Supplementary Material for
Machine Learning with High-Cardinality Categorical Features in Actuarial Applications

Benjamin Avanzi[a], Greg Taylor[b], Melantha Wang[b,*], Bernard Wong[b]

[a] *Centre for Actuarial Studies, Department of Economics, University of Melbourne VIC 3010, Australia*
[b] *School of Risk and Actuarial Studies, UNSW Australia Business School, UNSW Sydney NSW 2052, Australia*

A. GLMMNet Implementation

Below we outline a few practical considerations that arise from the implementation of GLMMNet.

A.1. Fixed Prior

The random effects layer of GLMMNet (Section 2.3.2) requires the specification of a prior distribution. We mentioned that we would use a Gaussian prior as per common practice with GLMMs, but we did not specify the exact parameters for it. With GLMMNet, it is possible to have a trainable prior whose parameters can be found by gradient descent with respect to the loss function in (2.12); this approach of estimating the prior from the data is known as *empirical Bayes* (Casella, 1985). We experimented with both fixed and trainable priors in our implementation. We found that having a trainable prior leads to worse performance in general. Blundell et al. (2015) also reached the same conclusion from their experiments with Bayesian neural networks and they speculated that the algorithm would be more tempted to update the prior parameters than the posterior when the prior were trainable. We choose to work with fixed priors.

As to what values to use for the fixed prior, we reference the Stan documentation by Gelman (2020), which recommends the use of weakly informative priors, i.e. priors that will give way to the likelihood in the presence of sufficient data but will dominate in the absence of data. The exact prior parameters to use will depend on the data and the task at hand.

A.2. Reparametrisation of variational parameters

We use the softplus bijector function, defined as $\sigma(\lambda) = \log(1 + \exp(\lambda)),\ \lambda \in \mathbb{R}$, to reparametrise the scale parameters $(\sigma_j,\ j = 1, \cdots, q)$ in the diagonal Gaussian distribution (i.e. the surrogate posterior). This ensures that the surrogate scale parameters always stay within their support (i.e. in the positive region).

Furthermore, we found that it is important to shrink the value of the σ_j's at the start of the learning process to help guide the algorithm in the right direction. In our implementation, this is achieved by adding a constant multiplier (e.g. 0.01) to the parametrisation of σ_j's. In the absence of such a constraint, the algorithm tends to return unreasonably large σ_j values, which significantly deteriorates the performance of GLMMNet. It appears that the unguided GLMMNet converges to strange local minima where the model attributes all the variation to the noise.

We note that, in theory, adding a constant multiplier as we did does not modify the solution space and thus should return the same results regardless. We were able to empirically verify that if we gave the model enough training time, it was able to figure out the right range of values for the parameters at the end. The tweak we made, however, helps the algorithm immediately converge to a better local minimum. One possible explanation is that by reducing the size of the gradients on the scale parameters, it helps the model focus on learning the more important location (mean parameters) of the posterior random effects.

*Corresponding author.
Email addresses: b.avanzi@unimelb.edu.au (Benjamin Avanzi), gregory.taylor@unsw.edu.au (Greg Taylor), wang.melantha@gmail.com (Melantha Wang), bernard.wong@unsw.edu.au (Bernard Wong)

A.3. Initialisation of trainable parameters

The network weights and biases in GLMMNet are initialised by Tensorflow's default Glorot uniform initialiser (Glorot and Bengio, 2010). The dispersion parameter for the ED family, which is also learned as part of GLMMNet, is initialised with a fixed estimate roughly calibrated to the data at hand. It does not seem to have a major impact on GLMMNet's performance, but we still consider it good practice to run some small scale experiments with this choice of initial value.

B. Notes on Numerical Experiments

B.1. GLMM Encoding

GLMM encoding can be regarded as an easier and more flexible alternative to the machine learning mixed models—e.g. GPBoost (Sigrist, 2021, 2022) or GLMMNet, which often present a convoluted estimation procedure. While simple enough, Pargent et al. (2022) found that this encoding scheme performed very well across a range of prediction tasks and a variety of datasets.

In the numerical experiments, we implemented a cross-validated version of GLMM encoding, as presented in Algorithm 1. The encoded values $\mathbf{z}'$ then take the place of the original categorical feature to enter any subsequent ML model, e.g. a gradient boosting machine.

Algorithm 1: GLMM Encoding, adapted from Pargent et al. (2022)

Data: $\mathcal{D} = (y_i, \mathbf{z}_i)_{i=1}^n$
Result: $\mathbf{z}' = \psi(\mathbf{z}) \in \mathbb{R}^n$, a numeric representation of the categorical feature $\mathbf{z}$
Randomly partition $\mathcal{D}$ into K subsets of equal size $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_K\}$;
for $k = 1$ *to* K **do**
  Fit a random intercept model $\boldsymbol{\eta} = \beta_0\mathbf{1} + \mathbf{Zu}$ with $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \sigma_u^2\mathbf{I})$ on $\mathcal{D}_k^{\text{train}} = \mathcal{D} \backslash \mathcal{D}_k$;
  for $\{y_i, \mathbf{z}_i\} \in \mathcal{D}_k$ **do**
    if *category of the i-th observation* $j[i]$ *is in* $\mathcal{D}_k^{train}$ **then**
      Predict $z_i' = \hat{y}_i = g^{-1}(\beta_0 + u_{j[i]})$;
    else
      Predict $z_i' = \hat{y}_i = g^{-1}(\beta_0)$;
    end
  end
end

B.2. Neural Networks

Training a network involves making many specific choices. Below we briefly describe the specifications we use for the two network models in the main document, `NN_ee` and `GLMMNet`. To allow a fair comparison, where possible, we keep those choices consistent across both networks.

— *Architecture for the hidden layers.* We choose to use three hidden layers and $[64, 32, 16]$ hidden units (neurons) in each layer. This choice was made following the recommendation of Ferrario et al. (2020), where the authors suggested that the first hidden layer should be large enough to allow new features to be constructed from the raw input covariates and that successive layers should compress information. Some quick experimentation confirms that this choice of the hidden units works relatively well.

— *Activation functions.* We use the ReLU function defined by $f(x) = \max(x, 0)$ for hidden layer activations and the inverse of the link function for final layer activation (see Section 2.3.3).

— *Optimiser.* For both networks we use Tensorflow's default Adam optimiser with learning rate 0.001 (Kingma and Ba, 2014), which is a state-of-the-art optimisation algorithm with demonstrated superior performance for a range of predictive tasks. This has also been used in many actuarial applications, e.g. Richman and Wüthrich (2021).

— *Early stopping.* In fitting the networks, we further split the training data into an inner-training set and a validation set. We decide the number of epochs to train the networks based on when the validation performance (as captured by the validation loss) stops improving for a fixed number of epochs.
— *Loss function.* Section 2.4 discusses in detail the loss function we use to optimise `GLMMNet`. For `NN_ee`, we follow standard practice and use the squared error loss function to optimise the network. While it is possible to design a likelihood-based loss function for `NN_ee`, this involves further complications (such as reparametrisation of the dispersion parameters) above the scope of this project.
— *Embedding dimension* (for `NN_ee` only). The dimension d of the embedding space is a network hyperparameter that should be chosen through experimentation. Lakshmanan et al. (2020) suggest to use "the fourth root of the total number of unique categorical elements" (p.48), which we used as a reference.

B.3. Learning Pipeline

As per common practice, the data are split into training and testing sets. The training set is used for learning (i.e. fitting of the models) and the testing set is used for assessing and comparing the performance of the optimised candidate models.

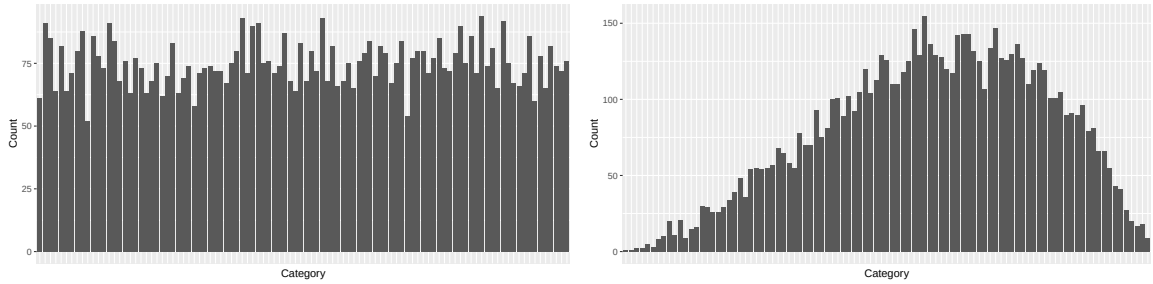
In this work, we only perform a minimal search for hyperparameters, as hyperparameter tuning is not the main purpose of this study. We use the validation set approach. For models that involve hyperparameters, we split the training set into an inner-training set and a validation set. Models are fitted on the inner-training set, and hyperparameters are selected based on their performance on the validation set. This approach is less systematic than cross validation, but from experience, it usually yields reasonably similar results at a much more sustainable computational cost.

B.4. Simulation Environments

We set up the desired simulation environments in Section 3 by adjusting the following parameters:
— `n_categories`: number of categories; fixed at 100.
— `signal_to_noise`: a three-dimensional vector that captures the relative ratio of signal strength (as measured by μ_f, the mean of $f(\mathbf{x})$), random effects variance (σ_u^2), and variability of the response (σ_ϵ^2; noise, or equivalently the irreducible error, as this component captures the unexplained inherent randomness in the response). The vector will be normalised to sum to 1, e.g. when `signal_to_noise` $= [3, 1, 1]^\top$, the vector is first normalised to $[0.6, 0.2, 0.2]^\top$. Data points are generated as follows:
  1. Generate a sample $\mathbf{u} \in \mathbb{R}^q$ from $u_j \overset{iid}{\sim} \mathcal{N}(0, \sigma_u^2 = 0.2^2)$.
  2. Rescale $f(\mathbf{X})$, which has been pre-calculated from a deterministic formula (e.g. the Friedman function in (3.2)), such that $\overline{f(\mathbf{X})} = \mu_f = 0.6$ where $\overline{x}$ denotes the sample mean across all observations $i = 1, \cdots, n$.
  3. Set the conditional mean as $\boldsymbol{\mu} = \mathbb{E}(\mathbf{y}|\mathbf{u}) = g^{-1}(f(\mathbf{X}) + \mathbf{Z}^\top \mathbf{u})$ and $\phi = \sigma_\epsilon^2 = 0.2^2$, where ϕ is the dispersion parameter for the ED family, such that $\text{Var}(\mathbf{y}|\mathbf{u}) = \phi V(\boldsymbol{\mu})$ where $V(\cdot)$ is the variance function for the family.
  4. Generate samples $\mathbf{y}|\mathbf{u}$ from the conditional mean $\boldsymbol{\mu}$ and conditional variance $\text{Var}(\mathbf{y}|\mathbf{u})$.
— `y_dist`: distributional assumption for the response, e.g. Gaussian, gamma, or any other member of the ED family.
— `inverse_link`: inverse of the link function, i.e. $g^{-1}(\cdot)$.
— `cat_dist`: to use balanced or skewed distribution for the allocation of categories. A "balanced" distribution allocates approximately equal number of observations to each category; a "skewed" distribution generates categories from a (scaled) beta distribution; see Figure B.1.

B.5. Model Evaluation Criteria

For all experiments presented in this paper, we randomly split the data into a training and a test set. The training set is used to select hyperparameters (by further splitting into an inner-training and a validation set) and fit the pool of candidate models. The different models are then evaluated and compared based on their test performance. Specifically, to quantify the predictive performance, we consider the metrics listed below:



(a) Balanced **(b)** Skewed

Figure B.1: Number of observations by category for balanced *versus* skewed distribution of categories

— *Accuracy of point predictions.* We report the root mean squared error (RMSE) and the mean absolute error (MAE), which are respectively given by

$$\text{RMSE} = \sqrt{\frac{1}{n^*}\sum_{i=1}^{n^*}(y_i^* - \hat{y}_i^*)^2}, \quad \text{MAE} = \frac{1}{n^*}\sum_{i=1}^{n^*}|y_i^* - \hat{y}_i^*|,$$

where n^* is the number of test observations, $\mathbf{y}^*$ is the target output and $\hat{\mathbf{y}}^*$ is the (point) prediction of the mean response.

— *Average accuracy of point predictions per category.* In order to gain insights into the category-specific accuracy of point predictions, we consider the RMSE of average prediction for each category:

$$\text{RMSE\_avg} = \sqrt{\frac{1}{q}\sum_{j=1}^{q}(\overline{y}_j^* - \overline{\widehat{y}}_j^*)^2}, \quad \overline{y}_j^* = \frac{1}{n_j^*}\sum_{i:j[i]=j}^{n^*} y_i^*, \quad \overline{\widehat{y}}_j^* = \frac{1}{n_j^*}\sum_{i:j[i]=j}^{n^*} \widehat{y}_i^*,$$

where q is the number of categories, n_j^* is the number of test observations in the j-th category, $j[i]$ indicates the category to which the i-the observation belongs, $\overline{y}_j^*$ and $\overline{\widehat{y}}_j^*$ respectively denote the average of the response variable y and its prediction $\hat{y}$ for all observations in the j-th category. This metric serves to measure and compare how well a model is able to capture the between-category differences. This can be interpreted as, for example, the average accuracy of loss predictions on each sub-portfolio, which has practical significance.

— *Accuracy of probabilistic predictions.* The quantification of probabilistic accuracy is especially relevant to actuarial applications (Embrechts and Wüthrich, 2022). In this work, we consider two *proper scoring rules*, the continuous ranked probability score ("CRPS") and the negative log-likelihood ("NLL"), both of which are commonly used for assessing probabilistic forecasts (Gneiting and Raftery, 2007; Al-Mudafer et al., 2022; Delong et al., 2021). The CRPS is defined as

$$\text{CRPS} = \frac{1}{n^*}\sum_{i=1}^{n^*}\text{CRPS}(\hat{F}_i, y_i), \text{ where } \text{CRPS}(\hat{F}_i, y_i) = \int_{-\infty}^{\infty}(\hat{F}_i(z) - \mathbb{1}_{\{z \geq y_i\}})^2 \mathrm{d}z, \tag{B.1}$$

and where $\hat{F}_i(\cdot)$ represents the distribution function (df) of a probabilistic forecaster for the i-th response value and y_i represents its observed value. The CRPS is a quadratic measure of the difference between the forecast predictive df and the empirical df of the observation. Hence, the smaller it is the better. The integral in (B.1) has been evaluated analytically and implemented in R for many distributions; we refer to Jordan et al. (2019) for details.

Computation of the NLL is more straightforward:

$$\text{NLL} = \frac{1}{n^*} \sum_{i=1}^{n^*} \text{NLL}(\hat{F}_i, y_i), \text{ where } \text{NLL}(\hat{F}_i, y_i) = -\log \hat{f}_i(y_i). \tag{B.2}$$

and where $\hat{f}_i(\cdot)$ is the density of the probabilistic forecaster.
For models that do not produce a distributional forecast, we construct an artificial predictive distribution by using the point forecast as the mean and estimating a dispersion parameter for the assumed distribution (see Section 4.6 of Denuit et al., 2019).

B.6. Description of the SME Building Insurance Data

B.6.1. Variable List

Variable	Description
`anzsic4_desc`	ANZSIC occupation code description that describes the business's economic activity (at the Class level).
`year_incurred`	Calendar year of the accident associated with the claim; value between 2010 and 2015.
`peril`	Cause of the claim, e.g. impact by third party, malicious damage, or fire.
`years_insured`	Tenure of the policy.
`state_risk`	State of the insured's business premises.
`occupancy`	Type of occupancy of the insured's business premises, e.g. property owner or tenant.
`locality`	Location of the insured's business premises, e.g. industrial or retail.
`roof_type`	Primary roof material of the insured's business premises; re-processed into a binary indicator of whether the material is fire resistant or not.
`wall_type`	Primary wall material of the insured's business premises.
`floor_type`	Primary floor material of the insured's business premises.
`log_si`	Log of total sum insured for building and contents.
`firep_detect`	A binary indicator of whether the insured's business premises have any fire detection equipment, e.g. a fire alarm.
`firep_stop`	A binary indicator of whether the insured's business premises have any fire suppression equipment, e.g. a fire extinguisher.

Table B.1: (Engineered) input features used in modelling

C. Supplementary Results

C.1. Experiment Settings

Table C.1 gives a full list of the simulation environments considered in our experiment. Note that we excluded experiments 3 and 4 from the main document for the sake of brevity. The results from experiments 3–4, along with additional results for the four experiments presented in the main document, are discussed below in Sections C.2–C.3.

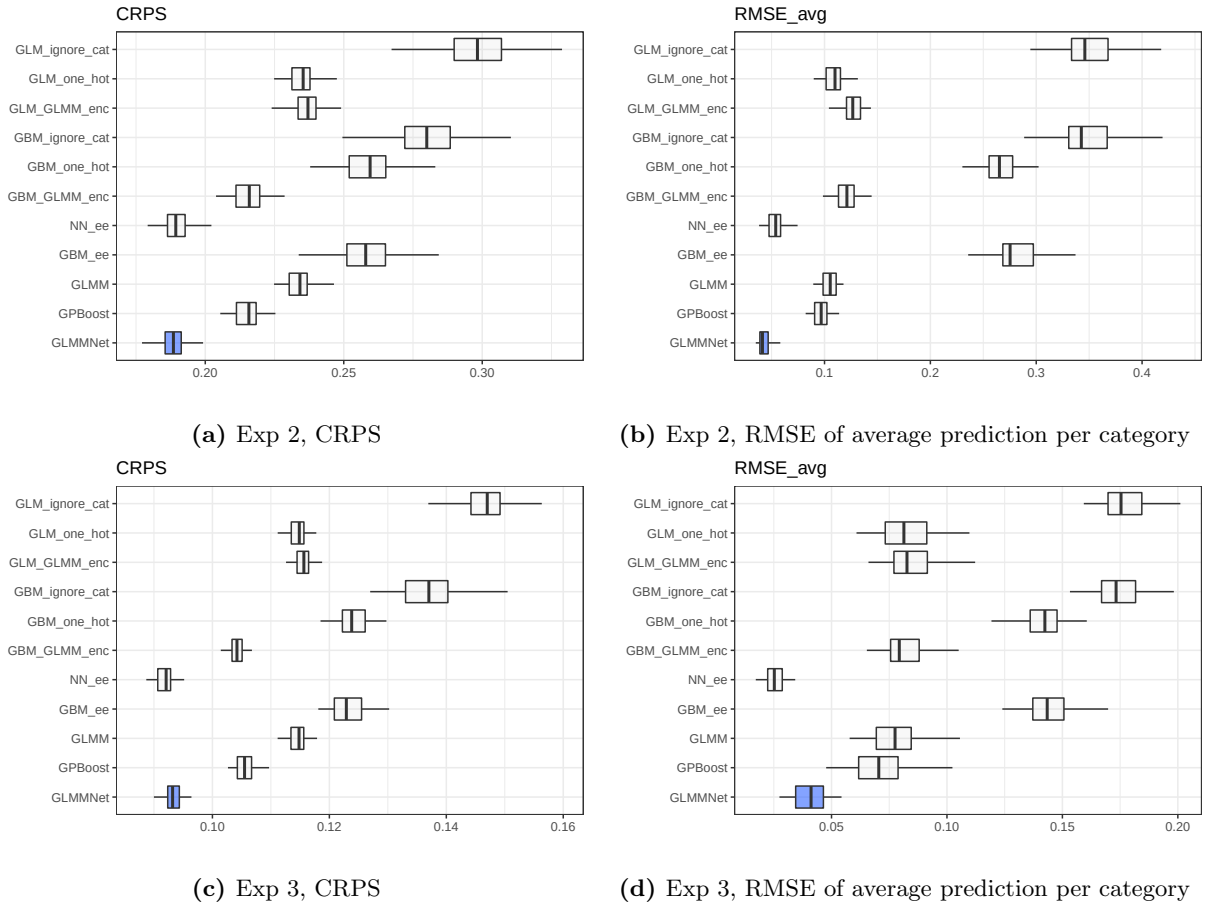
Exp ID	Signal-to-noise		Response distribution	Inverse link	Distribution of categories
1 (base)	$[4, 1, 1]$	(high)	Gaussian	Identity	Balanced
2	$[4, 1, 1]$	(high)	**Gamma**	**Exponential**	Balanced
3	$[4, 1, 1]$	(high)	Gaussian	Identity	**Skewed**
4	$[\mathbf{4}, \mathbf{1}, \mathbf{2}]$	(medium)	Gaussian	Identity	Balanced
5	$[\mathbf{8}, \mathbf{1}, \mathbf{4}]$	(low)	Gaussian	Identity	Balanced
6	$[\mathbf{8}, \mathbf{1}, \mathbf{4}]$	(low)	**Gamma**	**Exponential**	**Skewed**

Table C.1: Parameters used for the different simulation environments.
Bold face indicates changes from the base scenario (i.e. experiment 1).

— *Experiment 1* simulates the base scenario that adheres to the assumptions of a Gaussian GLMMNet.
— *Experiment 2* simulates a gamma-distributed response, which is often used to model claim severity in lines of business (LoB) such as auto insurance or general liability.
— *Experiment 3* simulates a skewed distribution of categories, which is a common characteristic of high-cardinality categorical features (e.g. car make, injury code).
— *Experiments 4–5* incrementally increase the level of noise in the data and simulate LoBs that are difficult to model by covariates, such as commercial building insurance, catastrophic events and cyber risks.
— *Experiment 6* represents the most challenging scenario, incorporating the complexities of all the other experiments.

C.2. Experiments 2–3: Increasing Complexity within a Low Noise Environment

Figure C.1 displays the out-of-sample performance metrics of the candidate models from experiments 2 and 3, which respectively consider a gamma-distributed response variable and a skewed distribution of categories (see Table 3.2). Note that we chose not to show in Figure C.1 the MAE or RMSE plots for brevity of presentation; the results from the two metrics are in perfect alignment with the CRPS results.



(a) Exp 2, CRPS

(b) Exp 2, RMSE of average prediction per category

(c) Exp 3, CRPS

(d) Exp 3, RMSE of average prediction per category

Figure C.1: Boxplots of out-of-sample performance metrics of the different models in experiment 2 (top; gamma-distributed response) and experiment 3 (bottom; skewed distribution of categories); GLMMNet highlighted in blue. Each experiment is repeated 50 times, with 5,000 training observations and 2,500 testing observations each.

C.3. Experiments 4–6: Dialling up the Noise

In the remaining three experiments, we test the stability of GLMMNet under increasing noise in the environment, where "noise" refers to the unmodellable inherent uncertainty in the response due to its stochastic nature. Experiments 5–6 share the same signal-to-noise ratio of $[8, 1, 4]$ which is derived from the estimated parameters in the real data case study in Section 4. Experiment 4 has a signal-to-noise ratio of $[4, 1, 2]$, such that the signal strength relative to the random noise is also at $2:1$ but the random effects are stronger (more variable) than in the case of experiments 5–6.

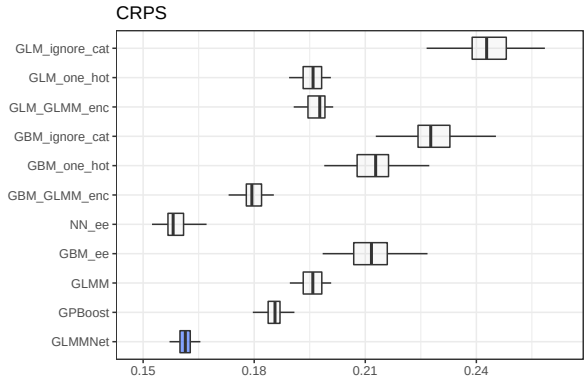
Figure C.2 plots the out-of-sample metrics for each of the three experiments. The results are discussed in Section 3.3. In general, as the noise level increases, the predictive advantage of the GLMMNet starts to fade away. Figure C.3 contrasts the random effects predictions under experiments 1 and 6: the GLMMNet is able to recover the random effects very well under low noise (experiments 1–3, plots for 2 and 3 are omitted), but the high level of noise in experiment 6 creates more difficulties for the model to accurately predict the random effects.

C.4. Application to Real Insurance Data

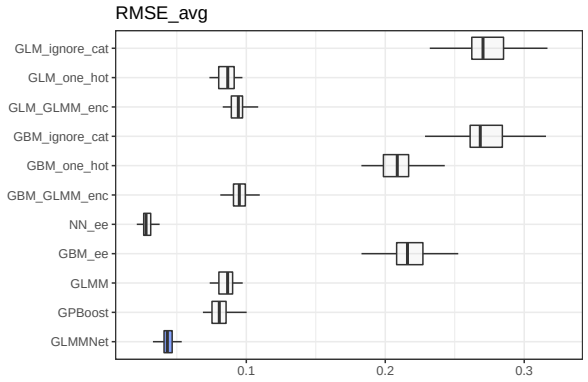
Tables C.2 and C.3 present the in-sample and out-of-sample metrics for the lognormal and loggamma models, respectively. The results for the top performing models are included in Table 4.2 of Section 4.

	Training			**Test (out-of-sample)**		
	MedAE	CRPS	NLL	MedAE	CRPS	NLL
GLM_ignore_cat	4349	0.7934	9.643	4323	0.8010	9.635
GLM_one_hot	4174	0.7791	9.623	4108	0.7931	9.623
GLM_GLMM_enc	4332	0.7919	9.642	4304	0.7982	9.632
GBM_ignore_cat	3893	0.7676	9.607	3952	0.7707	9.589
GBM_one_hot	3846	0.7653	9.604	3903	0.7682	9.586
GBM_GLMM_enc	3838	0.7644	9.603	3870	0.7666	9.584
NN_ee	3776	0.7571	9.595	3802	**0.7624**	**9.578**
GBM_ee	3825	0.7646	9.604	3828	0.7665	9.584
GLMM	3835	0.7650	9.604	3864	0.7666	9.584
GPBoost	3834	0.764	9.603	3978	0.7691	9.587
GLMMNet	3587	0.7535	9.589	3736	0.7681	9.587
GLMMNet_l2	3555	0.7617	9.600	**3545**	0.7626	9.579

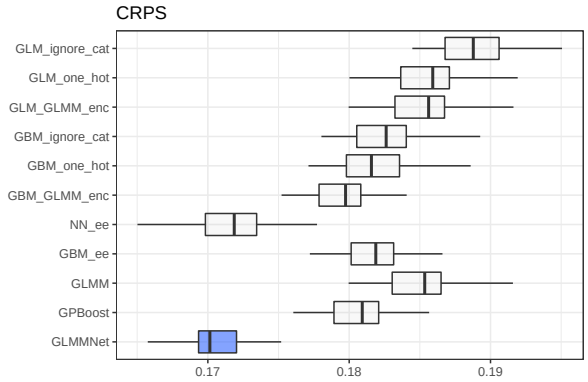
Table C.2: Comparison of **lognormal** model performance (median absolute error, CRPS, negative log-likelihood) on training and test sets. The best values for each out-of-sample metric are bolded.



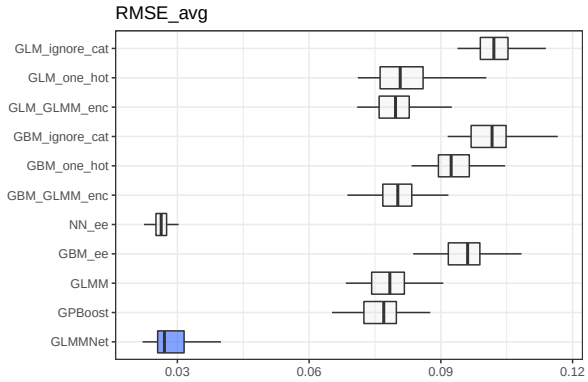
(a) Exp 4, CRPS



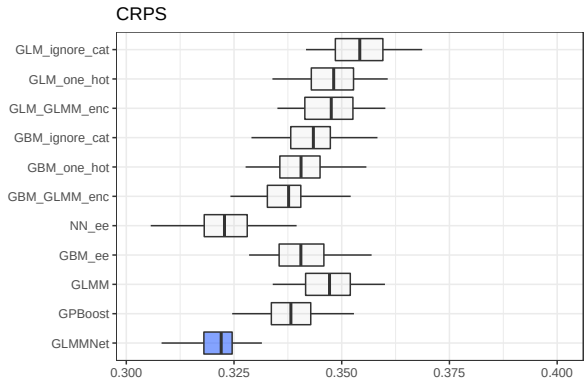
(b) Exp 4, RMSE of average prediction per category



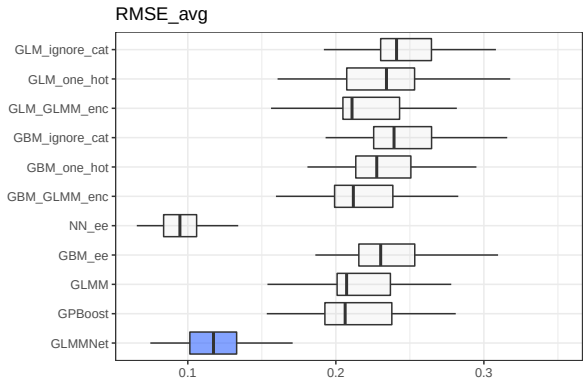
(c) Exp 5, CRPS



(d) Exp 5, RMSE of average prediction per category

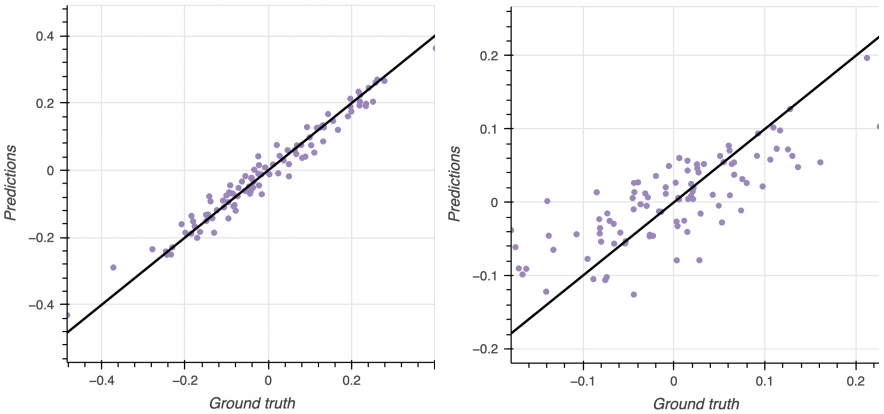


(e) Exp 6, CRPS



(f) Exp 6, RMSE of average prediction per category

Figure C.2: Boxplots of out-of-sample performance metrics of the different models in experiment 4 (top; medium noise Gaussian), experiment 5 (middle; high noise Gaussian) and experiment 6 (bottom; high noise, skewed categorical distribution, Gamma response); GLMMNet highlighted in blue. Each experiment is repeated 50 times, with 5,000 training observations and 2,500 testing observations each.



(a) Experiment 1 **(b)** Experiment 6

Figure C.3: Random effects predictions by GLMMNet (posterior mode) against the ground truth under (a) experiment 1, i.e. base scenario, (b) experiment 6, i.e. high complexity, high noise scenario

	Training			**Test (out-of-sample)**		
	MedAE	CRPS	NLL	MedAE	CRPS	NLL
GLM_ignore_cat	2036	0.8612	9.787	2000	0.8733	9.792
GLM_one_hot	1942	0.8334	9.730	1946	0.8557	9.751
GLM_GLMM_enc	2043	0.8606	9.787	2011	0.8722	9.792
GBM_ignore_cat	1591	0.7641	9.606	1568	0.7668	9.583
GBM_one_hot	1578	0.7619	9.603	1545	0.7643	9.580
GBM_GLMM_enc	1594	0.7610	9.602	**1536**	0.7626	9.578
NN_ee	1573	0.7517	9.591	1566	**0.7600**	9.575
GBM_ee	1589	0.7607	9.602	1549	0.7629	9.579
GLMM	1598	0.7618	9.602	1570	0.7629	9.577
GPBoost	1618	0.7631	9.604	1615	0.7666	9.583
GLMMNet	1549	0.7548	9.593	1537	0.7706	9.588
GLMMNet_l2	1583	0.7550	9.594	1551	**0.7600**	**9.574**

Table C.3: Comparison of **loggamma** model performance (median absolute error, CRPS, negative log-likelihood) on training and test sets. The best values for each out-of-sample metric are bolded.

Figure C.4 supplements the discussion in Section 4.3. It shows that, as expected, occupations with a larger number of observations generally have smaller posterior standard deviations for the random effects, indicating higher confidence in the estimates.

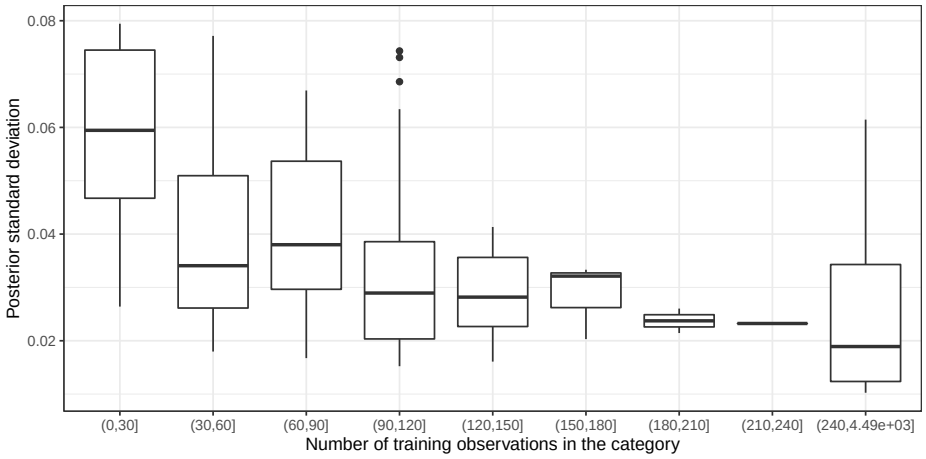


Figure C.4: Posterior standard deviations plotted against the number of training observations from the occupation class

References

Al-Mudafer, M.T., Avanzi, B., Taylor, G., Wong, B., 2022. Stochastic loss reserving with mixture density neural networks. Insurance: Mathematics and Economics 105, 144–174. URL: https://www.sciencedirect.com/science/article/pii/S0167668722000373, doi:10.1016/j.insmatheco.2022.03.010.

Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D., 2015. Weight uncertainty in neural networks, in: Proceedings of the 32nd International Conference on Machine Learning, PMLR. pp. 1613–1622. URL: https://proceedings.mlr.press/v37/blundell15.pdf.

Casella, G., 1985. An introduction to empirical bayes data analysis. The American Statistician 39, 83–87.

Delong, Ł., Lindholm, M., Wüthrich, M.V., 2021. Gamma mixture density networks and their application to modelling insurance claim amounts. Insurance: Mathematics and Economics , S0167668721001232URL: https://linkinghub.elsevier.com/retrieve/pii/S0167668721001232, doi:10.1016/j.insmatheco.2021.08.003.

Denuit, M., Hainaut, D., Trufin, J., 2019. Effective Statistical Learning Methods for Actuaries I: GLMs and Extensions. Springer Actuarial, Springer International Publishing, Cham. URL: http://link.springer.com/10.1007/978-3-030-25820-7, doi:10.1007/978-3-030-25820-7. DOI: 10.1007/978-3-030-25820-7.

Embrechts, P., Wüthrich, M., 2022. Recent challenges in actuarial science. Annual Review of Statistics and Its Application 9. URL: https://www.annualreviews.org/doi/10.1146/annurev-statistics-040120-030244, doi:10.1146/annurev-statistics-040120-030244.

Ferrario, A., Noll, A., Wüthrich, M., 2020. Insights from inside neural networks. SSRN. URL: https://papers.ssrn.com/abstract=3226852, doi:10.2139/ssrn.3226852. DOI: 10.2139/ssrn.3226852.

Gelman, A., 2020. Prior choice recommendations. URL: https://github.com/stan-dev/stan.

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings. pp. 249–256. URL: https://proceedings.mlr.press/v9/glorot10a.html. iSSN: 1938-7228.

Gneiting, T., Raftery, A.E., 2007. Strictly proper scoring rules, prediction, and estimation. Journal of the American Statistical Association 102, 359–378. URL: https://doi.org/10.1198/016214506000001437, doi:10.1198/016214506000001437. publisher: Taylor & Francis.

Jordan, A., Krüger, F., Lerch, S., 2019. Evaluating probabilistic forecasts with **scoringrules**. Journal of Statistical Software 90. URL: http://www.jstatsoft.org/v90/i12/, doi:10.18637/jss.v090.i12.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv:1412.6980 [Cs] URL: http://arxiv.org/abs/1412.6980.

Lakshmanan, V., Robinson, S., Munn, M., 2020. Machine learning design patterns: solutions to common challenges in data preparation, model building, and MLOps. First edition ed., O'Reilly Media, Sebastopol, CA. OCLC: on1178649818.

Pargent, F., Pfisterer, F., Thomas, J., Bischl, B., 2022. Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. Computational Statistics URL: https://doi.org/10.1007/s00180-022-01207-6, doi:10.1007/s00180-022-01207-6.

Richman, R., Wüthrich, M.V., 2021. A neural network extension of the Lee–Carter model to multiple populations. Annals of Actuarial Science 15, 346–366. URL: https://www.cambridge.org/core/product/identifier/S1748499519000071/type/journal_article, doi:10.1017/S1748499519000071.

Sigrist, F., 2021. Gaussian process boosting. arXiv:2004.02653 [cs.LG] URL: http://arxiv.org/abs/2004.02653. arXiv:2004.02653.

Sigrist, F., 2022. Latent gaussian model boosting. IEEE Transactions on Pattern Analysis and Machine Intelligence In press, 1–12. URL: https://doi.org/10.1109/TPAMI.2022.3168152, doi:10.1109/TPAMI.2022.3168152. arXiv:2105.08966.