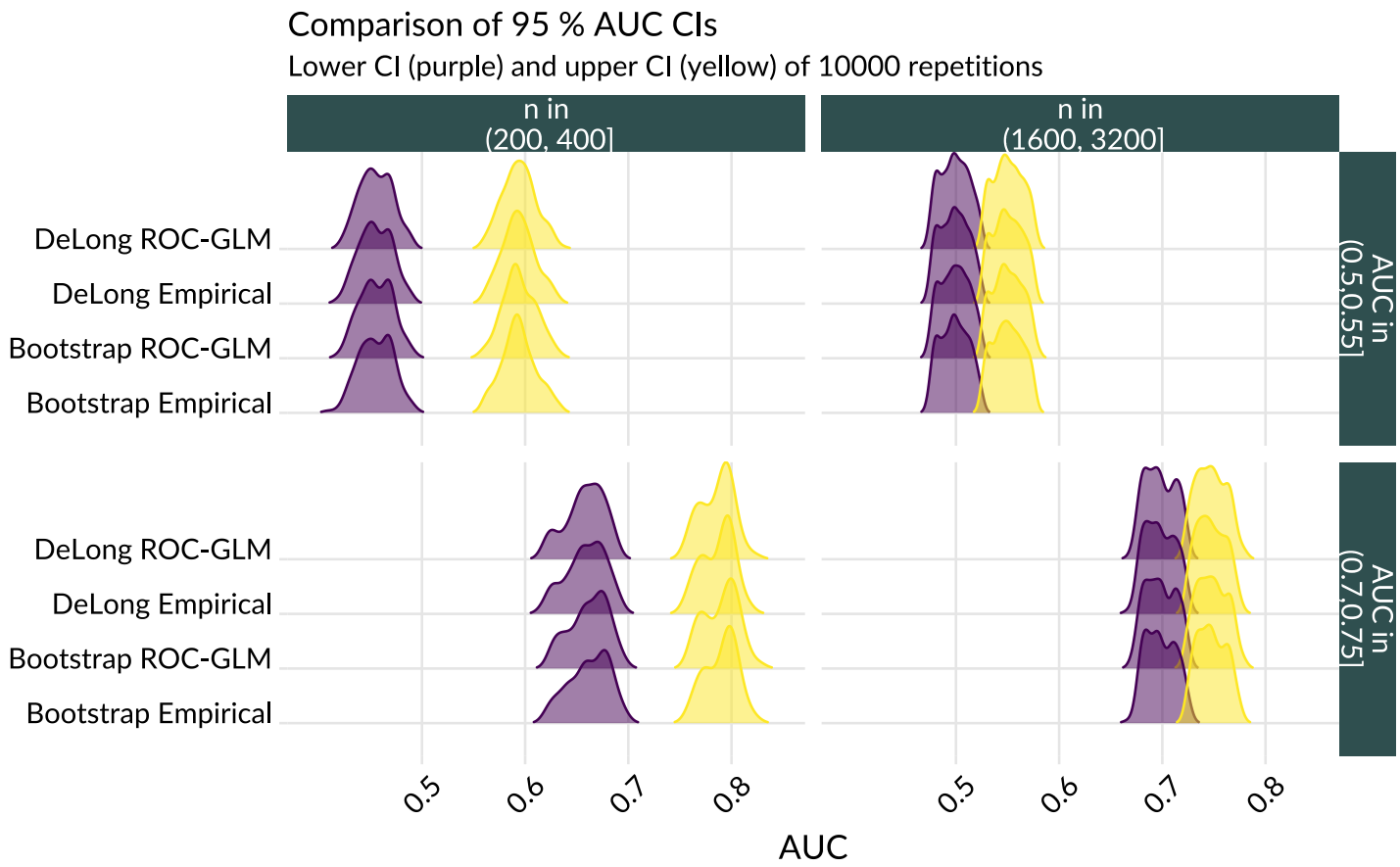
Results I: ROC-GLM





	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Empirical AUC	0.3804	0.5488	0.6392	0.6740	0.7813	1.0000
Approximated AUC	0.3810	0.5457	0.6289	0.6620	0.7611	0.9959
Difference	-0.0186	-0.0018	-0.0004	-0.0009	0.0007	0.0080

Results II: CI



	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Difference lower CI	-0.0131	-0.0001	0.0000	0.0009	0.0016	0.0243
Difference upper CI	-0.0108	-0.0002	0.0000	0.0009	0.0014	0.0221

Use case

Setup

- Data are simulated and distributed over 5 servers.
- Logistic regression as model which we want to evaluate (dummy for the TDS).
- DataSHIELD [2] is used as software framework for evaluating this model on the distributed data with new libraries specifically written for this purpose:
  - Package to push models from local machine to servers.
  - Package to predict on the servers.
  - Package to calculate the distributed ROC-GLM and CIs.

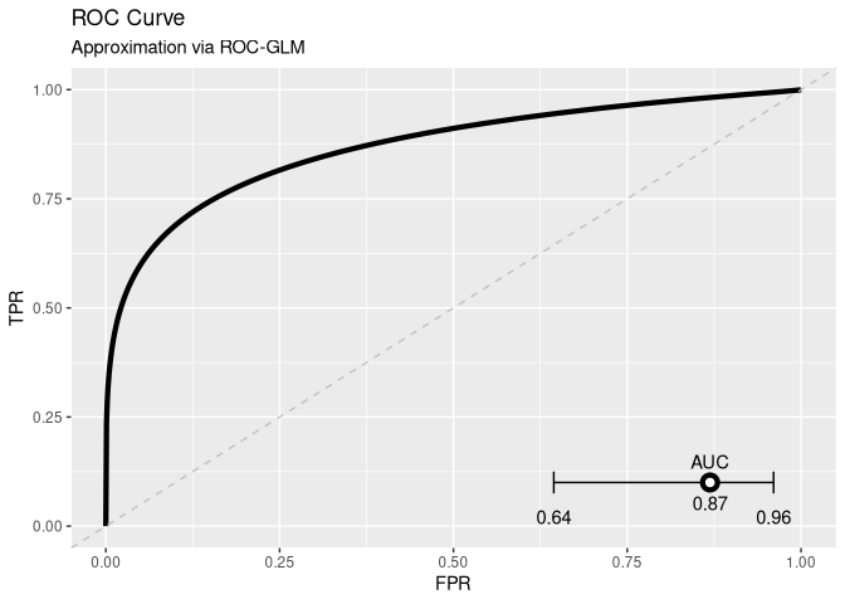
Calculate the ROC-GLM

```
## Log into DataSHIELD servers.
...

## Push and predict model
load("dts.Rda") # Gives logistic regression model "mod"

pushModel(connections, mod)
predictModel(connections, mod, "pred", "D",
  predict_fun = "predict(mod, newdata = D, type = 'response')")

## Calculate ROC-GLM
roc_glm = dsROCGLM(connections, "D$gender", "pred")
plot(roc_glm)
```



Final Remarks

Conclusion

- We are able to evaluate models, especially the TDS, in a distributed setup using the AUC without sharing patient data.
- The approximated AUC is very close to the true empirical AUC with small exceptions at the borders.
- The confidence interval for the approximated AUC is very close to the one for the empirical AUC.
- We already implemented the presented methods in DataSHIELD (packages will be publicy available soon).

Thanks for your attention!
Any Questions?

References

[1] E. R. DeLong, D. M. DeLong, and D. L. Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, pages 837–845, 1988.

[2] A. Gaye, Y. Marcon, J. Isaeva, P. LaFlamme, A. Turner, E. M. Jones, J. Minion, A. W. Boyd, C. J. Newby, M.-L. Nuotio, et al. Datashield: taking the analysis to the data, not the data to the analysis. *International journal of epidemiology*, 43(6):1929–1944, 2014.

[3] M. S. Pepe. The Statistical Evaluation of Medical Tests for Classification and Prediction. *Journal of the American Statistical Association*, 2003.

Backup slides

Backup - Probit regression i

- The probit regression models a Bernoulli distributed outcome $Y_i \sim \text{Ber}(h(\eta_i)$, response function $h(\eta_i) = \Phi(\eta_i)$, and $\eta_i = x_i^T \beta$ the linear predictor with model parameters β.
- Hence, the likelihood L and log-likelihood l is given by

$$L_\beta(y, X) = \prod_{i=1}^{n} h(\eta_i)^{y_i} \left(1 - h(\eta_i)\right)^{1-y_i} \tag{15}$$

$$\ell_\beta(y, X) = \log(L_\beta(y, X)) = \sum_{i=1}^{n} \log\left(h(\eta_i)^{y_i} \left(1 - h(\eta_i)\right)^{1-y_i}\right) \tag{16}$$

Backup - Probit regression ii

- Fitting the model, or in general GLMs, is done by maximizing the log-likelihood ℓ_β using the Fisher scoring algorithm based on the score vector s and Fisher information I:

$$s(\beta) = s(\beta \mid y, X) = \frac{\partial\, \ell_\beta(y, X)}{\partial \beta} \tag{17}$$

$$I(\beta) = I(\beta \mid y, X) = \frac{\partial\, s(\beta \mid y, X)}{\partial \beta} \tag{18}$$

- Finding the parameter estimate $\hat{\beta}$ is done by repeatedly executing

$$\hat{\beta} \leftarrow \hat{\beta} + I(\hat{\beta})^{-1} s(\beta) \tag{19}$$

until convergence is reached.

Backup - Distributed probit regression

Idea for K clients

- Split log-likelihood in client-based sums:

$$\ell_\beta(y, X) = \sum_{k=1}^{K} \ell_\beta(y_k, X_k) \tag{20}$$

- Due to linearity of the derivation we have the same structure for the Fisher information and the score vector:

$$s(\beta) = s(\beta \mid y, X) = \sum_{k=1}^{K} s(\beta \mid y_k, X_k) = \sum_{k=1}^{K} s_k(\beta)$$

$$I(\beta) = I(\beta \mid y, X) = \sum_{k=1}^{K} I(\beta \mid y_k, X_k) = \sum_{k=1}^{K} I_k(\beta)$$

Distributed ROC-GLM - Procedure

Servers are reporting the negative scores:

1. Step 1 of the ROC-GLM procedure (slide 6) is applied on all servers.
2. While on the server, random noise is added to the scores. The noise is big enough to not be able to identify original scores but not too much to break the ROC-GLM.
3. The scores are shared to the analyst.

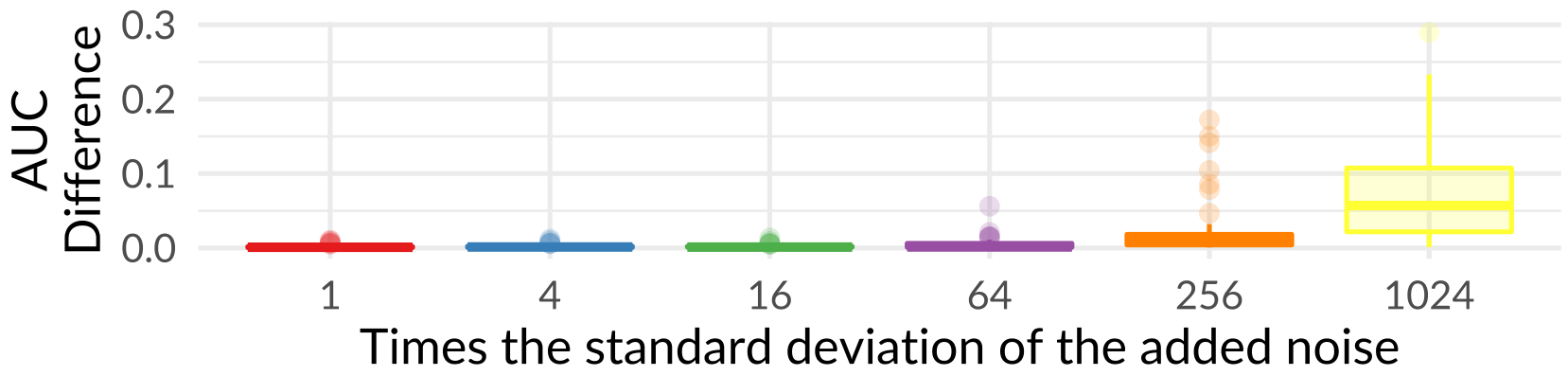


Figure 1: Approximation error when adding more noise.

Results II: CI