Investigating the impact of missing value handling on Boosted trees and Deep learning for Tabular data: A Claim Reserving case study

Anonymous authors
Paper under double-blind review

Abstract

While deep learning (DL) performance is exceptional for many applications, there is no consensus on whether DL or gradient boosted decision trees (GBDTs) are superior for tabular data. We compare TabNet (a DL model for tabular data), ~~a~~two simple neural networks inspired by ResNet (a DL model) and Catboost (a GBDT model) on a large UK insurer dataset for the task of claim reserving. This dataset contains a high amount of informative missing values. We use this application to shed light on the impact of missing value handling on accuracy. Under certain missing value schemes a carefully optimised simple neural network model performed comparably to Catboost with default settings. However, using less-than-minimum imputation, Catboost with default settings substantially outperformed carefully optimised DL models - achieving the best overall accuracy. We conclude that handling missing values is an important, yet often overlooked, step when comparing DL to GBDT algorithms for tabular data.

1 Introduction

Many machine learning problems involve regressing or classifying with *tabular* or *structured* data. Since their introduction, GBDTs have performed well on such tabular data (Friedman, 2001). Meanwhile, DL has become the state of the art in many problems that involve *unstructured* data, e.g. images (He et al., 2016; Simonyan & Zisserman, 2015; Tao et al., 2020), audio (Ao et al., 2021), text (Baktha & Tripathy, 2017; Ziegler et al., 2019; Touvron et al., 2023), and their combinations (Radford et al., 2021; Rombach et al., 2022; Ramesh et al., 2021).

Naturally, with the rise of research in DL, architectures have been proposed that claim to outperform GBDTs on tabular data (Somepalli et al., 2021; Shavitt & Segal, 2018; Huang et al., 2020; Kadra et al., 2021). However, there is a lack of consensus on whether these architectures really are more accurate. Large studies (Grinsztajn et al., 2022; McElfresh et al., 2024; Borisov et al., 2022) have compared DL to GBDTs across many datasets, many tasks and different computational budgets and suggest that GBDTs are on average the more accurate model for tabular data. The proposed reasons for the performance edge of GBDTs in these large studies remains an area of active research. Theories include the ability of GBDTs to ignore irrelevant variables and model discontinuous functions (Grinsztajn et al., 2022). Importantly, neither the proposed DL architectures nor the large studies investigate the impact of missing values. As missing values are common in real tabular data (Van Ness et al., 2023), and missing value handling can significantly impact the results of analyses (Jin et al., 2021), this is a significant gap in the literature.

In the remainder of this paper we use a claim reserving application with data from a large UK car insurer to shed light on the comparison of GBDTs to DL. We especially focus on the impact of missing value handling in the comparison. Specifically, we investigate Catboost (Prokhorenkova et al., 2018) and ~~two~~ three DL architectures: TabNet (Arik & Pfister, 2021) and ~~a~~two ResNet-inspired multi layer perceptron (MLP)s. TabNet was chosen as a specialised tabular DL architecture with prior validation in insurance (McDonnell et al., 2023). Catboost and a ResNet MLP were chosen as the respective best GBDT and DL model from McElfresh et al. (2024).

In Section 2 we present background on our application as well as describe the car insurance dataset that we analyse. In Section 3 we give a more detailed description of the modelling strategies we compare. In Section 4 we start with our approach to hyperparameter tuning, where extra care was taken to avoid bias, and then describe the experiments used to investigate the impact of missing value handling. In Section 5 we present and discuss the results, not only finding Catboost is the superior model for our data, but also highlighting the importance of missing value handling in model accuracy.

2 Background

Car insurance is an important financial service with a 2024 global value of over 1.9 trillion USD, which is estimated to reach over 2 trillion USD by 2028 (Statista, 2024). Car insurance works on the principle that insurers charge customers a *premium* in return for obligations to provide financial support in the event of contractually agreed risks. Accurate pricing is vital for both the sustainable profit of the insurer and fair prices for customers. The process of determining a price for a prospective customer in car insurance is complex (Olivieri & Pitacco, 2015; Werner & Modlin, 2010). It comprises of three core steps i) estimating the expected value of payments to the customer over the duration of the contract ii) estimating current liabilities for claims that are *reported but not settled (RBNS)* and iii) somehow sensibly combining the two prior estimates into a price. The first step typically comprises finding a model for *claim frequency* and a model for *claim severity*; the latter estimating the cost of a claim conditional on an accident. The second step comprises modelling the cost of claims conditional on them having already occurred and is called *claim reserving.* The third step combines the claim frequency, claim severity and claim reserve estimates using risk models and business considerations: such as profit margins, legal requirements, risk appetite and operational costs.

The focus within this work will be on the second step: claim reserving. Specifically, we focus on *outstanding claim reserve* modelling which is the process of predicting costs for claims that have been RBNS.

Typically, outstanding claim reserve modelling is mainly done on a portfolio level. In other words, insurance companies predict the overall reserve requirement for a given time period, say a quarter, across all customers. Importantly, these forms of claim reserve modelling use no individual claim information, instead using historic data on portfolio claim settlements. This is done with deterministic algorithms such as run-off triangles, the chain ladder (CL) method and the Bornhuetter-Ferguson algorithm (Bornhuetter & Ferguson, 1972); or stochastic extensions of said algorithms.

We focus instead on *individual claim reserve modelling*, or *micro-level reserving*, an alternative method of reserving. Individual claim reserve modelling predicts portfolio reserves from aggregating estimates per incident. There is not yet a consensus that individual claim reserving is more or less accurate than aggregate modelling. Still, the hypothesised benefits of micro-level reserving are: greater insight into exposure profiles within a portfolio; more signal (i.e. relevant covariates) should produce more accurate models; and the ability to adapt to trends that can be captured by covariates (Blier-Wong et al., 2021; Lopez et al., 2019; Delong & Wüthrich, 2020).

There is literature investigating the use of older machine learning (ML) algorithms such as CART (Breiman et al., 1984) and generalized linear models for individual claim reserving (Lopez et al., 2019; De Felice & Moriconi, 2019; Taylor et al., 2008; Wuthrich, 2018). Newer ML methods, such as neural networks (Delong & Wüthrich, 2020; Delong et al., 2022; Kuo, 2020) and gradient boosted trees (Duval & Pigeon, 2019) have also had some, limited, research. These works analysing micro-level reserving strategies broadly conclude that their respective models are either on par or better than an aggregate CL method, validating micro-level reserving in principle. However, there are only a few such works; their insurance fields vary; they use small sets of covariates and some use simulated data. This makes it difficult to know whether the results are relevant to car insurance micro-level reserving. Furthermore, of considerable practical importance is that missing data is endemic to real insurance data (Fauzan & Murfi, 2018; Hanafy & Ming, 2021) and none of these works give any special focus to missing data. Finally, these works often report benchmarks against the CL method instead of overall accuracy which complicates the interpretation of results as disagreement with CL could be the consequence of more accurate modelling. To our knowledge, also noted by the survey of Blier-Wong et al. (2021), none compare modern ML methods directly to each other on real data. The lack of direct comparison means no conclusion can be drawn about the relative performance of newer ML methods for claim reserving.

Ultimately, both research in insurance and ML more broadly paints a blurry picture on the relative merits of GBDTs and DL for micro-level reserving using tabular data. Furthermore, treatment or influence of missing values on accuracy is not investigated when comparing the methods. Although missing value handling has been shown to be important in other fields (Herring et al., 2004) and as such could be important to reserving. This leaves reserving actuaries dealing with tabular data unclear on whether it is worth the investment to investigate and deploy these more modern ML algorithms nor the impact of missing data for said algorithms.

The most relevant work to ours, comparing DL to GBDTs in insurance, is McDonnell et al. (2023). They compare the DL architecture TabNet (Arik & Pfister, 2021) to the GBDT implementation XGBoost. They model discretised *claim severity* classification on a dataset with hundreds of thousands of claims. They find TabNet to be comparable to XGBoost, with marginally better F1 score. Although this is modelling claim *severity*, not claim *reserving*, we note that claim severity and outstanding claim reserve are both costs conditional on an accident occurring. However, the severity is estimated *before* the accident occurs and the reserve *after*. From the perspective of regression, the only difference between micro-level reserving and claim severity modelling is the number of covariates. In the work of McDonnell et al. (2023), although TabNet performs comparatively well to XGBoost, the models were evaluated on synthetic data generated using a neural network (So et al., 2021) thus potentially biasing performance towards DL as the model class was more likely to be correct. Furthermore, the casting of the regression problem into discretised classification and lack of missing data makes the findings less interpretable and transferable.

2.1 Data description

The tabular data we model in Section 4 consists of many hundreds of thousands of insurance ~~claims~~claim feature vectors as rows, with hundreds of features as columns. This dataset has never been previously studied. The data is a combination of information available at policy issue (e.g. make and model of the car) and information available just after the time of claim reporting (e.g. accident date). The settlement value (SV) variable gives how much the insurer paid overall to settle a claim; inclusive of vehicle, personal and property damage. We aim to accurately predict SV for each claim to build a micro-level reserve, as described in Section 1. As we use supervised ML methods we only consider closed claims, i.e. there exists a SV to be used as a label.

Commercial confidentiality prevents us from giving a more detailed description of the data. However, we present the missing data properties in the next section. We present other data characteristics and their implications for modelling and data processing in Appendix A.1.Appendix A.1 includes time varying properties and handling of high cardinality categoricals, such as postcode information.

2.1.1 Missing data

The dataset we study has extensive missing values. Over 50% of features contain missing values, therefore ignoring all features with missing values would drop the number of features by over half. This could drop highly informative features, e.g. details of additional drivers on a policy, which are missing in the majority of claims.

Furthermore, due to the interaction of missing values in multiple features there is no complete feature vectors , i.e. every row has at least one missing value. ~~This suggests if we wish to~~Therefore if we directly apply a strategy such as complete-case analysis (Little & Rubin, 2019, p. 47), where any row with missing values is dropped, the whole dataset would be dropped. ~~More sensible~~Instead, we can first drop features that are missing in more than a certain proportion of cases and then run a complete-case analysis. This latter approach is also used to deal with missing values by Grinsztajn et al. (2022), ~~in~~ one of the broad comparative studies mentioned in Section 1. We explore this method, along with alternative imputation approaches, calling this missing value handling strategy `Drop` in Section 4.

Beyond the extent of missing values, the data presents a dependence of the response, SV, on the missing value structure. This can be shown by a large shift in the mean and standard deviation of the SV when using `Drop` at various missing value proportion thresholds. Smaller proportions of missing values in a feature vector are associated with substantially higher SV. This suggests that the data is not missing completely at random (MCAR) (Little & Rubin, 2019, p. 13-23). Therefore, fitting a model under a `Drop` strategy will result in biased predictions, above and beyond any bias introduced by the model or training algorithm. This bias also means the accuracy results of a model fit on `Drop` are not comparable to those of a model fit on imputed data.

To summarise, missing values represent a large portion of our insurance dataset and the SV varies substantially conditional on the missing value structure. This highlights the importance of investigating and choosing appropriate missing value handling strategies.

3 Models

In this section we start by defining notation, outlining some DL terms and then briefly give background on the models used: i) Catboost, a GBDT model; ii) ~~our implementation of a ResNet multi layer perceptron, a general purpose feed forward DL architecture~~two ResNet multi layer perceptrons, a general purpose feed forward DL model: ours and that of Gorishniy et al. (2021); and iii) TabNet, a DL architecture specifically designed to accommodate tabular data. Within this section we do not aim to provide comprehensive details. Instead we aim to describe methods in sufficient detail to follow the hyperparameters tuned in Section 4.1.

3.1 Notation

We denote the dataset $\mathcal{D}$, as a set of tuples, $\mathcal{D} = \{(\mathbf{x}_k, y_k)\}_{k=1}^{N}$, where $y \in \mathbb{R}^+$ denotes the target settlement value, N denotes the number of claims and $\mathbf{x}$ denotes a feature vector with D features. Subscripts denote indexing on an arbitrary ordering of data tuples from the overall dataset.

We seek a model, $F(\mathbf{x})$, to predict the claim settlement value y. The accuracy of this model is measured by some loss function $L(y, F(\mathbf{x}))$, that we wish to minimise.

3.2 Catboost model

Catboost (Prokhorenkova et al., 2018) is a GBDT (Friedman, 2001) with a special procedure for categorical encoding and gradient estimation. Note that the Catboost algorithm details are complex and have many configurable options. Here we only cover the relevant details of the base GBDT algorithm and briefly mention the core novel concepts proposed by Prokhorenkova et al. (2018). For removal of ambiguity, as the default behaviour can vary depending on the execution hardware, we present details and use defaults for running on a CPU opposed to a GPU.

3.2.1 Gradient boosted decision tree

Boosted models learn an additive ensemble of ‘weak learner’ models. If T is the total number of weak learners we want to use, the boosted model would be:

$$F_T(\mathbf{x}) = G_0(\mathbf{x}) + \sum_{t=1}^{T} \eta G_t(\mathbf{x}), \tag{1}$$

where $G_t(\mathbf{x})$ is the tth ‘weak learner’, η is a weighting factor, and $G_0(\mathbf{x})$ is an initial estimate, such as the mean response of the training set.

Usually boosted models are built in a sequential fashion, e.g. the ith model incorporating i weak learners would be $F_i = F_{i-1} + G_P$ for $i = 1, ..., T$. The sequential construction of the model enables the procedure to be terminated early if validation performance is not improving i.e. return $F_i(\mathbf{x})$ with $i < T$.

For gradient boosting, the summands $G_t(\mathbf{x})$, $t \in \{1, ..., T\}$, are chosen from within a hypothesis class of functions $\mathcal{G}$ to approximate $-\frac{\partial L}{\partial F}(y, F(\mathbf{x}))\big|_{F_{t-1}}$, the negative functional derivative of the loss. This negative functional derivative of the loss is also called a pseudo-residual and denoted $r_{t-1}(y, \mathbf{x})$ (Friedman, 2001).

To evaluate $r_{t-1}(y, \mathbf{x})$ requires knowledge of both $\mathbf{x}$ and y. As y is unavailable outside the training set, $r_{t-1}(y, \mathbf{x})$ can only be evaluated on the training data. However we can approximate r_{t-1} with a given summand $G_t(\mathbf{x})$ and measure of function fit, $L'\big(r_{t-1}(y, \mathbf{x})$:

$$G_t = \underset{G \in \mathcal{G}}{\arg\min} \sum_{(\mathbf{x}, y) \in \mathcal{D}} L'\big(r_{t-1}(\mathbf{x}, y), G(\mathbf{x})\big). \tag{2}$$

In practice finding the true arg min is infeasible so G_t is some approximation learned following a standard algorithm to minimise L'.

L' can be different from L as it is used to fit $G_t(\mathbf{x})$ to $r_{t-1}(y, F(\mathbf{x}))$ to enable derivative evaluation on data outside the training set. It is the addition of G_t to the ensemble that contributes to the minimisation of L given a small enough step size η.

This results in the boosted ensemble approximating a gradient descent of the loss functional (in the space of linear combinations of $\mathcal{G}$) with constant learning rate η:

$$F_t = F_{t-1} + \eta G_t \approx F_{t-1} - \eta \left.\frac{\partial L}{\partial F}\right|_{F_{t-1}} \tag{3}$$

In the context of GBDTs; the weak learner is a decision tree (Breiman et al., 1984). The choice of step size η and desired ensemble size T are among the hyperparameters tuned in Section 4.1.

3.2.2 Catboost: Pseudo-residual calculation and categorical encoding

Catboost aims to improve performance on unseen data by reducing overfitting. The key innovations of Catboost are twofold: i) how the pseudo-residuals, r_{t-1}, are approximated using G_t and ii) how categorical variables are encoded. Although we will describe the core idea of the improvement, there are further technicalities and engineering modifications present in Prokhorenkova et al. (2018), e.g. to improve speed, that we do not describe.

For the alteration to G_t fitting, the core idea is to fit G_t on data excluding the data point for which it will predict r_{t-1}, i.e. to calculate $G_t(\mathbf{x}_k)$ Catboost would fit G_t on data $\{\mathbf{x}_j : j < k\}$. This excludes the data point $\mathbf{x}_k$, and also generates different G_t for different data points.

Likewise, Catboost follows this procedure for generating a categorical encoding. For a given data point $\mathbf{x}_k$, Catboost fits a target mean encoding (Pargent et al., 2022) on a discretized target for $\{\mathbf{x}_j : j < k\}$ that is before the point encoded. Furthermore, when processing categoricals Catboost uses a novel algorithm to redefine category labels as the algorithm runs (‘feature combinations’ in the original work).

We note that there are further important implementation details regarding the categorical encoding, such as how the target is discretised prior to mean encoding, that are absent from the original publication. Full details can be found in the tool’s documentation (Catboost, 2024b) and codebase (Catboost, 2024a).

We will compare Catboost with two different categorical encoding schemes: first with target mean encoding, and second with Catboost’s novel debiased target encoding, that also employs category redefinition.

3.3 ResNet MLP

A ResNet, short for residual network, MLP is a feed-forward neural network (Murphy, 2022, p. 419) with additive *residual* connections that skip layers. Without additional knowledge about the underlying structure of the data, an MLP is a simple general purpose DL architecture; and skip connections make training more stable (Murphy, 2022, p. 445).

We implement a ResNet by using residual connections across building blocks, along with skip connections to the output. We use a building block layout of `BatchNorm`, `ReLU` and `Dense` as in He et al. (2016). We add `Dropout` following the example of Gorishniy et al. (2021) – which proposed the best performing DL model (a ResNet) from McElfresh et al. (2024). For the sake of clarity, our ResNet MLP is *not* identical in

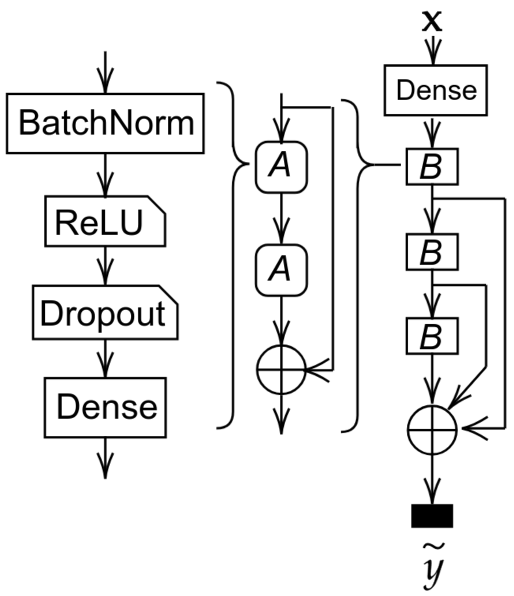

Figure 1: ResNet MLP model architecture. The architecture consists of a number of blocks, B. The skip connections in the diagram indicate that the outputs from each block B are all summed together and passed through a final Dense layer to produce the scalar output $\tilde{y}$. Each layer B consists of three sub-blocks denoted as A in the diagram. Each B contains a single residual connection so that the block output is produced by summing the outputs of the final two sub-blocks A. Each sub-block A consists of feed forward `BatchNorm`, `ReLU`, `Dropout` and `Dense` layers.

architecture to Gorishniy et al. (2021), and by extension McElfresh et al. (2024). For details of the differences with Gorishniy et al. (2021) see Appendix A.3. To contextualise our findings we also present results using the ResNet architecture from Gorishniy et al. (2021), referring to it as 'RTDL ResNet MLP'.

Figure 1 shows the layout of our ResNet MLP layers on the far left, with their combination into a sub-block denoted by A. The ResNet sub-block, A, is repeated in a residual pattern to form a high level block B – shown on the right of Figure 1. This higher level block B is in turn composed using skip connections into an overall model. Each block has independently trainable parameters. In the context of deep learning, choosing the architecture size (such as number of blocks, size of `Dense` layers in units etc.) is a part of the broader problem of hyperparameter tuning. Our approach involves choosing the number of B blocks to vary depth; and choosing the number of units used in every `Dense` layer to vary the width of the network. This tuning is further described in Section 4.1. Apart from architecture, we also use grid search in Section 4.1.3 to select the optimiser used, the initial learning rate of the optimiser, learning rate schedule (Murphy, 2022, p. 288), `Dense` layer weight regularisation strategy and regularisation intensity.

3.4 TabNet model

TabNet (Arik & Pfister, 2021) is a DL architecture specifically designed for tabular data. TabNet works by learning a *step* that multiplies a subset of features by zero, conditional on the input. Then TabNet uses DL layers on the remaining non-zero parts to produce an intermediate *decision* vector. The architecture sequentially applies multiple steps. Each step can determine a different subset of features to set to zero – so a feature that is set to zero for one step does not need to be zero for the following steps. In fact, the hyperparameters described below control how many distinct features can be selected and their potential for reuse across steps. As each step can select different subsets, each step can produce a different decision vector. Finally, the decisions from all steps are combined through a DL layer into a final prediction.

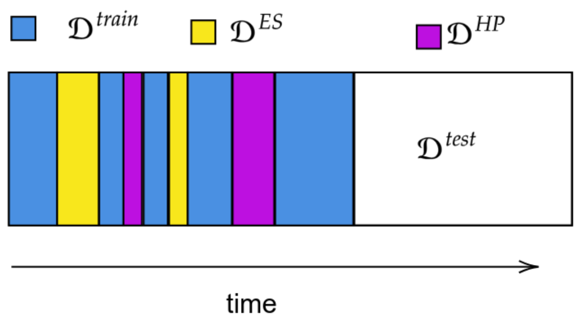

Figure 2: Dataset partitioning strategy for both hyperparameter tuning and final evaluation, where $\mathcal{D}^{\text{ES}}$ and $\mathcal{D}^{\text{train}}$ are shuffled per replication of a given experiment. $\mathcal{D}^{ES}$ is a split of data used for early stopping. $\mathcal{D}^{HP}$ is a split of data used for evaluation of hyperparameters. $\mathcal{D}^{HP}$ is sampled randomly in time, $\mathcal{D}^{test}$ is exclusively future data.

TabNet is a complex architecture for which we defer the detailed description to the original paper (Arik & Pfister, 2021). However, there are some key hyperparameters which we are required to tune.

The number of steps, S, determines the number of different feature subsets that are modelled to produce a prediction. With $S = 1$ only a single subset of the features is used, with more steps resulting in more feature subsets. Intuitively, more steps increases the overall number of features used, but also increases the depth of the network and destabilises training.

The so-called 'relaxation parameter', $\gamma \geq 1$, is designed to encourage different feature subsets to be selected at each step. When $\gamma = 1$ TabNet has the special property of being able to prevent reuse of features between steps. As γ increases, TabNet is more able to reuse features between steps.

The sparsity regularisation coefficient, $\lambda \geq 0$, is used to encourage more input features to be zeroed out in each step. As λ increases the network can multiply more features by zero, even if it decreases training accuracy.

4 Experimental method

To investigate the impact of preprocessing schemes for handling missing values, we first tuned the hyperparameters of each model. Preliminary analysis, described in Appendix A.5, suggested the best hyperparameters did not vary with preprocessing scheme. Therefore, the hyperparameter tuning process was done independently of later missing value investigation.

To prevent data leakage, the last 15% of the data was set aside into a test set, $\mathcal{D}^{\text{test}}$, shown in the top panel of Figure 2. This $\mathcal{D}^{\text{test}}$ was always withheld from training or validation procedures and only used to report the metrics presented in Section 5.

Early stopping (Murphy, 2022, p. 448) was applied to improve training speed and prevent overfitting in both hyperparameter tuning and final model training. Early stopping is a form of regularisation where out-of-sample model performance is evaluated at regular intervals on a dataset withheld from training. When the performance on the withheld dataset decreases, the training algorithm is terminated. Preliminary analysis confirmed there was no decrease in accuracy from using early stopping.

Section 4.1 describes the tuning of hyperparameters discussed in Section 3. Section 4.2 describes the experimental setup used to investigate the impact of missing value handling and categorical encoding. ~~The left and right panels of Figure 2 illustrate how data was allocated for Section 4.1 and Section 4.2 respectively, and will be described in more detail in the relevant sections. Finally, Section ?? touches on training speeds as a practical consideration.~~ Practical commentary on the training speeds of the algorithms can be found in Appendix A.2.

4.1 Hyperparameter tuning

Hyperparameter tuning on our ResNet and TabNet consisted of a grid search optimising for accuracy. For each modelling strategy a hyperparameter (HP) grid was subjectively chosen after initial trial and error. Although grid search may be a common practice in industry, it is also well known to theoretically underperform more principled methods of HPO such as Optuna (Akiba et al., 2019). As such, we also reran analyses for our ResNet using Optuna with the same upper and lower bounds on numeric hyperparameter values. Those results can be found in Appendix A.4 but we note that in this instance Optuna hyperparameter optimisation (HPO) did not significantly change performance. For consistency with McElfresh et al. (2024), Optuna was used for HPO on the RTDL ResNet model and presented in the results Table 1.

A single data subset, $\mathcal{D}^{\text{HP}}$, was sampled once for accuracy evaluation of *all* models under any given HP configuration. For clarity, $\mathcal{D}^{\text{HP}}$ was *not* a future partition of the data, but rather randomly sampled in time.

For each node in the grid, we randomly split the remaining data (after removing $\mathcal{D}^{\text{test}}$ and $\mathcal{D}^{\text{HP}}$) into $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{ES}}$ subsets, illustrated in Figure 2. Training was performed solely on $\mathcal{D}^{\text{train}}$ whilst $\mathcal{D}^{\text{ES}}$ was used to trigger early stopping (ES). Once training was completed, the root mean squared error (RMSE) for a given node was evaluated on $\mathcal{D}^{\text{HP}}$. This random splitting of $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{ES}}$, and subsequent evaluation of RMSE on $\mathcal{D}^{\text{HP}}$, was independently repeated 10 times for each HP node in the grid. This repeated split sampling is called Monte-Carlo cross-validation (Kuhn et al., 2013, p. 71-72).

The best HPs for a given model were chosen on the basis of the lowest RMSE averaged across the 10 Monte-Carlo cross-validation samples. This best HP configuration for a given model was then used for the model-to-model comparison as described in Section 4.2. ~~See the left panel of Figure 2 for an illustration of this data partitioning scheme.~~

The use of separate data subsets for early stopping, $\mathcal{D}^{\text{ES}}$, and accuracy evaluation, $\mathcal{D}^{\text{HP}}$, allowed us to remove bias associated with evaluation on data that was indirectly used for training. This is an especially rigorous process in contrast to what is often done in practice, where one validation set would be used for both early stopping and evaluation. However, as we had sufficient data, we opted for separate subsets to minimise potential bias.

Further details of the training algorithm and hyperparameter selection for each modelling strategy follow.

4.1.1 Catboost hyperparameter tuning

Initial manual exploration of the hyperparameters gave no significant improvements in $\mathcal{D}^{\text{HP}}$ RMSE. The step size, η, was varied between 0.005 and 0.018. The ensemble size, T, was varied between 1000 and 10000. Mean squared error was used as the loss function, L. As no noticeable improvement came from heuristic tuning, a complete grid search was not performed and all hyperparameters were left as defaults for evaluation of Catboost in model-to-model comparison. The only non-default choice was inclusion of early stopping which was used to speed up training, and had no noticeable effect on accuracy in the hyperparameter tuning stage.

4.1.2 TabNet tuning

For TabNet we follow the original paper (Arik & Pfister, 2021) in using the Adam optimiser (Kingma & Ba, 2014) and a learning rate schedule with a fixed initial learning rate of 0.001. The sparsity regularisation coefficient, λ; number of steps, S; and relaxation parameter γ were tuned. Following the recommendations and ablations of Arik & Pfister (2021): λ was varied between 0.0001 and 0.01, γ was varied between 1.3 and 2 and S was varied between 3 and 10.

4.1.3 ResNet MLP tuning

Currently, there is no principled way to select a deep learning architecture, such as depth and width, beyond intuition ~~and trial and error. Although there is research on generating architectures following an algorithm these approaches are computationally expensive and give only slight performance improvements. Since these methods are seldom used in practice, the depth and width of our ResNet MLP was selected heuristically.~~ ;trial and error; and neural architecture search (Ren et al., 2021). As neural architecture search was prohibitively

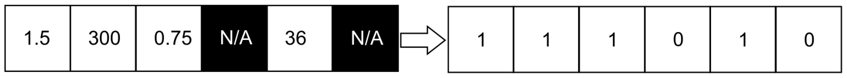

Figure 3: Example row undergoing `Binarize` transformation; where 'N/A' represents missing values. Present covariates map to 1; missing to 0.

expensive from a computational perspective we instead opted for a heuristic trial and error approach for choosing out ResNet architecture.

We began by arbitrarily choosing some expressive high level block, denoted B in Figure 1. We built this block from sub-blocks, denoted A in Figure 1. Sub-blocks A utilise a ResNet (He et al., 2016) layout of layers as described in Section 3.

We then chose the depth: i.e. the number of B blocks composed together prior to a `Dense` layer with a single output unit, with no activation, for prediction. A depth of 3 was chosen; subjectively balancing simplicity of the model with expressive power.

We then parameterised the width of a block with d, the number of units in each `Dense` layer of the ResNet block, denoted A in Figure 1. We varied d between heuristically identified limits wherein the model exhibited underfitting and the capacity to overfit training data. Underfitting was identified by both the training and validation error being similar and approximately constant per training epoch. Capacity to overfit was identified by the ability for the model to keep reducing training error whilst validation error is constant or getting worse. These criteria were considered fulfilled when the validation loss was not improving whilst training loss was still decreasing after 1000 epochs.

Varying d within the bounds of under and overfitting did not noticeably impact accuracy. As such, d was chosen for a total model size of four hundred thousand parameters. This was between the number of parameters which under- or overfitted. After the architecture was heuristically selected, grid search hyperparameter optimisation followed. The optimisers compared were Adam (Kingma & Ba, 2014) and RMSProp (Tieleman & Hinton, 2012). Both L1 and L2 regularisation of kernel weights were ~~compared~~used; coefficients for each were varied between 0.01 and 1. ~~An~~ Exponential Decay (Murphy, 2022, p. 288) and Cosine Decay (Loshchilov & Hutter, 2016) learning rate schedules were both compared; where initial learning rate was varied between 0.001 and 0.01 for each.

4.2 Investigating different ~~preprocessing~~imputation schemes

We aimed to investigate the impact of missing value handling through the comparison of each model under different preprocessing schemes.

To enable a pairwise comparison of trained models, the data subsets for these experiments were different from those of the hyperparameter tuning in Section 4.1. Instead of sampling different subsets per node in a grid search *all* experiments used the same set of 20 Monte-Carlo cross-validation samples of $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{ES}}$ ~~(illustrated in the right panel of Figure 2)~~. These were sampled once prior to running any experiments, enabling pairwise model comparison. As evaluation was performed on $\mathcal{D}^{\text{test}}$, there was no need for another withheld evaluation partition, like $\mathcal{D}^{\text{HP}}$.

A given $\mathcal{D}^{\text{train}}$ subset was used for training all models with all missing value handling schemes, described below, and the corresponding $\mathcal{D}^{\text{ES}}$ was used to trigger early stopping. After the models were trained, RMSE was evaluated on $\mathcal{D}^{\text{test}}$ for each of the 20 $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{ES}}$ samples. The mean and standard error of the RMSE across the 20 Monte-Carlo cross-validation samples for each model and preprocessing scheme are reported in Table 1. These results are discussed in Section 5.

4.2.1 Missing value handling

Four missing value handling strategies were investigated. Two of the methods, `Drop` and `Binarize`, were used in an attempt to disentangle the capability of a model to successfully extract signal from observed covariates and from missing value structure. The other two, `LT Min Impute` and `Mean Impute`, ~~was~~ were used to investigate model capability to jointly use covariates and missing value structure through imputation. The missing value handling strategies are summarised:

- `Drop`: Rows with any missing values were dropped after first removing features with more than 30% missing values – as ~~also~~ described in Section 2.1. From this scheme we aimed to isolate the ability of each modelling strategy to extract signal from observed covariates.
- `Binarize`: The whole dataset was converted into a binary ~~encoding~~representation of whether the covariate was observed or not, see Figure 3. This retained only the signal inherent in the missing value structure. With this scheme we isolate the ability of model strategies to regress against missing value structure.
- `LT Min Impute`: Missing values were imputed using less than the minimum of the numeric feature. This is the default missing imputation strategy of Catboost. The use of this strategy allows the weak learner trees, G_t in Section 3, to separate the missing values completely from observed values. However, this also means that should the tree split on an observed value it will include all missing values on the lesser side of the split.
- `Mean Impute`: Missing values were imputed using the mean of the corresponding numeric feature. This is the default approach used in McElfresh et al. (2024).

4.2.2 Categorical encoding

Categorical encoding was performed using target mean encoding for the majority of the experiments, this compared the modelling strategies fairly by holding potential confounders in categorical encoding constant.

In the interests of identifying the best performance, we also evaluated the original Catboost categorical encoding with the Catboost GBDT algorithm, described in Section 3.2. ~~We report the model names in Table 1 with the suffixes (ME) and (CE) indicating that mean encoding and Catboost encoding were used respectively.~~ We report the model names in Table 1 with the suffix(CE) indicating that Catboost encoding was used; otherwise mean encoding was used.

4.2.3 Effects of imputation scheme on other datasets

To study the broader relevance of results obtained from our insurance dataset, further analyses were performed on two other datasets under `Drop`, `Mean Impute` and `LT Min Impute`. The leading two GBDT and leading two DL algorithms from McElfresh et al. (2024) were compared on two datasets from the same paper. The datasets did not contain missing values; so MCAR missing values were simulated by removal. As MCAR data are not the focus of this case study we present the results in Appendix A.6. Overall, these preliminary TabZilla analyses demonstrate the relevance of our work.

5 Results and discussion

The results for various preprocessing strategies are presented in Table 1. As a benchmark, we include a `Mean prediction` row which shows the performance that is obtained by using the mean SV of a given training dataset as a constant prediction, using no feature information. Note the performance evaluated with the `Drop` preprocessing scheme is not directly comparable to the performance for other preprocessing schemes, i.e. `Drop` cannot be compared with other strategies across rows in Table 1. This is because the missingness mechanism is not missing completely at random and causes a distributional shift within both the training and evaluation data when dropping rows. However, due to consistency of the training datasets, these results are comparable within modelling strategies, i.e. one can compare down columns of Table 1.

The TabNet row of Table 1 shows that TabNet substantially underperforms both Catboost and the ResNet MLP. Notably, TabNet performs on par with the `Mean prediction` benchmark. This indicates that TabNet is either not suitable for micro-level reserving with this dataset or, at best, that TabNet is very difficult to

Table 1: $\mathcal{D}^{\text{test}}$ RMSE mean and standard error, se, over cross validation partitions. **Lower RMSE is better**. RMSE is to the nearest integer, standard error is to 2 significant figures. ~~(ME) and (CE) indicate mean encoding and Catboost encoding were used for categorical encoding respectively.~~(CE) indicates Catboost encoding was used for categorical encoding respectively. † replicates the HPO, architecture and imputation scheme of McElfresh et al. (2024).‡ corresponds to an Optuna HPO scheme.

Model	`Drop`	($\pm se$)	`LT Min Impute`	($\pm se$)	`Mean Impute`	($\pm se$)	`Binarize`	($\pm se$)
Catboost (CE)	1969	(±10)	1452	(±6.3)	1519	(±17)	2302	(±0.18)
Catboost	2060	(±1.7)	1574	(±2.1)	1524	(±5.2)	2268	(±0.28)
Our ResNet	2046	(±1.2)	1872	(±4.5)	7855	(±1600)	2288	(±1.8)
RTDL ResNet‡	2439	(±24)	2007	(±20)	2387	(±270)†	2282	(±2.4)
TabNet	2702	(±8.1)	2748	(±5.3)	3081	(±180)	2754	(±5.1)
Mean prediction	2813	(±0.34)	2764	(±0.020)	2764	(±0.020)	2764	(±0.020)

train. This runs counter to published work on the use of TabNet for claim severity modelling (McDonnell et al., 2023), but in line with surveys of DL for structured data (Grinsztajn et al., 2022; McElfresh et al., 2024) where TabNet performed poorly. As mentioned in Section 1, this could potentially be explained by the fact that the results of McDonnell et al. (2023) could be biased because the modelled data is itself generated from a neural network.

The relative accuracy of the RTDL ResNet MLP row of Table 1 agrees with McElfresh et al. (2024) in outperforming TabNet and underperforming CatBoost. Although for `Drop`, `LT Min Impute` and `Binarize` the RTDL ResNet has lower accuracy than our ResNet it is interesting to note that under `Mean Impute` the RTDL ResNet performs substantially better than any other DL algorithm. This highlights the sensitivity of algorithms to imputation schemes. However, we note that the performance of `Mean Impute` is still generally worse than that of `LT Min Impute` across most models. `Mean Impute` demonstrates moderately high standard error for the RTDL ResNet and high mean RMSE and standard error for our ResNet and TabNet. The cause of the exceptionally high RMSE for our ResNet appears to be the rare prediction of low value claims many orders of magnitude larger than they were. It is unclear why this would be the case specifically for `Mean Impute` and our ResNet. We hypothesise this could be due to `Mean Impute` making it difficult to distinguish meaningfully missing data, where absence of data is not due to random lack of records but from the reality of the data generating process, e.g. no additional drivers on the policy.

The `Drop` and `Binarize` columns of Table 1 show that both the covariates and the missing value structure are important in the performance of both Catboost and ResNet, respectively offering approximately a 27% and 18% improvement over a `Mean prediction` benchmark. Interestingly, neither ResNet nor Catboost is practically more accurate than the other when trained on either covariates (`Drop`) or missing value structure (`Binarize`) alone. This similarity of performance on `Drop` suggests studies comparing GBDTs and DL, mentioned in Section 1, do not have conclusions that are necessarily transferable to the context of claim reserve modelling. Those studies found a performance edge for GBDTs, notably either ignoring or in the absence of missing data. Whereas we find, for this micro-level reserving dataset, Catboost and ResNet are practically the same in terms of accuracy with missing data ignored, due to `Drop`.

However, the `LT Min Impute` and `Mean Impute` columns of Table 1 shows that Catboost with `LT Min Impute` both achieves the best overall performance and has a ~~clearly superior~~better accuracy over ~~ResNet with `LT Min Impute`.~~ both Our ResNet and the RTDL ResNet under either imputation scheme. We note that this ~~superior performance~~better accuracy is between ~~a~~ well-tuned ResNets against a default Catboost, underscoring the robustness of Catboost. The improved accuracy may be because Catboost better leverages both covariate signal and missing value structure, at least under ~~a~~ `LT Min Impute` or `Mean Impute` strategy. ~~This performance improvement could also be due to the fact that minimum imputation intuitively lends itself to the partitioning strategies of a GBDT algorithm.~~ The strongest performance being achieved under `LT Min Impute` could also be due to the fact that minimum imputation intuitively lends itself to the partitioning strategies of a GBDT algorithm. It is unclear how minimum imputation would interact with the ResNet MLP, nor DL more broadly. This discrepancy in performance suggests an indicator of when Catboost may outperform ResNet for micro-level reserving: when there is a high proportion of missing values. This indicator is particularly relevant as missing values are pervasive in real world datasets, and are currently not studied in their role in micro-level reserving nor their contribution to GBDT and DL performance.

These results do not rule out the potential for ~~the ResNet~~ResNets to perform comparably or even better than Catboost given a suitable imputation strategy. However, there does not appear to be a scientific consensus on simple and robust neural network appropriate imputation strategies. Some generative imputation approaches exist (Yoon et al., 2018) but these involve considerable extra complication in the modelling and training and have not shown much industry adoption.

6 Conclusion

~~In this paper we investigated `Drop`, `Binarize`, and `LT Min impute`, described in Section 4.2.1, for handling missing values using gradient boosted decision tree and deep learning models, in car insurance reported but not settled reserving. A thorough experimental method was used to tune hyperparameters and under two of the missing value handling schemes there is no noticeable difference between Catboost and ResNet. However, using imputation, Catboost substantially outperforms the other models. This result highlights the importance of missing value handling in reported but not settled reserving.~~ In this paper we investigated four different imputation schemes, described in Section 4.2.1, for handling missing values using gradient boosted decision tree and deep learning models. We investigated the impact of these imputation schemes using a large, real world insurance dataset with not MCAR missing values. Under two of the missing value handling schemes there is no noticeable difference between Catboost and our ResNet. However, using imputation, Catboost substantially outperforms the other models. This result highlights the importance of missing value handling in claim reserving.

More broadly, this result adds a case study to the body of evidence that gradient boosted decision trees can outperform deep learning for tabular data, but emphasises the importance of data handling in drawing this conclusion. The significant impact of missing value handling on accuracy also suggests that when analysing datasets with missing values, extra care should be taken choosing the missing value handling method and not to just focus on model selection. Furthermore, our results suggest research comparing gradient boosted decision trees to deep learning for tabular data could benefit from including more datasets with missing values, especially missing values that are not MCAR.

Based on the analysis in this work we recommend exercising caution when using deep learning models for claim reserving as they require thorough tuning and their interaction with imputation schemes is not understood. Furthermore, using Catboost for claim reserving has some practical advantages, beyond potentially better accuracy with the correct missing value handling. Catboost is fast, robust and easy to use off-the-shelf. In comparison, deep learning methods require more expertise to deploy successfully: requiring both architecture selection and hyperparameter tuning. Even with the expertise required to select, implement and optimise deep learning models there is no compelling empirical or theoretical evidence that they are likely to produce better results for claim reserving.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019.

Junyi Ao, Rui Wang, Long Zhou, Chengyi Wang, Shuo Ren, Yu Wu, Shujie Liu, Tom Ko, Qing Li, Yu Zhang, Zhihua Wei, Yao Qian, Jinyu Li, and Furu Wei. Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing. arXiv preprint arXiv:2110.07205, 2021.

Sercan Arik and Tomas Pfister. TabNet: Attentive interpretable tabular learning. Proceedings of the AAAI Conference on Artificial Intelligence, 35(8):6679–6687, 2021.

Kiran Baktha and BK Tripathy. Investigation of recurrent neural networks in the field of sentiment analysis. 2017 International Conference on Communication and Signal Processing, pp. 2047–2050, 2017.

Christopher Blier-Wong, Hélène Cossette, Luc Lamontagne, and Etienne Marceau. Machine learning in P&C insurance: A review for pricing and reserving. Risks, 9(1):4, January 2021. ISSN 2227-9091. doi: 10.3390/risks9010004.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. IEEE Transactions on Neural Networks and Learning Systems, pp. 1–21, 2022.

Ronald L Bornhuetter and Ronald E Ferguson. The actuary and IBNR. Casualty Actuarial Society, 59(112): 181–195, 1972.

Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. Classification and Regression Trees. Wadsworth and Brooks/Cole Monterey, CA, USA, 1984.

Catboost. Codebase tutorial on Catboost encoding, Feb 2024a. `https://github.com/catboost/catboost/blob/master/catboost/tutorials/categorical_features/categorical_features_parameters.ipynb`, Accessed on April 2024.

Catboost. Transforming categorical features to numerical features | Catboost, Feb 2024b. `https://catboost.ai/en/docs/concepts/algorithm-main-stages_cat-to-numberic`, Accessed on April 2024.

Massimo De Felice and Franco Moriconi. Claim watching and individual claims reserving using classification and regression trees. Risks, 7(4):102, December 2019. ISSN 2227-9091. doi: 10.3390/risks7040102.

Łukasz Delong and Mario V. Wüthrich. Neural networks for the joint development of individual payments and claim incurred. Risks, 8(2):33, June 2020. ISSN 2227-9091. doi: 10.3390/risks8020033.

Łukasz Delong, Mathias Lindholm, and Mario V. Wüthrich. Collective reserving using individual claims data. Scandinavian Actuarial Journal, 2022(1):1–28, January 2022. ISSN 0346-1238. doi: 10.1080/03461238.2021.1921836.

Francis Duval and Mathieu Pigeon. Individual loss reserving using a gradient boosting-based approach. Risks, 7(3):79, September 2019. ISSN 2227-9091. doi: 10.3390/risks7030079.

Muhammad Arief Fauzan and Hendri Murfi. The accuracy of XGBoost for insurance claim prediction. International Journal of Advances in Soft Computing and its Applications, 10(2):159–171, 2018.

Jerome Friedman. Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29(5), October 2001. ISSN 0090-5364. doi: 10.1214/aos/1013203451.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. Advances in Neural Information Processing Systems, 34:18932–18943, 2021.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? Advances in Neural Information Processing Systems, 35:507–520, 2022.

Mohamed Hanafy and Ruixing Ming. Machine learning approaches for auto insurance big data. Risks, 9(2), 2021. ISSN 2227-9091. doi: 10.3390/risks9020042. URL `https://www.mdpi.com/2227-9091/9/2/42`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. European Conference on Computer Vision, pp. 630–645, 2016.

Amy H. Herring, Joseph G. Ibrahim, and Stuart R. Lipsitz. Non-ignorable missing covariate data in survival analysis: A case-study of an international breast cancer study group trial. Journal of the Royal Statistical Society Series C: Applied Statistics, 53(2):293–310, 03 2004. ISSN 0035-9254. doi: 10.1046/j.1467-9876.2003.05168.x. URL `https://doi.org/10.1046/j.1467-9876.2003.05168.x`.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. arXiv preprint arXiv:2012.06678, 2020.

Liang Jin, Yingtao Bi, Chenqi Hu, Jun Qu, Shichen Shen, Xue Wang, and Yu Tian. A comparative study of evaluating missing value imputation methods in label-free proteomics. Scientific Reports, 11(1):1760, January 2021. ISSN 2045-2322. doi: 10.1038/s41598-021-81279-4.

Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. Advances in Neural Information Processing Systems, 34:23928–23941, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Max Kuhn, Kjell Johnson, et al. Applied predictive modeling, volume 26. Springer, 2013.

Kevin Kuo. Individual Claims Forecasting with Bayesian Mixture Density Networks. arXiv preprint arXiv:2003.02453, March 2020. doi: 10.48550/arXiv.2003.02453.

Roderick JA Little and Donald B Rubin. Statistical analysis with missing data, 3rd Edition. John Wiley & Sons, 2019.

Olivier Lopez, Xavier Milhaud, and Pierre-E Thérond. A tree-based algorithm adapted to microlevel reserving and long development claims. ASTIN Bulletin: The Journal of the International Actuarial Association, 49(3):741–762, 2019.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983, 2016.

Kevin McDonnell, Finbarr Murphy, Barry Sheehan, Leandro Masello, and German Castignani. Deep learning in insurance: Accuracy and model interpretability using TabNet. Expert Systems with Applications, 217: 119543, 2023. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2023.119543. URL `https://www.sciencedirect.com/science/article/pii/S0957417423000441`.

Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? Advances in Neural Information Processing Systems, 36, 2024.

Kevin P. Murphy. Probabilistic machine learning: An introduction. MIT Press, 2022. URL `probml.ai`.

Robert Nisbet, John Elder, and Gary D Miner. Handbook of statistical analysis and data mining applications. Academic press, 2009.

Annamaria Olivieri and Ermanno Pitacco. Introduction to insurance mathematics: technical and financial features of risk transfers. Springer, 2015.

ONS. Postal geographies - office for national statistics, Feb 2024. `https://www.ons.gov.uk/methodology/geography/ukgeographies/postalgeography`, Accessed Feb 2024.

Florian Pargent, Florian Pfisterer, Janek Thomas, and Bernd Bischl. Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. Computational Statistics, 37(5):2671–2692, November 2022. ISSN 1613-9658. doi: 10.1007/s00180-022-01207-6.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. Advances in Neural Information Processing Systems, 31, 2018.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. International Conference on Machine Learning, pp. 8748–8763, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. International Conference on Machine Learning, 139: 8821–8831, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/ramesh21a.html`.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. ACM Computing Surveys, 54(4), may 2021. ISSN 0360-0300. doi: 10.1145/3447582. URL `https://doi.org/10.1145/3447582`.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjarn Ommer. High-resolution image synthesis with latent diffusion models. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2022. URL `https://github.com/CompVis/latent-diffusionhttps://arxiv.org/abs/2112.10752`.

Ira Shavitt and Eran Segal. Regularization learning networks: Deep learning for tabular datasets. Advances in Neural Information Processing Systems, 31, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. International Conference on Learning Representations, 2015.

Banghee So, Jean-Philippe Boucher, and Emiliano A. Valdez. Synthetic dataset generation of driver telematics. Risks, 9(4), 2021. ISSN 2227-9091. doi: 10.3390/risks9040058. URL `https://www.mdpi.com/2227-9091/9/4/58`.

Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. SAINT: Improved neural networks for tabular data via row attention and contrastive pre-training. arXiv preprint arXiv:2106.01342, 2021.

Statista. Motor vehicle insurance - global: Statista market forecast, Feb 2024. `https://www.statista.com/outlook/fmo/insurances/non-life-insurances/motor-vehicle-insurance/worldwide`, Accessed on Feb 2024.

Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. arXiv preprint arXiv:2005.10821, 2020.

Greg Taylor, Gráinne McGuire, and James Sullivan. Individual claim loss reserving conditioned by case estimates. Annals of Actuarial Science, 3(1-2):215–256, 2008.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. Coursera: Neural networks for machine learning, 4(2):26–31, 2012.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023.

Mike Van Ness, Tomas M. Bosschieter, Roberto Halpin-Gregorio, and Madeleine Udell. The Missing Indicator Method: From Low to High Dimensions. Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 5004–5015, August 2023. doi: 10.1145/3580305.3599911.

Geoff Werner and Claudine Modlin. Basic ratemaking. Casualty Actuarial Society, 4:1–320, 2010.

Mario V. Wuthrich. Machine learning in individual claims reserving. Scandinavian Actuarial Journal, 2018 (6):465–480, July 2018. ISSN 0346-1238. doi: 10.1080/03461238.2018.1428681.

Jinsung Yoon, James Jordon, and Mihaela van der Schaar. GAIN: Missing data imputation using generative adversarial nets. International Conference on Machine Learning, 80:5689–5698, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/yoon18a.html`.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. arXiv preprint arXiv:1909.08593, 2019.

A Appendices

A.1 Data properties

This appendix presents further properties of the data.

A.1.1 Missing value concentration

Tables 2 and 3 characterise the concentration of missing values in our dataset.

Fraction f missing	Percentage of columns with given fraction missing
$f = 0$	45
$0 < f \leq 0.2$	17
$0.2 < f \leq 0.4$	13
$0.4 < f \leq 0.6$	11
$0.6 < f \leq 0.8$	7
$0.8 < f \leq 1.0$	6

Table 2: Missingness distribution by column, to nearest %

Fraction f missing	Percentage of rows with given fraction missing
$0 < f \leq 0.2$	46
$0.2 < f \leq 0.4$	19
$0.4 < f \leq 0.6$	12
$0.6 < f \leq 0.8$	11
$0.8 < f \leq 1.0$	11

Table 3: Missingness distribution by row, to nearest %

A.1.2 Time-varying properties

Figure 4 depicts the mean claim value per month, with claim value and year of incident anonymised for commercial confidentiality. The figure shows that mean claim value is clearly dependent on the time of the claim – exhibiting both seasonality and trend. This has implications for both the evaluation of model accuracy and encoding of time data.

When evaluating the accuracy of the ML models, described in Section 3, the time series nature of the data requires the definition of the test set data to be in the future relative to all training and validation data. This is to prevent bias in the estimation of performance, a form of data leakage (Nisbet et al., 2009, p. 742).

When considering how to encode the time variables, we aim to encode in such a way as to make it easier to fit seasonality and trend. To explicitly encode cyclic timestamp properties, we encode month and day-of-month variables separately. Having cyclic values in the input data, like month and day-of-month, intuitively makes it easier to fit conditional on cyclic seasons. In order to fit overall trend, we chose to also encode timestamps as a monotonic value, enabling the model to order claims in time from a single numeric value. We did this at two resolutions: i) year and ii) seconds since the epoch, also known as *Unix time*. The goal of encoding in years is to allow a fit to large scale trends such as inflation. The goal of encoding in seconds is to enable more granular ordering in time. Therefore, overall we encode a single timestamp feature into 4 variables: year, month, date and seconds since the epoch.

A.1.3 High-cardinality categorical variables

The dataset contains some high-cardinality categorical variables: insuree car make with hundreds of categories, insuree car model with thousands of categories and *first half* of insuree postcode with thousands of categories. The typical one-hot encoding (Murphy, 2022, p. 23) of these features would dramatically

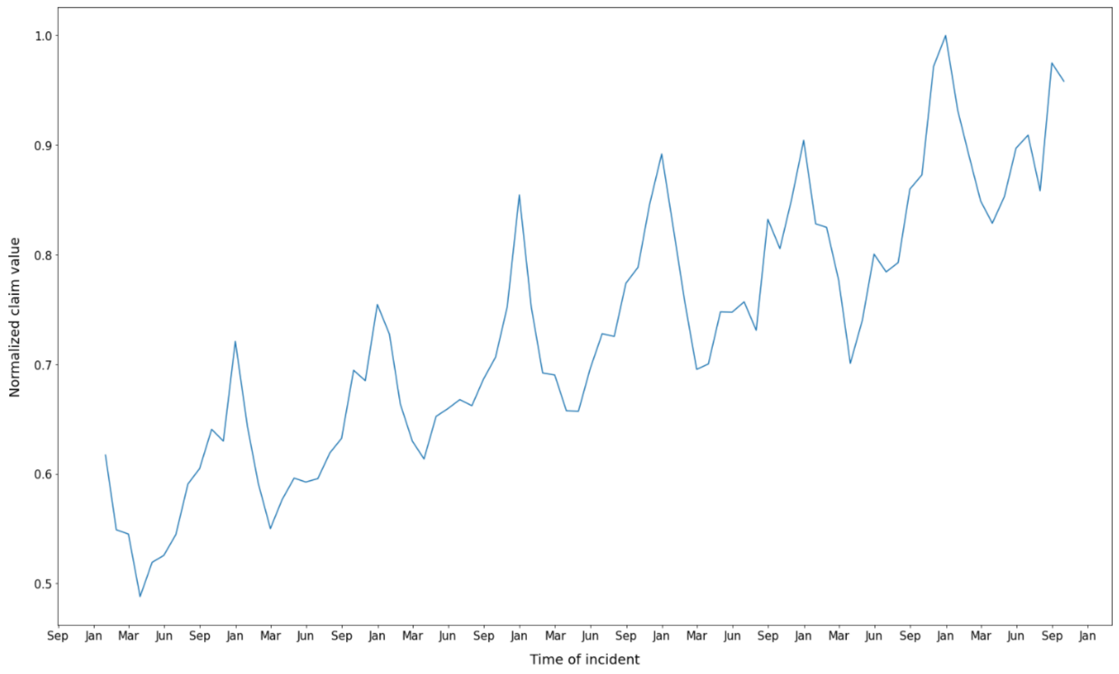

Figure 4: Time series plot of mean standardised claim values against anonymised time. Claim values were aggregated on a month. Claim values and years of data are anonymised for confidentiality.

blow up the dataset size in number of features. Therefore, we applied numeric encoding strategies such as impact encoding (Pargent et al., 2022), also known as *target mean encoding*, and Catboost's novel categorical encoding, described in Section 3.2.2.

As the UK has 1.7 million postcodes (ONS, 2024), attempting to treat full postcodes as a categorical variable would give rise to at least hundreds of thousands of categories. Even target mean encoding such a high cardinality categorical could exhibit high bias and high variance due to the low number of samples per category; following the intuition outlined in Prokhorenkova et al. (2018). Only modelling the first half of postcodes was used to decrease the cardinality of the variable to thousands of categories to potentially reduces the bias and variance of the mean encoding.

A.2 Training speed

Using 12 Intel Xeon 6136 CPUs and approximately 60GB of RAM: Catboost trains on our dataset in the order of 3 minutes per training run and the ResNet MLP architecture trains in the order of 45-120 minutes per training run. TabNet trains in the order of 300 minutes per training run. Although both Catboost and ResNet exhibited training times low enough to be retrained many times a day to keep up with changes in underlying distribution; the significantly faster training time of Catboost enables greater experimentation. This is potentially relevant for other applications as it has been shown that the difference in modelling strategies without hyperparameter tuning is just as great as the difference within a model class with hyperparameter tuning when training on tabular data (Kadra et al., 2021).

A.3 Differences with McElfresh

There are a few key differences between Our ResNet MLP and the RTDL ResNet MLP:

- Our ResNet MLP HPO tunes more HPs; McElfresh et al. (2024) only tunes learning rate whereas our ResNet uses an HPO grid containing learning rate, weight decay, regularisation coefficient, learning rate schedule and optimizer.

- Our ResNet MLP does not attempt to embed categorical features, as target mean encoding was used to convert all categories into scalar representations.
- Our ResNet MLP has skip/residual connections from output of each block to the overall model output; whereas residual connection only skip blocks in McElfresh et al. (2024).
- We use three repeating blocks prior to the regression head layer; McElfresh et al. (2024) use two.
- Our ResNet MLP does not vary dimensionality of the hidden state - we keep constant dimensionality of 256, in comparison to McElfresh et al. (2024) who step down to 128 and back up to 256 throughout the network. Ultimately, this and the above point result in our model having approximately 2.25x the parameters of McElfresh et al. (2024).

A.4 Optuna HPO Results

Table 4 shows a table analogous to Table 1 comparing results obtained using Optuna for the HPO instead of grid search. It can be seen that there is no substantive difference to the conclusions in Table 1 of the work from swapping to Optuna HPO; as core points are made in comparison of Our ResNet MLP and Catboost (which did not undergo HPO).

Table 4: Results when using Optuna HPO

Model	`Drop`	($\pm se$)	`LT Min Impute`	($\pm se$)	`Mean Impute`	($\pm se$)	`Binarize`	($\pm se$)
Our ResNet MLP + Grid Search (ME)	2046	(±1.2)	1872	(±4.5)	7855	(±1600)	2288	(±1.8)
Our ResNet MLP + Optuna HPO (ME)	2039	(±5.1)	1887	(±6.5)	7673	(±1600)	2283	(±1.3)

A.5 Impact of imputation scheme on HPO

This appendix presents preliminary experimental data used to justify the process of performing HPO independently of imputation scheme; as described in the beginning of Section 4. The closeness in parameter magnitude value and the robustness of HPO performance to varying hyperparameters suggested it was acceptable to reduce the computational complexity of experiments by performing HPO under one imputation scheme and evaluating on all. The `LT Min Impute` scheme was chosen as initial results suggested it would be the scheme with the best performance; and as such chosen in an attempt to give a level playing field for best performance across models.

A.5.1 Grid search based

The optimum hyperparameters obtained using grid search with `LT Min Impute` and `Drop` for our ResNet from preliminary analysis are presented in Table 5. Due to similarity in the optimal hyperparameters obtained it was concluded from this preliminary analysis that HPO could be performed independent of imputation scheme; keeping overall computational cost down.

Table 5: Best performing hyperparameters under different imputation schemes; following a grid search HPO procedure.

Hyperparameter	`LT Min Impute`	`Drop`
Regularisation coefficient	0.01	0.01
Initial learning rate	0.01	0.01
Learning rate schedule	CosineDecay	CosineDecay
Optimiser	RMSProp	RMSProp
Weight Decay	0.01	0.0001

A.5.2 Optuna based

The optimum hyperparameters obtained using Optuna with `LT Min Impute` and `Drop` for our ResNet are presented in Table 6. With corresponding final outcomes presented in Table 7. It can be seen from 7 that although tuning with `Drop` gives somewhat different hyperparameters; and gives our ResNet more stable results for `Mean Impute`; the positioning in the final Table 1 would be unaffected.

Table 6: Best performing hyperparameters under different imputation schemes; following an Optuna HPO procedure.

Hyperparameter	`LT Min Impute`	`Drop`
Regularisation coefficient	0.017	0.68
Initial learning rate	0.009	0.006
Learning rate schedule	CosineDecay	ExponentialDecay
Optimiser	RMSProp	RMSProp
Weight Decay	0.0002	0.0002

Table 7: Results when using Optuna HPO

Model	`Drop` $(\pm se)$	`LT Min Impute` $(\pm se)$	`Mean Impute` $(\pm se)$	`Binarize` $(\pm se)$
Our ResNet MLP + `LT Min Impute` Optuna HPO	2039 (± 5.1)	1887 (± 6.5)	7673 (± 1600)	2283 (± 1.3)
Our ResNet MLP + `Drop` Optuna HPO	2047 (± 4.3)	1885 (± 4.6)	4548 (± 680)	2292 (± 2.0)

A.6 TabZilla replication

This appendix details the results obtained from comparing `LT Min Impute`, `Mean Impute` and `Drop` on some extra datasets and algorithms from the TabZilla Benchmark (McElfresh et al., 2024). Source code can be found at `https://github.com/paper3193/tabzilla`.

30% of numeric values were randomly replaced with missing values from each dataset and then imputed using the imputation schemes studied. Then the correpsonding OpenML task was performed using Catboost, XGBoost, FTTransformer (Gorishniy et al., 2021) and the RTDL ResNet. We present the results by dataset in Table 8.

From Table 8 we can see that the overall impact of imputation schemes is not large and in some cases causes no performance difference between certain imputation schemes. However, the changes in accuracy rankings of the datasets under different imputation schemes demonstrates that in principle it is possible for the rankings, and therefore results such as those in McElfresh et al. (2024), to be influenced by imputation scheme. We note that, by design, Table 8 shows results from injected MCAR missing values and as such we do not attempt to interpret the absolute performance of any imputation scheme as MCAR data is not the focus of our work.

In summary, the TabZilla replication with different imputation schemes demonstrates the principle that handling missing values could be important; but the MCAR nature of the injected missing values and low signal value of the numerics in the chosen datasets produces a less pronounced effect as in our work with non MCAR claim reserving. This indicates the importance of analyses using real world large datasets with not MCAR missing value structure in comparing GBDTs and DL models.

Table 8: Relative accuracy and ranking per dataset per algorithm using the TabZilla repository under different imputation schemes. We report TabZilla mean 10-fold cross-validation test accuracy; as extracted from the tuned aggregated results output.

Dataset	Model	`Mean Impute`		`Drop`		`LT Min Impute`	
		Accuracy	Rank	Accuracy	Rank	Accuracy	Rank
`ada_agnostic`	CatBoost	0.857	1	0.854	2	0.857	1
`ada_agnostic`	FTTransformer	0.844	4	0.845	3	0.846	3
`ada_agnostic`	RTDL ResNet	0.846	3	0.842	4	0.841	4
`ada_agnostic`	XGBoost	0.855	2	0.855	1	0.855	2
`LED-display`	CatBoost	0.716	2	0.714	3	0.716	3
`LED-display`	FTTransformer	0.708	4	0.728	1	0.728	1
`LED-display`	RTDL ResNet	0.724	1	0.724	2	0.722	2
`LED-display`	XGBoost	0.710	3	0.706	4	0.710	4