Assignment 1

Jakob og Oskar

2025

Assignment 1 in interpretable machine learning

We are tasked with building a predictive regression model, with the best possible prediction. This submission will be split up into several parts:

- Initial data preprocessing and overview
- Introduction into the mathematics
- Modelling and justification
- Comparative discussion

Initial data preprocessing and overview

The given data has the form:

```
##  [1] "ID"                   "Date_start_contract"  "Date_last_renewal"
##  [4] "Date_next_renewal"    "Date_birth"           "Date_driving_licence"
##  [7] "Distribution_channel" "Seniority"            "Policies_in_force"
## [10] "Max_policies"         "Max_products"         "Lapse"
## [13] "Date_lapse"           "Payment"              "Premium"
## [16] "Cost_claims_year"     "N_claims_year"        "N_claims_history"
## [19] "R_Claims_history"     "Type_risk"            "Area"
## [22] "Second_driver"        "Year_matriculation"   "Power"
## [25] "Cylinder_capacity"    "Value_vehicle"        "N_doors"
## [28] "Type_fuel"            "Length"               "Weight"
```

We are asked to predict the *Cost_claims_year* given the rest of the covariate-vector. Initially it is important to note that out data is a classical insurance dataset, where we are given rows corresponding to insurance periode for a given contract. There are several issues with this, since some contracts might overlap into multiple contracts, which can be identified by the ID. There are numerous char. vectors in the data, which can be seen here:

```
## Classes 'data.table' and 'data.frame':   105555 obs. of  30 variables:
##  $ ID                 : int  1 1 1 1 2 2 3 3 3 3 ...
##  $ Date_start_contract : chr  "05/11/2015" "05/11/2015" "05/11/2015" "05/11/2015" ...
##  $ Date_last_renewal   : chr  "05/11/2015" "05/11/2016" "05/11/2017" "05/11/2018" ...
##  $ Date_next_renewal   : chr  "05/11/2016" "05/11/2017" "05/11/2018" "05/11/2019" ...
##  $ Date_birth          : chr  "15/04/1956" "15/04/1956" "15/04/1956" "15/04/1956" ...
##  $ Date_driving_licence: chr  "20/03/1976" "20/03/1976" "20/03/1976" "20/03/1976" ...
##  $ Distribution_channel: chr  "0" "0" "0" "0" ...
```

```
##  $ Seniority           : int  4 4 4 4 4 4 15 15 15 15 ...
##  $ Policies_in_force   : int  1 1 2 2 2 2 1 1 1 1 ...
##  $ Max_policies        : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ Max_products        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Lapse               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Date_lapse          : chr  "" "" "" "" ...
##  $ Payment             : int  0 0 0 0 1 1 0 0 0 0 ...
##  $ Premium             : num  223 214 215 217 214 ...
##  $ Cost_claims_year    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ N_claims_year       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ N_claims_history    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ R_Claims_history    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Type_risk           : int  1 1 1 1 1 1 3 3 3 3 ...
##  $ Area                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Second_driver       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Year_matriculation  : int  2004 2004 2004 2004 2004 2004 2013 2013 2013 2013 ...
##  $ Power               : int  80 80 80 80 80 80 85 85 85 85 ...
##  $ Cylinder_capacity   : int  599 599 599 599 599 599 1229 1229 1229 1229 ...
##  $ Value_vehicle       : num  7068 7068 7068 7068 7068 ...
##  $ N_doors             : int  0 0 0 0 0 0 5 5 5 5 ...
##  $ Type_fuel           : chr  "P" "P" "P" "P" ...
##  $ Length              : num  NA NA NA NA NA ...
##  $ Weight              : int  190 190 190 190 190 190 1105 1105 1105 1105 ...
##  - attr(*, ".internal.selfref")=<externalptr>
## NULL
```

One could, model some of the char. vectors like proposed in the lectures, by

$$\begin{aligned} X_i &= E(Y \mid X_i) \\ &= \int_Y y \ \mu(x, dy) \end{aligned}$$

where μ is the appropriate probability kernel, and we let y be our response. However, we choose to take the chararacter vectors and round them to yearly values, which then has a ordinal ordering and can thus be used as features. Further we take and one-hot encode *Distribution_channel* by creating three new features which are either 1 or 0. Same for the type of fuel.

Next there are some missing values.

It becomes apparent that there are missing values in the length and type_fuel variable. We can see that the missing is overlapping in 1764 rows. However the length variable suffers way heavier from missing compared to type_fuel. In order to impute values, we assume the missing it completely at random for both features. We consider correlated features to do imputation. We see from the correlation plot (fig. 3) that type_fuel is mostly correlated with Cylinder_capacity, Value_vehicle and Weight. The same goes for the feature Length. Therefore, we fit a multivariate linear regression model with these 3 covariates to predict both type_fuel and Length (using cbind in the response formula). Finally, we predict the missing values using this trained model.

Our response variable *Cost_claims_year* suffers from a few extreme values. We decide to remove these values to later on achieve a better model fit. In fig. 2, *N_claims_year* is plotted against *Cost_claims_year*, where at least 5-10 extreme values of *Cost_claims_year* are spotted.

Finally we look at the correlation between the covariates.

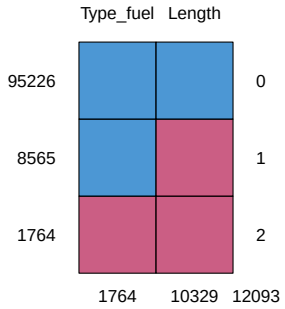

Figure 1: Missing data plot, right axis shows numer of missing columns in that row, and the left axis show how many rows have this missingness pattern

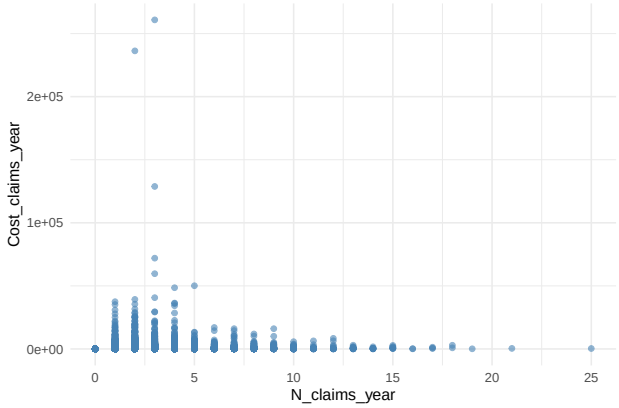

Figure 2: Claim costs over number of claims

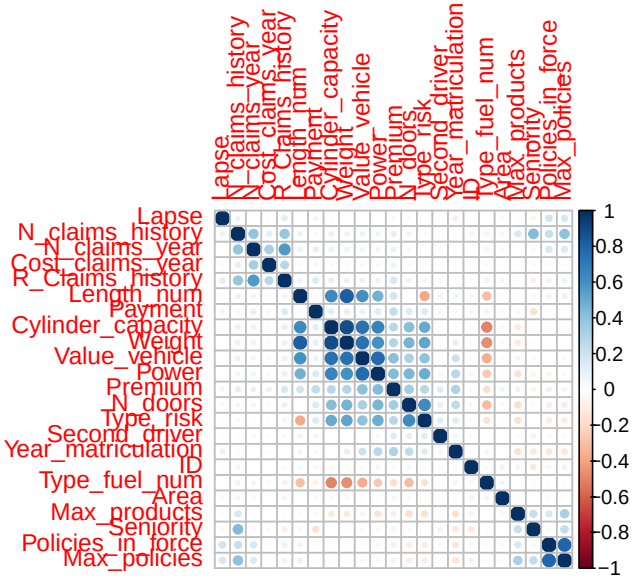

Figure 3: Correlation plot between the continious covariates

We notice some clusters, most meaningfull between *Cylinder_capacity*, *Value_vehicle* and *Weight* which is expected. We deem these to have significant predictive ability, and thus we choose to not remove these. The top left cluster, will be ignored for now, since we will later introduce a data-transformation which affects this cluster in a high degree.

Data Aggregation For our data aggregation we want the ability to assume independence, which we have if we aggregate the rows into one policy. There are several problems to tackle to achieve data in the desired format. We have to both choose an appropriate aggregation, and we have to consider events where the insured object change. We start off by identifying some unique-identifiers. If these columns change, we deem the contract to insure a new car, or at least insure something that is independent of what was previously insured. Among these unique identifiers are *Length*, *Weight*, *Power* etc. If these change, it probably means that the insured object has changed.

Next, we look at how we have aggregate data. Firstly we sum the exposure for each contract, since we will use this to weight in the models later on. Next we use the max function on the number of policies, lapse, date of birth, date of driving licence, start of contract, products, payments, ratio of claims history, type of risk,. For the premium we use the mean of the premium without the forward premium, that is the premium for the time we want to predict for. For the value of the vehicle, and number of doors we use mean. For the length we use sum since this is just a scaling. For the variable *cost_claims_this_year* we use the cost of the claims in $\mathcal{F}_t$ where we are to predict data on $\mathcal{F}_{t+1}$.

So we have $Z_i(\omega_i) := (X_i(\omega_i), Y_i(\omega_i)) : (\Omega_i, \mathcal{F}_i) \to (\mathcal{X} \times \mathcal{Y}, \mathcal{B}(\mathcal{X} \times \mathcal{Y}))$, where by aggregating data row-wise we obtain independence on every set in the Borel-set induced. However this is an empirical assumption which can be violated by catastrophe events.

Introduction into the mathematical framework

Note the classical derivation of how to decompose a total claim sum.

$$\begin{aligned} E(S(t) \mid Z = z) &= E\left(\sum_{i=1}^{N(t)} X_i \mid Z = z\right) \\ &= E\left(\left(\sum_{i=1}^{N(t)} X_i \mid Z = z \cap \mathcal{F}(t)\right)\right) \\ &= E\left(NE\left(X_i \mid Z = z\right) \mid Z = z\right) \\ &= E(N \mid Z = z)E(X_1 \mid Z = z) \end{aligned}$$

We would however like to modify this slightly, later on. Further the general definition of the Tweedie law is

$$P_{\theta,\sigma^2}(Y \in A) = \int_A \exp\left\{\frac{\theta z - \kappa_p(\theta)}{\sigma^2}\right\} \nu_y(dz)$$

Where $A \subset \mathbb{R}$ and measurable. θ is the canonical parameter, and $\kappa_p(\theta)$ is the cumulative function dependent on p the so called tweedie power parameter. σ^2 is the dispersion parameters, and ν_λ is some sigma-finite base measure depending on the Lebesgue measure.

since it can accommodate a zero-inflated distribution very well, we feel the need to introduce this model here. Lastly we introduce the so-called *bbrier* measure, for classification models.

$$\frac{1}{n}\sum_{i=1}^{n} w_i(1\{X_i = 1\} - P(X_i = 1))^2$$

Which is the mean weighted square euclidean distance between the probability and the true value.

Modelling and justification

In this section, we train and evauluate different models. For all the proposed models, we have caried out nested cross-validition. Also, we stick to 5 folds for both inner and outer resampling, with 2 evaluations. We do stratified resampling in both the inner and outer cross-validation-loops, to ensure both zero and non-zero claims over the training and test set. I final note, that goes for all our trained models, is that we introduce offset on the exposure. Meaning: Give more importance to observations with more exposure since, they represent more information.

```
#Common trans dataset to skip rerunning trans

data_trans <- data_trans(data)
```

Tweedie Initily we want to fit a Tweedie model on the entire data, since the Tweedie model is known for handeling zero-inflated distributions well. We plot the distribution of the claim size.

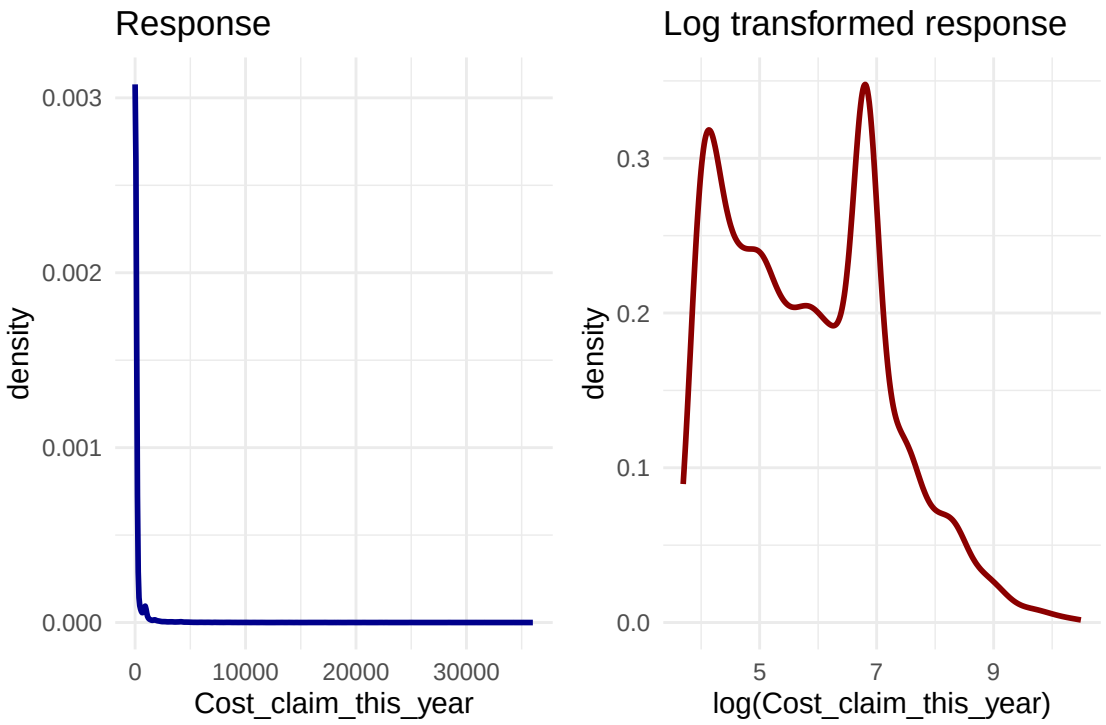

Figure 4: Distribution of claim size pr. unique contract

We apply both the Xgboost, Ligth-gbm and glmnet. For each of these we supply search-space information in the "**Parameter search space tweedie**" Table.

We define the workflow as shown in figure 5.

The results of the nested cross-validation are shown in figure 6.

And finally the chosen hyperparameters are presented in Table "**Best parameters tweedie**".

The tweedie power parameter makes the model a compound gamma.

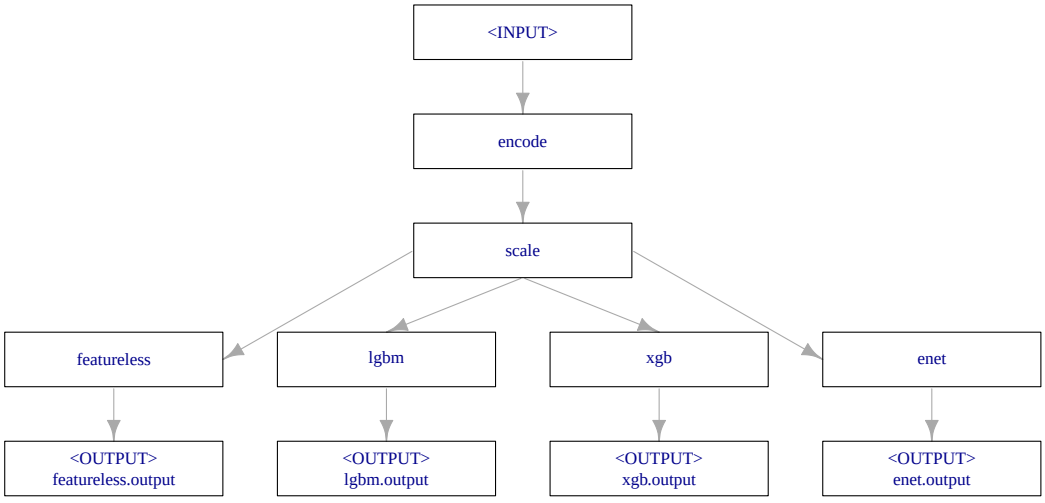

Figure 5: Input and model flow

Parameter search space tweedie

Model	tweedie_power	learning_rate_eta	leaves	iterations	s	alpha
Xgboost	1-1.9	1e-3 - 0.2	16 - 32	200 - 1000		
lgbm	1-1.9	1e-3 - 0.2	3 - 9	200 - 1000		
glmnet					1e-4 - 1	0 - 1

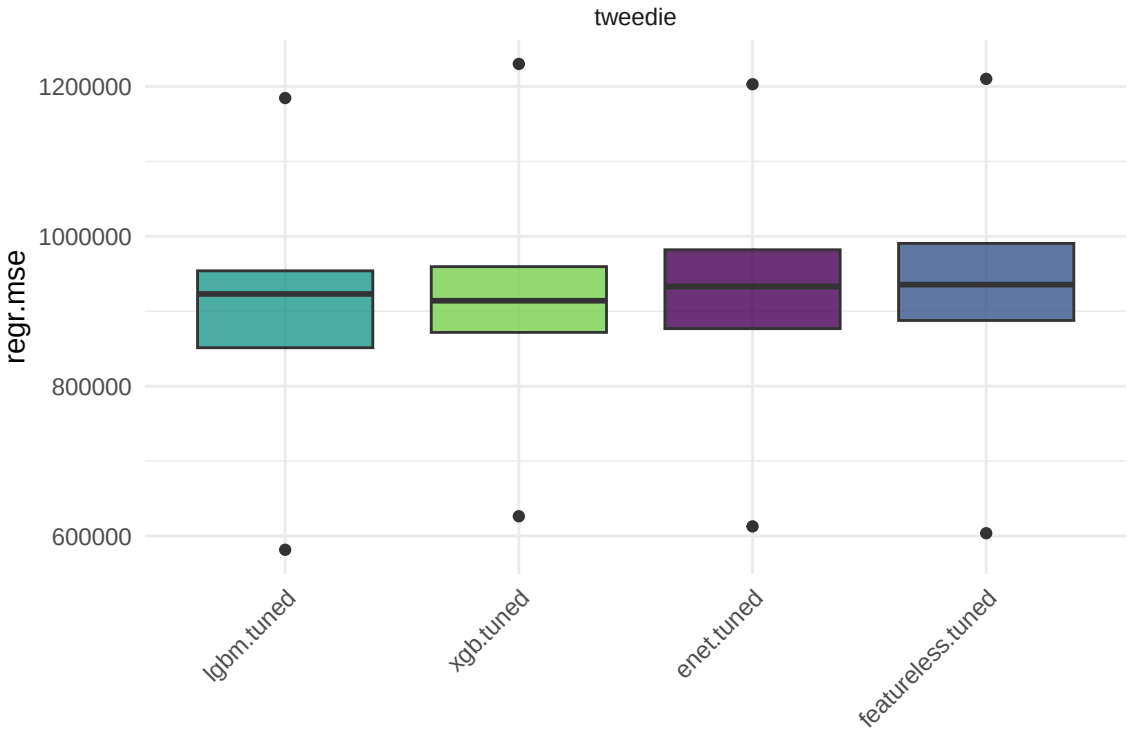

Figure 6: MSE boxplot

Split model Next we can look at the split model proposed in the mathematical frame, where we aim to create two models, and then predict in the following manner:

$$P(N > 0 \mid Z) \cdot E(X \mid Z)$$

This is unlike the traditional and proposed $E(N)E(X)$, but since we are asked to predict the next year, we feel like we can provide a suitable approximation with the above prediction. This entails fitting both a frequency model and a severity model. We use the same ML models for fitting as used to fit the tweedie model, adding random forest to the group of learners. Hence we begin with the frequency regression model.

Next we show the parameter space in table "**Parameter search space frequency**".

The results of the nested cross-validation are shown in figure 8.

We show the best parameters in table "**Best parameters frequency model**".

We have fitted the model with nested cross-validation since it provides a more stable estimate of the generalisation error. Next we look at some model diagnostics. We can look at the combined *bbrier* score for the aggregated model:

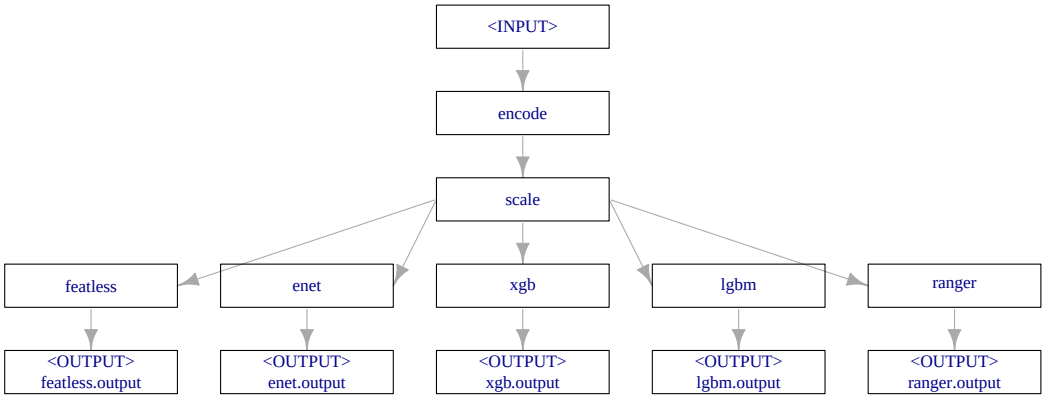

Figure 7: Input and model flow

Best parameters tweedie

parameter	value
encode.method	one-hot
scale.robust	FALSE
regr.lightgbm.objective	tweedie
regr.lightgbm.verbose	-1
regr.lightgbm.learning_rate	0.01438868
regr.lightgbm.num_leaves	28
regr.lightgbm.num_threads	1
regr.lightgbm.tweedie_variance_power	1.729677
regr.lightgbm.num_iterations	796

Parameter search space frequency

Model	learning_rate / eta	leaves / depth	iterations / trees	subsample/ min_nodes	alpha /
XGBoost	0.01 – 0.3	3 – 7 (depth)	200 – 800	0.6 – 1	
LightGBM	1e-3 – 0.2	16 – 32 (leaves)	200 – 1000		
glmnet					0 – 1
Ranger				5	5

```
## classif.bbrier
##     0.09266676
```

which seems quite small.

Which in general show that the accuracy and precision is very high, meaning that we are usually correct when predicting a claim. We see that the True positive ratio is lower than the $PPV = \frac{\text{TP}}{\text{TP+FP}}$, indicating that we miss a fair share of claims. Overall we note that our model seems stable through the cross-validation and that it performs somewhat well.

Here we see the *acc* error based on the choosen probability threshold, which as expected shows that for example a threshold at 0.5 results in a pretty good model. By setting the threshold higher we could risk introducing more false negatives.

Next we look at the severity model. For this model, we choose to log transform the response.

We show the parameter space in table "**Parameter search space severity**".

The results of the nested cross-validation are shown in figure 9.

We show the best parameters in table "**Best parameters severity model**".

Then we can combine the models, and calculate the mean square error for the combined model.

```
## INFO  [18:01:40.282] [bbotk] Starting to optimize 4 parameter(s) with '<OptimizerBatchRandomSearch>'
## INFO  [18:01:40.356] [bbotk] Evaluating 1 configuration(s)
## INFO  [18:01:40.378] [mlr3] Running benchmark with 5 resampling iterations
## INFO  [18:01:40.492] [mlr3] Applying learner 'encode.scale.regr.lightgbm' on task 'twd' (iter 1/5)
## INFO  [18:01:47.097] [mlr3] Applying learner 'encode.scale.regr.lightgbm' on task 'twd' (iter 2/5)
## INFO  [18:01:53.499] [mlr3] Applying learner 'encode.scale.regr.lightgbm' on task 'twd' (iter 3/5)
## INFO  [18:01:59.888] [mlr3] Applying learner 'encode.scale.regr.lightgbm' on task 'twd' (iter 4/5)
```

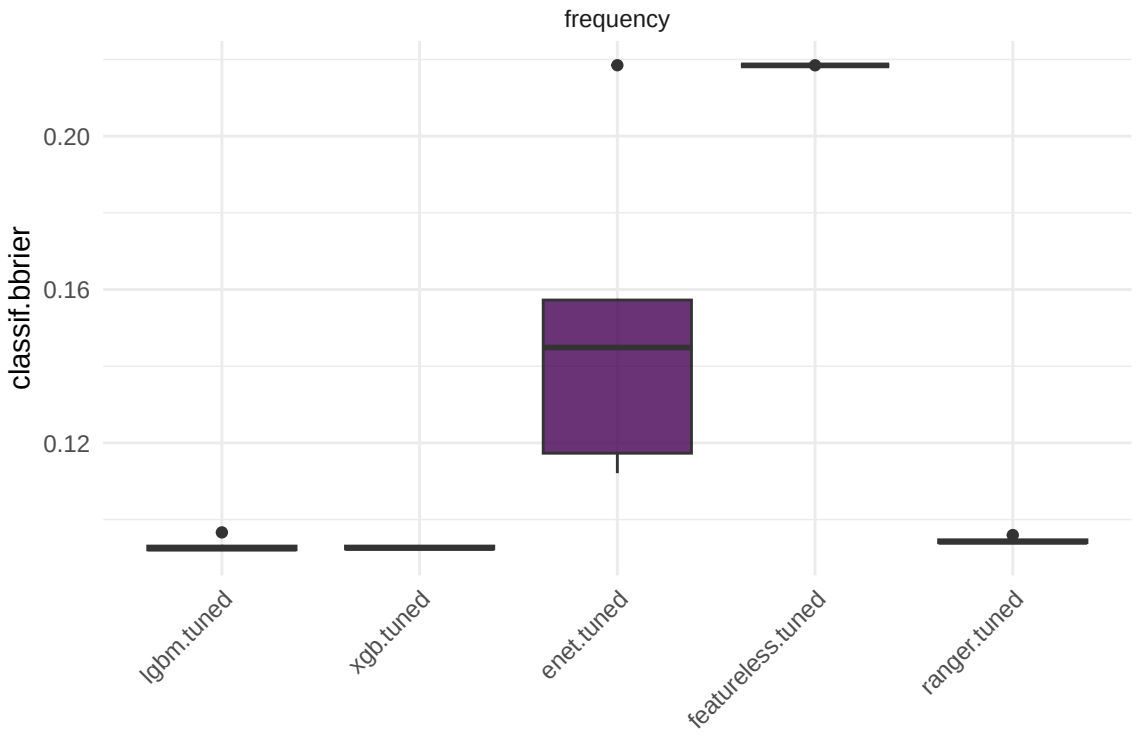

Figure 8: MSE boxplot

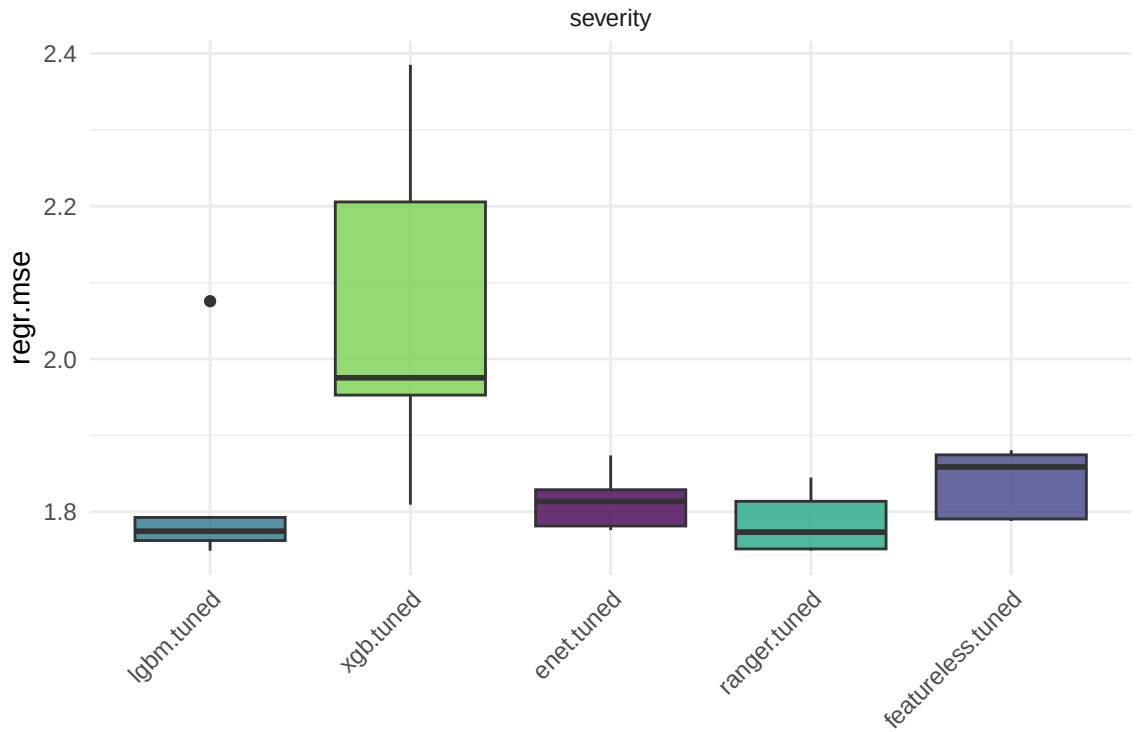

Figure 9: MSE boxplot

Best parameters frequency model

parameter	value
encode.method	one-hot
scale.robust	FALSE
classif.xgboost.eta	0.02677119
classif.xgboost.eval_metric	logloss
classif.xgboost.max_depth	4
classif.xgboost.nrounds	406
classif.xgboost.nthread	1
classif.xgboost.subsample	0.6542777
classif.xgboost.verbose	0

Parameter search space severity

Model	learning_rate / eta	leaves / depth	iterations / trees	subsample/ min_nodes	alpha /
XGBoost	1e-3 – 0.2	3 – 9 (depth)	200 – 1000		
LightGBM	1e-3 – 0.2	16 – 32 (leaves)	200 – 1000		
glmnet					0 – 1
Ranger				5	5

```
## INFO  [18:02:06.800] [mlr3] Applying learner 'encode.scale.regr.lightgbm' on task 'twd' (iter 5/5)
## INFO  [18:02:12.777] [mlr3] Finished benchmark
## INFO  [18:02:12.825] [bbotk] Result of batch 1:
## INFO  [18:02:12.829] [bbotk]  regr.lightgbm.learning_rate regr.lightgbm.num_leaves
## INFO  [18:02:12.829] [bbotk]                    -2.019363                      30
## INFO  [18:02:12.829] [bbotk]  regr.lightgbm.tweedie_variance_power regr.lightgbm.num_iterations regr
## INFO  [18:02:12.829] [bbotk]                                1.432426                          943 9105
## INFO  [18:02:12.829] [bbotk]  warnings errors runtime_learners                                uhash
## INFO  [18:02:12.829] [bbotk]         0      0            32.16 975a98ce-31aa-4a2b-a41c-66fa0efb8e23
## INFO  [18:02:12.865] [bbotk] Finished optimizing after 1 evaluation(s)
## INFO  [18:02:12.866] [bbotk] Result:
## INFO  [18:02:12.868] [bbotk]  regr.lightgbm.learning_rate regr.lightgbm.num_leaves
## INFO  [18:02:12.868] [bbotk]                          <num>                   <int>
## INFO  [18:02:12.868] [bbotk]                    -2.019363                      30
## INFO  [18:02:12.868] [bbotk]  regr.lightgbm.tweedie_variance_power regr.lightgbm.num_iterations
## INFO  [18:02:12.868] [bbotk]                                  <num>                         <int>
## INFO  [18:02:12.868] [bbotk]                                1.432426                           943
## INFO  [18:02:12.868] [bbotk]  learner_param_vals  x_domain regr.mse
## INFO  [18:02:12.868] [bbotk]             <list>    <list>    <num>
## INFO  [18:02:12.868] [bbotk]          <list[9]> <list[4]> 910547.9
## INFO  [18:02:21.373] [bbotk] Starting to optimize 4 parameter(s) with '<OptimizerBatchRandomSearch>'
## INFO  [18:02:21.428] [bbotk] Evaluating 1 configuration(s)
## INFO  [18:02:21.438] [mlr3] Running benchmark with 5 resampling iterations
## INFO  [18:02:21.444] [mlr3] Applying learner 'encode.scale.classif.xgboost' on task 'frek_binary' (i
## INFO  [18:03:12.671] [mlr3] Applying learner 'encode.scale.classif.xgboost' on task 'frek_binary' (i
## INFO  [18:04:03.116] [mlr3] Applying learner 'encode.scale.classif.xgboost' on task 'frek_binary' (i
## INFO  [18:04:55.299] [mlr3] Applying learner 'encode.scale.classif.xgboost' on task 'frek_binary' (i
## INFO  [18:05:47.621] [mlr3] Applying learner 'encode.scale.classif.xgboost' on task 'frek_binary' (i
```

Best parameters severity model

parameter	value
encode.method	one-hot
scale.robust	FALSE
regr.ranger.importance	permutation
regr.ranger.min.node.size	5
regr.ranger.mtry	5
regr.ranger.num.threads	1

```
## INFO  [18:06:40.367] [mlr3] Finished benchmark
## INFO  [18:06:40.420] [bbotk] Result of batch 1:
## INFO  [18:06:40.423] [bbotk]  classif.xgboost.eta classif.xgboost.max_depth classif.xgboost.nrounds
## INFO  [18:06:40.423] [bbotk]            0.1423624                         7                     633
## INFO  [18:06:40.423] [bbotk]  classif.xgboost.subsample classif.bbrier warnings errors runtime_learne
## INFO  [18:06:40.423] [bbotk]                  0.7413903      0.1052119        0      0           258
## INFO  [18:06:40.423] [bbotk]                                     uhash
## INFO  [18:06:40.423] [bbotk]  42c635b1-4143-44d0-a293-d0ed9bec5596
## INFO  [18:06:40.477] [bbotk] Finished optimizing after 1 evaluation(s)
## INFO  [18:06:40.478] [bbotk] Result:
## INFO  [18:06:40.481] [bbotk]  classif.xgboost.eta classif.xgboost.max_depth classif.xgboost.nrounds
## INFO  [18:06:40.481] [bbotk]                <num>                     <int>                   <int>
## INFO  [18:06:40.481] [bbotk]            0.1423624                         7                     633
## INFO  [18:06:40.481] [bbotk]  classif.xgboost.subsample learner_param_vals  x_domain classif.bbrier
## INFO  [18:06:40.481] [bbotk]                      <num>            <list>    <list>          <num>
## INFO  [18:06:40.481] [bbotk]                  0.7413903         <list[9]> <list[4]>      0.1052119
## INFO  [18:07:43.569] [bbotk] Starting to optimize 0 parameter(s) with '<OptimizerBatchRandomSearch>'
## INFO  [18:07:43.576] [bbotk] Evaluating 1 configuration(s)
## INFO  [18:07:43.582] [mlr3] Running benchmark with 5 resampling iterations
## INFO  [18:07:43.588] [mlr3] Applying learner 'encode.scale.regr.ranger' on task 'skade' (iter 1/5)
## INFO  [18:07:55.989] [mlr3] Applying learner 'encode.scale.regr.ranger' on task 'skade' (iter 2/5)
## INFO  [18:08:07.703] [mlr3] Applying learner 'encode.scale.regr.ranger' on task 'skade' (iter 3/5)
## INFO  [18:08:21.313] [mlr3] Applying learner 'encode.scale.regr.ranger' on task 'skade' (iter 4/5)
## INFO  [18:08:34.206] [mlr3] Applying learner 'encode.scale.regr.ranger' on task 'skade' (iter 5/5)
## INFO  [18:08:47.972] [mlr3] Finished benchmark
## INFO  [18:08:48.012] [bbotk] Result of batch 1:
## INFO  [18:08:48.014] [bbotk]  regr.mse warnings errors runtime_learners
## INFO  [18:08:48.014] [bbotk]  1.793924        0      0            64.31 18f7b9ea-0daf-45cf-a401-0e06
## INFO  [18:08:48.032] [bbotk] Finished optimizing after 1 evaluation(s)
## INFO  [18:08:48.033] [bbotk] Result:
## INFO  [18:08:48.034] [bbotk]  learner_param_vals  x_domain regr.mse
## INFO  [18:08:48.034] [bbotk]              <list>    <list>    <num>
## INFO  [18:08:48.034] [bbotk]           <list[6]> <list[0]> 1.793924
## [1] "mse_tweedie"
## [1] 967218.4
## [1] "mse_combined_model"
## [1] 968942
## [1] "mse_baseline"
## [1] 981095.1
```

We see here that our final model for the combined model is barely better than the baseline model predicting

just the mean of the response. It is clear that both our models (tweedie and frequency/severity model) perform extremely poorly, and one should probably have split up the claims into small and large claims so we could model the large and small claims independently. Since both models are somewhat lacking we stick with the split model, since this is more common in the actuarial business.

Discussion

Here we touch upon the various modeling problems. ##### Preprocessing We took a somewhat minimalist approach: obvious date parsing, encoding etc. some slight imputation. Since we impute, we impute off random values which adds noise. We could have used better imputation, and our data aggregation could have been slightly better.

Model choices In the frequency model, despite the good accuracy, we miss a lot of false negatives. In the severity model, we severely miss the tail of the distribution. Instead of splitting into two models, we could have made splitted into three models and used the variable in order to approximate:

$E(\frac{L}{E}) = E(F \mid Y > 0) \cdot E\left(Y \mid Y > 0\right) \cdot P(Y > 0)$

Where let $\frac{L}{E}$, F is the Cost claims year.

Also, we could have used the variable *R_claims_history* directly as a response variable in the frequency model, instead of creating our own binary claim indicator response variable.

Evaluation protocol We did nested cross-validation, but we could have increased the iterations and the number of folds for more stable results. For severity we could have chosen a better metric to optimize for, which will hopefully result in a better model, especially in the tail. We could have fit an EVT distribution on the heavy tailed data, but again this is quite strenious for this assignment.