Comparative Analysis of Implementations of Gradient Boosting for Decision Trees in Insurance

59th Actuarial Research Conference

Work from

Dominik Chevalier and
Marie-Pier Côté

École d'actuariat, Université Laval

July 18th, 2024

Murfreesboro, TN

Université Laval

Gradient boosting proliferates in actuarial science

Henckaerts et al. (2021) conclude that gradient boosting machines (GBM) improve the performance over GLMs for general insurance **ratemaking**.

Gradient boosting proliferates in actuarial science

Henckaerts et al. (2021) conclude that gradient boosting machines (GBM) improve the performance over GLMs for general insurance **ratemaking**. Other uses in insurance applications :

- GBM in Wirawan and Gunardi (2022) and Clemente et al. (2023),
- XGBoost in Gao et al. (2023).

Gradient boosting proliferates in actuarial science

Henckaerts et al. (2021) conclude that gradient boosting machines (GBM) improve the performance over GLMs for general insurance **ratemaking**. Other uses in insurance applications :

- GBM in Wirawan and Gunardi (2022) and Clemente et al. (2023),
- XGBoost in Gao et al. (2023).

Crevecoeur et al. (2022)

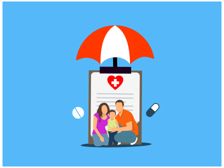
Images by Azam Ishaq and Mohamed Hassan from Pixabay

Gradient boosting proliferates in actuarial science

Henckaerts et al. (2021) conclude that gradient boosting machines (GBM) improve the performance over GLMs for general insurance **ratemaking**. Other uses in insurance applications :

- GBM in Wirawan and Gunardi (2022) and Clemente et al. (2023),
- XGBoost in Gao et al. (2023).

Crevecoeur et al. (2022)



Hartman et al. (2020)

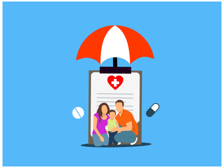
Images by Azam Ishaq and Mohamed Hassan from Pixabay

Gradient boosting proliferates in actuarial science

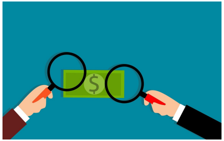
Henckaerts et al. (2021) conclude that gradient boosting machines (GBM) improve the performance over GLMs for general insurance **ratemaking**. Other uses in insurance applications :

- GBM in Wirawan and Gunardi (2022) and Clemente et al. (2023),
- XGBoost in Gao et al. (2023).

Crevecoeur et al. (2022)



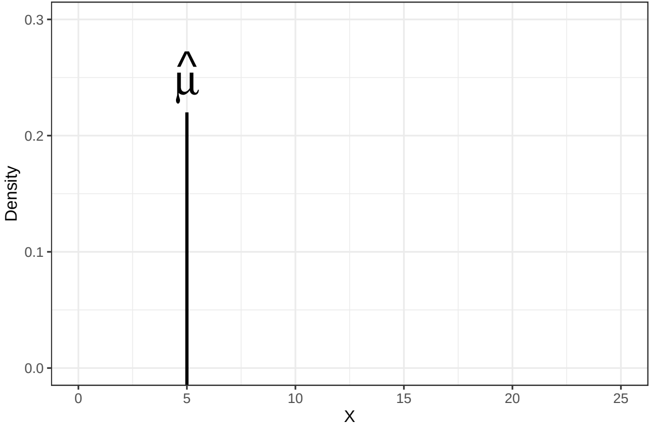
Hartman et al. (2020)



Hancock and Khoshgoftaar (2020)

Images by Azam Ishaq and Mohamed Hassan from Pixabay

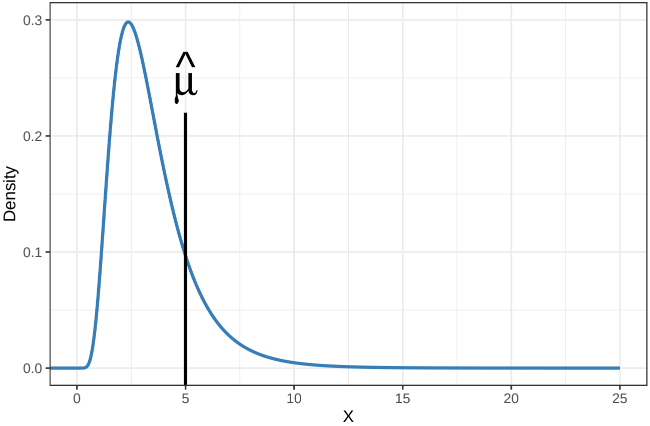
Point prediction



Model output is

$$f(\mathbf{x}) = \widehat{E[Y|\mathbf{x}]}.$$

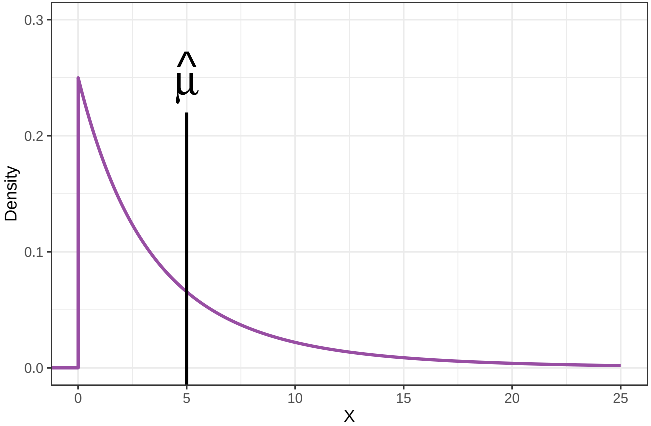
Point prediction



Model output is

$$f(\mathbf{x}) = \widehat{E[Y|\mathbf{x}]}.$$

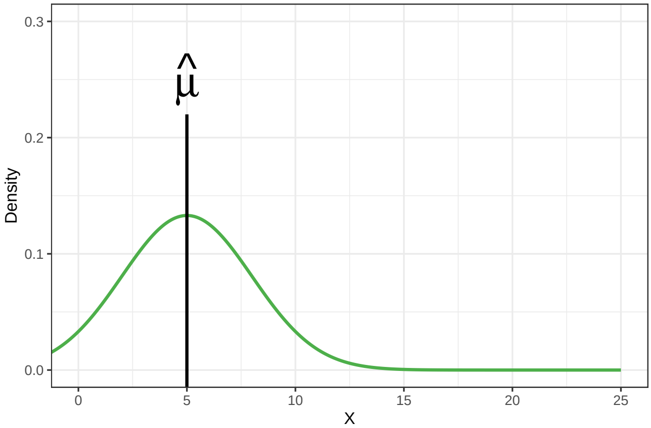
Point prediction



Model output is

$$f(\mathbf{x}) = \widehat{E[Y|\mathbf{x}]}.$$

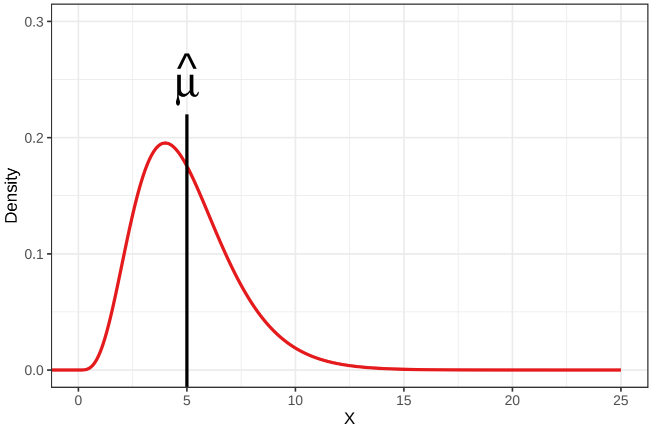
Point prediction



Model output is

$$f(\mathbf{x}) = \widehat{E[Y|\mathbf{x}]}.$$

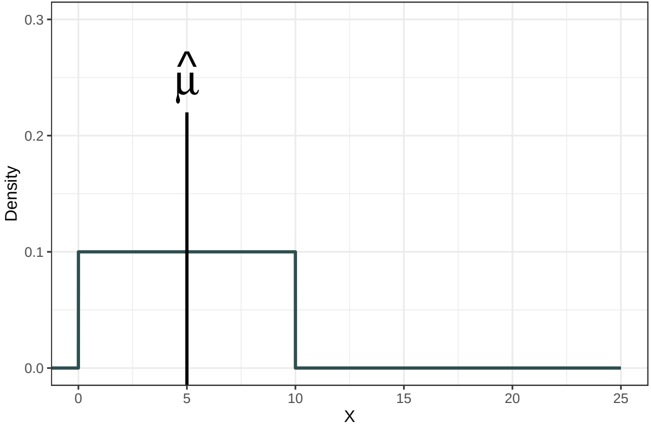
Point prediction



Model output is

$$f(\mathbf{x}) = \widehat{E[Y|\mathbf{x}]}.$$

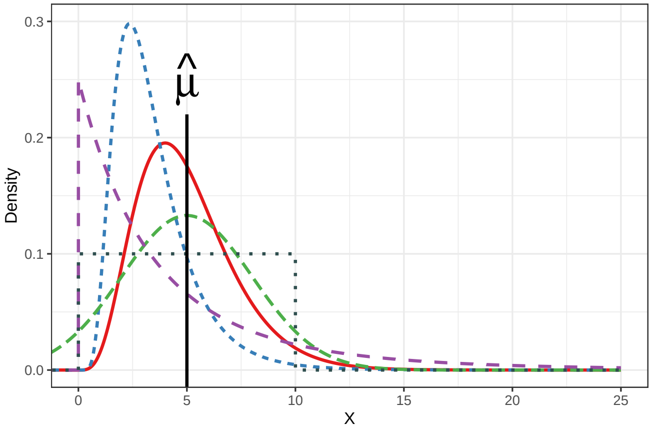
Point prediction



Model output is

$$f(\mathbf{x}) = \widehat{E[Y|\mathbf{x}]}.$$

Point prediction



Model output is

$$f(\mathbf{x}) = \widehat{E[Y|\mathbf{x}]}.$$

Insufficient for risk management!

From point predictions to probabilistic predictions

Point prediction

$$\widehat{E[Y|\mathbf{x}]} = \hat{\mu}(\mathbf{x})$$

From point predictions to probabilistic predictions

Point prediction

$$\widehat{E[Y|\mathbf{x}]} = \hat{\mu}(\mathbf{x})$$

Probabilistic prediction

$$\widehat{\Pr[Y \leq y|\mathbf{x}]}$$

From point predictions to probabilistic predictions

Point prediction

$$\widehat{E[Y|\mathbf{x}]} = \hat{\mu}(\mathbf{x})$$

Probabilistic prediction

$$\Pr[\widehat{Y \leq y}|\mathbf{x}] = F_Y\{y; \hat{\mu}(\mathbf{x}), \hat{\sigma}\}$$

- From **point predictions**, we can get **probabilistic predictions** by assuming some parameters as constant.

From point predictions to probabilistic predictions

Point prediction

$$\widehat{E[Y|\mathbf{x}]} = \hat{\mu}(\mathbf{x})$$

- From **point predictions**, we can get **probabilistic predictions** by assuming some parameters as constant.

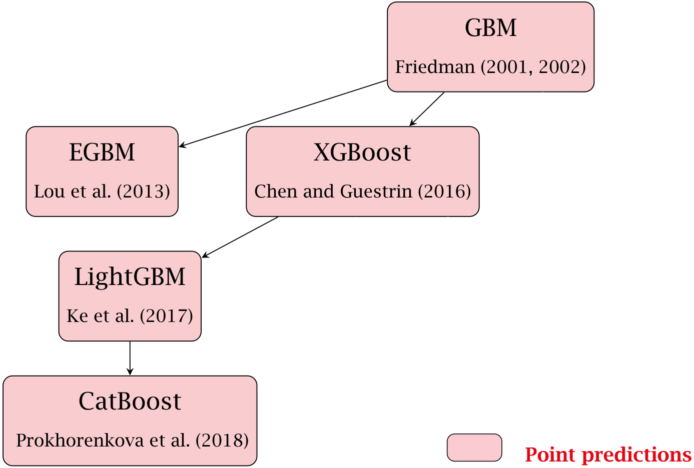
Probabilistic prediction

$$\Pr[\widehat{Y \leq y}|\mathbf{x}] = F_Y\{y; \hat{\mu}(\mathbf{x}), \hat{\sigma}\}$$

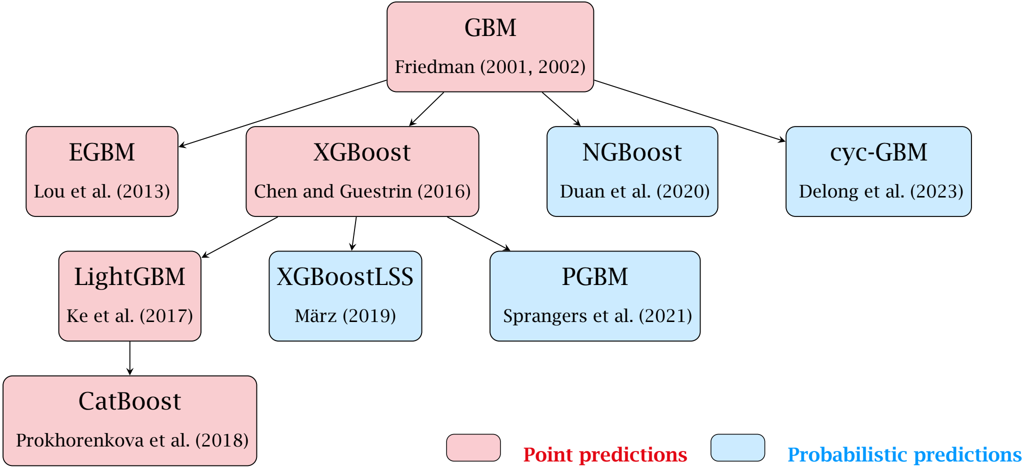
$$\Pr[\widehat{Y \leq y}|\mathbf{x}] = F_Y\{y; \hat{\mu}(\mathbf{x}), \hat{\sigma}(\mathbf{x})\}$$

- Recent **probabilistic boosting** algorithms relax this assumption.

Recent implementations of gradient boosting for decision trees



Recent implementations of gradient boosting for decision trees



Which algorithm should we use ?

So (2024) compares **XGBoost**, **LightGBM** and **CatBoost**, and concludes that **CatBoost performs best** with Poisson and ZIP on the French MTPL data and a simulated dataset.

Which algorithm should we use ?

So (2024) compares **XGBoost**, **LightGBM** and **CatBoost**, and concludes that **CatBoost performs best** with Poisson and ZIP on the French MTPL data and a simulated dataset.

Power et al. (2024) construct a multivariate claim severity model with **XGBoost** or **XGBoostLSS** marginals linked in a copula.

Which algorithm should we use ?

So (2024) compares **XGBoost**, **LightGBM** and **CatBoost**, and concludes that **CatBoost performs best** with Poisson and ZIP on the French MTPL data and a simulated dataset.

Power et al. (2024) construct a multivariate claim severity model with **XGBoost** or **XGBoostLSS** marginals linked in a copula.

What about **probabilistic boosting** ?

- Is there a compromise between model adequacy and predictive performance ?

Research objectives

1. To compare **point** and **probabilistic** gradient boosting algorithms for frequency and severity data in terms of :
   1. computational efficiency,
   2. predictive performance, and
   3. **model adequacy**.

Research objectives

1. To compare **point** and **probabilistic** gradient boosting algorithms for frequency and severity data in terms of :
   1. computational efficiency,
   2. predictive performance, and
   3. **model adequacy**.
2. To understand the relationship between these elements.

Outline

1. Algorithms
   - Point algorithms
   - Probabilistic algorithms
2. Applications in insurance
   - Datasets and metrics
   - Computational efficiency
   - Predictive performance
   - Model adequacy

Algorithms

Algorithm : Gradient boosting machine (GBM)

Input : n observations of target variable y_i and features $\mathbf{x}_i$

Algorithm : Gradient boosting machine (GBM)

Input : n observations of target variable y_i and features $\mathbf{x}_i$

1. **Initialization :** $f^0_{GBM}(\mathbf{x}_i) \leftarrow \text{argmin}_b\{\sum_{i=1}^n \mathcal{L}(y_i, b)\}$

Algorithm : Gradient boosting machine (GBM)

Input : n observations of target variable y_i and features $\mathbf{x}_i$

1. **Initialization :** $f^0_{GBM}(\mathbf{x}_i) \leftarrow \text{argmin}_b\{\sum_{i=1}^n \mathcal{L}(y_i, b)\}$
2. For $m = 1, \ldots, M$,
   - **Sampling :** $\mathcal{D}' \leftarrow$ Sample of size $\delta \times n$ observations without replacement

Algorithm : Gradient boosting machine (GBM)

Input : n observations of target variable y_i and features $\mathbf{x}_i$

1. **Initialization :** $f^0_{GBM}(\mathbf{x}_i) \leftarrow \text{argmin}_b\{\sum_{i=1}^{n} \mathcal{L}(y_i, b)\}$
2. For $m = 1, \dots, M$,
   - **Sampling :** $\mathcal{D}' \leftarrow$ Sample of size $\delta \times n$ observations without replacement
   - **Evaluation :** for $i \in \mathcal{D}'$, set

$$g_{m,i} \leftarrow -\nabla_a \mathcal{L}(y_i, a)\Big|_{a=f^{m-1}_{GBM}(\mathbf{x}_i)}$$

Algorithm : Gradient boosting machine (GBM)

Input : n observations of target variable y_i and features $\mathbf{x}_i$

1. **Initialization :** $f^0_{GBM}(\mathbf{x}_i) \leftarrow \text{argmin}_b\{\sum_{i=1}^n \mathcal{L}(y_i, b)\}$
2. For $m = 1, \ldots, M$,
   - **Sampling :** $\mathcal{D}' \leftarrow$ Sample of size $\delta \times n$ observations without replacement
   - **Evaluation :** for $i \in \mathcal{D}'$, set

$$g_{m,i} \leftarrow -\nabla_a \mathcal{L}(y_i, a)\Big|_{a=f^{m-1}_{GBM}(\mathbf{x}_i)}$$

   - **Learning :** Fit tree of depth d to $(\mathbf{x}_i, g_{m,i})$ to get regions $R_{1,m}, \ldots, R_{J_m,m}$

Algorithm : Gradient boosting machine (GBM)

Input : n observations of target variable y_i and features $\mathbf{x}_i$

1. **Initialization :** $f^0_{GBM}(\mathbf{x}_i) \leftarrow \text{argmin}_b\{\sum_{i=1}^{n} \mathcal{L}(y_i, b)\}$
2. For $m = 1, \ldots, M$,
   - **Sampling :** $\mathcal{D}' \leftarrow$ Sample of size $\delta \times n$ observations without replacement
   - **Evaluation :** for $i \in \mathcal{D}'$, set

$$g_{m,i} \leftarrow -\nabla_a \mathcal{L}(y_i, a)\Big|_{a=f^{m-1}_{GBM}(\mathbf{x}_i)}$$

   - **Learning :** Fit tree of depth d to $(\mathbf{x}_i, g_{m,i})$ to get regions $R_{1,m}, \ldots, R_{J_m,m}$
   - **Optimization :** $\hat{b}_{j,m} \leftarrow \text{argmin}_b \sum_{i:\mathbf{x}_i \in R_{j,m}} \mathcal{L}\{y_i, f^{m-1}_{GBM}(\mathbf{x}_i) + b\}$, for $j \in \{1, \ldots, J_m\}$

Algorithm : Gradient boosting machine (GBM)

Input : n observations of target variable y_i and features $\mathbf{x}_i$

1. **Initialization :** $f^0_{GBM}(\mathbf{x}_i) \leftarrow \text{argmin}_b\{\sum_{i=1}^n \mathcal{L}(y_i, b)\}$
2. For $m = 1, \dots, M$,
   - **Sampling :** $\mathcal{D}' \leftarrow$ Sample of size $\delta \times n$ observations without replacement
   - **Evaluation :** for $i \in \mathcal{D}'$, set

$$g_{m,i} \leftarrow -\nabla_a \mathcal{L}(y_i, a)\Big|_{a=f^{m-1}_{GBM}(\mathbf{x}_i)}$$

   - **Learning :** Fit tree of depth d to $(\mathbf{x}_i, g_{m,i})$ to get regions $R_{1,m}, \dots, R_{J_m,m}$
   - **Optimization :** $\hat{b}_{j,m} \leftarrow \text{argmin}_b \sum_{i:\mathbf{x}_i \in R_{j,m}} \mathcal{L}\{y_i, f^{m-1}_{GBM}(\mathbf{x}_i) + b\}$, for $j \in \{1, \dots, J_m\}$
   - **Update :** $f^m_{GBM}(\mathbf{x}_i) \leftarrow f^{m-1}_{GBM}(\mathbf{x}_i) + \lambda \sum_{j=1}^{J_m} \hat{b}_{j,m} \mathbf{1}(\mathbf{x}_i \in R_{j,m})$

Algorithm : Gradient boosting machine (GBM)

Input : n observations of target variable y_i and features $\mathbf{x}_i$

1. **Initialization :** $f^0_{GBM}(\mathbf{x}_i) \leftarrow \text{argmin}_b\{\sum_{i=1}^n \mathcal{L}(y_i, b)\}$
2. For $m = 1, \dots, M$,
   - **Sampling :** $\mathcal{D}' \leftarrow$ Sample of size $\delta \times n$ observations without replacement
   - **Evaluation :** for $i \in \mathcal{D}'$, set

$$g_{m,i} \leftarrow -\nabla_a \mathcal{L}(y_i, a)\Big|_{a=f^{m-1}_{GBM}(\mathbf{x}_i)}$$

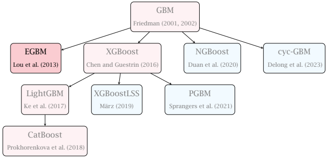
   - **Learning :** Fit tree of depth d to $(\mathbf{x}_i, g_{m,i})$ to get regions $R_{1,m}, \dots, R_{J_m,m}$
   - **Optimization :** $\hat{b}_{j,m} \leftarrow \text{argmin}_b \sum_{i:\mathbf{x}_i \in R_{j,m}} \mathcal{L}\{y_i, f^{m-1}_{GBM}(\mathbf{x}_i) + b\}$, for $j \in \{1, \dots, J_m\}$
   - **Update :** $f^m_{GBM}(\mathbf{x}_i) \leftarrow f^{m-1}_{GBM}(\mathbf{x}_i) + \lambda \sum_{j=1}^{J_m} \hat{b}_{j,m} \mathbf{1}(\mathbf{x}_i \in R_{j,m})$

Output : $\hat{y}_i = f^M_{GBM}(\mathbf{x}_i)$

Explainable GBM (EGBM, also known as GA^{2}M)

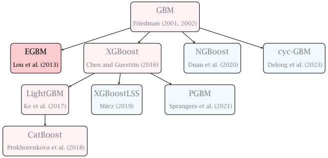
EGBM (Lou et al., 2012, 2013) is a GAM with selected two-way interactions

$$f_{EGBM}(\mathbf{x}_i) = \sum_{j=1}^{d} f_j(x_{i,j}) + \sum_{(k,\ell)\in\mathcal{S}} f_{k,\ell}(x_{i,k}, x_{i,\ell}).$$



Explainable GBM (EGBM, also known as GA^{2}M)

EGBM (Lou et al., 2012, 2013) is a GAM with selected two-way interactions



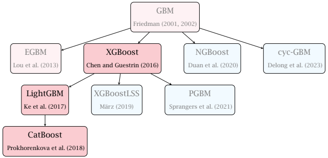
$$f_{EGBM}(\mathbf{x}_i) = \sum_{j=1}^{d} f_j(x_{i,j}) + \sum_{(k,\ell)\in\mathcal{S}} f_{k,\ell}(x_{i,k}, x_{i,\ell}).$$

- The univariate f_j's are built with gradient boosted stumps (Lou et al., 2012)
- Lou et al. (2013) propose the **FAST algorithm** to select the pairs in $\mathcal{S}$.
- Doumont (2024) finds that its predictive performance is comparable to that of **GBM** on frequency data.

XGBoost, LightGBM and CatBoost

In **XGBoost**, Chen and Guestrin (2016) improve the **computational efficiency** with :

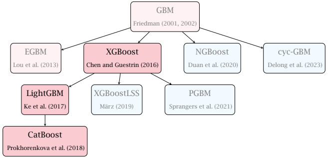
- a second-order Taylor approximation of the loss,
- hyperparameters to prevent overfitting,
- a simplified split-finding algorithm.



XGBoost, LightGBM and CatBoost

In **XGBoost**, Chen and Guestrin (2016) improve the **computational efficiency** with :

- a second-order Taylor approximation of the loss,
- hyperparameters to prevent overfitting,
- a simplified split-finding algorithm.



LightGBM (Ke et al., 2017) and **CatBoost** (Prokhorenkova et al., 2018) are refinements of **XGBoost** on :

- handling of categorical features,
- sampling step,
- tree growth strategies.

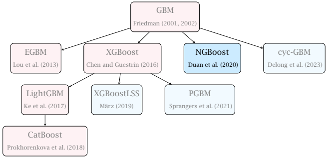
▶▶A comparative analysis can be found in So (2024).

NGBoost (Duan et al., 2020)

The natural gradient is $\tilde{\nabla}\mathcal{L}(y,\mathbf{p}) \propto I_{\mathcal{L}}^{-1}(\mathbf{p})\nabla_{\mathbf{p}}\mathcal{L}(y,\mathbf{p})$, where $\mathbf{p} = (p_1, p_2, \dots, p_k)$.

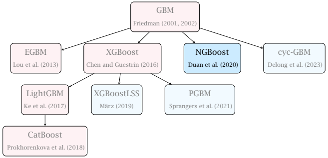
1. **Initialization** at MLE :

$$\mathbf{f}^0_{NGB}(\mathbf{x}_i) \leftarrow \operatorname{argmin}_{\mathbf{p}} \left\{ \sum_{i=1}^{n} \mathcal{L}(y_i, \mathbf{p}) \right\}$$



NGBoost (Duan et al., 2020)

The natural gradient is $\tilde{\nabla}\mathcal{L}(y, \mathbf{p}) \propto I_{\mathcal{L}}^{-1}(\mathbf{p})\nabla_{\mathbf{p}}\mathcal{L}(y, \mathbf{p})$, where $\mathbf{p} = (p_1, p_2, \ldots, p_k)$.



1. **Initialization** at MLE :

$$\mathbf{f}^0_{NGB}(\mathbf{x}_i) \leftarrow \text{argmin}_{\mathbf{p}} \left\{ \sum_{i=1}^{n} \mathcal{L}(y_i, \mathbf{p}) \right\}$$

2. For $m = 1, \ldots, M$,
   - **Evaluation :** $\mathbf{g}_{m,i} \leftarrow \tilde{\nabla}\mathcal{L}(y, \mathbf{p})$ evaluated at $\mathbf{p} = \mathbf{f}^{m-1}_{NGB}(\mathbf{x}_i)$, for $i \in \mathcal{D}$

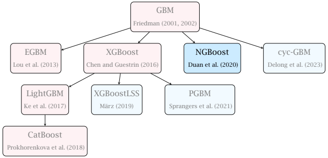
NGBoost (Duan et al., 2020)

The natural gradient is $\tilde{\nabla}\mathcal{L}(y, \mathbf{p}) \propto I_{\mathcal{L}}^{-1}(\mathbf{p})\nabla_{\mathbf{p}}\mathcal{L}(y, \mathbf{p})$, where $\mathbf{p} = (p_1, p_2, \ldots, p_k)$.

1. **Initialization** at MLE :

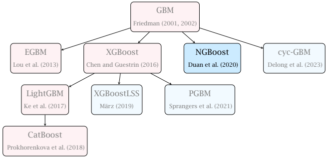
$$\mathbf{f}^0_{NGB}(\mathbf{x}_i) \leftarrow \text{argmin}_{\mathbf{p}} \left\{ \sum_{i=1}^{n} \mathcal{L}(y_i, \mathbf{p}) \right\}$$

2. For $m = 1, \ldots, M$,
   - **Evaluation :** $\mathbf{g}_{m,i} \leftarrow \tilde{\nabla}\mathcal{L}(y, \mathbf{p})$ evaluated at $\mathbf{p} = \mathbf{f}^{m-1}_{NGB}(\mathbf{x}_i)$, for $i \in \mathcal{D}$
   - **Learning :** Fit k trees of depth d, one for each element of vector $\mathbf{g}_{m,i}$, resulting in the vector of prediction functions $\mathbf{t}^m(\mathbf{x}_i)$



NGBoost (Duan et al., 2020)

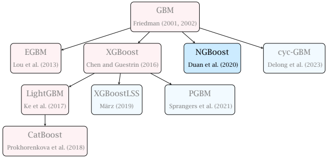
The natural gradient is $\tilde{\nabla}\mathcal{L}(y,\mathbf{p}) \propto I_{\mathcal{L}}^{-1}(\mathbf{p})\nabla_{\mathbf{p}}\mathcal{L}(y,\mathbf{p})$, where $\mathbf{p} = (p_1, p_2, \dots, p_k)$.



1. **Initialization** at MLE :
   $$\mathbf{f}^0_{NGB}(\mathbf{x}_i) \leftarrow \text{argmin}_{\mathbf{p}} \left\{ \sum_{i=1}^{n} \mathcal{L}(y_i, \mathbf{p}) \right\}$$
2. For $m = 1, \dots, M$,
   - **Evaluation :** $\mathbf{g}_{m,i} \leftarrow \tilde{\nabla}\mathcal{L}(y,\mathbf{p})$ evaluated at $\mathbf{p} = \mathbf{f}^{m-1}_{NGB}(\mathbf{x}_i)$, for $i \in \mathcal{D}$
   - **Learning :** Fit k trees of depth d, one for each element of vector $\mathbf{g}_{m,i}$, resulting in the vector of prediction functions $\mathbf{t}^m(\mathbf{x}_i)$
   - **Optimization :** $\hat{\rho}^m \leftarrow \text{argmin}_{\rho} \left[\sum_{i=1}^{n} \mathcal{L}\{y_i, \mathbf{f}^{m-1}_{NGB}(\mathbf{x}_i) + \rho \mathbf{t}^m(\mathbf{x}_i)\} \right]$

NGBoost (Duan et al., 2020)

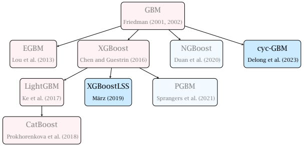
The natural gradient is $\tilde{\nabla}\mathcal{L}(y,\mathbf{p}) \propto I_{\mathcal{L}}^{-1}(\mathbf{p})\nabla_{\mathbf{p}}\mathcal{L}(y,\mathbf{p})$, where $\mathbf{p} = (p_1, p_2, \dots, p_k)$.



1. **Initialization** at MLE :
   $$\mathbf{f}^0_{NGB}(\mathbf{x}_i) \leftarrow \text{argmin}_{\mathbf{p}} \left\{ \sum_{i=1}^{n} \mathcal{L}(y_i, \mathbf{p}) \right\}$$
2. For $m = 1, \dots, M$,
   - **Evaluation :** $\mathbf{g}_{m,i} \leftarrow \tilde{\nabla}\mathcal{L}(y,\mathbf{p})$ evaluated at $\mathbf{p} = \mathbf{f}^{m-1}_{NGB}(\mathbf{x}_i)$, for $i \in \mathcal{D}$
   - **Learning :** Fit k trees of depth d, one for each element of vector $\mathbf{g}_{m,i}$, resulting in the vector of prediction functions $\mathbf{t}^m(\mathbf{x}_i)$
   - **Optimization :** $\hat{\rho}^m \leftarrow \text{argmin}_{\rho} \left[\sum_{i=1}^{n} \mathcal{L}\{y_i, \mathbf{f}^{m-1}_{NGB}(\mathbf{x}_i) + \rho \mathbf{t}^m(\mathbf{x}_i)\} \right]$
   - **Update :** $\mathbf{f}^m_{NGB}(\mathbf{x}_i) \leftarrow \mathbf{f}^{m-1}_{NGB}(\mathbf{x}_i) + \lambda \hat{\rho}^m \mathbf{t}^m(\mathbf{x}_i)$
3. **Prediction :** $\mathbf{f}^M_{NGB}(\mathbf{x}_i)$

XGBoostLSS (März, 2019) and cyc-GBM (Delong et al., 2023)

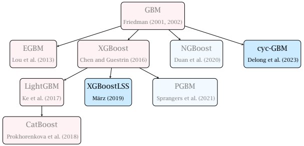
Both predict $\mathbf{p} = (p_1, p_2, \dots, p_k)$ with multiple boosting sequences.



XGBoostLSS (März, 2019) and cyc-GBM (Delong et al., 2023)

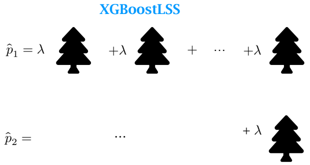
Both predict $\mathbf{p} = (p_1, p_2, \ldots, p_k)$ with multiple boosting sequences.

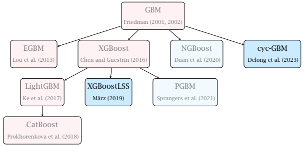
XGBoostLSS



XGBoostLSS (März, 2019) and cyc-GBM (Delong et al., 2023)

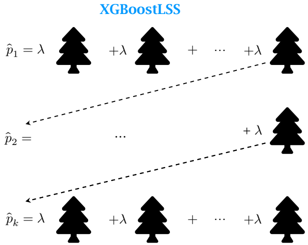
Both predict $\mathbf{p} = (p_1, p_2, \dots, p_k)$ with multiple boosting sequences.

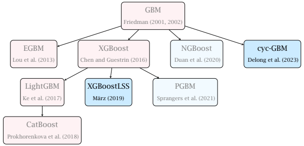




XGBoostLSS (März, 2019) and cyc-GBM (Delong et al., 2023)

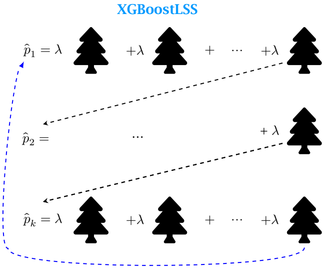
Both predict $\mathbf{p} = (p_1, p_2, \ldots, p_k)$ with multiple boosting sequences.

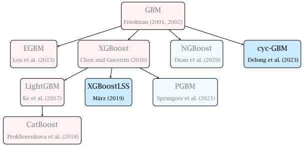




XGBoostLSS (März, 2019) and cyc-GBM (Delong et al., 2023)

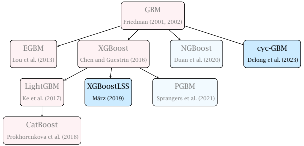
Both predict $\mathbf{p} = (p_1, p_2, \dots, p_k)$ with multiple boosting sequences.

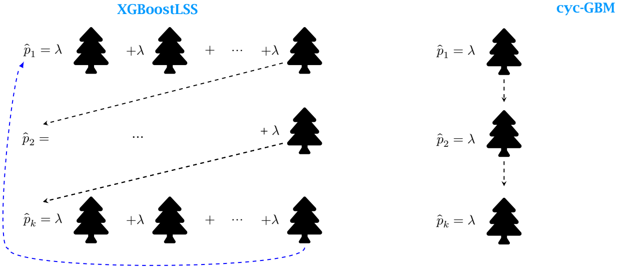




XGBoostLSS (März, 2019) and cyc-GBM (Delong et al., 2023)

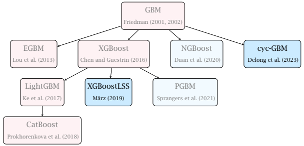
Both predict $\mathbf{p} = (p_1, p_2, \dots, p_k)$ with multiple boosting sequences.

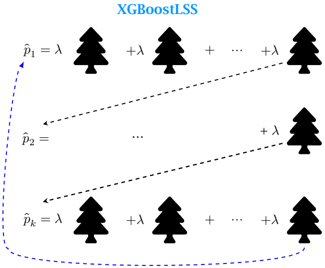


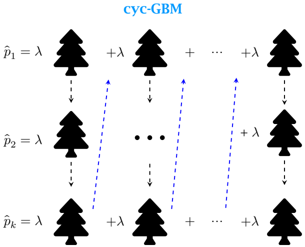


XGBoostLSS (März, 2019) and cyc-GBM (Delong et al., 2023)

Both predict $\mathbf{p} = (p_1, p_2, \ldots, p_k)$ with multiple boosting sequences.







Applications in insurance

Datasets for Poisson Frequency

We consider Poisson distribution on these **four datasets**.

Dataset	Sample size	# of features	# of cat. variables	Max # of levels for cat. variable
Belgian MTPL	163 212	12	6	583
pg15training [1]	50 021	13	8	471
freMPL	165 200	10	6	46
swauto	62 436	6	4	7

We split the dataset in 68% for training, 17% for validation and 15% for test.

Sources are Denuit and Lang (2004) for Belgian MTPL and CASDatasets (Dutang and Charpentier, 2020) for the others.

1. Subset for which `CalYear`=2009.

Datasets for Severity

We consider both Gamma and lognormal distributions on these **four datasets**.

Dataset	Sample size	# of features	# of cat. variables	Max # of levels for cat. variable
Belgian MTPL	17 910	12	6	583
pg15training	12 256	13	8	471
French MTPL	21 611	9	4	21
Emcien	9 134	17	5	9

We split the dataset in 68% for training, 17% for validation and 15% for test.

Sources are Denuit and Lang (2004) for Belgian MTPL, CASDatasets (Dutang and Charpentier, 2020) for French MTPL and Emcien Patterns (2017) for Emcien.

Performance comparison

We compare the algorithms based on :

- Computational efficiency : **run time** in seconds on laptop computer (IntelCore i7-1195G7 @ 2.90 GHz CPU)

Performance comparison

We compare the algorithms based on :

- Computational efficiency : **run time** in seconds on laptop computer (IntelCore i7-1195G7 @ 2.90 GHz CPU)
- Predictive performance : **RMSE** and **deviance** on test set

Performance comparison

We compare the algorithms based on :

- Computational efficiency : **run time** in seconds on laptop computer (IntelCore i7-1195G7 @ 2.90 GHz CPU)
- Predictive performance : **RMSE** and **deviance** on test set
- Model adequacy :
  - Level of **CI** on test set
  - Uniform **Q-Q plots**
  - **Proper scoring rules** : log-score and continuous ranked probability score (CRPS) as implemented by Jordan et al. (2017)
  - For Poisson, **randomized quantile residuals (RQR)** as in So (2024)

Performance comparison

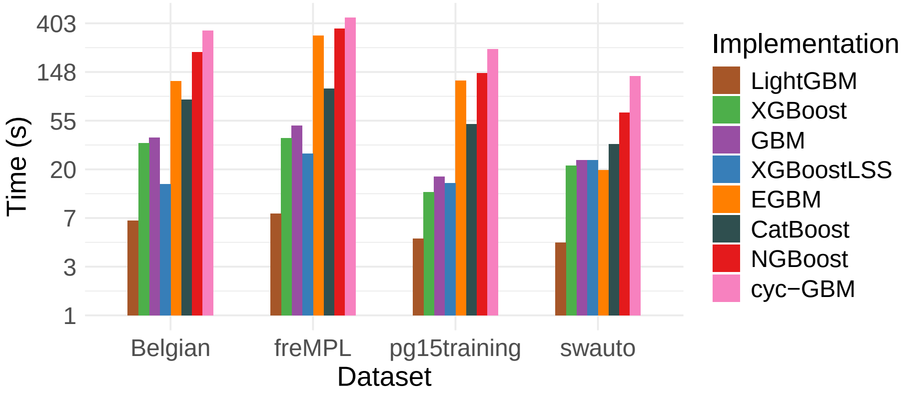
We compare the algorithms based on :

- Computational efficiency : **run time** in seconds on laptop computer (IntelCore i7-1195G7 @ 2.90 GHz CPU)
- Predictive performance : **RMSE** and **deviance** on test set
- Model adequacy :
  - Level of **CI** on test set
  - Uniform **Q-Q plots**
  - **Proper scoring rules** : log-score and continuous ranked probability score (CRPS) as implemented by Jordan et al. (2017)
  - For Poisson, **randomized quantile residuals (RQR)** as in So (2024)

Implementations :

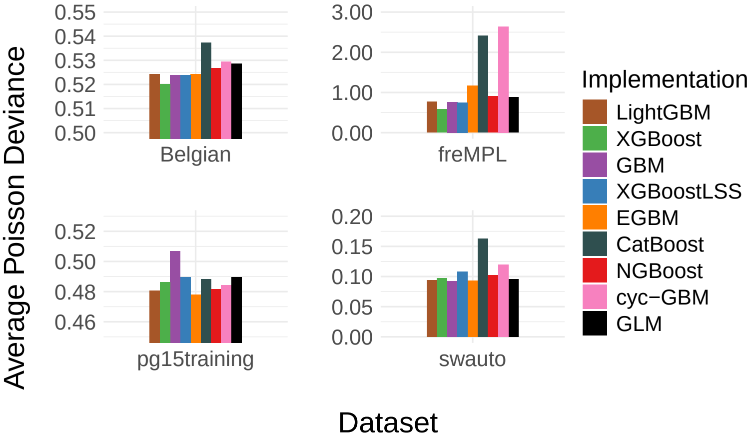
- XGBoostLSS, EGBM, and NGBoost are in **Python**.
- Other algorithms are in **R**.

Computational efficiency : Poisson distribution

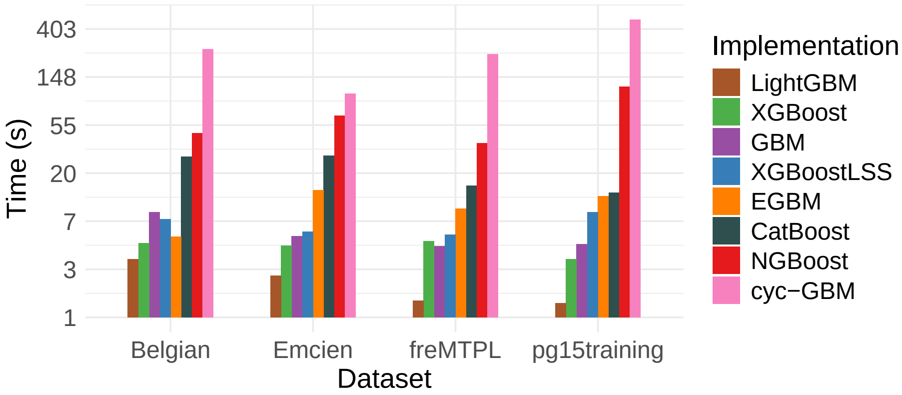


Y−axis is on a logarithmic scale.

Predictive performance on Poisson



Computational efficiency : severity with lognormal distribution



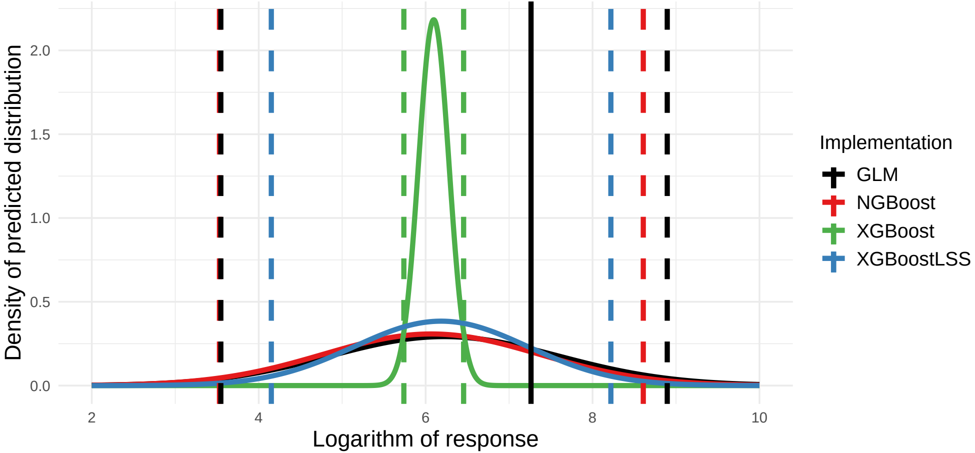
Y−axis is on a logarithmic scale.

Predictive performance on lognormal and Gamma

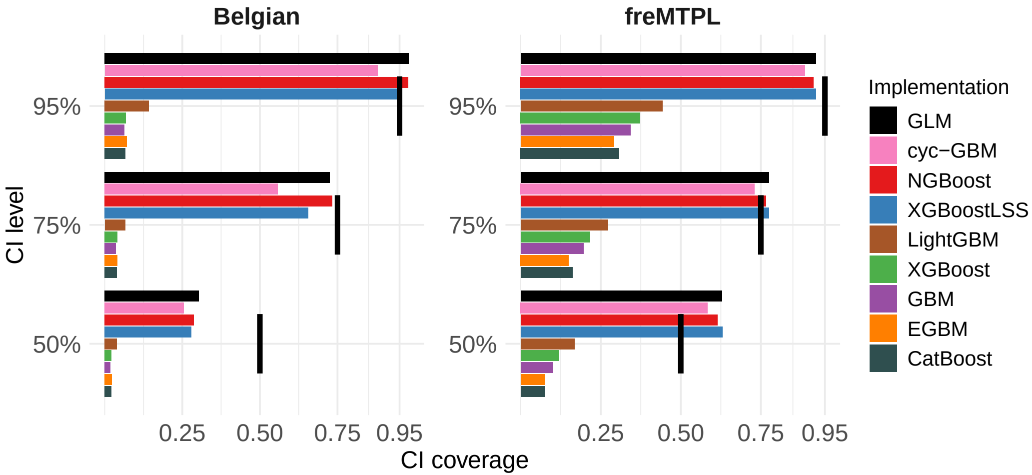
	Lognormal RMSE				Gamma Deviance			
	Belgian	freMTPL	Emcien	pg15training	Belgian	freMTPL	Emcien	pg15training
XGBoostLSS	1.401	1.220	0.462	1.231	1.622	**1.513**	0.151	1.061
NGBoost	1.389	1.220	0.463	1.232	**1.618**	1.532	0.154	1.060
cyc-GBM	1.402	1.220	0.467	1.236	1.657	1.533	**0.150**	1.067
CatBoost	1.389	**1.219**	**0.445**	**1.228**				
XGBoost	**1.389**	1.220	0.463	1.232	1.620	1.524	0.151	1.058
GBM	1.389	1.220	0.463	1.239	1.624	1.525	0.151	1.099
LightGBM	1.398	1.224	0.464	1.232	1.656	1.572	0.155	**1.058**
EGBM	1.390	1.221	0.463	1.233	1.627	1.525	0.152	1.071

Using CI to assess model adequacy - Explained

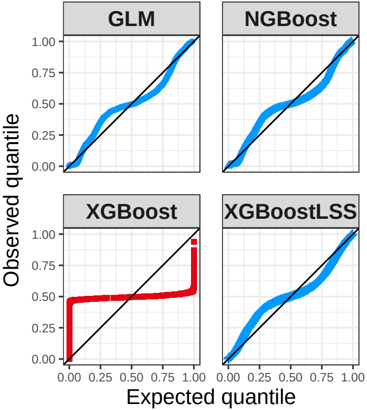
For one observation (solid vertical line) of the test set of BelgianMTPL :



Coverage of confidence intervals for lognormal

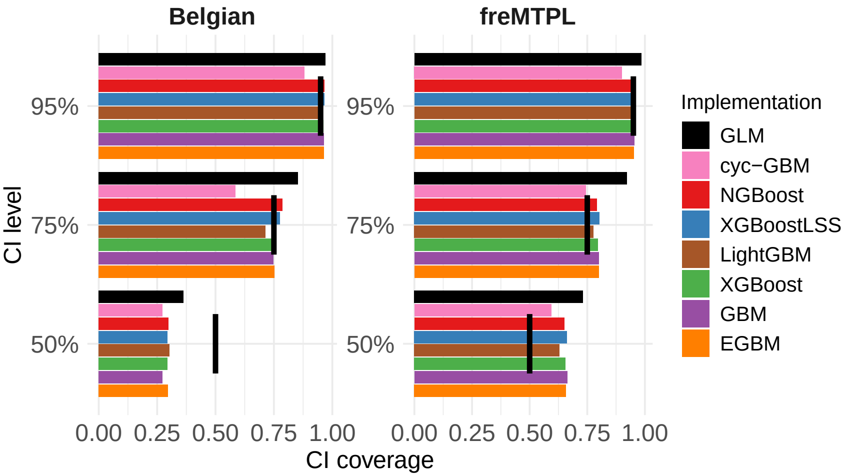


Q-Q plots for Lognormal - Belgian MTPL dataset

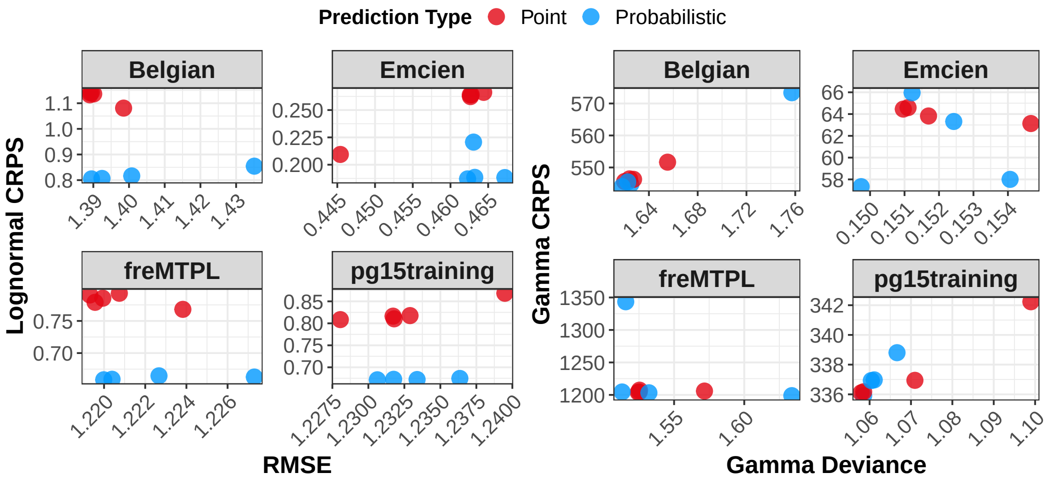


Implementation	RMSE
GLM	1.3924
NGBoost	1.3895
XGBoost	1.3890
XGBoostLSS	1.4008

Coverage of confidence intervals for gamma

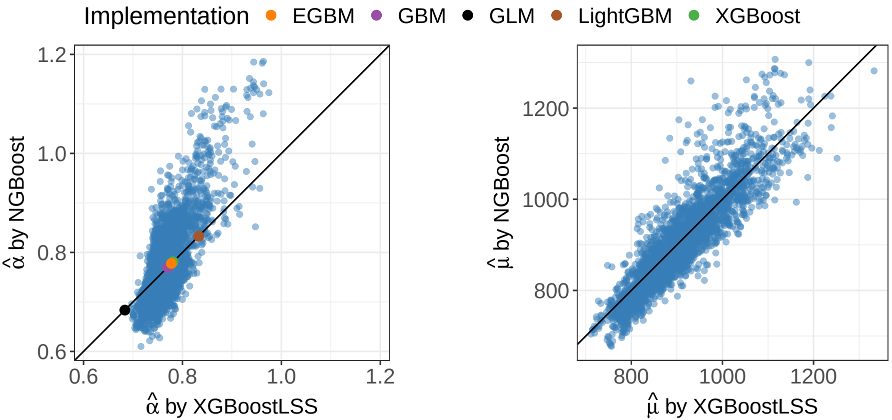


Predictive performance vs Model adequacy



Varying shape parameter α for the Gamma distribution

We have Y, a Gamma r.v. with $E[Y] = \mu$ and $Var[Y] = \mu^2/\alpha$.



Finally, which algorithm should we use ?

No free lunch !

Finally, which algorithm should we use?

No free lunch!

- Need for a fully interpretable model
  - **EGBM** or tree depth $d = 1$ in your favorite algorithm
- Large dataset, computational time is an issue, focus on point prediction
  - **LightGBM**
- Need for a computationally efficient and precise probabilistic model
  - **XGBoostLSS**

Finally, which algorithm should we use ?

No free lunch !

- Need for a fully interpretable model
  - **EGBM** or tree depth $d = 1$ in your favorite algorithm
- Large dataset, computational time is an issue, focus on point prediction
  - **LightGBM**
- Need for a computationally efficient and precise probabilistic model
  - **XGBoostLSS**

It is possible to improve model adequacy without hurting predictive performance.

Thank you!

References i

Chen, T. and Guestrin, C. (2016). XGBoost : A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 785–794.

Clemente, C., Guerreiro, G. R., and Bravo, J. M. (2023). Modelling motor insurance claim frequency and severity using gradient boosting. *Risks*, 11(9).

Crevecoeur, J., Robben, J., and Antonio, K. (2022). A hierarchical reserving model for reported non-life insurance claims. *Insurance : Mathematics and Economics*, 104 :158–184.

Delong, L., Lindholm, M., and Zakrisson, H. (2023). On cyclic gradient boosting machines. *Available at SSRN 4352505*.

Denuit, M. and Lang, S. (2004). Non-life rate-making with bayesian gams. *Insurance : Mathematics and Economics*, 35(3) :627–647.

Doumont, M. (2024). Non-life insurance pricing under ethical constraints (interpretability, non-discrimination and fairness). Master's thesis, Université catholique de Louvain.

References ii

Duan, T., Anand, A., Ding, D. Y., Thai, K. K., Basu, S., Ng, A., and Schuler, A. (2020). NGBoost : Natural gradient boosting for probabilistic prediction. *International conference on machine learning*, pages 2690–2700.

Dutang, C. and Charpentier, A. (2020). *CASdatasets : Insurance datasets.* R package version 1.0-11.

Emcien Patterns (2017). Automobile insurance claims including location, policy type and claim amount. Page consultée le 21 novembre 2023.

Friedman, J. H. (2001). Greedy function approximation : a gradient boosting machine. *Annals of statistics*, 29 :1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4) :367–378.

Gao, Y., Huang, Y., and Meng, S. (2023). Evaluation and interpretation of driving risks : Automobile claim frequency modeling with telematics data. *Statistical Analysis and Data Mining : The ASA Data Science Journal*, 16(2) :97–119.

References iii

Hancock, J. and Khoshgoftaar, T. M. (2020). Performance of catboost and xgboost in medicare fraud detection. In *2020 19th IEEE international conference on machine learning and applications (ICMLA)*, pages 572–579. IEEE.

Hartman, B., Owen, R., and Gibbs, Z. (2020). Predicting high-cost health insurance members through boosted trees and oversampling : An application using the hcci database. *North American Actuarial Journal*, 25(1) :53–61.

Henckaerts, R., Côté, M.-P., Antonio, K., and Verbelen, R. (2021). Boosting insights in insurance tariff plans with tree-based machine learning methods. *North American Actuarial Journal*, 25(2) :255–285.

Jordan, A., Krüger, F., and Lerch, S. (2017). Evaluating probabilistic forecasts with scoringRules. *arXiv preprint arXiv :1709.04743*.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). LightGBM : A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30 :3147–3155.

Lou, Y., Caruana, R., and Gehrke, J. (2012). Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158.

References iv

Lou, Y., Caruana, R., Gehrke, J., and Hooker, G. (2013). Accurate intelligible models with pairwise interactions. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631.

März, A. (2019). XgboostLSS–an extension of XGBoost to probabilistic forecasting. *arXiv preprint arXiv :1907.03178*.

Power, J., Côté, M.-P., and Duchesne, T. (2024). A flexible hierarchical insurance claims model with gradient boosting and copulas. *North American Actuarial Journal*, pages 1–29.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018). CatBoost : unbiased boosting with categorical features. *Advances in neural information processing systems*, 31 :6639–6649.

So, B. (2024). Enhanced gradient boosting for zero-inflated insurance claims and comparative analysis of catboost, xgboost, and lightgbm. *Scandinavian Actuarial Journal*, pages 1–23.

References v

Sprangers, O., Schelter, S., and de Rijke, M. (2021). Probabilistic gradient boosting machines for large-scale probabilistic regression. *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 1510–1520.

Wirawan, D. B. and Gunardi (2022). Determining auto insurance pure premium based on mileage (pay-as-you-drive insurance) using tree-based machine learning. In *International Conference on Educational Technology and Administration*, pages 317–342. Springer.