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. It is however very difficult to locate these policies, and we overlook this issue.

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 ...
```

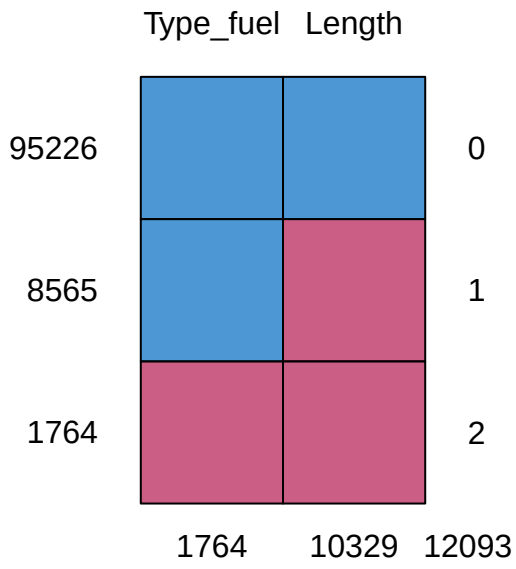
```
##  $ Power               : int  80 80 80 80 80 80 85 85 85 85 ...
##  $ Cylinder_capacity   : int  599 599 599 599 599 599 1229 1229 1229 1229 ...
##  $ Value_vehicle       : num  7068 7068 7068 7068 7068 ...
##  $ N_doors             : int  0 0 0 0 0 0 5 5 5 5 ...
##  $ Type_fuel           : chr  "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

$$\int_Y y \ \kappa(x, dy)$$

where κ is the appropriate probability kernel, and we let y be our response. Due to simplicity this is not performed.

Next there are some missing values.




```
##       Type_fuel Length
## 95226         1      1     0
## 8565          1      0     1
## 1764          0      0     2
##            1764  10329 12093
```

It becomes apparent that there are missing values in the *length* and *type_fuel* variable. We can see that the missingness is overlapping in 1764 rows, but since the total missingness is substantial for the Length variable, we choose to impute these. We impute by drawing realizations from the empirical law of the *length* variable. We ignore type fuel since it is char.

Finally we look at the correlation between the covariates.

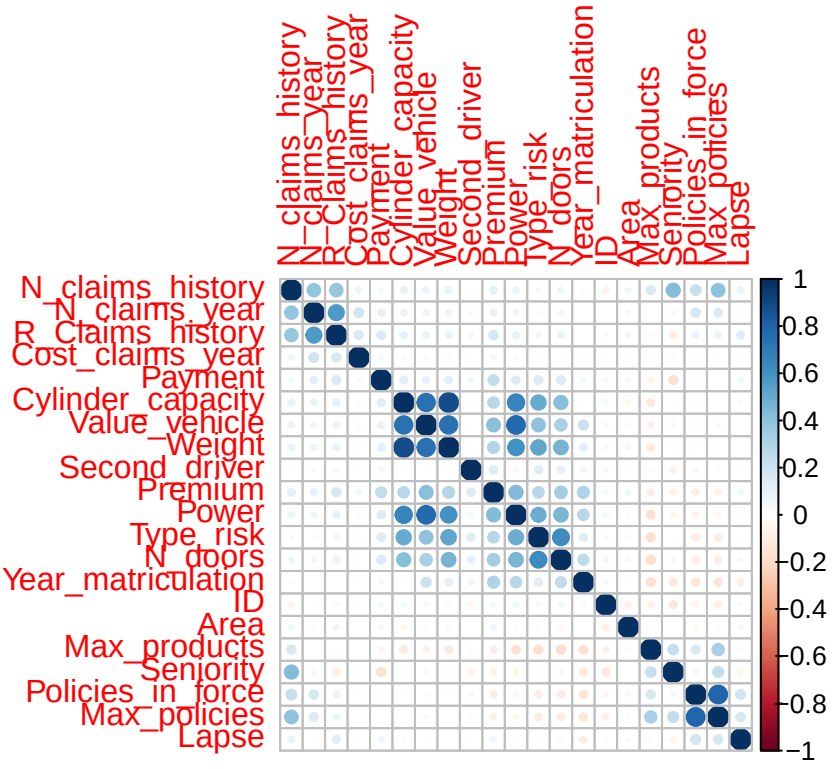


Figure 1: Correlation plot between the continious covariates

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 to have the ability to assume independence, which we have if we aggregate the rows into being one policy. 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

$$P\big(Z_i \in A, Z_k \in B\big) = P\big(Z_i \in A\big)\, P\big(Z_k \in B\big), \quad A, B \in \mathcal{B}(\mathcal{X} \times \mathcal{Y}).$$

However this is an empirical assumption which can be violated by catastrophe events.

The aggregating is done by formatting the date covariates such that we can calculate the contract exposure, where we note that one year means $E = 1$, and then sum over the entire *ID*. Futher we take the premium, for the second to last entry, since we dont want to have forward-leakage in our data. The same is done for *cost_claims_history*, and related rows. We apply a mix of sum and max functions on the remaining rows to ensure that we aggregate data.

Introduction into the mathematical framework

Here we desire some mathematics before we proceed. It could be desireable to look at a classical insurance related result

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

Further we define the Tweedie law as

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

since it can accommodate a zero-inflated distribution very well. 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

Tweedie Initially we want to fit a Tweedie model on the entire data, since the Tweedie model is known for handeling zero-inflated distributions well. We model with a gradient boosting machine, since we have a couple of parameters and the *lightgbm* provides a solid bias-variance trade-off. The additive structure of the gradient boosting is preferable over the random forest. We minimize the mse, and find that the optimal variance power through cross-validation is approximately 1.82. This makes the tweedie a compound Poisson-Gamma. This optimal model is found with 19 leaves.

The random search is not optimal, and one could have used a better optimization algorithm. We can look at the model fit, on the whole dataset.

This fit is somewhat disappointing since we severely underestimate the larger claims. It is also evident that our data is very heavy tailed, and the tweedie does not fully capture the more extreme events.

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:

$$1\{P(N > 0 \mid Z) > t\} \cdot E(X \mid Z)$$

Which 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 classification model and a severity model. Hence we begin with the frequency classification model. Again we use a boosting machine. More specifically an XGBoost classifier based on the Brier score. We note that we introduce offset on the exposure, so we can weights accordingly to the insurance period. We note that

We have fitted the model with nested cross-validation since it provides a more stable estimate of the generalisation error. We use $K = 2$ folds in both the inner and outer loops. Next we look at some model diagnostics. We can look at the combined *bbrier* score for the aggregated model:

```
## classif.bbrier
##     0.06244365
```

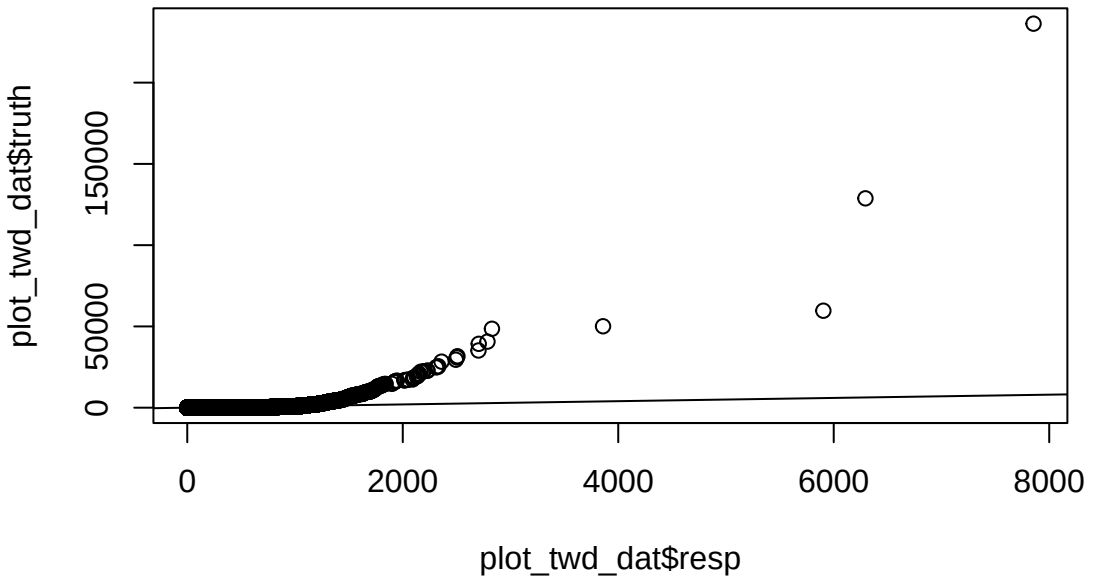


Figure 2: Residual plot, histogram for residuals, QQ-plot

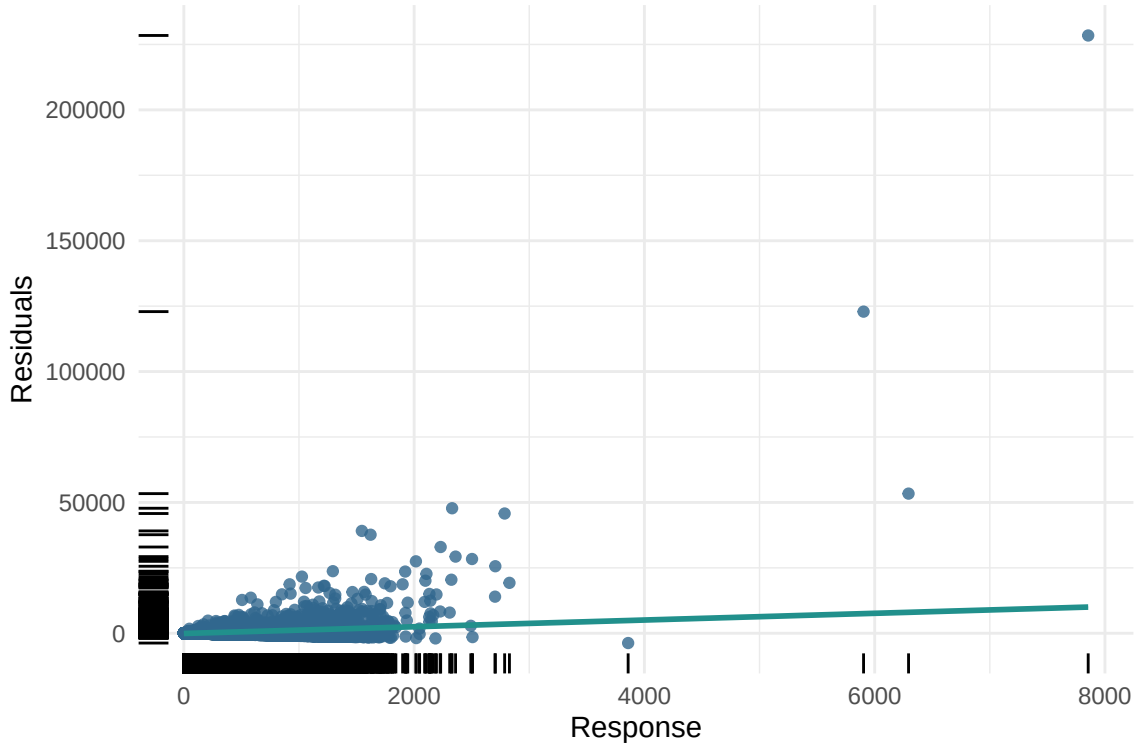


Figure 3: Residual plot, histogram for residuals, QQ-plot

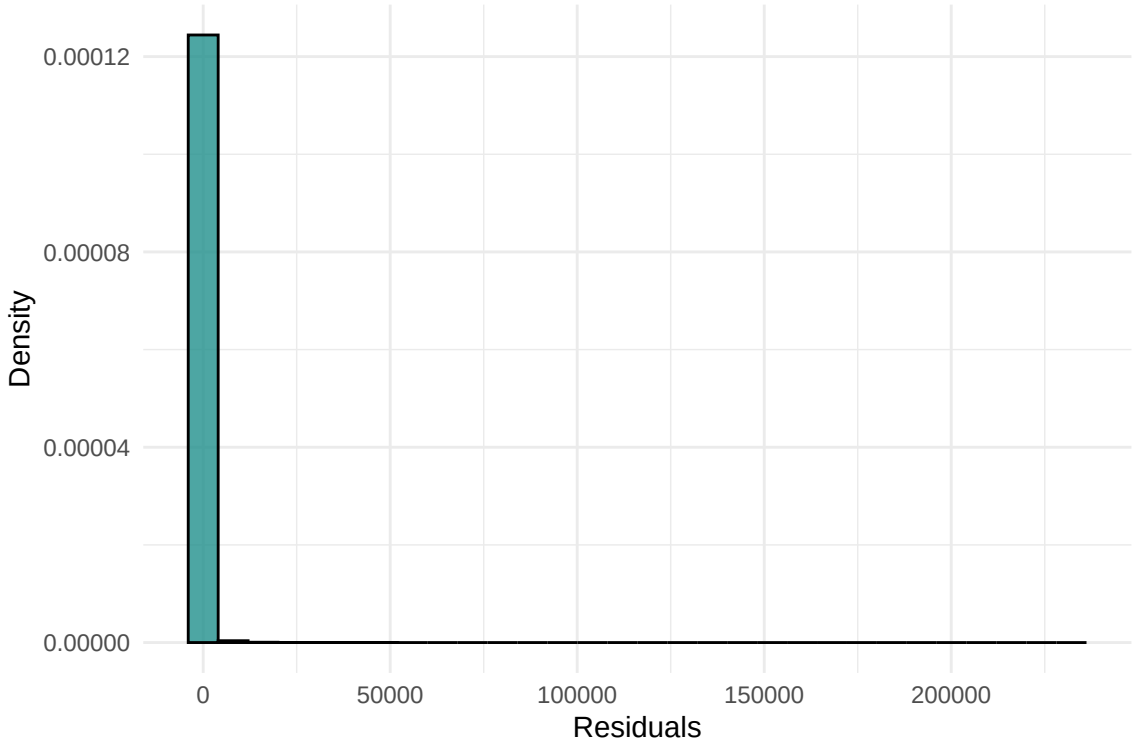


Figure 4: Residual plot, histogram for residuals, QQ-plot

which seems quite small. Further we have some confusion matrices for each of the folds.

```
## Fold:  1        truth
## response     1     0
##        1  5668     0
##        0  2156 18499
## acc :  0.9181; ce  :  0.0819; dor :  NaN; f1  :  0.8402
## fdr :  0.0000; fnr :  0.2756; fomr:  0.1044; fpr :  0.0000
## mcc :  0.8055; npv :  0.8956; ppv :  1.0000; tnr :  1.0000
## tpr :  0.7244
## Fold:  2        truth
## response     1     0
##        1  5751   321
##        0  1980 18270
## acc :  0.9126; ce  :  0.0874; dor :  165.3148; f1  :  0.8333
## fdr :  0.0529; fnr :  0.2561; fomr:  0.0978; fpr :  0.0173
## mcc :  0.7856; npv :  0.9022; ppv :  0.9471; tnr :  0.9827
## tpr :  0.7439
```

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.

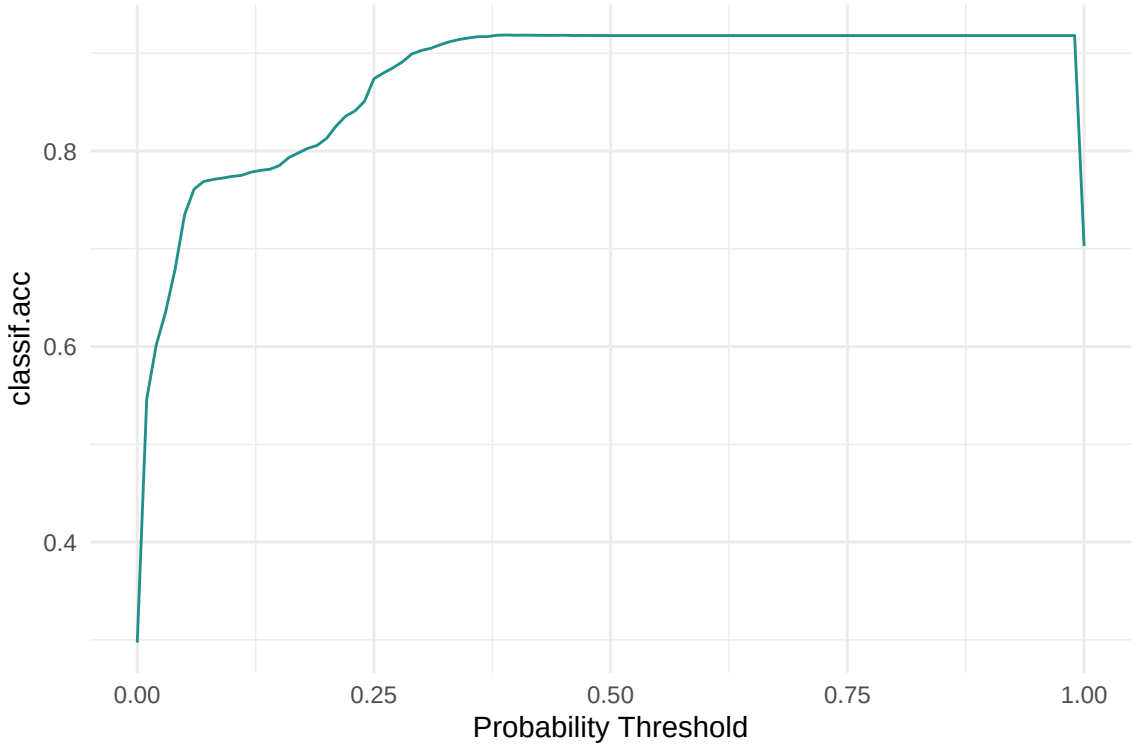


Figure 5: the bbrier error for each choice of probability threshold in classification

Next we look at the severity model. For severity we use LightGBM for quantile regression with $\alpha = 0.85$. We do 5-fold cross-validation, optimizing for mean squared error (regr.mse) and limited to 5 evaluations.

Then we can combine the models, and calculate the mean square error for the combined model. We have shown that our models are somewhat acceptable, and we do not fit them on the entire dataset and find the out of sample *MSE* for 20 percent of data.

```
## INFO  [21:13:00.743] [bbotk] Starting to optimize 4 parameter(s) with '<OptimizerBatchRandomSearch>'
## INFO  [21:13:00.840] [bbotk] Evaluating 1 configuration(s)
## INFO  [21:13:00.852] [mlr3] Running benchmark with 2 resampling iterations
## INFO  [21:13:00.861] [mlr3] Applying learner 'encode.scale.classif.xgboost' on task 'frek_binary' (i
## INFO  [21:13:21.751] [mlr3] Applying learner 'encode.scale.classif.xgboost' on task 'frek_binary' (i
## INFO  [21:13:42.901] [mlr3] Finished benchmark
## INFO  [21:13:42.960] [bbotk] Result of batch 1:
## INFO  [21:13:42.964] [bbotk]  classif.xgboost.eta classif.xgboost.max_depth classif.xgboost.nrounds
## INFO  [21:13:42.964] [bbotk]           0.2805049                         7                     417
## INFO  [21:13:42.964] [bbotk]  classif.xgboost.subsample classif.bbrier warnings errors runtime_learne
## INFO  [21:13:42.964] [bbotk]                  0.8720876     0.08843578        0      0            41
## INFO  [21:13:42.964] [bbotk]                                  uhash
## INFO  [21:13:42.964] [bbotk]  7ee4ca40-22eb-4d4e-a07b-7ffa49c583e0
## INFO  [21:13:42.994] [bbotk] Evaluating 1 configuration(s)
## INFO  [21:13:43.004] [mlr3] Running benchmark with 2 resampling iterations
## INFO  [21:13:43.014] [mlr3] Applying learner 'encode.scale.classif.xgboost' on task 'frek_binary' (i
## INFO  [21:14:00.029] [mlr3] Applying learner 'encode.scale.classif.xgboost' on task 'frek_binary' (i
## INFO  [21:14:17.094] [mlr3] Finished benchmark
## INFO  [21:14:17.162] [bbotk] Result of batch 2:
## INFO  [21:14:17.167] [bbotk]  classif.xgboost.eta classif.xgboost.max_depth classif.xgboost.nrounds
```

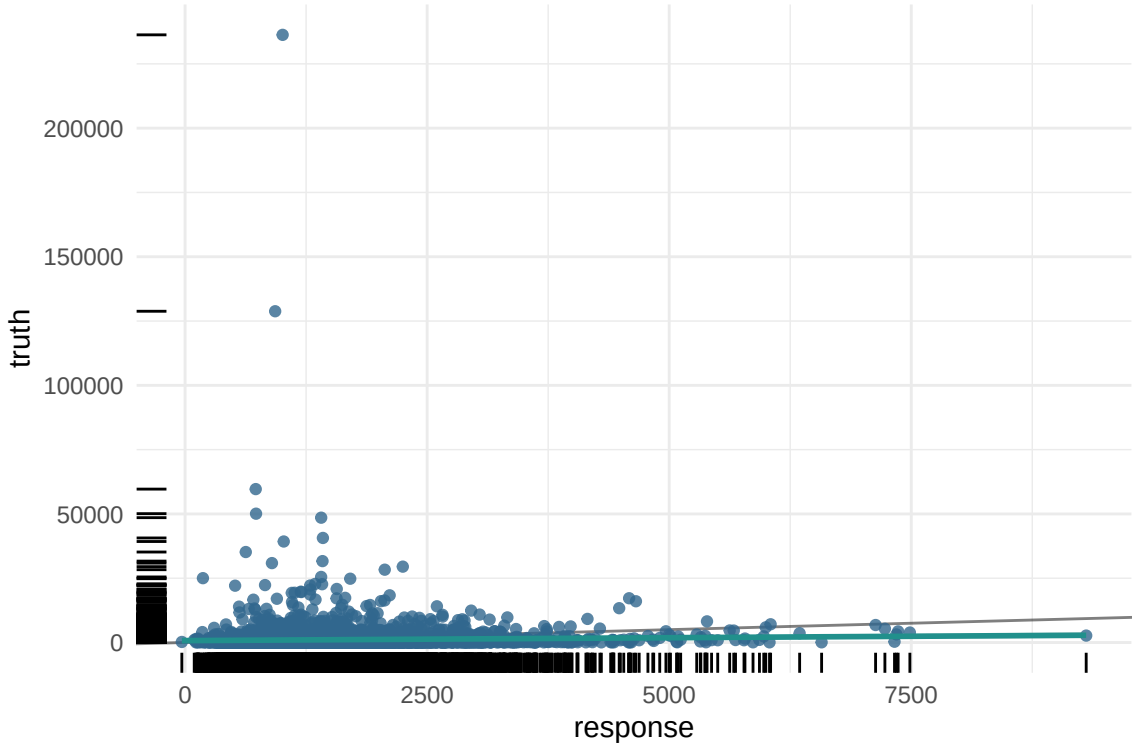


Figure 6: Shows the predicted values along the x-axis vs the real values on the y-axis

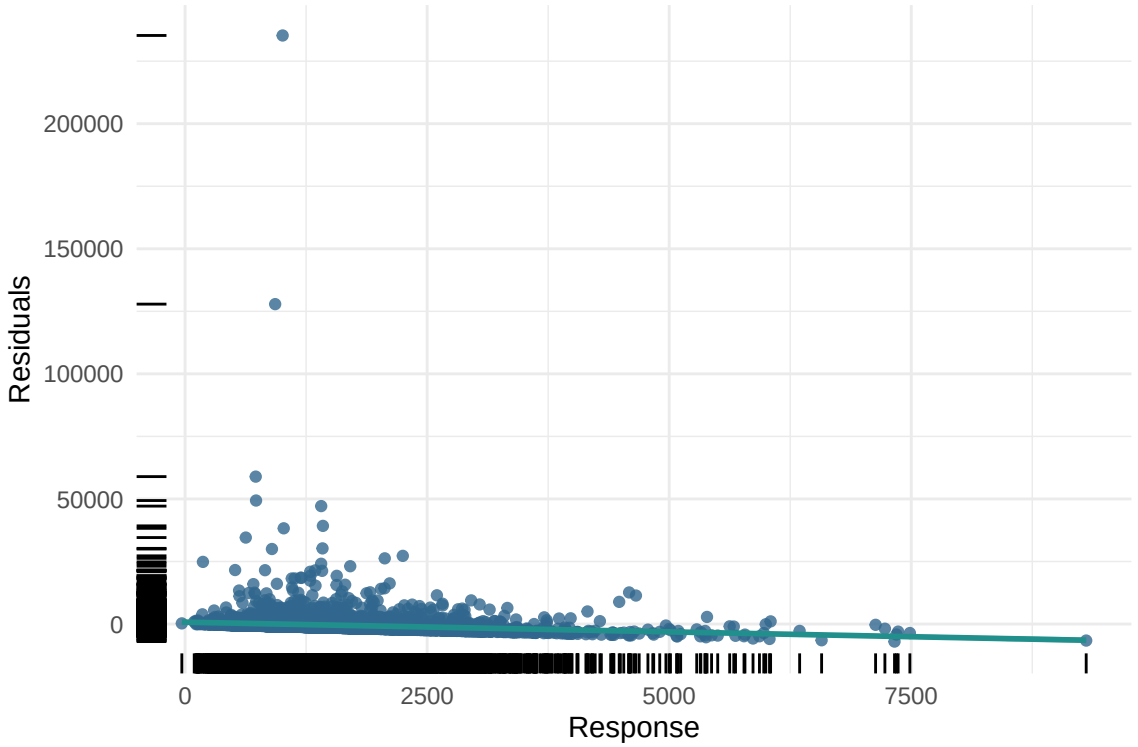


Figure 7: Shows the predicted values along the x-axis vs the real values on the y-axis

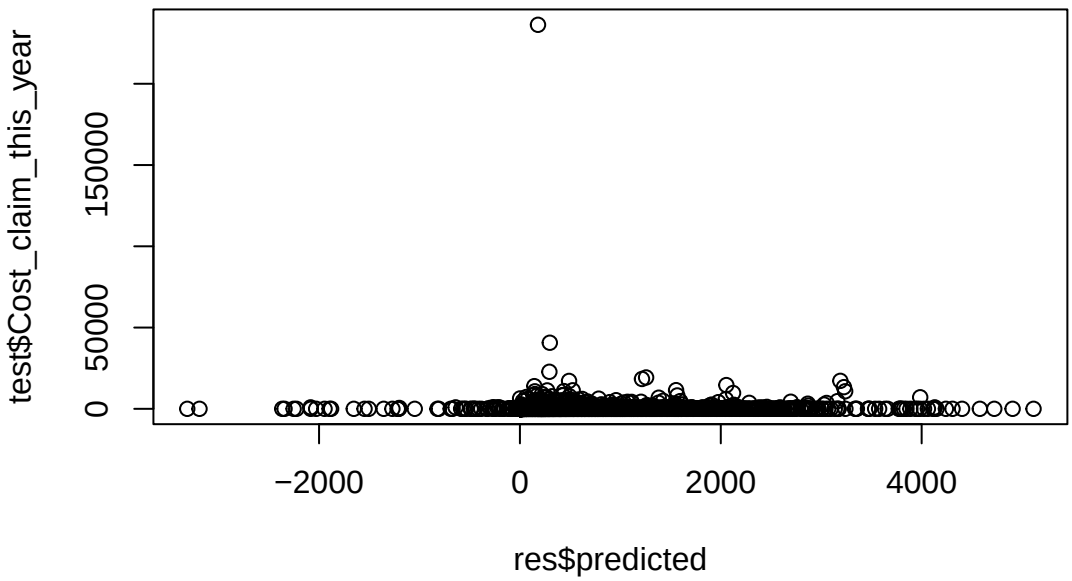


Figure 8: Predicted vs. actual

```
## INFO  [21:14:17.167] [bbotk]            0.1954272                         3                    745
## INFO  [21:14:17.167] [bbotk]  classif.xgboost.subsample classif.bbrier warnings errors runtime_learne
## INFO  [21:14:17.167] [bbotk]                  0.7626313     0.07765571        0      0            34
## INFO  [21:14:17.167] [bbotk]                                    uhash
## INFO  [21:14:17.167] [bbotk]  99d8c52b-baab-41e1-bd1a-8699d9f208fa
## INFO  [21:14:17.214] [bbotk] Finished optimizing after 2 evaluation(s)
## INFO  [21:14:17.215] [bbotk] Result:
## INFO  [21:14:17.218] [bbotk]  classif.xgboost.eta classif.xgboost.max_depth classif.xgboost.nrounds
## INFO  [21:14:17.218] [bbotk]               <num>                     <int>                   <int>
## INFO  [21:14:17.218] [bbotk]           0.1954272                         3                     745
## INFO  [21:14:17.218] [bbotk]  classif.xgboost.subsample learner_param_vals  x_domain classif.bbrier
## INFO  [21:14:17.218] [bbotk]                      <num>             <list>    <list>          <num>
## INFO  [21:14:17.218] [bbotk]                  0.7626313          <list[9]> <list[4]>     0.07765571
## INFO  [21:14:50.697] [bbotk] Starting to optimize 0 parameter(s) with '<OptimizerBatchRandomSearch>'
## INFO  [21:14:50.702] [bbotk] Evaluating 1 configuration(s)
## INFO  [21:14:50.711] [mlr3] Running benchmark with 2 resampling iterations
## INFO  [21:14:50.718] [mlr3] Applying learner 'imputeoor.encode.scale.regr.lightgbm' on task 'skade'
## INFO  [21:14:51.653] [mlr3] Applying learner 'imputeoor.encode.scale.regr.lightgbm' on task 'skade'
## INFO  [21:14:52.559] [mlr3] Finished benchmark
## INFO  [21:14:52.616] [bbotk] Result of batch 1:
## INFO  [21:14:52.621] [bbotk]  regr.mse warnings errors runtime_learners
## INFO  [21:14:52.621] [bbotk]  10605105        0      0             1.78 16ff5cfb-395f-4527-a031-1b10
## INFO  [21:14:52.626] [bbotk] Evaluating 1 configuration(s)
## INFO  [21:14:52.635] [mlr3] Running benchmark with 2 resampling iterations
## INFO  [21:14:52.644] [mlr3] Applying learner 'imputeoor.encode.scale.regr.lightgbm' on task 'skade'
## INFO  [21:14:53.707] [mlr3] Applying learner 'imputeoor.encode.scale.regr.lightgbm' on task 'skade'
```

```
## INFO  [21:14:54.602] [mlr3] Finished benchmark
## INFO  [21:14:54.747] [bbotk] Result of batch 2:
## INFO  [21:14:54.750] [bbotk]  regr.mse warnings errors runtime_learners
## INFO  [21:14:54.750] [bbotk]  10605105        0      0             1.92 4349312b-ffd4-4d16-9763-51c4
## INFO  [21:14:54.780] [bbotk] Finished optimizing after 2 evaluation(s)
## INFO  [21:14:54.781] [bbotk] Result:
## INFO  [21:14:54.785] [bbotk]  learner_param_vals  x_domain regr.mse
## INFO  [21:14:54.785] [bbotk]              <list>    <list>    <num>
## INFO  [21:14:54.785] [bbotk]           <list[9]> <list[0]> 10605105
## [1] 6338512
```

We see here that our final model for the combined model. It is clear that both our models 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 severly miss the tail of the distribution.

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.

```
rmd <- knitr::opts_knit$get("input.file") %||% knitr::current_input()
stopifnot(!is.null(rmd) && nzchar(rmd))
tmp <- knitr::purl(input = rmd,
                   output = tempfile(fileext = ".R"),
                   documentation = 0, quiet = TRUE)
code_lines <- readLines(tmp, warn = FALSE, encoding = "UTF-8")
cat("```r\n", paste(code_lines, collapse = "\n"), "\n```")
```

```
## ```r
##
## knitr::opts_chunk$set(
##   echo   = TRUE,
##   message = FALSE,
##   warning = FALSE,
##   fig.align = "center",
##   dpi = 300,
##   fig.width = 6, fig.height = 4
## )
##
##   train_severity <- function(data, folds, n_evals){
##     data <- data %>% dplyr::select(-claim_indicator)
##     data <- data %>% dplyr::filter(Cost_claim_this_year > 0)
##
```

```
##     task_S = as_task_regr(
##       data,
##       target = "Cost_claim_this_year",
##       weights = data$Exposure,
##       id     = "skade"
##     )
##
##     task_S$set_col_roles("Exposure", "weight")
##
##     graph_S <- po("imputeoor") %>>%
##       po("encode")    %>>%
##       po("scale")     %>>%
##       po("learner",
##          lrn("regr.lightgbm",
##              objective       = "quantile",
##              alpha           = 0.85
##          ))
##
##
##     glrn_S = GraphLearner$new(graph_S)
##
##     at_severity = AutoTuner$new(
##       learner    = glrn_S,
##       resampling = rsmp("cv", folds = folds),
##       measure    = msr("regr.mse"),
##       tuner      = tnr("random_search"),
##       terminator = trm("evals", n_evals = n_evals)
##     )
##
##
##     at_severity <- at_severity$train(task_S)
##     return(at_severity)
##   }
##
##   train_freq <- function(data, folds, n_evals){
##
##     data <- data %>% dplyr::select(-Cost_claim_this_year)
##
##     task_freq = as_task_classif(
##       data,
##       target = "claim_indicator",
##       positive = "1",
##       weights = data$Exposure,
##       id     = "frek_binary"
##     )
##   task_freq$set_col_roles("Exposure", "weight")
##
##   graph_freq  = po("encode") %>>%
##     po("scale")  %>>%
##     lrn("classif.xgboost",
##         predict_type = "prob",
##         eval_metric  = "logloss", #Bedre loss funktion?
##         nrounds      = to_tune(200, 800),
##         max_depth    = to_tune(3, 7),
```

```
##         eta          = to_tune(0.01, 0.3),
##         subsample    = to_tune(0.6, 1))
##
##   at_freq = auto_tuner(
##     learner    = as_learner(graph_freq),
##     resampling = rsmp("cv", folds = folds), #ÆNDRET 5 -> 2
##     measure    = msr("classif.bbrier"),
##     tuner      = tnr("random_search"),
##     term_evals = n_evals #Ændret 5 til 1
##   )
##   model_freq <- at_freq$train(task_freq)
##     return(model_freq)
##   }
##   data_trans <- function(data){
##
##     today <- Sys.Date()
##
##     date_cols <- c("Date_start_contract",
##                    "Date_last_renewal",
##                    "Date_next_renewal",
##                    "Date_lapse")
##     data[, (date_cols) := lapply(.SD, as.IDate, format = "%d/%m/%Y"),
##          .SDcols = date_cols]
##     data[, Period_end :=
##            fifelse(
##              !is.na(Date_lapse) &
##                Date_lapse >= Date_last_renewal &
##                Date_lapse <= Date_next_renewal,
##              Date_lapse,
##              pmin(Date_next_renewal, today, na.rm = TRUE)
##            )]
##
##
##     data[, Exposure_days  := as.numeric(Period_end - Date_last_renewal)]
##     data[, Exposure_days  := pmax(Exposure_days, 0)]
##     data[, Exposure_unit  := fcase(
##       between(Exposure_days, 364, 367), 1,
##       default = Exposure_days/365.25
##     )]
##
##
##     #data[, Cost_claims_sum := sum(Cost_claims_year), by = ID]
##
##     # Add a column that flags the latest row per group
##
##     setorder(data, ID, -Date_last_renewal)
##     data[, is_second_latest := FALSE]
##     data[, is_second_latest := .I == .I[2], by = ID]
##     # data$Cost_claims_year <- NA
##
##     key_vars <- c("ID","Year_matriculation","Power",
##                   "Cylinder_capacity","N_doors","Type_fuel","Weight")
##
##     agg <- data[, .(
```

```
##       Exposure               = sum(Exposure_unit),
##       Max_policies_max       = max(Max_policies),
##       Policies_in_force_second_latest = Policies_in_force[is_second_latest],
##
##       Max_products_max       = max(Max_products),
##       Payment_max            = max(Payment),
##       Premium_second_latest = Premium[is_second_latest],
##       Cost_claim_this_year = Cost_claims_year[which.max(Date_last_renewal)],
##       Cost_claims_sum_history = sum(Cost_claims_year) -(Cost_claims_year[which.max(Date_last_renewal
##       R_claims_history   = max(N_claims_history - N_claims_year)/sum(Exposure_unit),
##       # N_claims_year_sum      = sum(N_claims_year),
##       N_claims_history   = max(N_claims_history - N_claims_year),
##       # R_claims_history_sum   = sum(R_Claims_history),
##       Type_risk_max          = max(Type_risk),
##       Value_vehicle_mean     = mean(Value_vehicle),
##       N_doors_mean           = mean(N_doors),
##       Length_sum             = sum(Length, na.rm = TRUE)
##     ), by = key_vars]
##
##
##
##
##
##     agg <- agg %>%
##       dplyr::mutate(claim_indicator = dplyr::if_else(Cost_claims_sum_history + Cost_claim_this_year >
##
##     tmp <- agg %>%
##       dplyr::filter(is.na(Type_fuel))
##
##     NA_fuel_Y <- sum(tmp$Cost_claims_sum_history + tmp$Cost_claims_this_year)
##     total_Y <- sum(agg$Cost_claims_sum_history + agg$Cost_claims_this_year)
##
##     na_Y_ratio <- NA_fuel_Y / total_Y
##
##     agg <- agg %>%
##       dplyr::filter(!is.na(Type_fuel))
##
##     agg <- agg %>%
##       dplyr::mutate(Type_fuel = dplyr::if_else(Type_fuel == "P", 1, 0))
##     return(agg)
##   }
##
##
##
## library(rsample)
## library(skimr)
## library(patchwork)
## library(mlr3viz)
## library(GGally)
## library(tweedie)
## library(mlr3extralearners)
## library(Hmisc)
## library(mice)
## library(mlr3)
```

```
## library(mlr3learners )
## library(mlr3tuning)
## library(mlr3mbo)
## library(glmnet)
## library(OpenML)
## library(mlr3pipelines)
## library(future)
## library(magrittr)
## library(data.table)
## library(tidyverse)
## library(corrr)
## library(corrplot)
## library(mlr3verse)
##
##
## data <- fread("Motor vehicle insurance data.csv", sep = ";")
## print(colnames(data))
##
##
## print(str(data))
##
##
##
##
##
## data_with_na <- data %>%
##   dplyr::select(dplyr::where(~base::anyNA(.x)))
##
##
##
##
## mice::md.pattern(data_with_na, plot = TRUE)
##
##
## data <- data %>%
##   dplyr::mutate(Length = Hmisc::impute(Length, fun = "random"))
##
##
## num_data <- data %>%
##   dplyr::select(-Date_start_contract,
##                 -Date_last_renewal,
##                 -Date_next_renewal,
##                 -Date_birth,
##                 -Distribution_channel,
##                 -Date_lapse,
##                 -Date_driving_licence,
##                 -Type_fuel,
##                 -Length)
## M <- stats::cor(num_data)
## corrplot::corrplot(M, order = 'AOE')
##
##
##
## data_trans <- function(data){
```

```
##
##   today <- Sys.Date()
##
##   date_cols <- c("Date_start_contract",
##                  "Date_last_renewal",
##                  "Date_next_renewal",
##                  "Date_lapse")
##   data[, (date_cols) := lapply(.SD, as.IDate, format = "%d/%m/%Y"),
##        .SDcols = date_cols]
##   data[, Period_end :=
##          fifelse(
##            !is.na(Date_lapse) &
##              Date_lapse >= Date_last_renewal &
##              Date_lapse <= Date_next_renewal,
##            Date_lapse,
##            pmin(Date_next_renewal, today, na.rm = TRUE)
##          )]
##
##
##   data[, Exposure_days  := as.numeric(Period_end - Date_last_renewal)]
##   data[, Exposure_days  := pmax(Exposure_days, 0)]
##   data[, Exposure_unit  := fcase(
##     between(Exposure_days, 364, 367), 1,
##     default = Exposure_days/365.25
##   )]
##
##
## #data[, Cost_claims_sum := sum(Cost_claims_year), by = ID]
##
##   # Add a column that flags the latest row per group
##
##   setorder(data, ID, -Date_last_renewal)
##   data[, is_second_latest := FALSE]
##   data[, is_second_latest := .I == .I[2], by = ID]
##
##   key_vars <- c("ID","Year_matriculation","Power",
##                 "Cylinder_capacity","N_doors","Type_fuel","Weight")
##
##   agg <- data[, .(
##     Exposure               = sum(Exposure_unit),
##     Max_policies_max       = max(Max_policies),
##     Policies_in_force_second_latest = Policies_in_force[is_second_latest],
##
##     Max_products_max       = max(Max_products),
##     Payment_max            = max(Payment),
##     Premium_second_latest = Premium[is_second_latest],
##     Cost_claim_this_year = Cost_claims_year[which.max(Date_last_renewal)],
##     Cost_claims_sum_history = sum(Cost_claims_year) -(Cost_claims_year[which.max(Date_last_renewal)]
##     R_claims_history   = max(N_claims_history - N_claims_year)/sum(Exposure_unit),
##     # N_claims_year_sum      = sum(N_claims_year),
##     N_claims_history   = max(N_claims_history - N_claims_year),
##     # R_claims_history_sum   = sum(R_Claims_history),
##     Type_risk_max          = max(Type_risk),
##     Value_vehicle_mean     = mean(Value_vehicle),
```

```
##     N_doors_mean           = mean(N_doors),
##     Length_sum             = sum(Length, na.rm = TRUE)
##   ), by = key_vars]
##
##
##
##
##
##   agg <- agg %>%
##     dplyr::mutate(claim_indicator = dplyr::if_else(Cost_claims_sum_history + Cost_claim_this_year > 0
##
##   tmp <- agg %>%
##     dplyr::filter(is.na(Type_fuel))
##
##   NA_fuel_Y <- sum(tmp$Cost_claims_sum_history + tmp$Cost_claims_this_year)
##   total_Y <- sum(agg$Cost_claims_sum_history + agg$Cost_claims_this_year)
##
##   na_Y_ratio <- NA_fuel_Y / total_Y
##
##   agg <- agg %>%
##     dplyr::filter(!is.na(Type_fuel))
##
##   agg <- agg %>%
##     dplyr::mutate(Type_fuel = dplyr::if_else(Type_fuel == "P", 1, 0))
##   return(agg)
## }
##
## data_twd <- data_trans(data)
##
## task_twd = as_task_regr(data_twd, target = "Cost_claim_this_year",
##                         id = "tweedie")
##
##
##
## graph_twd = po("encode") %>>%
##   po("scale") %>>%
##   po("learner", lrn("regr.lightgbm",
##     objective              = "tweedie",
##     tweedie_variance_power = to_tune(1, 1.9),
##     learning_rate          = to_tune(1e-3, 0.2, logscale = TRUE),
##     num_leaves             = to_tune(16L, 64L),
##
##     num_iterations         = to_tune(200L, 1000L)
##
##   ))
##
## glrn_twd = GraphLearner$new(graph_twd, id = "tweedie_lgbm")
## resampling = rsmp("cv", folds = 5)
## tuner = tnr("random_search")
## terminator = trm("evals", n_evals = 5)
## measure = msr("regr.mse")
##
##
##
```

```
## at_twd = AutoTuner$new(
##   learner    = glrn_twd,
##   resampling = resampling,
##   measure    = measure,
##   terminator = terminator,
##   tuner      = tuner
## )
##
## at_twd$train(task_twd)
##
##
## at_twd$archive
##
##
##
##
## twd_p1 <- mlr3verse::autoplot(at_twd$predict(task_twd), type = "residual")
## twd_p2 <- mlr3verse::autoplot(at_twd$predict(task_twd), type = "histogram")
##
## plot_twd_dat <- tibble::tibble(
## resp = at_twd$predict(task_twd)$response,
## truth = at_twd$predict(task_twd)$truth
## )
##
##
##
## twd_p3 <- qqplot(x = plot_twd_dat$resp, y = plot_twd_dat$truth)
## abline(a = 0,b = 1)
##
##
## par(mfrow = c(1,2))
## twd_p1
## twd_p2
##
##
## data_F <- data_trans(data)
## data_F <- data_F %>%
##   dplyr::select(-Cost_claim_this_year)
##
##
## inner_folds <- 2
## outer_folds <- 2
## n_evals     <- 4
##
## skim(data_F)
##
## task_freq <- as_task_classif(
##   data_F,
##   target = "claim_indicator",
##   positive = "1",
##   weights = data_F$Exposure,
##   id = "frek_classif"
## )
##
```

```
## graph_freq <- po("encode") %>>%
##   po("scale") %>>%
##   lrn("classif.xgboost",
##       predict_type = "prob",
##       eval_metric = "logloss",
##       nrounds      = to_tune(200, 800),
##       max_depth    = to_tune(3, 7),
##       eta          = to_tune(0.01, 0.3),
##       subsample    = to_tune(0.6, 1)
##       )
##
## at_inner <- auto_tuner(
##   learner = as_learner(graph_freq),
##   resampling = rsmp("cv", folds = inner_folds),
##   measure = msr("classif.bbrier"),
##   tuner = tnr("random_search"),
##   term_evals = n_evals
## )
##
## outer_rsmp <- rsmp("cv", folds = outer_folds)
##
## rr <- resample(
##   task         = task_freq,
##   learner      = at_inner,
##   resampling   = outer_rsmp,
##   store_models = TRUE
## )
##
##
##
##
##
##
##
##
##
## print(rr$aggregate(msr("classif.bbrier")))
##
##
## ncs_plot_dat <- rr$predictions()
##
##
##
## for (i in 1:length(ncs_plot_dat)){
##   cat("Fold: ", i)
##   print(mlr3measures::confusion_matrix(
##     truth = ncs_plot_dat[[i]]$truth,
##     response = ncs_plot_dat[[i]]$response,
##     positive = "1"
##
##   ))
## }
##
##
##
```

```
##
## autoplot(ncs_plot_dat[[1]], type = "threshold", measure = msr("classif.acc"))
##
##
##
## data_S <- data_trans(data)
##
## data_S <- data_S %>% dplyr::select(-claim_indicator)
## data_S <- data_S %>% dplyr::filter(Cost_claim_this_year > 0)
##
## data_S_imputed <- data_S %>%
##   # 1) Sæt Inf → NA for alle numeriske kolonner
##   mutate(across(
##     where(is.numeric),
##     ~ ifelse(is.infinite(.x), NA, .x)
##   )) %>%
##   # 2) Imputer hver kolonne; for integer kolonner rundes mean() af til integer
##   mutate(
##     across(
##       .cols = where(is.integer),
##       .fns  = ~ replace_na(
##         .x,
##         as.integer(round(mean(.x, na.rm = TRUE)))
##       )
##     ),
##     across(
##       .cols = where(is.double),
##       .fns  = ~ replace_na(
##         .x,
##         mean(.x, na.rm = TRUE)
##       )
##     )
##   )
##
##
##
##
## folds_inner <- 5
## folds_outer <- 5
## n_evals     <- 5
##
##
## task_S = as_task_regr(
##   data_S_imputed,
##   target = "Cost_claim_this_year",
##   weights = data_S$Exposure,
##   id     = "skade"
## )
##
##
## graph_S = po("encode") %>>%
##   po("learner", lrn("regr.ranger"))
##
##
```

```
##
##
## graph_S <- po("imputeoor") %>>%
##            po("encode")    %>>%
##            po("scale")     %>>%
##            po("learner",
##               lrn("regr.lightgbm",
##                   objective       = "quantile",
##                   alpha           = 0.85
##               ))
##
##
##
## glrn_S <- GraphLearner$new(graph_S)
##
##
## inner_tuner_S = AutoTuner$new(
##   learner    = glrn_S,
##   resampling = rsmp("cv", folds = folds_inner),
##   measure    = msr("regr.mse"),
##   tuner      = tnr("random_search"),
##   terminator = trm("evals", n_evals = n_evals)
## )
##
## outer_cv <- rsmp("cv", folds = folds_outer)
## rr_S <- resample(
##   task_S,
##   inner_tuner_S,
##   outer_cv
## )
##
##
## rr_S$aggregate(msr("regr.mse"))
##
##
##
##
##
##
## autoplot(rr_S$prediction(), type = "xy")
## autoplot(rr_S$prediction(), type = "residual")
##
##
##
## DATA <- data_trans(data)
## N <- nrow(DATA)
## train_idx <- sample(seq_len(N), size = 0.8 * N)
##
## train <- DATA[train_idx, ]
## test <- DATA[-train_idx, ]
##
##
## frekvens_model <- train_freq(train, folds = 2, n_evals = 2)
##
```

```
##
## skades_model <- train_severity(train, folds = 2, n_evals = 2)
##
##
## E_N <- frekvens_model$predict_newdata(test)
## E_X <- skades_model$predict_newdata(test)
##
## res <- tibble(
##   E_N = E_N$prob[, 1],
##   E_X =  E_X$response
##   )
##
## res <- res %>% dplyr::mutate(predicted = E_N * E_X )
##
##
## plot(res$predicted, test$Cost_claim_this_year)
## print(mse_oos <- mean((res$predicted-test$Cost_claim_this_year)^2))
##
##
##
##
## rmd <- knitr::opts_knit$get("input.file") %||% knitr::current_input()
## stopifnot(!is.null(rmd) && nzchar(rmd))
## tmp <- knitr::purl(input = rmd,
##                    output = tempfile(fileext = ".R"),
##                    documentation = 0, quiet = TRUE)
## code_lines <- readLines(tmp, warn = FALSE, encoding = "UTF-8")
## cat("```r\n", paste(code_lines, collapse = "\n"), "\n```")
## ```
```