Toward a Predictive Analytics Framework

P. Adams, C. Edwalds, B. Holland, E. Hui, M. Niemerg, H. Yu

July 10, 2024

Table of Contents

# Executive Summary

Predictive modeling is a powerful tool that leverages statistical techniques to forecast outcomes. It is widely used across various industries, from healthcare to finance, to make informed decisions based on historical data. This tutorial aims to introduce a framework for developing predictive models in the context of actuarial experience studies. To ground this framework within the context of real actuarial problems, we will also specifically look to understand and model the differences in mortality by product (whole life, term, etc.) With the modeling approach we can see that

* The relative spread of preferred mortality differs by product.
  + For 2-class preferred systems, the residual standard mortality is much higher than preferred for term than for other products.
  + The spread for UL/VL/ULSG/VLSG for 4-class preferred systems is much wider than for other products.
* There are divergences in the spread of face amount factors for xl, Perm, and Term, with xL narrowing relative to Term.
* The issue age slope appears to be steeper for Term than Perm and xL under age 65. However, differences emerge above issue age 65, with the slope for Perm steepening relative to xL.
* There are differences among the products in durations 1 and 2.
* Since issues years 1990-1999, there has been a small but steady increase in relative mortality for xL vs Term, with xL now approaching Term.

# Data

For what follows, we used a filtered and summarized subset of the Society of Actuaries’ Individual Life Experience Committee mortality data. Columns included in the extract were

* Number of Preferred Classes
* Preferred Class
* Smoker Status
* Face Amount Band
* Observation\_Year
* Duration
* Issue Age
* Insurance Plan
* Anticipated Level Term Period
* Issue Year
* Sex
* Death Count
* Death Claim Amount
* Tabular Expected Mortality by Count - 2015VBT
* Tabular Expected Mortality by Amount - 2015VBT

The data were filtered as

* Issue ages 18 and greater
* Durations 25 and less
* Experience years 2013-2017

and then grouped or combined as

* Underwriting: concatenation of smoker status, number of preferred classes, and preferred class, in that order
* Duration: 1, 2, 3, 4-5, 6-15, 16-25
* Issue Age: 18-24, 25-34, 35-44, 45-54, 55-64, 65-74, 75-84, 85-99
* Issue Years: 1900-1989, 1990-1999, 2000-2009, 2010+
* Insurance Plan: UL, ULSG, VL, VLSG collapsed into category “xL”
* Face Amount Band: face amounts under 50,000 grouped into a single category, face amounts 1 million and greater grouped into a single category

The intent of this heavy grouping and summarization was to enable running this document with modest computing resources. The source data can be replaced with a similarly constructed dataset with finer grouped variables.

The code to generate these files can be found in the datafiles subfolder. It relies on an unpublished version of the ILEC dataset which has been restructured using the Arrow framework into a collection of Parquet files. A knowledgeable reader should be able to adapt the code to whatever environment in which they keep their own copy of the ILEC dataset.

# Machine Learning in Mortality Studies

Experience studies are a primary tool that actuaries use to quantify and understand historical experience. It is a natural next step to apply statistical techniques to experience studies to discover new and relevant insights. There are multiple advantages to this approach:

* Allows the actuary to avoid cumbersome and potentially misleading univariate analysis
* Allows the actuary to appropriately consider credibility and unlock all the credibility inherent in the data
* Makes it easier to discover and appropriately adjust for variable interactions
* Enables the actuary the ability to statistically control for the different sources of variation in any given cell of a mortality study.

## Problem Statement

One question of interest to actuaries is why different products exhibit different mortality outcomes. Even though they can be difficult to separately identify and quantify, it is known that underwriting, target market, policyholder behavior, and socioeconomic factors, among others, have direct bearing on mortality outcomes. With a statistical or machine learning model we have a possible solution to account for the impact of these variables. For this project, the key question we are trying to answer is how mortality varies by product in the Individual Life Experience Committee dataset. In the simplified dataset that is used herein, the product categories are Term, Perm, UL/VL, and Other. To understand the differences in mortality by product, we will construct machine learning models to predict the mortality outcomes and analyze the results for relevant insights.

## Methodology

The framework will guide the process of code setup, model creation, preprocessing, and validation. It will also address common challenges often encountered such as: incorporating nonlinear relationships, determining interactions, dealing with underfitting and overfitting (bias-variance trade-off), and model interpretability. The goal of this project is to provide useful techniques, code, and ideas, to actuaries to guide future analysis of mortality studies.

There are several common key steps in any modeling process: data preprocessing, data exploration, model selection, model validation, and model interpretation. Much more can be written on these topics than we have the space to explore, and we aim to address the key considerations as they pertain to experience studies.

## Modeling Approaches

When applying statistics and machine learning to experience studies, there are multiple different modeling approaches one might take. We will focus our attention on the most common approaches used: generalized linear models (GLMs), generalized linear models with penalization (also known as elastic net GLMs), and gradient boosting machines (GBMs or GBDTs). Many other approaches or variations on these approaches are also reasonable.

### Generalized Linear Models (GLM)

Generalized linear models have the most history of the methods that we will examine and in some sense are the simplest. One of the benefits of GLMs is that they allow statistical hypothesis testing. For instance, individual model coefficients can be statistically tested, and various statistical tests can be performed to validate results and compare models. The results of GLMs are also relatively simple to interpret. However, GLMs have a few disadvantages: due to their relative simplicity, they have lower predictive power than other methods. To get the best performance out of a GLM, additional effort is needed to capture nonlinear relationships and interactions. Ultimately, this can make them more time-intensive than other methodologies.

GLMs can be extended into regularized GLMs, such as LASSO or Ridge, which modifies the objective used to fit the model. This regularization term offers several advantages, disadvantages, and changes to the modeling process. First, the addition of penalization makes confidence intervals and hypothesis testing infeasible. Instead of using hypothesis testing on coefficients and likelihood ratio tests to evaluate relative fitness of models, we apply a machine learning paradigm by optimizing our model using cross-validation. Fortunately, a regularized GLM still maintains the nice interpretability of a linear model, and it can increase the overall predictive accuracy of the model. Additionally, by using a LASSO penalty, it can perform automatic feature selection.

### Gradient-Boosted Decision Trees (GBDT)

Gradient boosted decision trees are an ensemble of decision trees generated in a stage-wise fashion. Each decision tree is recursively trained on the residuals of the previous tree. The first tree is a decision tree on the outcome, the second the residuals on that, and so on. In this way, the model is continually refocusing on where its predictions are weakest. Popular frameworks for gradient boosted decision trees include LightGBM, CatBoost, and XGBoost. This model is one of the most effective methods for classification and regression for tabular data.

Gradient boosting machines (applied here with LightGBM) have become the go-to approach in many tabular machine learning tasks due to their very high accuracy, ease of use, and ability to seamlessly discover important interactions. However, they can also be the most complex to interpret. To aid in interpretation, we will discuss the use of SHAP values, which is a popular method of interpretation.

## Model Explanation

### Ordered Lorenz Plot and Gini

An ordered Lorenz curve and the associated Gini coefficient measure the ability of a model to stratify risk. An ordered Lorenz curve is created using the model prediction as an index. Using this index, we graph the cumulative percentage of claims vs the cumulative percentage of exposure. The more bowed this line, the better the model is able to predict the outcome. The Gini Index measures the difference between this line and perfect equality. The more your model is able to predict risk, the more unequal the distribution of claims is between the model prediction, and thus the larger the Gini coefficient.

### Lift Plot

There are several different varieties of lift plots used in connection with machine learning. These plots are used to help visually understand the risk stratification and accuracy of a model. As presented here, lift plots sort the model predictions into deciles based upon the predicted value. For each decile, the model’s average prediction for that cell is graphed vs the value seen empirically in the data. The more these two values are in agreement, the better the model is performing.

### SHAP

SHAP values are a method of model interpretation in machine learning and originally come from Shapley values in economics. SHAP values measure the impact each feature has on the prediction for a particular instance. This numeric score indicates how much each feature contributed to the prediction in terms of sign and magnitude.

### Feature Importance

Feature importance is a global measure of how much a variable contributes to the predictions of a target variable within a model. This can be helpful in interpreting a model to understand the key drivers in aggregate. However, unlike SHAP values, feature importance does not help you interpret individual predictions. Feature importance is usually presented in terms of percent contribution. When done so, a feature importance of 20% for a feature would implies that 20% of the overall reduction in prediction error is attributable to that particular feature. There are multiple ways of measuring feature importance. One of the simplest and most intuitive is permutation feature importance. Using this method, you scramble a particular feature so that it is no longer useful and measure the percent difference in model performance before and after this change. The change in error would be the importance.

The reader should be cautioned that a low relative importance does not imply lack of significance or of predictive value. For example, gender is a well-known predictor of mortality. The variation explainable by other factors of the data can greatly exceed the variation arising from gender, and interactions with other variables like age can further rob gender of importance attributed to it. The effect then is to push gender down the feature importance list.

### Goodness-of-Fit

No matter how well a model may behave on measures of feature importance, lift, Lorenz and Gini indices, mean square error, deviance, and so on, it is nonetheless important to check goodness-of-fit. Goodness-of-fit checks allow us to see how well a model reproduces the phenomena of interest. For our purposes, this is the same as checking ratios of actual claims to model predicted claims. In each model section, there are univariate and bivariate goodness-of-fit tables. Ideally, we should see 100% for all entries. For the GLM model and the univariate goodness-of-fit checks, we will see this throughout the tables of goodness-of-fit, as a non-penalized GLM will reproduce the margins for any included categorical variable or interaction of categorical variables. For space reasons, we omit a test for ratios significantly different from 100%. However, qualitatively, ratios far from 100%, perhaps +/- 5% or +/- 10%, should be deemed as evidence of poor fit for that cell.

# Framework Preparation

Before getting to the core data analysis task, we need to first prepare the R environment by configuring display and model options, loading necessary libraries. Then, we load the data and prep for running data analysis and modeling. This section also reads in the dataset and splits the data into training and testing sets based on the observation year. Additionally, most parameters are set here. If prototyping is enabled, it creates a smaller subset of the training data for quicker processing.

Here, we set display options.

#-----------------------------------------#  
##### Display Options #####  
#-----------------------------------------#  
## turn off scientific notation  
options(scipen = 999)  
  
## change how many digits to display  
options(digits=4)  
  
## Suppress warnings  
options(warn = -1)  
  
## Determine which output we are generating  
## This will be html, docx, or pdf  
if(interactive()) {  
 output\_format <- "html"  
} else {  
 output\_format <- knitr::opts\_knit$get("rmarkdown.pandoc.to")   
}

Here we set model options for GLMNET and LightGBM.

#-----------------------------------------#  
##### Model Options #####  
#-----------------------------------------#  
  
## When TRUE, only a fraction of the data is used, drastically reducing runtimes  
prototype <- FALSE  
prototype\_size <- 50000 ## number of records in data fraction  
nTrainSeed <- 42 ## seed to use when splitting the data for reproducibility  
  
## GLMNet parameters  
nGLMNetCores <- 10  
nInteractionDepth <- 1  
fGLMNetAlpha <- 0.5  
nUseTopLightGBMInteractions <- "ALL" # Integer or "ALL"  
bUseSparse <- TRUE  
nELSeed <- 13579  
  
## LightGBM parameters   
flgbm\_vis\_subset <- 0.1  
bFullInteractions <- FALSE # Very slow in default configuration  
nPlotTopFeatures <- 3  
nPlotTopInteractions <- 3  
nGBMSeed <- 1337  
  
## Flags for running specific models  
runGLM <- TRUE  
runLightGBM <- TRUE  
runGLMInt <- TRUE

Here we load all required libraries.

#-----------------------------------------#  
##### Required libraries #####  
#-----------------------------------------#  
## Less verbose tidyverse  
options (tidyverse.quiet = TRUE)  
  
## We use bUseGroundhog in the R codespaces to control versioning,  
## If you have your own setup, set to FALSE.  
## Things may nonetheless break, so use at your own risk.  
bUseGroundhog <- FALSE   
  
if(bUseGroundhog) {  
 pkgDate <- "2024-05-09"  
   
 suppressPackageStartupMessages( {  
 library(groundhog)  
   
 groundhog.library(pre,pkgDate)  
 groundhog.library(lightgbm,pkgDate)  
 groundhog.library(data.table,pkgDate)   
 groundhog.library(lmtest,pkgDate)  
 groundhog.library(glmnet,pkgDate)  
 groundhog.library(dplyr,pkgDate)  
 groundhog.library(EIX,pkgDate)  
 groundhog.library(ggplot2,pkgDate)  
 groundhog.library(tidyr,pkgDate)  
 groundhog.library(doParallel,pkgDate)  
 groundhog.library(tidyverse,pkgDate)  
 groundhog.library(magrittr,pkgDate)  
 groundhog.library(dtplyr,pkgDate)  
 groundhog.library(flextable,pkgDate)  
 groundhog.library(ftExtra,pkgDate)  
 groundhog.library(arrow,pkgDate)  
 groundhog.library(here,pkgDate)  
 groundhog.library(shapviz,pkgDate)  
 groundhog.library(patchwork,pkgDate)  
 groundhog.library(Matrix,pkgDate)  
 groundhog.library(MatrixModels,pkgDate)  
 groundhog.library(openxlsx,pkgDate)  
 groundhog.library(flexlsx,pkgDate)  
 }  
 )  
  
} else {  
 suppressPackageStartupMessages({  
 library(pre)  
 library(lightgbm)  
 library(data.table)   
 library(lmtest)  
 library(glmnet)  
 library(dplyr)  
 library(EIX)  
 library(ggplot2)  
 library(tidyr)  
 library(doParallel)  
 library(tidyverse)  
 library(magrittr)  
 library(dtplyr)  
 library(flextable)  
 library(ftExtra)  
 library(arrow)  
 library(here)  
 library(shapviz)  
 library(patchwork)  
 library(Matrix)  
 library(MatrixModels)  
 library(openxlsx2)  
 library(flexlsx)  
 })  
}  
  
#-----------------------------------------#  
##### Set Folder Locations #####  
#-----------------------------------------#  
  
source("R/functions.R")  
source("R/glmnet\_support.R")  
  
## set library location  
local\_libraries <- FALSE  
if(local\_libraries)  
{  
 library.dir <- 'D:\\Data\\Niemerg\\Life Predictive Mortality POG\\01 library'  
 .libPaths(new = library.dir)  
}  
  
bDebug <- FALSE  
  
## Save the expensive working objects and reload if they exist. If this is true   
## and they do not exist, the computations will rerun.  
bUseCache <- TRUE  
bInvalidateCaches <- FALSE  
  
#-----------------------------------------#  
##### Data Options #####  
#-----------------------------------------#  
  
src\_file <- 'http://finriskanalytics-ilecdata.s3-website-us-east-1.amazonaws.com/ilec13\_17\_framework\_light.parquet'  
  
cacheFileRoot <- file.path(  
 getwd(),  
 "objectcache",  
 tools::file\_path\_sans\_ext(  
 tail(  
 unlist(strsplit(src\_file,"/")),  
 1  
 )  
 )  
)  
  
exportsRoot <- file.path(  
 getwd(),  
 "render\_exports",  
 tools::file\_path\_sans\_ext(  
 tail(  
 unlist(strsplit(src\_file,"/")),  
 1  
 )  
 )  
)  
  
#-----------------------------------------#  
##### Modeling Parameters #####  
#-----------------------------------------#  
resp\_var <- "amount\_actual"  
resp\_offset <- "amount\_2015vbt"  
  
pred\_cols <- c("uw",  
 "face\_amount\_band",  
 "dur\_band1",  
 "ia\_band1",  
 "gender",  
 "insurance\_plan",  
 "ltp",  
 "iy\_band1")  
  
factor\_cols <- c("uw",  
 "face\_amount\_band",  
 "dur\_band1",  
 "ia\_band1",  
 "gender",  
 "insurance\_plan",  
 "ltp",  
 "iy\_band1")

Here we read the data, convert specified columns to categorical factors to ensure proper data handling and adjust the labels for the ‘face\_amount\_band’ factor to avoid issues in model outputs. The dataset is split into training and testing sets based on the observation year, with the year 2017 data used for validation. If prototyping is enabled, the code further subsets the training data to a smaller size for faster processing, ensuring reproducibility by setting a seed before shuffling and selecting the subset.

#-----------------------------------------#  
##### Load dataset #####  
#-----------------------------------------#  
  
## Determine the file extension of the source file  
file\_type <- tools::file\_ext(src\_file)  
  
## Read the dataset based on file extension  
if (file\_type == "csv") {  
 ## Load a CSV file  
 ds <- fread(src\_file)  
} else if (file\_type == "parquet") {  
 ## Load a Parquet file and convert it to a data table  
 if(src\_file %like% "amazonaws.com") {  
 s3b <- s3\_bucket(strsplit(urltools::domain(src\_file),".",fixed=T)[[1]][1])  
 ds <- read\_parquet(s3b$path(urltools::path(src\_file)))  
 } else {  
 ds <- arrow::read\_parquet(src\_file) %>% as.data.table()  
 }  
}  
  
#-----------------------------------------#  
##### Convert columns to factors #####  
#-----------------------------------------#  
  
## This step ensures that categorical data is appropriately treated as such.  
ds[, (factor\_cols) := lapply(.SD, factor), .SDcols = factor\_cols]  
  
## Adjust the labels for the 'face\_amount\_band' factor  
## Replace colons in factor names with hyphens to avoid issues in model outputs  
ds[, face\_amount\_band := fct\_relabel(  
 face\_amount\_band,  
 function(x) sub(":", " -", x, fixed = TRUE)  
)]  
  
#-----------------------------------------#  
##### Set train / test #####  
#-----------------------------------------#  
  
## Split the dataset into training and testing sets based on the observation year  
## The year 2017 is used as the validation set  
train <- ds[observation\_year != 2017]  
test <- ds[observation\_year == 2017]  
  
## Subset the data for prototyping purposes  
## This code block is executed if a prototype subset is requested  
if (prototype) {  
 ## Set the seed for reproducibility  
 set.seed(nTrainSeed)  
   
 ## Shuffle and select a subset of the training data  
 train <- train[sample(1:nrow(train)), ]  
 train <- train[1:prototype\_size]  
}

# Models

## GLM

Below is an analysis using a main-effects GLM to better understand the data. We integrate two modeling approaches:

* Standard GLM analysis with model calibration (but no model building), including presentation of coefficients and residuals analysis, and
* An approach to explore the interactions of the main effects model along selected dimensions of the data, checking the average main effects for those subsets, weighted according to the offset used in the analysis.

### Model Summary

## Construct the model formula from predictor columns  
modelFormula <- paste(pred\_cols, collapse = " + ")  
  
## Include offset in the model formula if it exists  
if (exists("resp\_offset")) {  
 modelFormula <- paste(  
 modelFormula,  
 paste0("offset(log(", resp\_offset, "))"),  
 sep = " + "  
 )  
}  
  
## Complete the model formula with the response variable  
modelFormula <- paste(resp\_var, modelFormula, sep = " ~ ")  
modelFormula <- as.formula(modelFormula)  
  
## Check if the cached model exists and is valid  
if (bUseCache & file.exists(paste0(cacheFileRoot, "\_glm\_model.rds")) & !bInvalidateCaches) {  
 ## Load the cached model if it exists  
 modelGLM <- readRDS(paste0(cacheFileRoot, "\_glm\_model.rds"))  
} else {  
 ## Fit the GLM model using the specified formula and data  
 modelGLM <- glm(  
 formula = modelFormula,   
 family = quasipoisson,   
 data = train,  
 x = FALSE,  
 y = FALSE,  
 model = FALSE  
 )  
 ## Remove the data from the model object to reduce size  
 modelGLM$data <- c()  
   
 ## Save the fitted model to cache if caching is enabled  
 if (bUseCache) {  
 saveRDS(modelGLM, paste0(cacheFileRoot, "\_glm\_model.rds"))  
 }  
}  
  
## Append predictions to the dataset  
ds[, predictions\_glm := predict(modelGLM, newdata = .SD, type = "response")]  
train[, predictions\_glm := predict(modelGLM, newdata = .SD, type = "response")]  
test[, predictions\_glm := predict(modelGLM, newdata = .SD, type = "response")]

#### Model, Table of Coefficients, ANOVA

##### Model Summary

Below is the table of coefficients for the fitted GLM. Each entry is a coefficient in the table for the level of the indicated variable. The estimate and standard errors are on the scale of the linear predictor. For a Poisson model with log link, this means they are on the log scale.

## Show summary of the model  
modelGLM %>%   
 as\_flextable() %>%  
 set\_table\_properties(opts\_html=list(  
 scroll=list(  
 add\_css="max-height: 500px;"  
 )  
 )  
 )

|  | Estimate | Standard Error | z value | Pr(>|z|) |  |
| --- | --- | --- | --- | --- | --- |
| (Intercept) | 0.978 | 0.091 | 10.743 | 0.0000 | \*\*\* |
| uwN/2/1 | -0.152 | 0.011 | -13.931 | 0.0000 | \*\*\* |
| uwN/2/2 | 0.248 | 0.010 | 23.945 | 0.0000 | \*\*\* |
| uwN/3/1 | -0.324 | 0.014 | -23.614 | 0.0000 | \*\*\* |
| uwN/3/2 | -0.180 | 0.012 | -14.802 | 0.0000 | \*\*\* |
| uwN/3/3 | 0.152 | 0.011 | 14.077 | 0.0000 | \*\*\* |
| uwN/4/1 | -0.330 | 0.014 | -23.555 | 0.0000 | \*\*\* |
| uwN/4/2 | -0.166 | 0.015 | -11.060 | 0.0000 | \*\*\* |
| uwN/4/3 | 0.011 | 0.017 | 0.630 | 0.5287 |  |
| uwN/4/4 | 0.208 | 0.016 | 12.618 | 0.0000 | \*\*\* |
| uwS/1/1 | 0.068 | 0.014 | 4.778 | 0.0000 | \*\*\* |
| uwS/2/1 | -0.179 | 0.021 | -8.637 | 0.0000 | \*\*\* |
| uwS/2/2 | 0.112 | 0.022 | 5.178 | 0.0000 | \*\*\* |
| uwU/1/1 | 0.261 | 0.038 | 6.794 | 0.0000 | \*\*\* |
| face\_amount\_band04 - 50,000 - 99,999 | -0.113 | 0.018 | -6.410 | 0.0000 | \*\*\* |
| face\_amount\_band05 - 100,000 - 249,999 | -0.236 | 0.015 | -15.413 | 0.0000 | \*\*\* |
| face\_amount\_band06 - 250,000 - 499,999 | -0.299 | 0.016 | -18.893 | 0.0000 | \*\*\* |
| face\_amount\_band07 - 500,000 - 999,999 | -0.322 | 0.016 | -20.258 | 0.0000 | \*\*\* |
| face\_amount\_band08 - 1,000,000+ | -0.349 | 0.015 | -23.004 | 0.0000 | \*\*\* |
| dur\_band102 | 0.010 | 0.030 | 0.341 | 0.7334 |  |
| dur\_band103 | 0.000 | 0.029 | 0.006 | 0.9951 |  |
| dur\_band104-05 | -0.044 | 0.026 | -1.686 | 0.0917 | . |
| dur\_band106-15 | -0.115 | 0.028 | -4.084 | 0.0000 | \*\*\* |
| dur\_band116-25 | -0.096 | 0.031 | -3.088 | 0.0020 | \*\* |
| ia\_band125-34 | -0.062 | 0.034 | -1.813 | 0.0699 | . |
| ia\_band135-44 | -0.045 | 0.033 | -1.347 | 0.1779 |  |
| ia\_band145-54 | -0.075 | 0.033 | -2.258 | 0.0239 | \* |
| ia\_band155-64 | -0.100 | 0.033 | -3.024 | 0.0025 | \*\* |
| ia\_band165-74 | -0.082 | 0.033 | -2.440 | 0.0147 | \* |
| ia\_band175-84 | -0.173 | 0.034 | -5.100 | 0.0000 | \*\*\* |
| ia\_band185-99 | -0.304 | 0.040 | -7.581 | 0.0000 | \*\*\* |
| genderM | 0.011 | 0.006 | 1.790 | 0.0735 | . |
| insurance\_planPerm | -0.134 | 0.062 | -2.159 | 0.0308 | \* |
| insurance\_planTerm | -0.275 | 0.069 | -3.977 | 0.0001 | \*\*\* |
| insurance\_planxL | -0.042 | 0.061 | -0.681 | 0.4958 |  |
| ltp10 yr | -0.090 | 0.034 | -2.650 | 0.0080 | \*\* |
| ltp15 yr | -0.097 | 0.034 | -2.826 | 0.0047 | \*\* |
| ltp20 yr | -0.207 | 0.033 | -6.286 | 0.0000 | \*\*\* |
| ltp25 yr | -0.247 | 0.049 | -5.077 | 0.0000 | \*\*\* |
| ltp30 yr | -0.161 | 0.036 | -4.502 | 0.0000 | \*\*\* |
| ltpNot Level Term | -0.396 | 0.045 | -8.750 | 0.0000 | \*\*\* |
| ltpUnknown | -0.207 | 0.036 | -5.804 | 0.0000 | \*\*\* |
| iy\_band11990-1999 | -0.073 | 0.023 | -3.209 | 0.0013 | \*\* |
| iy\_band12000-2009 | -0.148 | 0.026 | -5.706 | 0.0000 | \*\*\* |
| iy\_band12010+ | -0.206 | 0.030 | -6.857 | 0.0000 | \*\*\* |
| *Signif. codes: 0 <= '\*\*\*' < 0.001 < '\*\*' < 0.01 < '\*' < 0.05* | | | | | |
|  | | | | | |
| (Dispersion parameter for quasipoisson family taken to be 700118) | | | | | |
| Null deviance: 6.172e+10 on 249979 degrees of freedom | | | | | |
| Residual deviance: 5.662e+10 on 249935 degrees of freedom | | | | | |

##### ANOVA

The ANOVA table displays the analysis of deviance for the GLM. For each variable, we see the proportion of deviance explained by that variable and its associated degrees of freedom.

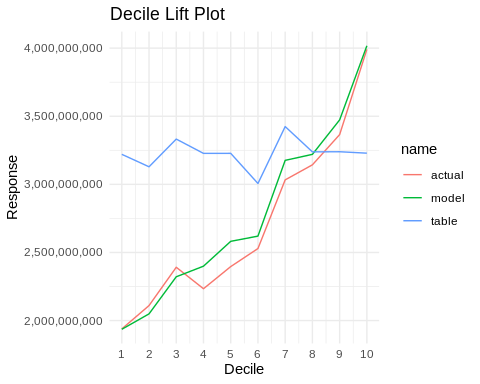
## Show the ANOVA table  
  
## Check if the cached ANOVA results exist and are valid  
if (bUseCache & file.exists(paste0(cacheFileRoot, "\_glm\_model\_anova.rds")) & !bInvalidateCaches) {  
 ## Load the cached ANOVA results if they exist  
 mod.glm.anova <- readRDS(paste0(cacheFileRoot, "\_glm\_model\_anova.rds"))  
} else {  
 ## Perform ANOVA on the GLM model  
 mod.glm.anova <- anova(modelGLM, test = "Chisq")  
   
 ## Save the ANOVA results to cache if caching is enabled  
 if (bUseCache) {  
 saveRDS(mod.glm.anova, paste0(cacheFileRoot, "\_glm\_model\_anova.rds"))  
 }  
}  
  
## Convert the ANOVA results to a data table, add feature names, and format the table  
mod.glm.anova %>%   
 as.data.table() %>% # Convert to data table  
 add\_column(rownames(mod.glm.anova), .before = 1) %>% # Add feature names as a new column  
 setnames(old = 1, new = "feature") %>% # Rename the new column to "feature"  
 flextable() %>% # Create a flextable for formatting  
 set\_formatter(  
 `Pr(>Chi)` = function(x) ifelse(x < 0.01, "< 0.1%", sprintf("%1.2f%%", 100 \* x)) # Format p-values  
 )

| feature | Df | Deviance | Resid. Df | Resid. Dev | Pr(>Chi) |
| --- | --- | --- | --- | --- | --- |
| NULL |  |  | 249,979 | 61,716,566,139 |  |
| uw | 13 | 3,858,650,451 | 249,966 | 57,857,915,688 | < 0.1% |
| face\_amount\_band | 5 | 701,275,400 | 249,961 | 57,156,640,288 | < 0.1% |
| dur\_band1 | 5 | 118,698,616 | 249,956 | 57,037,941,672 | < 0.1% |
| ia\_band1 | 7 | 162,999,155 | 249,949 | 56,874,942,517 | < 0.1% |
| gender | 1 | 3,395,713 | 249,948 | 56,871,546,804 | 2.76% |
| insurance\_plan | 3 | 57,762,926 | 249,945 | 56,813,783,878 | < 0.1% |
| ltp | 7 | 152,501,012 | 249,938 | 56,661,282,865 | < 0.1% |
| iy\_band1 | 3 | 41,049,822 | 249,935 | 56,620,233,043 | < 0.1% |

##### Lift

The lift plot compares the GLM against the underlying mortality table.

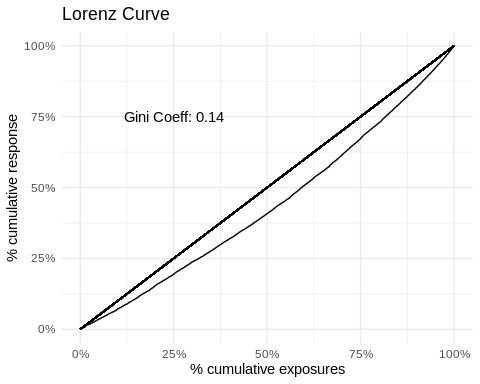
## generate lift plot  
test[,decile.table(get(resp\_var),predictions\_glm/get(resp\_offset),get(resp\_offset))] %>%  
 pivot\_longer(-c(decile,exposures)) %>%   
 as.data.table() %>%  
 ggplot(aes(x=decile, y=value, col=name)) +   
 geom\_line() +  
 scale\_x\_continuous(breaks=c(1:10)) +  
 labs(x="Decile",y="Response")+  
 ggtitle("Decile Lift Plot") +  
 theme\_minimal() +  
 scale\_y\_continuous(labels=scales::comma)



##### Lorenz Plot

The Lorenz plot demonstrates a model’s ability to stratify predictions against a null baseline.

## lorenz plot  
test[,lorenz(get(resp\_var), predictions\_glm / get(resp\_offset), get(resp\_offset))]



The table of coefficients shows a number of interesting phenomena and perhaps some surprises:

* Gender is not significant. Since we are using an offset of tabular expected rates, the interpretation is that the underlying differentials in the tabular expected rates are adequate for the current data, after adjusting for other factors.
* Underwriting is the most influential factor from the ANOVA perspective.
* Both the most recent issue years (2010+) and the most recent durations show significant mortality factors. Durations 1 and 2 are significantly higher than durations 3+
* While insurance plans other than “Other” are significantly different from 0, a quick glance at the effects plot shows that the UL/VL plans are not significantly different from each other and with Perm, while Term is borderline significantly different from UL/VL.
* Face amount bands 250K and greater have factors not significant from one another.

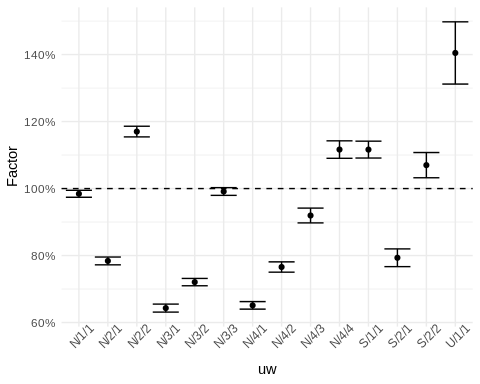
### Model Illustrations and Graphics

#### Effects Plots

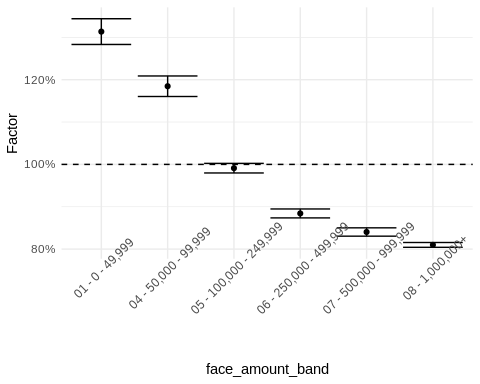
Because the model contains every column, this is equivalent to computing the marginal actual-to-tabular ratios. However, the model also provides standard errors, which is useful for assessing the significance of the marginal ratios.

## Plot amount model terms  
  
## Calculate the dispersion parameter for the GLM model  
glm\_disp <- sum(modelGLM$residuals^2 \* modelGLM$weights) / modelGLM$df.residual  
  
## Generate plots for each predictor column  
lapply(pred\_cols, function(s) {  
 ## Summarize the data for the current predictor  
 p <- ds[, .(  
 predicted = sum(predictions\_glm) / sum(amount\_2015vbt),  
 stde = sqrt(sum(predictions\_glm) \* glm\_disp) / sum(amount\_2015vbt)  
 ), by = c(s)] %>%  
 setnames(s, "x") %>% # Rename the grouping column to "x"  
 mutate(  
 x = fct\_relevel(  
 x,  
 sort(levels(x)) # Reorder factor levels  
 )  
 ) %>%  
 as.data.table() %>%  
 ## Create the plot  
 ggplot(aes(x = x, y = predicted)) +  
 geom\_point() + # Add points for the predicted values  
 geom\_errorbar(  
 aes(  
 ymin = predicted - 1.96 \* stde,  
 ymax = predicted + 1.96 \* stde  
 ) # Add error bars for the 95% confidence interval  
 ) +  
 geom\_hline(yintercept = 1, linetype = 2) + # Add a horizontal line at y = 1  
 scale\_y\_continuous(  
 name = "Factor",  
 labels = scales::percent # Format y-axis labels as percentages  
 ) +  
 scale\_x\_discrete(name = s) + # Set x-axis name to the current predictor  
 theme\_minimal() + # Use a minimal theme for the plot  
 theme(  
 axis.text.x = element\_text(angle = 45) # Rotate x-axis text for readability  
 )  
   
 return(p) # Return the plot  
}) %>%  
 purrr::set\_names(pred\_cols) %>% # Set names for each plot based on predictor columns  
 iwalk(~ {  
 cat('##### ', .y, '\n\n') # Print the plot title  
 print(.x) # Print the plot  
 cat('\n\n') # Add spacing after each plot  
 })

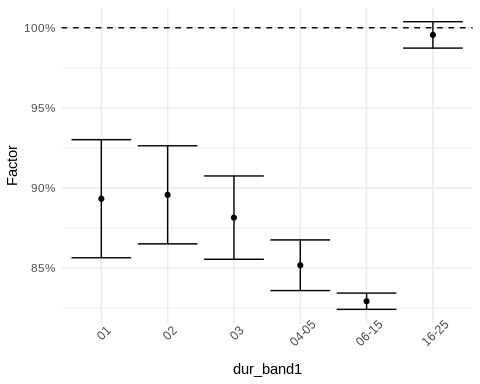
##### uw



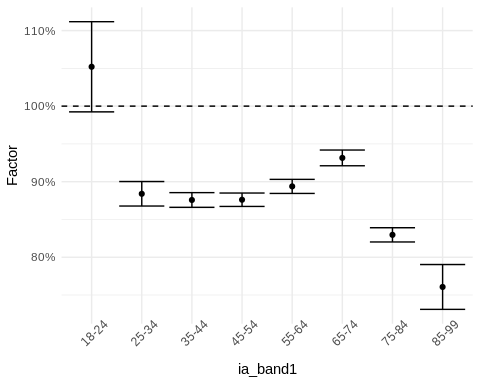
##### face\_amount\_band



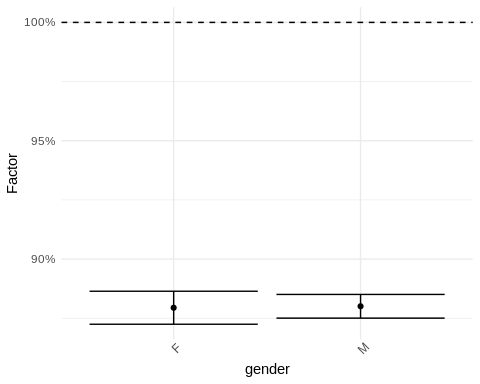
##### dur\_band1



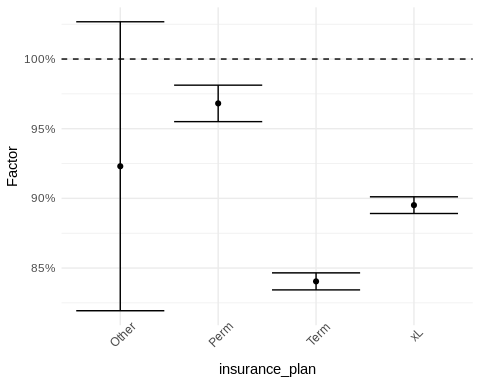
##### ia\_band1



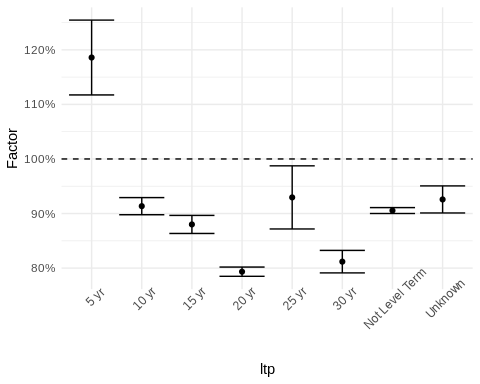
##### gender



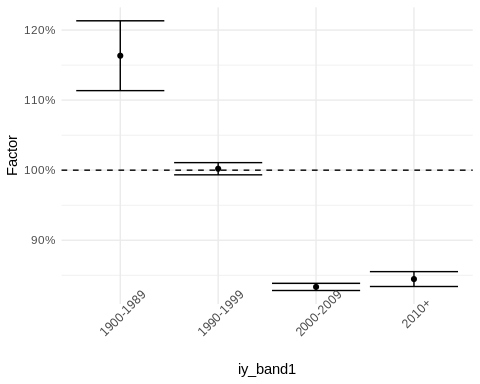
##### insurance\_plan



##### ltp



##### iy\_band1



#### Goodness-of-Fit

Goodness-of-fit tables are provided. Each table provides actual-to-model ratios for single variables and for 2-way combinations of variables. A model is qualitatively deemed to perform well if goodness-of-fit ratios are close to 100% in almost all situations. The quantitative assessment using significance testing is omitted here.

##### Unvariate Goodness-of-Fit

## Generate summary tables for each factor column and format them  
map(factor\_cols, .f = \(x) {  
 ## Convert column name to symbol for tidy evaluation  
 x <- sym(x)  
 resp\_var\_sym <- resp\_var  
   
 ## Summarize data by the current factor column  
 train %>%  
 group\_by(!!x) %>%  
 summarize(  
 Outcome = sum(amount\_actual), # Sum the actual amounts  
 AM = sum(amount\_actual) / sum(predictions\_glm) # Calculate the Actual-to-Model ratio  
 ) %>%  
 ## Create a flextable for the summarized data  
 flextable() %>%  
 ## Format the Actual-to-Model column as percentages  
 set\_formatter(  
 AM = function(x) {  
 if (is.numeric(x))  
 sprintf("%.1f%%", x \* 100)  
 else  
 x  
 }  
 ) %>%  
 ## Format the Outcome column as numbers  
 colformat\_num(j = "Outcome") %>%  
 ## Set header labels for the table  
 set\_header\_labels(  
 Outcome = "Outcome",  
 AM = "Actual-to-Model"  
 ) %>%  
 autofit() # %>%  
 ## Print the flextable  
 #knitr::knit\_print()  
}) %>%  
 ## Set names for each table based on the factor columns  
 purrr::set\_names(factor\_cols) ->   
 output\_tables  
  
if(output\_format == "html") {  
 output\_tables %>%  
 ## Print the flextable  
 map(.f=knitr::knit\_print) %>%  
 ## Generate a tabset from the list of tables  
 generate\_tabset(  
 tabtitle = "",  
 tablevel = 5  
 ) %>%  
 ## Print the generated tabset  
 cat()   
} else {  
 export\_tables\_to\_excel(  
 output\_tables,  
 paste0(  
 exportsRoot,  
 "\_glm\_univariate\_goodness\_of\_fit\_tables.xlsx"  
 )  
 )  
   
 cat("See included Excel table for additional information.\n")  
}

See included Excel table for additional information.

##### Bivariate Goodness-of-Fit

## Generate a list of unique pairs of factor columns  
pairlist <- data.table()  
for (i in 1:(length(factor\_cols) - 1)) {  
 for (j in (i + 1):length(factor\_cols)) {  
 ## Initialize or append to the pairlist  
 if (i == 1 & j == 2) {  
 pairlist <- data.table(F1 = factor\_cols[i], F2 = factor\_cols[j])  
 } else {  
 pairlist <- rbind(pairlist, data.table(F1 = factor\_cols[i], F2 = factor\_cols[j]))  
 }  
 }  
}  
  
## Generate and format summary tables for each pair of factor columns  
map2(.x = pairlist$F1, .y = pairlist$F2, .f = \(x, y) {  
 xs <- sym(x)  
 ys <- sym(y)  
   
 ## Choose grouping order based on the number of levels in each factor  
 if (length(train[, levels(get(x))]) >= length(train[, levels(get(y))])) {  
 fttmp <- train %>%  
 group\_by(!!xs, !!ys) %>%  
 summarize(  
 Outcome = sum(amount\_actual),  
 Ratio = sprintf("%.1f%%", 100 \* sum(amount\_actual) / sum(predictions\_glm))  
 ) %>%  
 pivot\_wider(  
 names\_from = !!ys,  
 values\_from = c(Outcome, Ratio),  
 names\_glue = paste0(y, ": {", y, "}.{.value}"),  
 names\_vary = "slowest"  
 )  
 } else {  
 fttmp <- train %>%  
 group\_by(!!ys, !!xs) %>%  
 summarize(  
 Outcome = sum(amount\_actual),  
 Ratio = sprintf("%.1f%%", 100 \* sum(amount\_actual) / sum(predictions\_glm))  
 ) %>%  
 pivot\_wider(  
 names\_from = !!xs,  
 values\_from = c(Outcome, Ratio),  
 names\_glue = paste0(x, ": {", x, "}.{.value}"),  
 names\_vary = "slowest"  
 )  
 }  
   
 ## Adjust column keys for the flextable  
 fttmp.colkeys <- names(fttmp)[1]  
 for (i in 1:((length(names(fttmp)) - 1) / 2)) {  
 fttmp.colkeys <- c(fttmp.colkeys, paste0("blank", i), names(fttmp)[(2 \* i):(2 \* i + 1)])  
 }  
   
 ## Create and print the flextable  
 fttmp %>%  
 flextable(col\_keys = fttmp.colkeys) %>%  
 ftExtra::span\_header(sep = "\\.") %>%  
 align(align = 'center', part = "all") %>%  
 empty\_blanks() %>%  
 autofit() #%>%  
 #knitr::knit\_print()  
}) %>%  
 ## Set names for each element in the list based on the factor column pairs  
 purrr::set\_names(pairlist[, paste0(F1, " x ", F2)]) ->  
 output\_tables  
  
if(output\_format == "html") {  
 output\_tables %>%  
 map(.f=knitr::knit\_print) %>%  
 ## Generate a tabset from the list of formatted tables  
 generate\_tabset(tabtitle = "", tablevel = 5) %>%  
 ## Print the generated tabset  
 cat()  
} else {  
 export\_tables\_to\_excel(  
 output\_tables,  
 paste0(  
 exportsRoot,  
 "\_glm\_bivariate\_goodness\_of\_fit\_tables.xlsx"  
 )  
 )  
   
 cat("See included Excel table for additional information.\n")  
}

See included Excel table for additional information.

#### Subgroup Variability

This section reproduces Brian Holland’s publication. For background on the tables generated below, please refer to the publication.

## Load custom functions  
source("R/functions\_BDHGLM.R")  
  
## Generate and format summary tables for each factor column  
factor\_cols %>%  
 map(.f = \(x) {  
 ## Call the mainF function to compute weighted averages for GLM factors  
 mainF(  
 df = ds,  
 model = modelGLM,  
 rf = x,  
 resp = resp\_var,  
 offset = resp\_offset  
 ) %>%  
 ## Create a flextable from the results  
 flextable() %>%  
 ## Set header labels for the table  
 set\_header\_labels("rowname" = "") %>%  
 ## Format table values as percentages if numeric  
 set\_formatter(values = function(x) {  
 if (is.numeric(x))  
 sprintf("%.1f%%", x \* 100)  
 else  
 x  
 }) %>%  
 ## Set a caption for the table  
 set\_caption(caption = paste0("Weighted Average GLM Factors for Variable: ", x)) %>%  
 ## Set table properties to enable scrolling  
 set\_table\_properties(opts\_html = list(  
 scroll = list(  
 add\_css = "max-height: 500px;"  
 )  
 )) %>%  
 autofit() #%>%  
 ## Print the flextable  
 #knitr::knit\_print()  
 }) %>%  
 ## Set names for each table based on the factor columns  
 purrr::set\_names(factor\_cols) ->  
 output\_tables   
  
if(output\_format == "html") {  
 output\_tables %>%  
 map(.f=knitr::knit\_print) %>%  
 ## Generate a tabset from the list of tables  
 generate\_tabset(  
 tabtitle = "Tables of Terms",  
 tablevel = 4  
 ) %>%  
 ## Print the generated tabset  
 cat()  
} else {  
 export\_tables\_to\_excel(  
 output\_tables,  
 paste0(  
 exportsRoot,  
 "\_glm\_subgroup\_variability.xlsx"  
 )  
 )  
 cat("See included Excel table for additional information.\n")  
}

See included Excel table for additional information.

## LightGBM

### Data Preparation

First, the data are prepared for LightGBM. LightGBM expects matrices for its inputs. Thereafter, the LightGBM model is trained. Factors are recast as their underlying integer indices.

### Model Fitting

The LightGBM model is fit to the training subset using a Poisson objective. The model response is the ratio of response variable and response offset, and the weights are the specified offset. Often, this might be “actual claims” as the response and “expected claims” as the offset.

#==============================================================================#  
#### Section 5: lightgbm ####  
#==============================================================================#  
  
#### Notes  
## Helpful Resources:  
## https://lightgbm.readthedocs.io/en/v3.3.2/  
## https://christophm.github.io/interpretable-ml-book/shapley.html  
  
## In this section we fit a lightgbm model, an implementation of gradient   
## boosting machines. We also extract interaction feature importance  
## This is a way to determine most likely interactions for a linear model  
## We also look at Shapley values which can be useful to decompose black-box   
## model predictions  
  
#-----------------------------------------#  
##### Fit model and make predictions #####  
#-----------------------------------------#  
  
## create lgbm dataset  
lgbm.train <- lgb.Dataset(train.x.lgbm,   
 label = train.y.lgbm/train.weight.lgbm,   
 weight = train.weight.lgbm)  
  
lgbm.test <- lgb.Dataset.create.valid(lgbm.train,  
 test.x.lgbm,  
 label=test.y.lgbm/test.weight.lgbm)  
  
## define parameters  
params <- list(  
 objective = "poisson",  
 metric = "poisson",  
 min\_data\_in\_leaf = 500,  
 learning\_rate = .3,  
 feature\_fraction = .75,  
 bagging\_fraction = 0.50,  
 seed = nGBMSeed  
)  
  
## train model   
if(bUseCache & file.exists(  
 paste0(cacheFileRoot,"\_lgb\_model.txt")  
 ) & !bInvalidateCaches)  
{  
 lgbm1 <- lgb.load(paste0(cacheFileRoot,"\_lgb\_model.txt"))  
} else {  
 lgbm1 <- lgb.train(  
 params = params,  
 data = lgbm.train,  
 nrounds = 2000L#, ## for demo purposes; switch back to 2000  
 #valids=list(test=lgbm.test),  
 #early\_stopping\_rounds = 10  
 )  
 if(bUseCache)  
 lgb.save(lgbm1,paste0(cacheFileRoot,"\_lgb\_model.txt"))  
}  
  
## generate predictions  
## note: predictions needed to be multiplied by weights, linear models do this automatically  
test[,predictions\_lgbm1:=predict(lgbm1, test.x.lgbm) \* get(resp\_offset)]  
train[,predictions\_lgbm1:=predict(lgbm1, train.x.lgbm) \* get(resp\_offset)]

### Model Illustrations and Graphics

From this, we can plot decile lift and Lorenz curves.

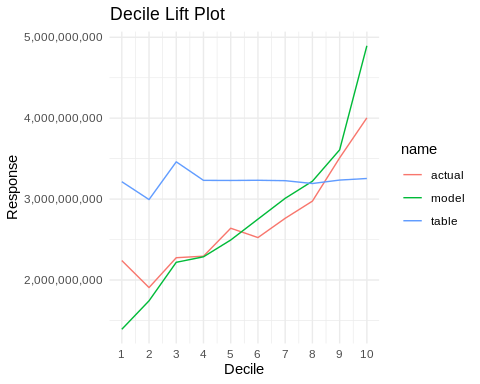
The decile lift plot can be interpreted as a way to visualize the effectiveness of a predictive model. It divides the data into ten parts (deciles) based on the model’s predictions, from the highest probability of an event occurring to the lowest. The steeper the plot against deciles, the better the segmentation or lift. We see three lines. The “table” line indicates that the expected mortality is relatively constant across these model deciles even though the “actual” mortality and the mortality predicted by the “model” vary substantially, indicating significant risk stratification.

The Lorenz curve describes is another way of visualizing the risk stratification of the model. The more bowed the line is from the y=x axis, the greater the Gini coefficient and the greater the risk stratification.

Understanding the behavior of the interactions as well as gain and cover can give us some macro insight into what the model is doing. The feature interaction table ranks and demonstrates the most important interactions in the model. ‘gain’ refers to the improvement in accuracy brought by a feature to the branches it is on, thus indicating the feature is important. ‘cover’ measures the number of times a feature is used to split the data across all trees regardless of the gain in accuracy achieved. A high gain with a high cover suggests a feature that is very useful across many parts of the dataset.

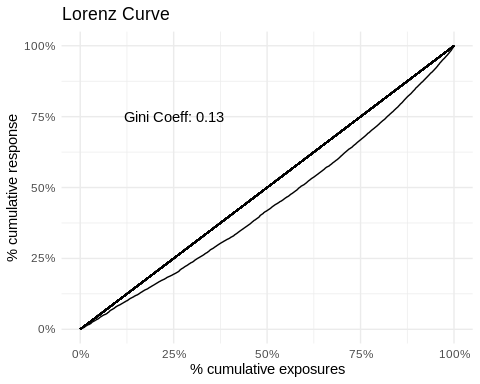
#### Lift Curve

#-----------------------------------------#  
##### Validation Metrics #####  
#-----------------------------------------#  
  
## generate plot  
test[,decile.table(get(resp\_var),predictions\_lgbm1/get(resp\_offset),  
 get(resp\_offset))] %>%  
 pivot\_longer(-c(decile,exposures)) %>%   
 as.data.table() %>%  
 ggplot(aes(x=decile, y=value, col=name)) +   
 geom\_line() +  
 scale\_x\_continuous(breaks=c(1:10)) +  
 labs(x="Decile",y="Response")+  
 ggtitle("Decile Lift Plot") +  
 theme\_minimal() +  
 scale\_y\_continuous(labels=scales::comma)



#### Lorenz Curve

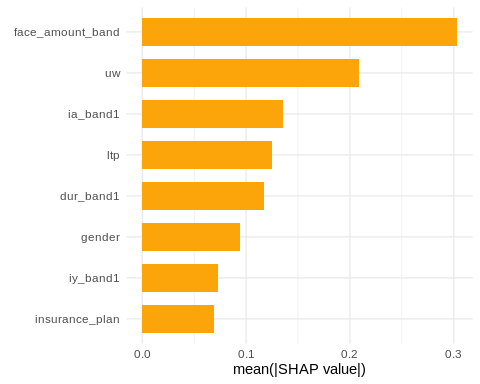
## lorenz plot  
test[,lorenz(get(resp\_var), predictions\_lgbm1 / get(resp\_offset),   
 get(resp\_offset))]



#### Feature Importance

The following plot is the feature importance plot which ranks the mean absolute SHAP value for a given feature. It should be noted that being low on the list does not automatically imply that a feature is unimportant. Due to phenomena such as aggregation bias, features with relatively higher numbers of levels can seemingly rank higher than those with lower numbers of levels. Here, the top three tend to have large numbers of levels versus the bottom four.

#-----------------------------------------#  
##### Feature Importance #####  
#-----------------------------------------#  
  
## get most important features  
if(bUseCache & file.exists(  
 paste0(cacheFileRoot,"\_lgb\_imp.rds")  
 ) & !bInvalidateCaches) {  
 imp <- readRDS(paste0(cacheFileRoot,"\_lgb\_imp.rds"))  
} else {  
 imp <- lgb.importance(lgbm1, percentage = TRUE)  
 if(bUseCache)  
 saveRDS(imp,paste0(cacheFileRoot,"\_lgb\_imp.rds"))  
}  
  
## get most important interactions from EIX library  
## warning: very slow  
if(bUseCache & file.exists(  
 paste0(cacheFileRoot,"\_lgb\_imp\_int.rds")  
 ) & !bInvalidateCaches) {  
 imp.int <- readRDS(paste0(cacheFileRoot,"\_lgb\_imp\_int.rds"))  
} else {  
 imp.int <- importance(lgbm1, sm, option = "interactions")  
 if(bUseCache)  
 saveRDS(imp.int,paste0(cacheFileRoot,"\_lgb\_imp\_int.rds"))  
}  
  
#-----------------------------------------#  
##### Shap Values #####  
#-----------------------------------------#  
  
## get shap values for lightgbm  
if(bUseCache & file.exists(  
 paste0(cacheFileRoot,"\_lgb\_shap.rds")  
 ) & !bInvalidateCaches) {  
 shap\_lgbm <- readRDS(paste0(cacheFileRoot,"\_lgb\_shap.rds"))  
} else {  
 shap\_lgbm <- as.data.table(  
 predict(lgbm1,   
 test.x.lgbm,   
 rawscore = FALSE,   
 predcontrib = TRUE)   
 ) %>%  
 setnames(names(.),  
 c(colnames(test.x.lgbm),"BIAS")  
 )  
 shap\_lgbm[,pred:=exp(Reduce('+',.SD))\*test.weight.lgbm] ## reproduce model predictions  
 if(bUseCache)  
 saveRDS(shap\_lgbm,paste0(cacheFileRoot,"\_lgb\_shap.rds"))  
}  
  
set.seed(1337)  
if(flgbm\_vis\_subset < 1 ) {  
 shp\_int\_subset <- sample.int(n=nrow(train),  
 size=nrow(train)\*flgbm\_vis\_subset)  
} else {  
 shp\_int\_subset <- 1:nrow(train)  
}  
  
if(bUseCache & file.exists(  
 paste0(cacheFileRoot,"\_lgb\_shapviz.rds")  
 ) & !bInvalidateCaches)  
{  
 shp <- readRDS(paste0(cacheFileRoot,"\_lgb\_shapviz.rds"))  
} else {  
 shp <- shapviz(  
 lgbm1,  
 X\_pred=train.x.lgbm[shp\_int\_subset,],  
 X=train[shp\_int\_subset]  
 )  
   
 setDT(shp$X)  
   
 if(bUseCache)  
 saveRDS(shp,paste0(cacheFileRoot,"\_lgb\_shapviz.rds"))  
}  
  
## Feature importance  
sv\_importance(shp) + theme\_minimal()



#### Feature Interaction Table

We also develop a table of interaction strengths, sorted by the total contribution to explaining variation in the data. Again, aggregation bias can distort the ranking, so interpreting the ranking should be taken with caution.

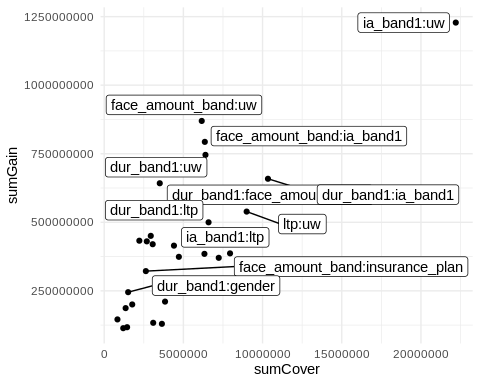
## Convert the 'imp.int' object to a data.table  
imp.int <- data.table(imp.int)  
  
## Create a new column 'Feature2' by sorting and collapsing elements of 'Feature'  
imp.int[, Feature2 := sapply(Feature, FUN = function(f) {  
 paste(sort(unlist(strsplit(f, ":"))), collapse = ":")  
})]  
  
## Aggregate 'sumGain', 'sumCover', and 'frequency' by the new 'Feature2' column  
imp.int2 <- imp.int[, .(  
 sumGain = sum(sumGain),  
 sumCover = sum(sumCover),  
 frequency = sum(frequency)  
), by = .(Feature = Feature2)]  
  
## Calculate additional metrics: meanCover, meanGain, sumGainPct, sumCoverPct  
imp.int2[, `:=`(  
 meanCover = sumCover / frequency,  
 meanGain = sumGain / frequency,  
 sumGainPct = sumGain / sum(sumGain),  
 sumCoverPct = sumCover / sum(sumCover)  
)]  
  
## Split 'Feature' into 'Feature1' and 'Feature2' columns  
imp.int2[, c("Feature1", "Feature2") := tstrsplit(Feature, ":")]  
  
## Order by 'sumGain' in descending order and create a flextable with scrollable properties  
imp.int2[order(-sumGain)] %>%  
 flextable() %>%  
 set\_table\_properties(opts\_html = list(  
 scroll = list(  
 add\_css = "max-height: 500px;"  
 )  
 ))

| Feature | sumGain | sumCover | frequency | meanCover | meanGain | sumGainPct | sumCoverPct | Feature1 | Feature2 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ia\_band1:uw | 1,228,300,000 | 22,200,000 | 3,003 | 7,393 | 409,024 | 0.101945 | 0.160579 | ia\_band1 | uw |
| face\_amount\_band:uw | 869,600,000 | 6,159,000 | 2,112 | 2,916 | 411,742 | 0.072174 | 0.044550 | face\_amount\_band | uw |
| face\_amount\_band:ia\_band1 | 793,300,000 | 6,354,000 | 2,185 | 2,908 | 363,066 | 0.065841 | 0.045960 | face\_amount\_band | ia\_band1 |
| dur\_band1:uw | 745,800,000 | 6,395,000 | 1,677 | 3,813 | 444,723 | 0.061899 | 0.046257 | dur\_band1 | uw |
| dur\_band1:ia\_band1 | 658,700,000 | 10,341,000 | 1,739 | 5,947 | 378,781 | 0.054670 | 0.074800 | dur\_band1 | ia\_band1 |
| dur\_band1:face\_amount\_band | 642,300,000 | 3,509,000 | 1,522 | 2,306 | 422,011 | 0.053309 | 0.025382 | dur\_band1 | face\_amount\_band |
| dur\_band1:ltp | 602,600,000 | 4,825,000 | 1,101 | 4,382 | 547,321 | 0.050014 | 0.034901 | dur\_band1 | ltp |
| ltp:uw | 539,000,000 | 8,995,000 | 1,432 | 6,281 | 376,397 | 0.044735 | 0.065064 | ltp | uw |
| ia\_band1:ltp | 499,900,000 | 6,590,000 | 1,294 | 5,093 | 386,321 | 0.041490 | 0.047667 | ia\_band1 | ltp |
| insurance\_plan:uw | 450,500,000 | 2,941,000 | 683 | 4,306 | 659,590 | 0.037390 | 0.021273 | insurance\_plan | uw |
| face\_amount\_band:gender | 432,800,000 | 2,219,600 | 1,018 | 2,180 | 425,147 | 0.035921 | 0.016055 | face\_amount\_band | gender |
| gender:uw | 430,400,000 | 2,691,000 | 1,119 | 2,405 | 384,629 | 0.035722 | 0.019465 | gender | uw |
| gender:ia\_band1 | 420,000,000 | 3,063,000 | 1,136 | 2,696 | 369,718 | 0.034859 | 0.022156 | gender | ia\_band1 |
| face\_amount\_band:ltp | 414,800,000 | 4,405,000 | 1,179 | 3,736 | 351,824 | 0.034427 | 0.031863 | face\_amount\_band | ltp |
| iy\_band1:uw | 386,600,000 | 7,947,000 | 1,133 | 7,014 | 341,218 | 0.032087 | 0.057483 | iy\_band1 | uw |
| ia\_band1:iy\_band1 | 384,800,000 | 6,330,000 | 1,041 | 6,081 | 369,645 | 0.031937 | 0.045787 | ia\_band1 | iy\_band1 |
| face\_amount\_band:iy\_band1 | 374,000,000 | 4,715,000 | 1,023 | 4,609 | 365,591 | 0.031041 | 0.034105 | face\_amount\_band | iy\_band1 |
| ia\_band1:insurance\_plan | 370,400,000 | 7,233,000 | 964 | 7,503 | 384,232 | 0.030742 | 0.052318 | ia\_band1 | insurance\_plan |
| face\_amount\_band:insurance\_plan | 321,900,000 | 2,630,400 | 734 | 3,584 | 438,556 | 0.026717 | 0.019026 | face\_amount\_band | insurance\_plan |
| dur\_band1:gender | 245,090,000 | 1,512,500 | 688 | 2,198 | 356,235 | 0.020342 | 0.010940 | dur\_band1 | gender |
| dur\_band1:iy\_band1 | 210,720,000 | 3,843,000 | 631 | 6,090 | 333,946 | 0.017489 | 0.027798 | dur\_band1 | iy\_band1 |
| dur\_band1:insurance\_plan | 200,320,000 | 1,770,400 | 529 | 3,347 | 378,677 | 0.016626 | 0.012806 | dur\_band1 | insurance\_plan |
| insurance\_plan:ltp | 186,950,000 | 1,353,500 | 164 | 8,253 | 1,139,939 | 0.015516 | 0.009790 | insurance\_plan | ltp |
| gender:insurance\_plan | 145,750,000 | 837,300 | 310 | 2,701 | 470,161 | 0.012097 | 0.006056 | gender | insurance\_plan |
| insurance\_plan:iy\_band1 | 133,400,000 | 3,095,000 | 427 | 7,248 | 312,412 | 0.011072 | 0.022387 | insurance\_plan | iy\_band1 |
| iy\_band1:ltp | 129,660,000 | 3,644,000 | 556 | 6,554 | 233,201 | 0.010761 | 0.026358 | iy\_band1 | ltp |
| gender:iy\_band1 | 117,310,000 | 1,447,700 | 421 | 3,439 | 278,646 | 0.009736 | 0.010472 | gender | iy\_band1 |
| gender:ltp | 113,770,000 | 1,203,000 | 471 | 2,554 | 241,550 | 0.009443 | 0.008702 | gender | ltp |

#### Gain vs. Cover

As noted above, ‘gain’ refers to the improvement in accuracy brought by a feature to the branches it is on, thus indicating the feature is important. ‘cover’ measures the number of times a feature is used to split the data across all trees regardless of the gain in accuracy achieved. A high gain with a high cover suggests a feature that is very useful across many parts of the dataset.

## Create scatter plot  
ggplot(imp.int2, aes(x = sumCover, y = sumGain, label = Feature)) +   
 geom\_point() +   
 scale\_size() +   
 ggrepel::geom\_label\_repel() + # Add labels with repulsion  
 theme\_minimal()



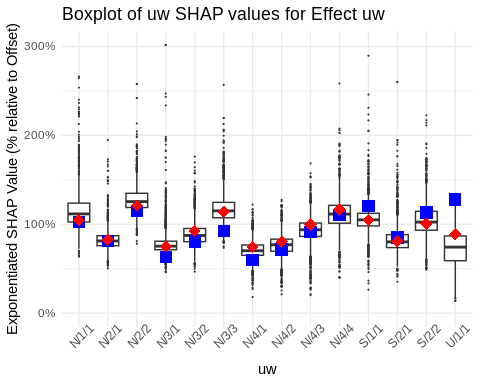
### Feature Plots

It is useful to plot SHAP values for their main effects (e.g., SHAP values for face amount band by face amount band) as well as interactions (e.g., same, but stratified in some way by other variables). Traditionally, scatter plots are used. However, due to overplotting, it is not clear what is going on with the SHAP values. Here we use boxplots of the SHAP values instead of scatter plotting. This provides a sense of the spread of the SHAP values along with the median and outliers. This is particularly useful for qualitatively evaluating whether there are any meaningful interactions.

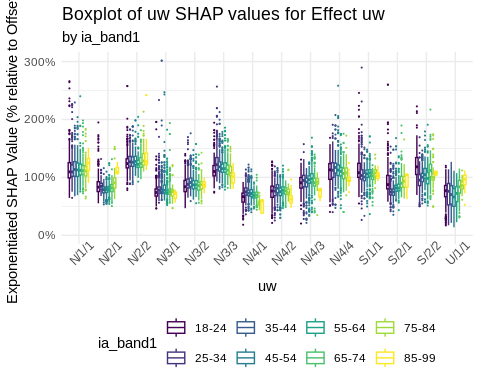
In what follows, red diamonds are mean SHAP values, while blue squares are mean mortality from a subset of the data. Note that SHAP values are partial effects which work in concern with the other features. Therefore, the mean actual mortality will not necessarily be captured by the variability of the feature SHAP values.

## Load external R script  
source("R/ilec\_shap\_plot.R")  
  
## Select top features to plot  
featurestoplot <- imp[1:nPlotTopFeatures, Feature]  
  
## Initialize plot list  
plist <- list()  
  
## Loop through top features  
for (i in 1:nPlotTopFeatures) {  
 ## Filter interactions for current feature  
 int.vars <- imp.int2[featurestoplot[i] == Feature1 | featurestoplot[i] == Feature2] %>%  
 head(nPlotTopInteractions) %>%  
 select(Feature1, Feature2) %>%  
 pivot\_longer(cols = c(Feature1, Feature2), values\_to = "Feature") %>%  
 distinct() %>%  
 filter(Feature != featurestoplot[i])  
   
 ## Add SHAP plot to list  
 plist <- c(plist,   
 ilec\_shap\_plot(  
 shp,  
 featurestoplot[i],  
 int.vars$Feature,  
 resp\_var = resp\_var,  
 resp\_offset = resp\_offset,  
 train.data = train[shp\_int\_subset]  
 )  
 )  
}  
  
## Print plots with headers  
plist %>%  
 iwalk(~ {  
 cat('#### ', .y, '\n\n')  
 print(.x)  
 cat('\n\n')  
 })

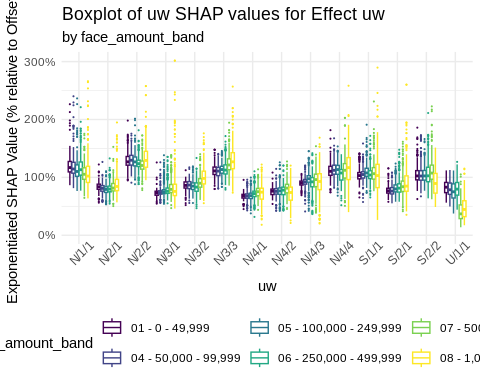
#### main effect: uw



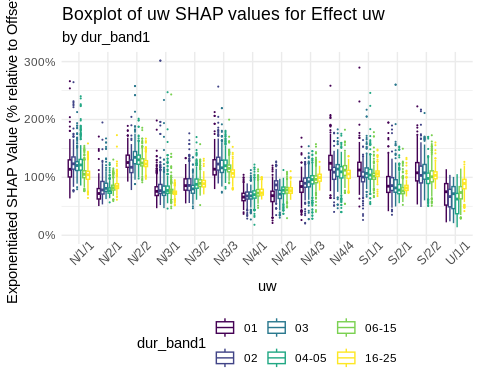
#### uw x ia\_band1



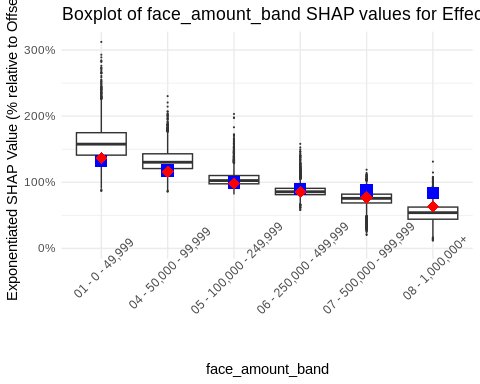
#### uw x face\_amount\_band



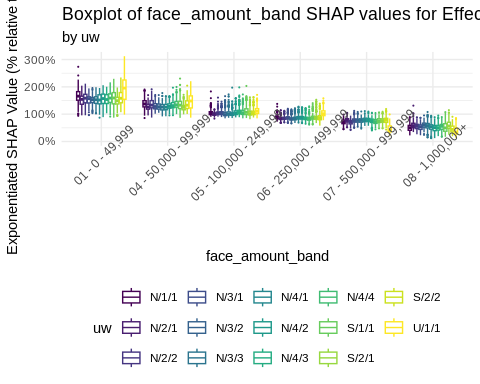
#### uw x dur\_band1



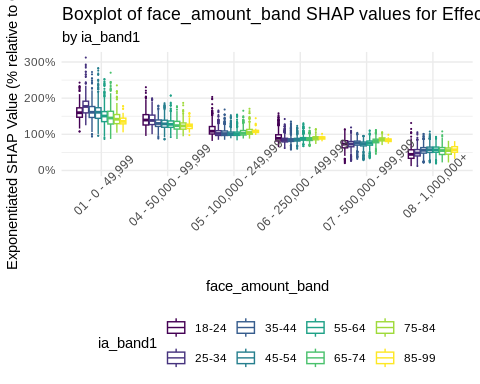
#### main effect: face\_amount\_band



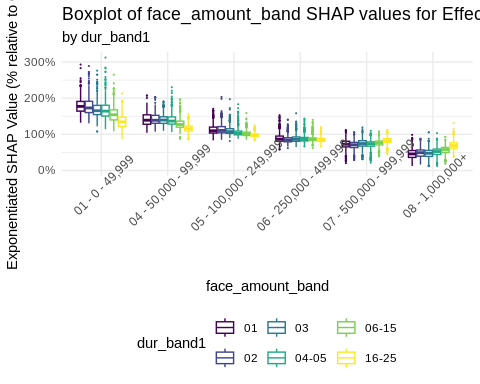
#### face\_amount\_band x uw



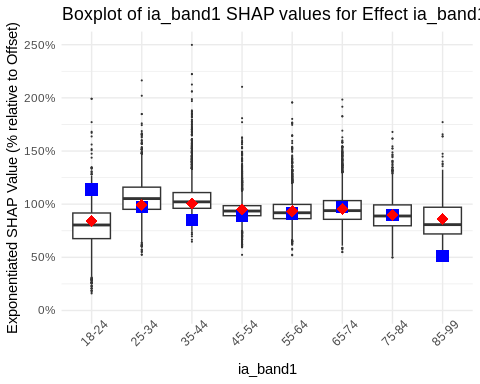
#### face\_amount\_band x ia\_band1



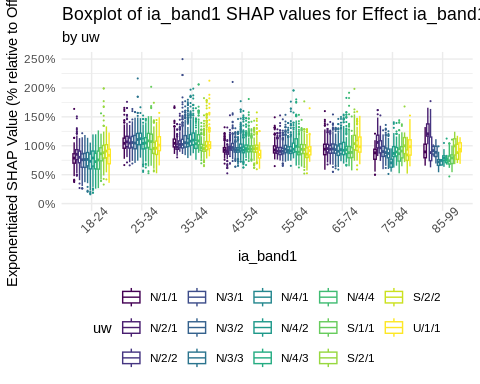
#### face\_amount\_band x dur\_band1



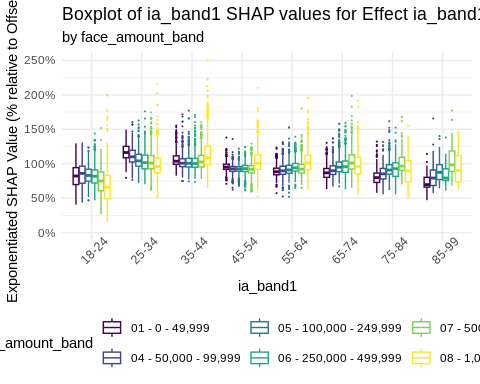
#### main effect: ia\_band1



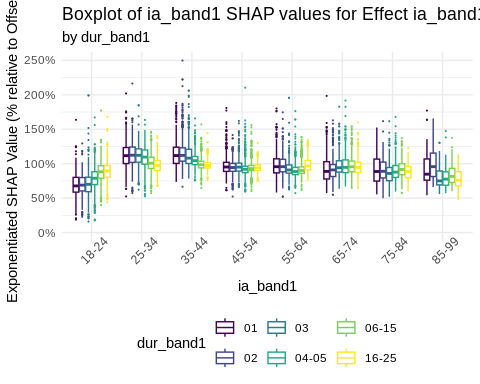
#### ia\_band1 x uw



#### ia\_band1 x face\_amount\_band



#### ia\_band1 x dur\_band1



Some patterns are noticeable. We discuss them for each group. You can visually detect an interaction by checking whether the box plots are all on the same level or not for a given subgroup.

Underwriting (uw)

1. Main effect
   1. The spreads from highest to lowest risk classes are similar across non-smoker class systems.
   2. Smoker differentiation is narrower than for 2-class non-smokers.
2. Interaction with face amount band
   1. The interaction between underwriting and face amount band, for the underwriting effect, appears confined mostly to 3-class non-smoker (N/3/\*). Higher face amount bands ($250K+ in the light dataset) appear to have larger spread of effects.
3. Interaction with issue age band: possible narrowing at older ages for 2- or 4-class non-smokers
4. Interaction with observation year: possible widening of spread of 4-class non-smokers with increasing observation year

Face Amount Band (face\_amount\_band)

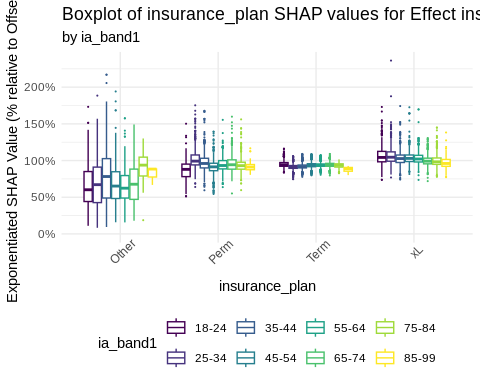
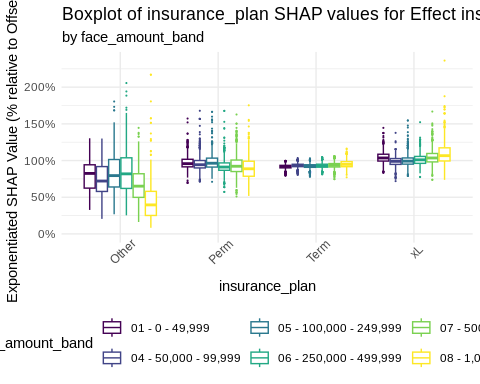
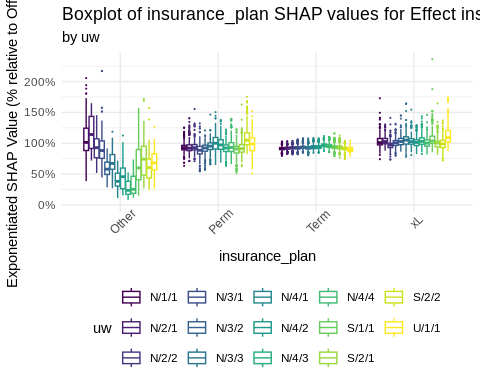
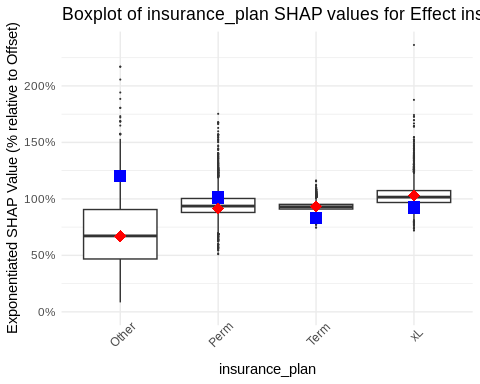
1. Main effect: expected decrease as face amount band increases
2. Interaction with underwriting: face amount effect may be interacting with the Unknown smoker category
3. Interaction with issue age band:
   1. Decreasing effect by issue age for lower bands, flipping to increasing effect by issue age for upper issue age bands
   2. Put another way, spread of face amount effects decreases with increasing issue age
4. Interaction with Observation Year: no obvious effect

Issue Age Band (ia\_band1)

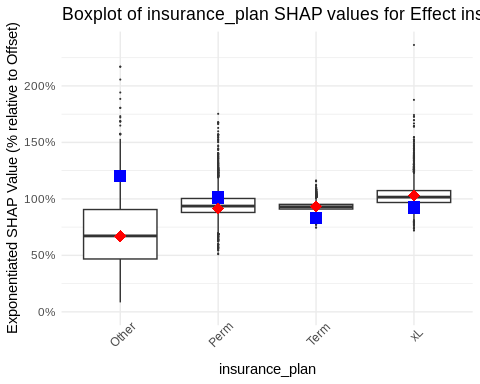
1. Main Effect: With the exception of ages 18-24, decreasing issue age effect by issue age
2. Interaction with face amount band: similar to face amount band, spread decreases with increasing issue age
3. Interaction with underwriting: substantial changes above issue age 75, qualitatively negligible below age 75
4. Interaction with observation year: no obvious interaction

Since the focus of this analysis is mortality differences by product, we also call attention to the same information for insurance\_plan.

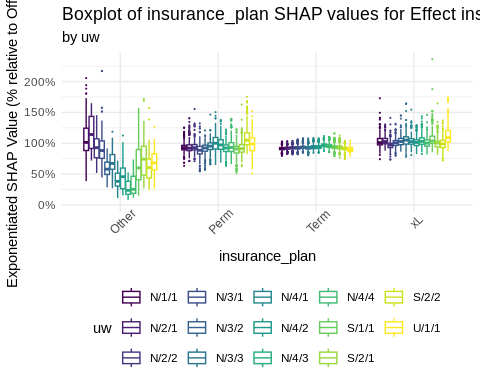
## Filter interactions for 'insurance\_plan'  
int.vars <- imp.int2["insurance\_plan" == Feature1 | "insurance\_plan" == Feature2] %>%  
 head(nPlotTopInteractions) %>%  
 select(Feature1, Feature2) %>%  
 pivot\_longer(cols = c(Feature1, Feature2), values\_to = "Feature") %>%  
 distinct() %>%  
 filter(Feature != "insurance\_plan")  
  
## Generate SHAP plot for 'insurance\_plan'  
plist <- ilec\_shap\_plot(  
 shp,  
 "insurance\_plan",  
 int.vars$Feature,  
 resp\_var = resp\_var,  
 resp\_offset = resp\_offset,  
 train.data = train[shp\_int\_subset]  
)  
  
## Plot each item in the plot list  
lapply(plist, plot)



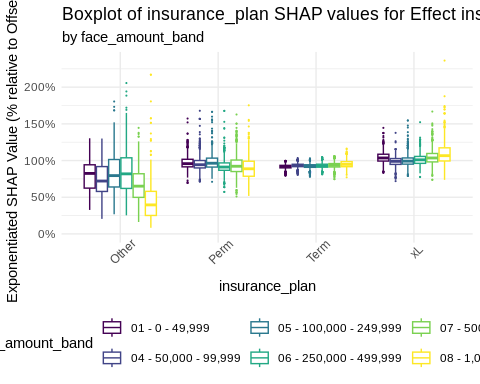
## $`main effect: insurance\_plan`



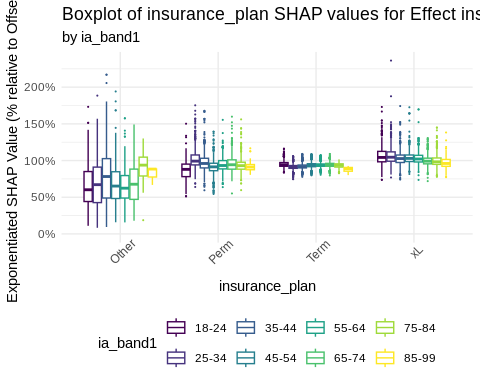
##   
## $`insurance\_plan x uw`



##   
## $`insurance\_plan x face\_amount\_band`



##   
## $`insurance\_plan x ia\_band1`



1. Main Effect: Perm and Term shap distributions are qualitatively similar, with the xL class higher.
2. Interaction with underwriting:
   1. Substantial interaction with the “Other” category
   2. No obvious interaction for Perm and Term
3. Interaction with face amount band:
   1. No obvious interactions with Perm and Term
   2. Evidence for interaction with “Other” and xL
4. Interaction with issue age
   1. No obvious interaction with Perm and Term
   2. Evidence for interaction with “Other” and xL

### Goodness-of-Fit

Goodness-of-fit tables are provided. Each table provides actual-to-model ratios for single variables and for 2-way combinations of variables. A model is qualitatively deemed to perform well if goodness-of-fit ratios are close to 100% in almost all situations. The quantitative assessment using significance testing is omitted here.

#### Unvariate Goodness-of-Fit

## Process each factor column and create formatted tables  
map(factor\_cols, .f = \(x) {  
 ## Convert column name to symbol  
 x <- sym(x)  
   
 ## Group data by factor column and calculate summary statistics  
 train %>%  
 group\_by(!!x) %>%  
 summarize(  
 Outcome = sum(amount\_actual),  
 AM = sum(amount\_actual) / sum(predictions\_lgbm1)  
 ) %>%  
 flextable() %>% # Create a flextable  
 set\_formatter( # Format 'AM' as a percentage  
 AM = function(x) {  
 if (is.numeric(x))  
 sprintf("%.1f%%", x \* 100)  
 else  
 x  
 }  
 ) %>%  
 colformat\_num(j = "Outcome") %>% # Format 'Outcome' column  
 set\_header\_labels( # Set custom header labels  
 Outcome = "Outcome",  
 AM = "Actual-to-Model"  
 ) %>%  
 autofit() #%>%  
 #knitr::knit\_print() # Print the table in a format suitable for knitting  
}) %>% # Set names for each element in the list  
 purrr::set\_names(factor\_cols) ->  
 output\_tables  
  
if(output\_format == "html") {  
 output\_tables %>%  
 map(.f=knitr::knit\_print) %>%  
 generate\_tabset( # Generate tabset from the list of tables  
 tabtitle = "",  
 tablevel = 4  
 ) %>%  
 cat() # Print the tabset  
} else {  
 export\_tables\_to\_excel(  
 output\_tables,  
 paste0(  
 exportsRoot,  
 "\_lightgbm\_univariate\_goodness\_of\_fit\_tables.xlsx"  
 )  
 )  
   
 cat("See included Excel table for additional information.\n")  
   
}

See included Excel table for additional information.

#### Bivariate Goodness-of-Fit

## Create a list of unique pairs of factor columns  
pairlist <- data.table()  
for (i in 1:(length(factor\_cols) - 1)) {  
 for (j in (i + 1):length(factor\_cols)) {  
 pairlist <- rbind(pairlist, data.table(F1 = factor\_cols[i], F2 = factor\_cols[j]))  
 }  
}  
  
## Generate summary tables and formatted outputs for each pair of factor columns  
map2(.x = pairlist$F1, .y = pairlist$F2, .f = \(x, y) {  
 ## Convert column names to symbols for tidy evaluation  
 xs <- sym(x)  
 ys <- sym(y)  
   
 ## Choose grouping order based on the number of levels in each factor  
 ## Group by the factor with more levels first for better summarization  
 if (length(train[, levels(get(x))]) >= length(train[, levels(get(y))])) {  
 fttmp <- train %>%  
 group\_by(!!xs, !!ys) %>%  
 summarize(  
 Outcome = sum(amount\_actual), # Calculate the total outcome  
 Ratio = sprintf("%.1f%%", 100 \* sum(amount\_actual) / sum(predictions\_lgbm1)) # Calculate the actual-to-model ratio  
 ) %>%  
 pivot\_wider(  
 names\_from = !!ys, # Pivot the data to widen by the second factor  
 values\_from = c(Outcome, Ratio), # Use Outcome and Ratio as values  
 names\_glue = paste0(y, ": {", y, "}.{.value}"), # Create new column names using glue syntax  
 names\_vary = "slowest" # Handle varying names by the slowest changing variable  
 )  
 } else {  
 fttmp <- train %>%  
 group\_by(!!ys, !!xs) %>%  
 summarize(  
 Outcome = sum(amount\_actual), # Calculate the total outcome  
 Ratio = sprintf("%.1f%%", 100 \* sum(amount\_actual) / sum(predictions\_lgbm1)) # Calculate the actual-to-model ratio  
 ) %>%  
 pivot\_wider(  
 names\_from = !!xs, # Pivot the data to widen by the first factor  
 values\_from = c(Outcome, Ratio), # Use Outcome and Ratio as values  
 names\_glue = paste0(x, ": {", x, "}.{.value}"), # Create new column names using glue syntax  
 names\_vary = "slowest" # Handle varying names by the slowest changing variable  
 )  
 }  
   
 ## Adjust column keys for the flextable  
 ## Start with the first column name  
 fttmp.colkeys <- names(fttmp)[1]  
 ## Add pairs of Outcome and Ratio columns, inserting a blank column between each pair  
 for (i in 1:((length(names(fttmp)) - 1) / 2)) {  
 fttmp.colkeys <- c(fttmp.colkeys, paste0("blank", i), names(fttmp)[(2 \* i):(2 \* i + 1)])  
 }  
   
 ## Create and print the flextable  
 fttmp %>%  
 flextable(col\_keys = fttmp.colkeys) %>%  
 ftExtra::span\_header(sep = "\\.") %>%  
 align(align = 'center', part = "all") %>%  
 empty\_blanks() %>%  
 autofit() #%>%  
 #knitr::knit\_print()  
}) %>%  
 ## Set names for each element in the list based on the factor column pairs  
 purrr::set\_names(pairlist[, paste0(F1, " x ", F2)]) ->  
 output\_tables  
  
if(output\_format == "html") {  
 output\_tables %>%  
 map(.f=knitr::knit\_print) %>%  
 ## Generate a tabset from the list of formatted tables  
 generate\_tabset(tabtitle = "", tablevel = 4) %>%  
 ## Print the generated tabset  
 cat()   
} else {  
 export\_tables\_to\_excel(  
 output\_tables,  
 paste0(  
 exportsRoot,  
 "\_lightgbm\_bivariate\_goodness\_of\_fit\_tables.xlsx"  
 )  
 )  
   
 cat("See included Excel table for additional information.\n")  
}

See included Excel table for additional information.

## Elastic Net GLM

### Background

Elastic net regularization allows the modeler to combine both LASSO and ridge penalties into a single model.

As one may recall, ordinary least squares regression requires minimizing the squared difference of the response variable and the predicted values. In symbols,

This is equivalent to maximum likelihood estimation, where one assumes that the response variable is normally distributed with mean and variance . The maximum is taken with respect to , and the variance parameter is assumed to be fixed but unknown.

The LASSO and ridge regression methods each add an additional penalty term on the coefficients . The LASSO adds the sum of the absolute values of the parameters subject to a tunable weight, . This term incentivizes the fitting algorithm to fit toward parameter values close to 0.

The ridge penalty adds the sum of the squares of the parameters , subject to a tunable weight, . This term also incentivizes the fitting algorithm to fit toward parameter values close to 0.

What may be new to some readers is that in both cases, for special or , the minimizers of these expressions correspond to the Bayesian maximum a posteriori (MAP) estimators for specific prior distributions for . In the ridge case, the prior is the normal distribution with mean 0 and covariance for some assumed .

For the LASSO, the prior is the double-exponential or Laplace distribution with mean 0 and parameter .

In either case, it can be shown that if and are known, the penalizing weights have unique solutions and are equivalent to the term in Bühlmann credibility. In practice, the penalizing weights are unknown and must be tuned. The resulting optimal is also credible from a Bayesian perspective. Moreover, it can be shown that these facts carry over to the GLM case.

### Data Preparation

Elastic net GLMs as implemented in the glmnet package require that the inputs be converted to model matrices.

### Model Fitting

Once the data are set up, we can calibrate a LASSO penalty, lambda, using n-fold cross-validation.

## In this section we fit a penalized linear regression model using glmnet.   
## We first fine-tune hyperparameter alpha using cross-validation  
## Then we refit the model using the optimal alpha  
  
#-----------------------------------------#  
##### Perform Cross-Validation #####  
#-----------------------------------------#  
## Load or fit the cross-validated glmnet model  
  
## Check if the cached model exists and is valid  
if (bUseCache & file.exists(paste0(cacheFileRoot, "\_glmnet\_int\_cv\_model.rds")) & !bInvalidateCaches) {  
 ## Load the cached model if it exists  
 cvfit <- readRDS(paste0(cacheFileRoot, "\_glmnet\_int\_cv\_model.rds"))  
} else {  
 ## Initialize a cluster for parallel processing  
 cl <- makeCluster(nGLMNetCores)  
 registerDoParallel(cl)  
   
 ## Set glmnet control options, enable iteration trace if debugging  
 glmnet.control(itrace = ifelse(bDebug, 1, 0))  
   
 ## Set random seed for reproducibility  
 set.seed(nELSeed)  
   
 ## Perform cross-validated glmnet model fitting  
 cvfit <- cv.glmnet(  
 train.x.net,   
 train.y.net,   
 offset = log(train.offset.net),   
 family = "poisson",   
 alpha = fGLMNetAlpha,  
 parallel = TRUE  
 )  
   
 ## Stop the parallel cluster  
 stopCluster(cl)  
   
 ## Save the model to cache if caching is enabled  
 if (bUseCache) {  
 saveRDS(cvfit, paste0(cacheFileRoot, "\_glmnet\_int\_cv\_model.rds"))  
 }  
}  
  
## Generate predictions for the test data (handling offsets correctly)  
test[, predictions\_glmnet := predict(  
 cvfit,  
 s = "lambda.min",  
 newx = test.x.net,   
 type = "response",   
 newoffset = log(test.offset.net)  
)]  
  
## Generate predictions for the training data (handling offsets correctly)  
train[, predictions\_glmnet := predict(  
 cvfit,  
 s = "lambda.min",  
 newx = train.x.net,   
 type = "response",   
 newoffset = log(train.offset.net)  
)]

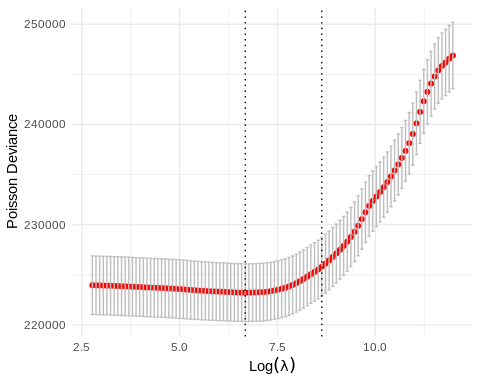
#### Model Plots and Tables

An important by-product of the n-fold cross validation is the plot of values. The optimal choice of is the lowest, and the model associated with that is the final model.

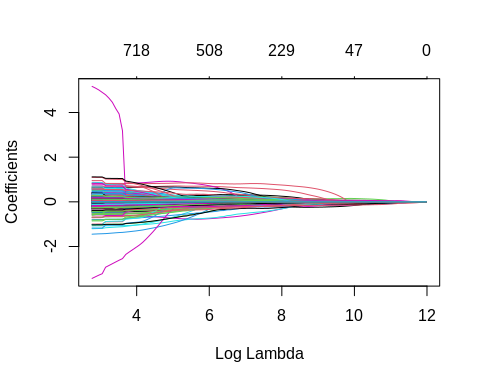
One can also plot the trajectory of coefficients as progressively higher imposes ever harsher penalties on the coefficients.

##### Lambda Plot

## plot lambda vs deviance  
  
data.table(lambda=cvfit$lambda,  
 cvm=cvfit$cvm,  
 cvlo=cvfit$cvlo,  
 cvup=cvfit$cvup,  
 nzero=cvfit$nzero) %>%  
 ggplot(aes(x=log(lambda))) +  
 geom\_point(aes(y=cvm),color="red") +  
 geom\_errorbar(aes(ymin=cvlo,ymax=cvup),color="grey") +  
 scale\_y\_continuous(name=cvfit$name,  
 limits = range(cvfit$cvup,cvfit$cvlo)) +  
 scale\_x\_continuous(name=expression(Log(lambda))) +  
 geom\_vline(xintercept=log(cvfit$lambda.min),linetype=3) +  
 geom\_vline(xintercept = log(cvfit$lambda.1se),linetype=3) +  
 theme\_minimal()

 ##### Coefficient Penalization

plot(cvfit$glmnet.fit,xvar="lambda")



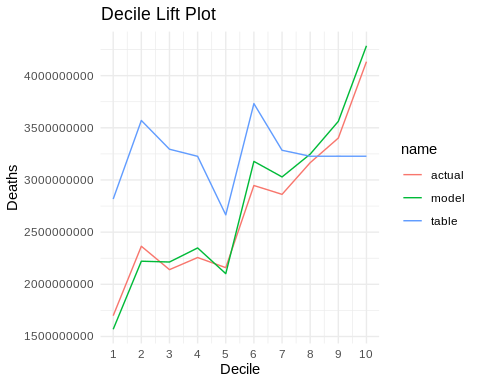
##### Final Model

#---------------------------------------------------------------#  
##### Examine model for minimum lambda and make predictions #####  
#---------------------------------------------------------------#  
  
## get coefficients for fitted version, reformatted and filtered  
reformatCoefs(cvfit, pred\_cols) %>%  
 filter(Coef != 0) %>%  
 select(Feature1Name,  
 Feature1Level,  
 Feature2Name,  
 Feature2Level,  
 Coef) %>%  
 mutate(Coef=exp(Coef)) %>%  
 flextable() %>%  
 highlight(j="Coef",color="yellow",part="body", i = ~ abs(log(Coef)) > log(1.05)) %>%  
 set\_formatter(  
 Coef=function(x) paste0(sprintf("%.01f", 100\*x),"%")  
 ) %>%  
 set\_table\_properties(opts\_html=list(  
 scroll=list(  
 add\_css="max-height: 500px;"  
 )  
 )  
 ) %>%  
 autofit() ->  
 output\_table  
  
if(output\_format == "html") {  
 output\_tables  
} else {  
 wb <- wb\_workbook()  
   
 wb$add\_worksheet("GLMNet Coefficients")  
   
 wb <- wb\_add\_flextable(  
 wb,  
 "GLMNet Coefficients",  
 output\_table  
 )  
   
 wb$save(  
 paste0(  
 exportsRoot,  
 "\_glmnet\_coefficients\_table.xlsx"  
 )  
 )  
   
 cat("See included Excel table for additional information.\n")  
}

## See included Excel table for additional information.

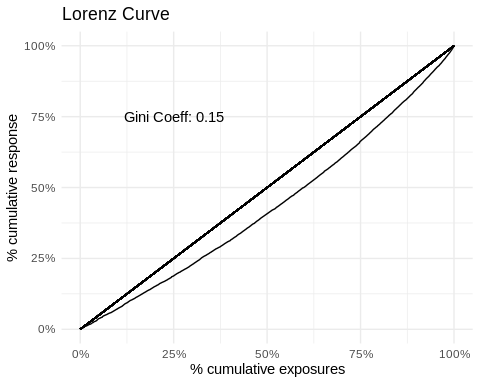
##### Lift

#-----------------------------------------#  
##### Calculate Validation Metrics #####  
#-----------------------------------------#  
  
## generate plot  
test[,decile.table(get(resp\_var), predictions\_glmnet/get(resp\_offset), get(resp\_offset))] %>%  
 pivot\_longer(-c(decile,exposures)) %>%  
 as.data.table() %>%  
 ggplot(aes(x=decile, y=value, col=name)) +   
 geom\_line() +  
 scale\_x\_continuous(breaks=c(1:10)) +  
 labs(x="Decile",y="Deaths") +  
 theme\_minimal() +  
 ggtitle("Decile Lift Plot")



##### Lorenz Curve

## lorenz plot  
test[,lorenz(get(resp\_var), predictions\_glmnet / get(resp\_offset), get(resp\_offset))]



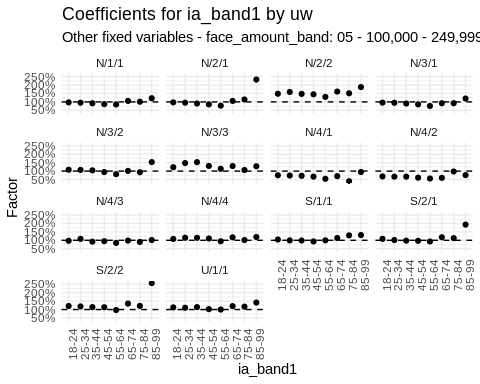
### Plots of Terms

## Generate a grid of predictor levels for the training data  
train[, ..pred\_cols] %>%  
 lapply(levels) %>%  
 expand.grid() %>%  
 setDT() -> train.grid  
  
## Create model matrix for the training grid  
train.grid %>%  
 model.Matrix(  
 object = glmnetFormula,  
 data = .,  
 sparse = bUseSparse  
 ) %>%  
 ## Predict new coefficients using the fitted cv.glmnet model  
 predict(  
 cvfit,  
 newx = .,  
 s = "lambda.min",  
 newoffset = rep(0, nrow(.))  
 ) %>%  
 as.vector() -> newCoef  
  
## Add the predicted factors to the training grid  
train.grid %>%  
 add\_column(Factor = exp(newCoef)) %>%  
 setDT() -> train.grid  
  
## Reformat coefficients and generate the list of interactions  
reformatCoefs(cvfit, pred\_cols) %>%  
 filter(Coef != 0 & !is.na(Feature2Name)) %>%  
 select(Feature1Name, Feature2Name) %>%  
 distinct() %>%  
 as.list() %>%  
 purrr::list\_transpose() -> glmnet.int.list

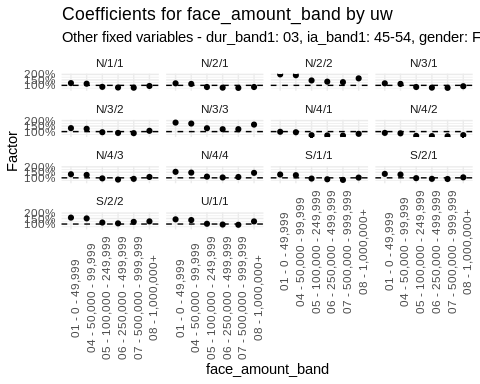
Below are plots of the 2-way interaction terms, with external factors fixed at their middle values.

## Generate and print plots for each interaction pair in glmnet.int.list  
glmnet.int.list %>%  
 map(.f = \(x) {  
 ## Generate a plot of cross-validated net coefficients  
 plotCVNetCoefs(  
 train.grid,  
 sort(x),  
 "Factor",  
 pred\_cols  
 )  
 }) %>%  
 ## Set names for each plot based on the interaction pairs  
 purrr::set\_names(  
 map(glmnet.int.list, \(x) paste0(x, collapse = " x "))  
 ) %>%  
 ## Print each plot with a corresponding title  
 iwalk(~ {  
 cat('#### ', .y, '\n\n') # Print the plot title  
 print(.x) # Print the plot  
 cat('\n\n') # Add spacing after each plot  
 })

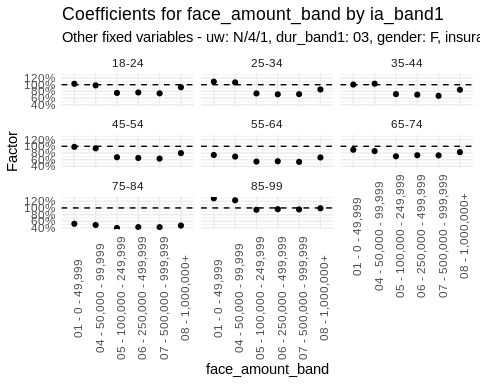
#### uw x ia\_band1



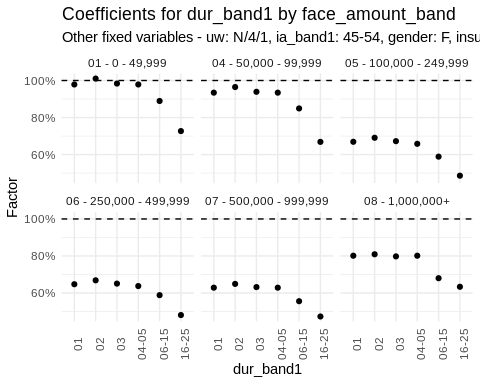
#### uw x face\_amount\_band



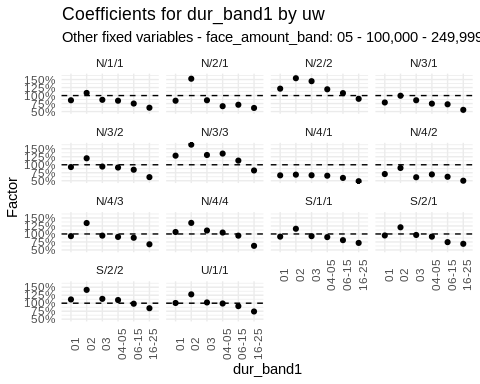
#### face\_amount\_band x ia\_band1



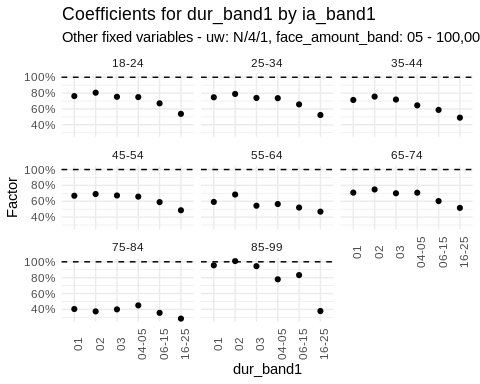
#### face\_amount\_band x dur\_band1



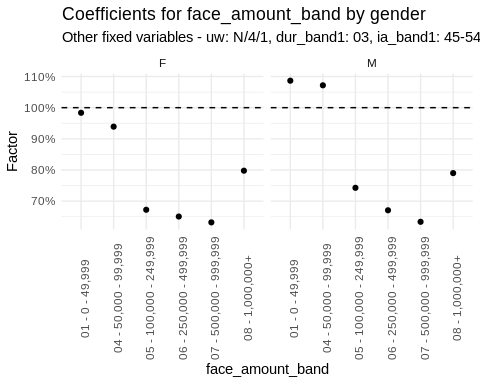
#### uw x dur\_band1



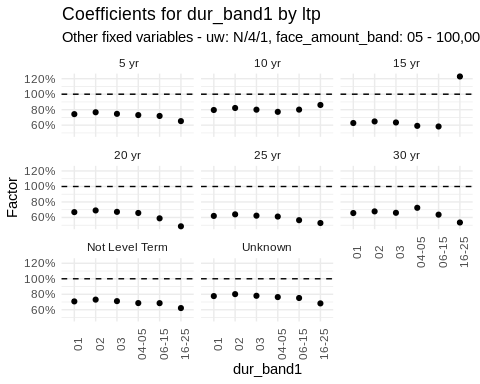
#### dur\_band1 x ia\_band1



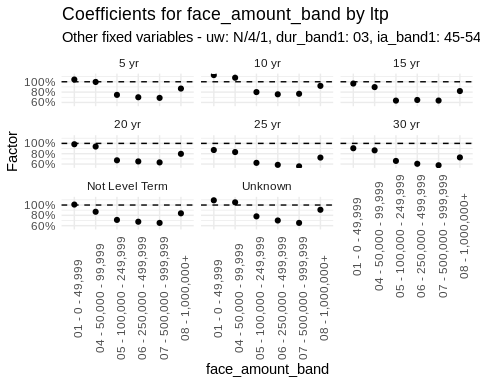
#### face\_amount\_band x gender



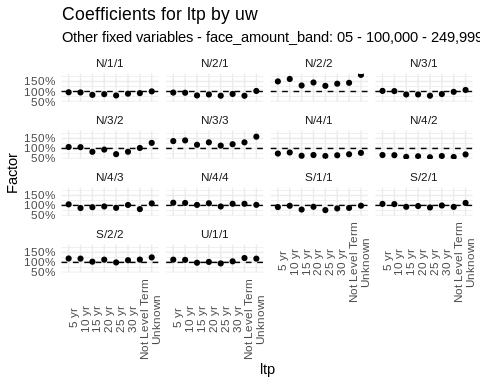
#### dur\_band1 x ltp



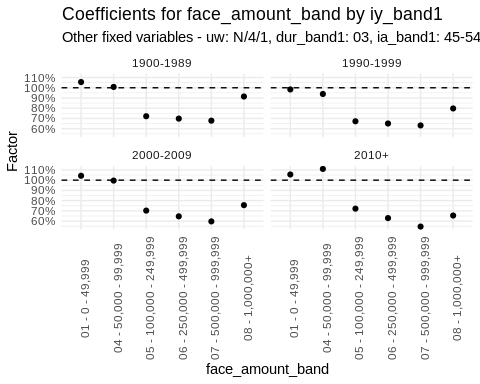
#### face\_amount\_band x ltp



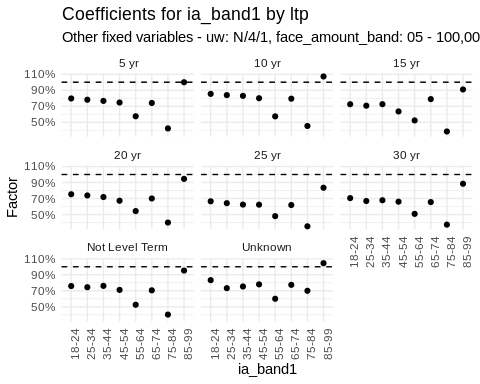
#### uw x ltp



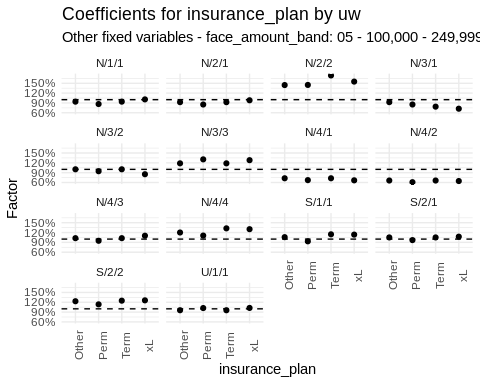
#### face\_amount\_band x iy\_band1



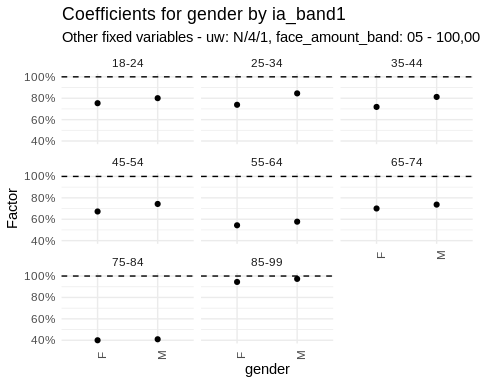
#### ia\_band1 x ltp



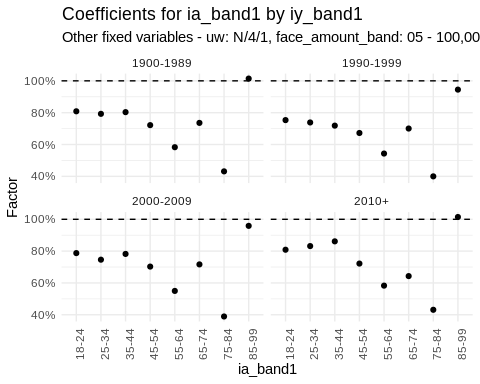
#### uw x insurance\_plan



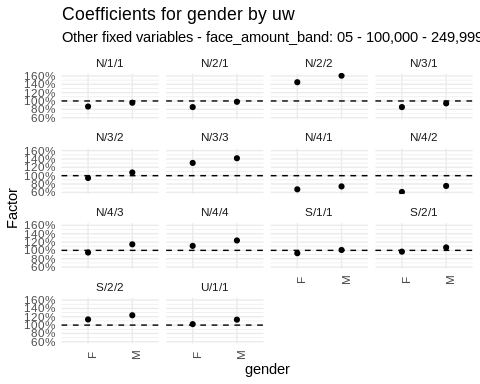
#### ia\_band1 x gender



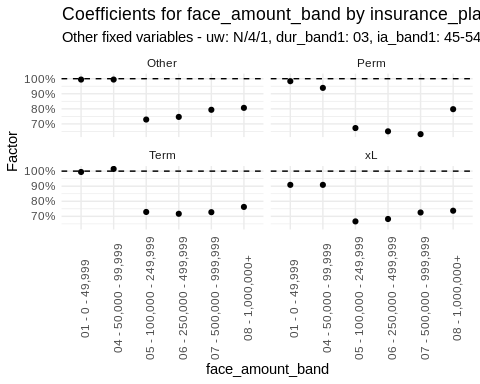
#### ia\_band1 x iy\_band1



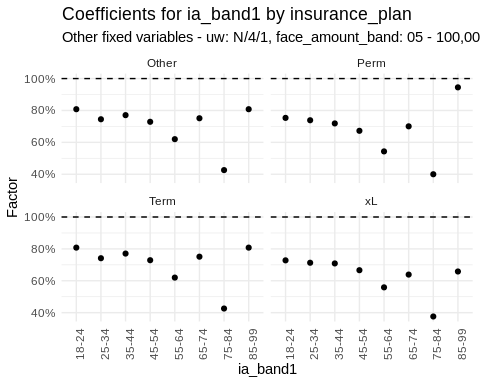
#### uw x gender



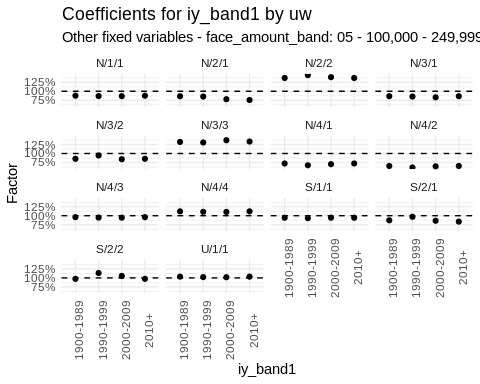
#### face\_amount\_band x insurance\_plan



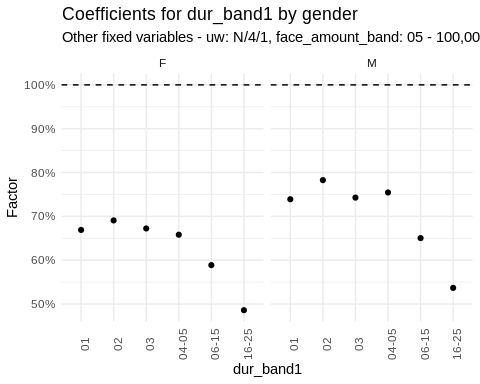
#### ia\_band1 x insurance\_plan



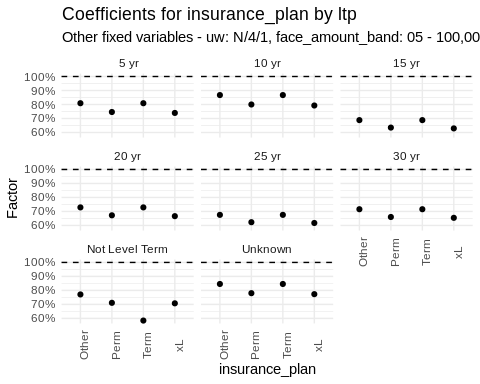
#### uw x iy\_band1



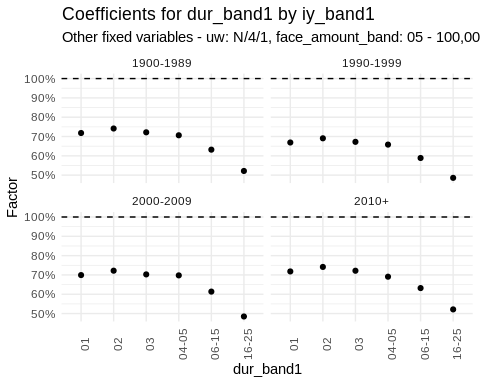
#### dur\_band1 x gender



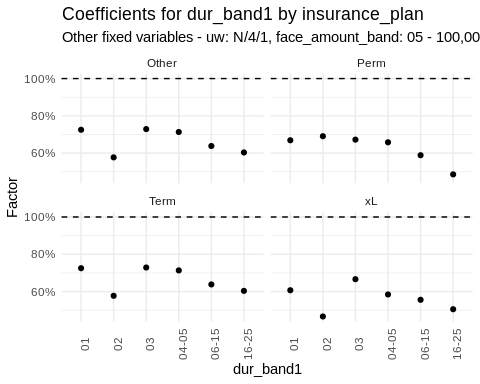
#### insurance\_plan x ltp



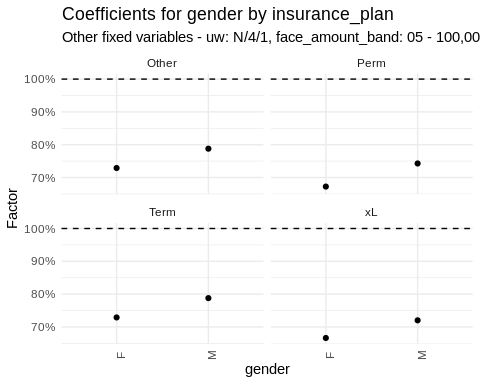
#### dur\_band1 x iy\_band1



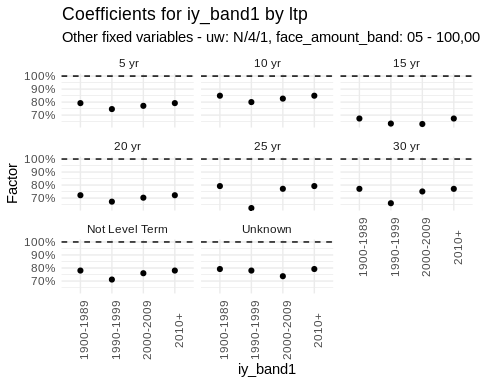
#### dur\_band1 x insurance\_plan



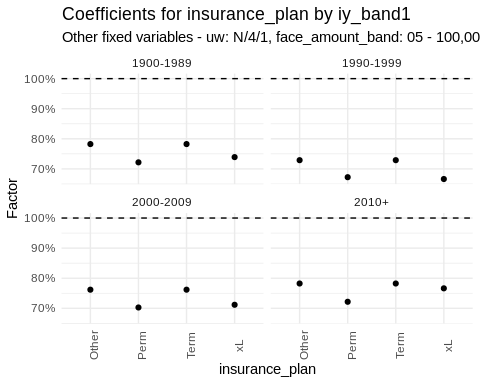
#### gender x insurance\_plan



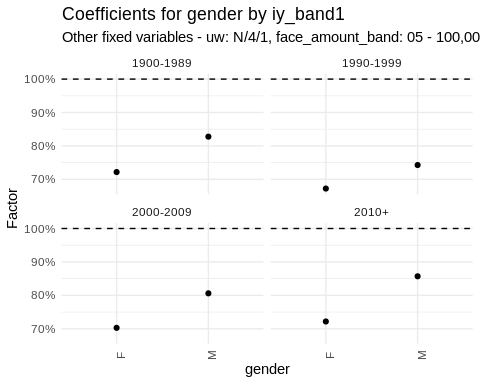
#### ltp x iy\_band1



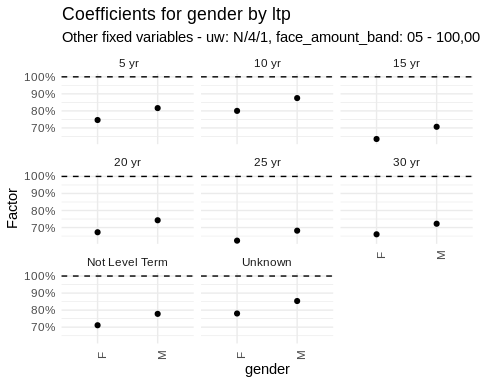
#### insurance\_plan x iy\_band1



#### gender x iy\_band1



#### gender x ltp



### Tables of Terms

Below are tables of 2-way interactions, with external factors fixed at their middle values.

## Generate and format summary tables for each interaction pair in glmnet.int.list  
glmnet.int.list %>%  
 map(.f = \(x) {  
 # Generate a table of cross-validated net coefficients  
 tableCVNetCoefs(  
 train.grid,  
 sort(x),  
 "Factor",  
 pred.cols = pred\_cols  
 ) %>%  
 ## Create a flextable from the generated table  
 flextable() %>%  
 ## Format the table values, converting numeric values to percentages  
 set\_formatter(values = function(x) {  
 if (is.numeric(x))  
 sprintf("%.1f%%", x \* 100)  
 else  
 x  
 }) %>%  
 ## Set a caption for the table  
 set\_caption(  
 paste0("Factors for ", x[1], " and ", x[2], ", all other factors fixed at middle levels")  
 ) %>%  
 ## Set table properties to enable scrolling  
 set\_table\_properties(opts\_html = list(  
 scroll = list(  
 add\_css = "max-height: 500px;"  
 )  
 )) %>%  
 autofit() #%>%  
 ## Print the flextable  
 #knitr::knit\_print()  
 }) %>%  
 ## Set names for each table based on the interaction pairs  
 purrr::set\_names(  
 map(glmnet.int.list, \(x) paste0(x, collapse = " x "))  
 ) ->   
 output\_tables  
  
if(output\_format == "html") {  
 output\_tables %>%  
 map(.f=knitr::knit\_print) %>%  
 ## Generate a tabset from the list of tables  
 generate\_tabset(  
 tabtitle = "",  
 tablevel = 3  
 ) %>%  
 ## Print the generated tabset  
 cat()  
} else {  
 export\_tables\_to\_excel(  
 output\_tables,  
 paste0(  
 exportsRoot,  
 "\_glmnet\_int\_twoway\_interaction\_tables.xlsx"  
 )  
 )  
   
 cat("See included Excel table for additional information.\n")  
}

See included Excel table for additional information.

### Goodness-of-Fit

Goodness-of-fit tables are provided. Each table provides actual-to-model ratios for single variables and for 2-way combinations of variables. A model is qualitatively deemed to perform well if goodness-of-fit ratios are close to 100% in almost all situations. The quantitative assessment using significance testing is omitted here.

#### Unvariate Goodness-of-Fit

## Generate and format summary tables for each factor column  
map(factor\_cols, .f = \(x) {  
 ## Convert column name to symbol for tidy evaluation  
 x <- sym(x)  
 resp\_var\_sym <- resp\_var  
   
 ## Summarize data by the current factor column  
 train %>%  
 group\_by(!!x) %>%  
 summarize(  
 Outcome = sum(amount\_actual), # Sum the actual amounts  
 AM = sum(amount\_actual) / sum(predictions\_glmnet) # Calculate the Actual-to-Model ratio  
 ) %>%  
 ## Create a flextable from the summarized data  
 flextable() %>%  
 ## Format the Actual-to-Model column as percentages  
 set\_formatter(  
 AM = function(x) {  
 if (is.numeric(x))  
 sprintf("%.1f%%", x \* 100)  
 else  
 x  
 }  
 ) %>%  
 ## Format the Outcome column as numbers  
 colformat\_num(j = "Outcome") %>%  
 ## Set header labels for the table  
 set\_header\_labels(  
 Outcome = "Outcome",  
 AM = "Actual-to-Model"  
 ) %>%  
 autofit() #%>%  
 # Print the flextable  
 #knitr::knit\_print()  
}) %>%  
 ## Set names for each table based on the factor columns  
 purrr::set\_names(factor\_cols) ->  
 output\_tables  
  
if(output\_format == "html") {  
 output\_tables %>%  
 map(.f=knitr::knit\_print) %>%  
 ## Generate a tabset from the list of tables  
 generate\_tabset(  
 tabtitle = "",  
 tablevel = 4  
 ) %>%  
 ## Print the generated tabset  
 cat()  
} else {  
 export\_tables\_to\_excel(  
 output\_tables,  
 paste0(  
 exportsRoot,  
 "\_glmnet\_int\_univariate\_goodness\_of\_fit\_tables.xlsx"  
 )  
 )  
   
 cat("See included Excel table for additional information.\n")  
}

See included Excel table for additional information.

#### Bivariate Goodness-of-Fit

## Create a list of unique pairs of factor columns  
pairlist <- data.table()  
for (i in 1:(length(factor\_cols) - 1)) {  
 for (j in (i + 1):length(factor\_cols)) {  
 if (i == 1 & j == 2) {  
 pairlist <- data.table(F1 = factor\_cols[i], F2 = factor\_cols[j])  
 } else {  
 pairlist <- rbind(pairlist, data.table(F1 = factor\_cols[i], F2 = factor\_cols[j]))  
 }  
 }  
}  
  
## Generate and format summary tables for each pair of factor columns  
map2(.x = pairlist$F1, .y = pairlist$F2, .f = \(x, y) {  
 xs <- sym(x)  
 ys <- sym(y)  
   
 ## Choose grouping order based on the number of levels in each factor  
 if (length(train[, levels(get(x))]) >= length(train[, levels(get(y))])) {  
 fttmp <- train %>%  
 group\_by(!!xs, !!ys) %>%  
 summarize(  
 Outcome = sum(amount\_actual),  
 Ratio = sprintf("%.1f%%", 100 \* sum(amount\_actual) / sum(predictions\_glmnet))  
 ) %>%  
 pivot\_wider(  
 names\_from = !!ys,  
 values\_from = c(Outcome, Ratio),  
 names\_glue = paste0(y, ": {", y, "}.{.value}"),  
 names\_vary = "slowest"  
 )  
 } else {  
 fttmp <- train %>%  
 group\_by(!!ys, !!xs) %>%  
 summarize(  
 Outcome = sum(amount\_actual),  
 Ratio = sprintf("%.1f%%", 100 \* sum(amount\_actual) / sum(predictions\_glmnet))  
 ) %>%  
 pivot\_wider(  
 names\_from = !!xs,  
 values\_from = c(Outcome, Ratio),  
 names\_glue = paste0(x, ": {", x, "}.{.value}"),  
 names\_vary = "slowest"  
 )  
 }  
   
 ## Adjust column keys for the flextable  
 fttmp.colkeys <- names(fttmp)[1]  
 for (i in 1:((length(names(fttmp)) - 1) / 2)) {  
 fttmp.colkeys <- c(fttmp.colkeys, paste0("blank", i), names(fttmp)[(2 \* i):(2 \* i + 1)])  
 }  
   
 ## Create and print the flextable  
 fttmp %>%  
 flextable(col\_keys = fttmp.colkeys) %>%  
 ftExtra::span\_header(sep = "\\.") %>%  
 align(align = 'center', part = "all") %>%  
 empty\_blanks() %>%  
 autofit() #%>%  
 #knitr::knit\_print()  
}) %>%  
 ## Set names for each element in the list based on the factor column pairs  
 purrr::set\_names(pairlist[, paste0(F1, " x ", F2)]) ->  
 output\_tables  
  
if(output\_format == "html") {  
 output\_tables %>%  
 map(.f=knitr::knit\_print) %>%  
 ## Generate a tabset from the list of formatted tables  
 generate\_tabset(  
 tabtitle = "",  
 tablevel = 4  
 ) %>%  
 ## Print the generated tabset  
 cat()  
} else {  
 export\_tables\_to\_excel(  
 output\_tables,  
 paste0(  
 exportsRoot,  
 "\_glmnet\_int\_bivariate\_goodness\_of\_fit\_tables.xlsx"  
 )  
 )  
   
 cat("See included Excel table for additional information.\n")  
}

See included Excel table for additional information.

# Comparison of Model Predictions

## Goodness of Fit

It is important to compare model performance on the test dataset. Models tend to fit well on the training data.

We compute the MSE, MAD, and Poisson deviance for each model on the test dataset. Models with lower values are considered qualitatively better.

Across all measures, the elastic net GLM model has the lowest deviation, with the LightGBM qualitatively not far behind. The main-effects GLM does not compete, which reinforces the need for some accommodation of interactions.

#-----------------------------------------#  
##### Table of Results #####  
#-----------------------------------------#  
  
## creates a table of results for mse, mae, dev for the models  
  
rbind(test[,val(get(resp\_var),predictions\_glm,get(resp\_offset))],  
 test[,val(get(resp\_var),predictions\_glmnet,get(resp\_offset))],  
 test[,val(get(resp\_var),predictions\_lgbm1,get(resp\_offset))]) %>%  
 set\_rownames(c("glm","glmnet","lgbm")) %>%  
 rownames\_to\_column(var="model") %>%  
 as.data.table() %>%  
 as\_flextable()

| model | mse | mae | dev |
| --- | --- | --- | --- |
| glm | 657,407,396,611,204 | 10,369,070 | 1,552,142,707,959,196 |
| glmnet | 285,309,606,550,480 | 8,470,467 | 723,646,527,740,728 |
| lgbm | 355,563,440,625,343 | 8,741,101 | 909,129,632,590,165 |

## Graphical Model Comparison

Unlike the GLM, neither the LightGBM nor the penalized GLM provide any information regarding parameter uncertainty. For elastic net GLMs, there are options to estimate parameter uncertainty:

1. Move to a fully Bayesian setting. This gives the modeler significant control, at the cost of complexity (e.g., how to choose reasonable priors) and computation cost. Stan and INLA are available for this purpose.
2. Apply the method in Tibshirani et al’s [A significance test for the lasso](https://arxiv.org/abs/1301.7161). This requires rerunning penalized GLMs and is thus potentially costly.
3. Apply the method in Lederer’s [Fundamentals of High-Dimensional Statistics](https://johanneslederer.com/hdbook/), Sec. 5.2. While technically involved, there does not seem to be a heavy computational lift.

To get around the limitations of assessing uncertainty for now, we plot the models versus the envelope of uncertainty arising from the data itself. This shifts the point of view from assessing parameter uncertainty to assessing goodness-of-fit.

Below are plots of how the model performs versus marginal effects, with performance tested on the test subset. Black dots with error bars are from the actual-to-2015VBT ratio, with error bar width based on the dispersion from the GLM model. (Caution: this is at best a crude approximation.)

The following colors denote specific predictive model ratios versus the 2015 VBT:

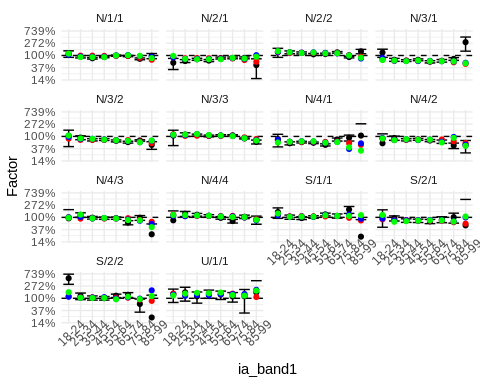
* Red uses GLM predicted claims
* Blue uses predicted claims from the elastic net GLM
* Green uses LightGBM predicted claims

Broad observations:

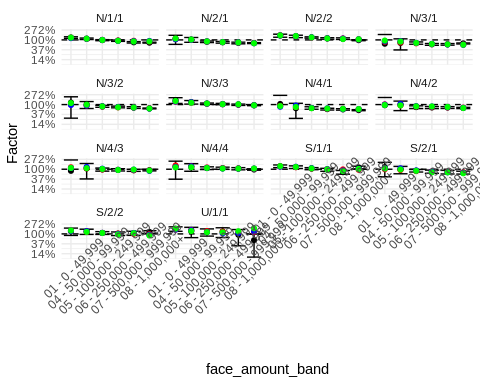
* In 2017, some of the average relationships shifted versus 2013-2016. This can be seen by noting the model dots resting outside the error bars.
* The elastic net model may be missing some higher order interactions.

#-----------------------------------------#  
##### Model Feature Plots #####  
#-----------------------------------------#  
  
## Select the top `nUseTopLightGBMInteractions` interactions from `imp.int2`  
## and transform the `Feature1` and `Feature2` columns into a long format,  
## ensuring each feature appears only once in the resulting list.  
  
imp.int2 %>%   
 head(nUseTopLightGBMInteractions) %>% # Select the top interactions  
 select(Feature1, Feature2) %>% # Select the relevant columns  
 pivot\_longer(cols = c(Feature1, Feature2), # Pivot to long format  
 names\_to = NULL,  
 values\_to = "Feature") %>%   
 distinct() %>% # Remove duplicate features  
 pull(Feature) -> int.subset # Extract the features into `int.subset`  
  
## Calculate the dispersion of the GLM model  
glm\_disp <- sum(modelGLM$residuals^2 \* modelGLM$weights)/modelGLM$df.residual  
  
## Select the top `nUseTopLightGBMInteractions` interactions from `imp.int2` again  
## to use as input for the `map2` function  
int.subset <- imp.int2 %>%   
 head(nUseTopLightGBMInteractions) %>%  
 select(Feature1,Feature2)  
  
## Apply a function to each pair of features in `int.subset` to generate plots  
map2(  
 .x = int.subset$Feature1, # First feature in the pair  
 .y = int.subset$Feature2, # Second feature in the pair  
 .f = function(s1, s2) { # Function to apply  
 test[, # Subset the data  
 .(  
 predicted\_glm = sum(predictions\_glm)/sum(amount\_2015vbt),  
 predicted\_glmnet= sum(predictions\_glmnet)/sum(amount\_2015vbt),  
 predicted\_lgbm1 = sum(predictions\_lgbm1)/sum(amount\_2015vbt),  
 a\_e = sum(amount\_actual)/sum(amount\_2015vbt),  
 stde = sqrt(sum(amount\_actual)\*glm\_disp)/sum(amount\_2015vbt)  
 ),  
 by=c(s1,s2)] %>%  
 setnames(s1,"x") %>%  
 setnames(s2,"byvar") %>%  
 mutate(  
 x = fct\_relevel(  
 x,  
 sort(levels(x))  
 ),  
 byvar = fct\_relevel(  
 byvar,  
 sort(levels(byvar))  
 )  
 ) %>%  
 as.data.table() %>%  
 ggplot(aes(x = x, y = a\_e)) + # Create a ggplot with `x` on the x-axis and `a\_e` on the y-axis  
 facet\_wrap(vars(byvar)) + # Create separate panels for each value of `byvar`  
 geom\_point() + # Add points for actual values  
 geom\_errorbar(aes(ymin = a\_e - 1.96 \* stde, ymax = a\_e + 1.96 \* stde)) + # Add error bars  
 geom\_hline(yintercept = 1, linetype = 2) + # Add a horizontal line at y = 1  
 geom\_point(aes(y = predicted\_glm), color = "red") + # Add points for GLM predictions  
 geom\_point(aes(y = predicted\_glmnet), color = "blue") + # Add points for GLMNet predictions  
 geom\_point(aes(y = predicted\_lgbm1), color = "green") + # Add points for LightGBM predictions  
 scale\_y\_continuous(name = "Factor", labels = scales::percent, trans = "log") + # Log scale for y-axis  
 scale\_x\_discrete(name = s1) + # Name the x-axis after the first feature  
 theme\_minimal() + # Use a minimal theme  
 theme(axis.text.x = element\_text(angle = 45)) # Rotate x-axis labels  
 }  
) %>%  
 purrr::set\_names(  
 int.subset %>%   
 mutate(Features=paste0( Feature1, " x ", Feature2)) %>%   
 select(Features) %>%  
 unlist  
 ) %>%  
 iwalk(~ {  
 cat('### ', .y, '\n\n') # Print the feature pair as a section header  
 print(.x) # Print the plot  
 cat('\n\n') # Add some spacing  
 } )

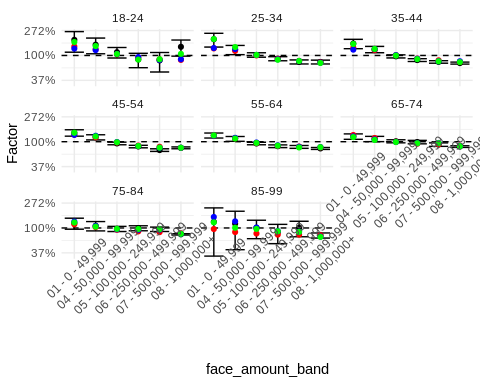
### ia\_band1 x uw



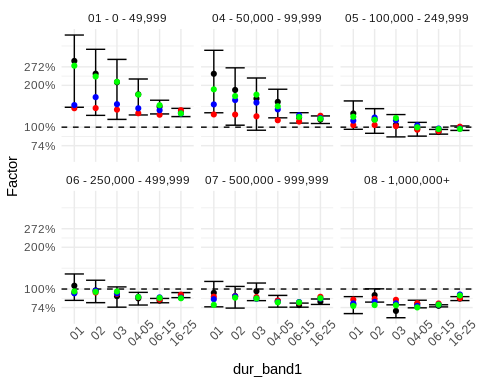
### face\_amount\_band x uw



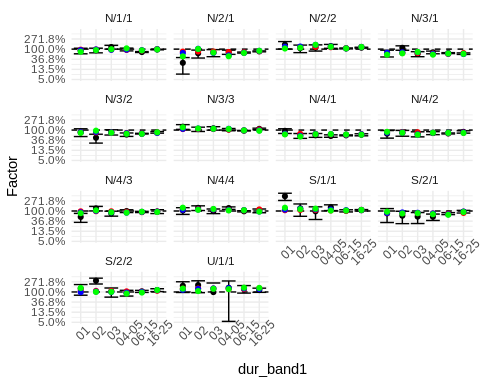
### face\_amount\_band x ia\_band1



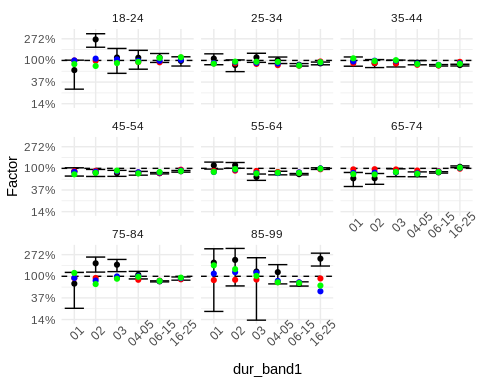
### dur\_band1 x face\_amount\_band



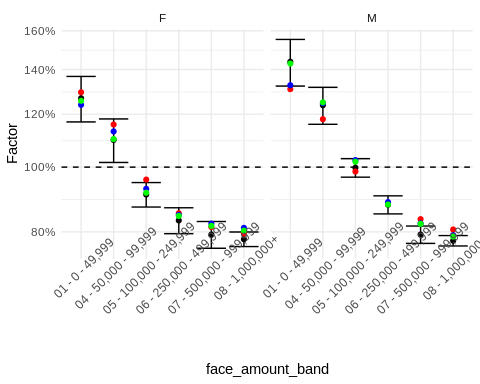
### dur\_band1 x uw



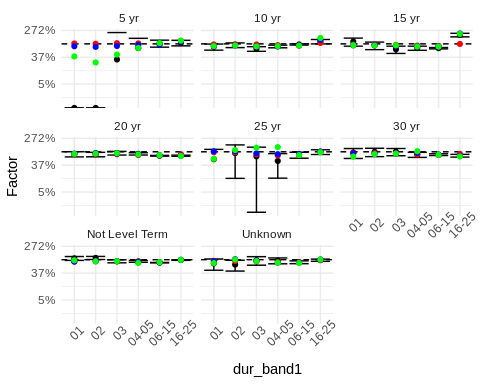
### dur\_band1 x ia\_band1



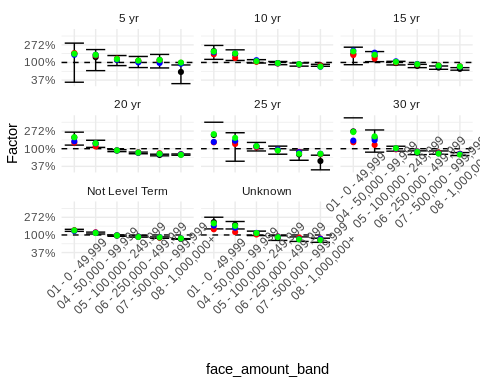
### face\_amount\_band x gender



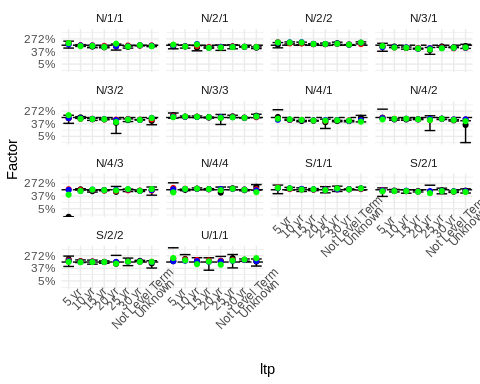
### dur\_band1 x ltp



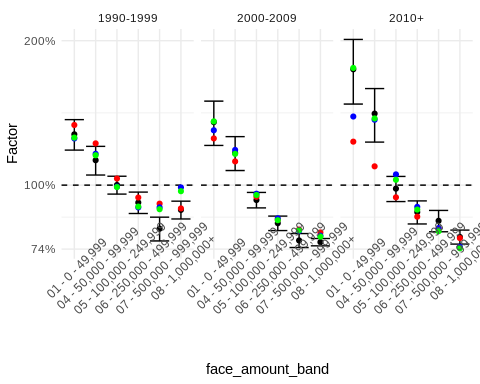
### face\_amount\_band x ltp



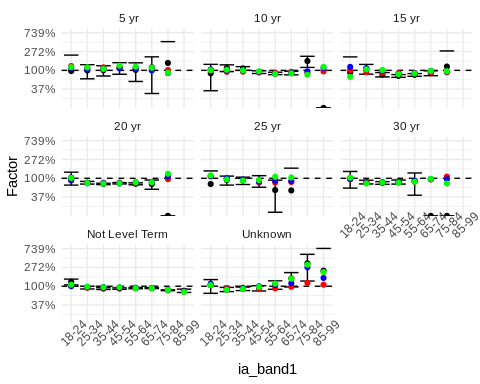
### ltp x uw



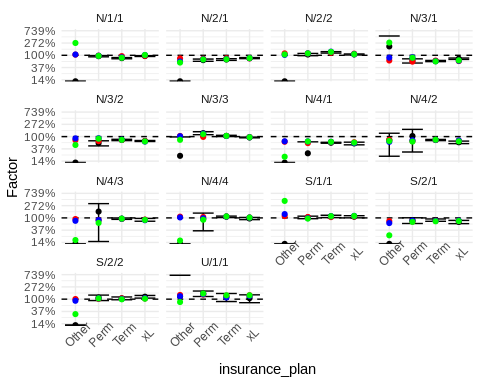
### face\_amount\_band x iy\_band1



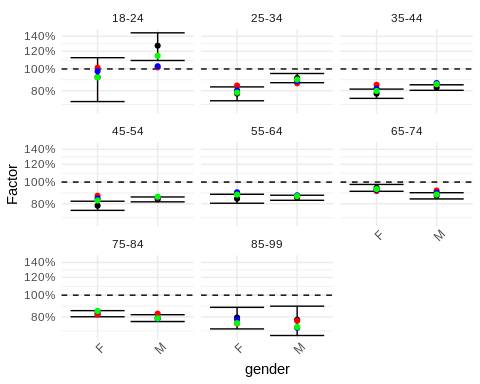
### ia\_band1 x ltp



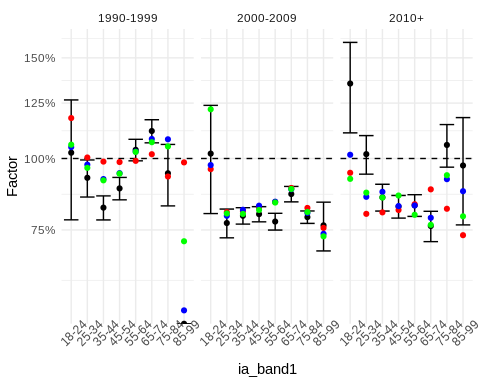
### insurance\_plan x uw



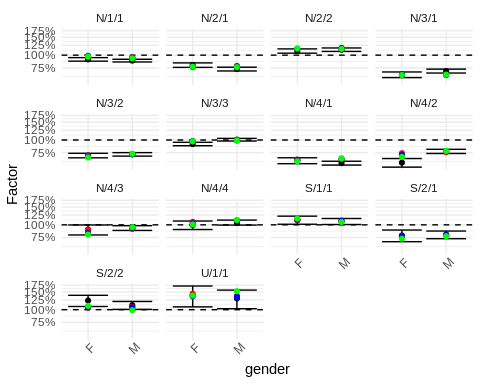
### gender x ia\_band1



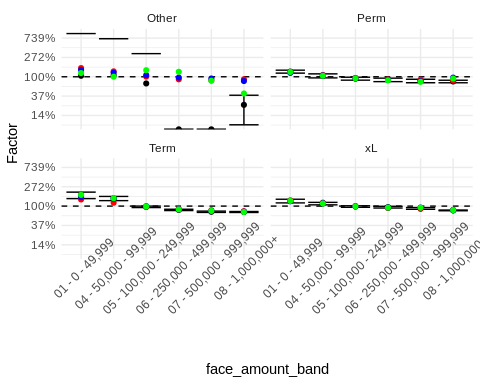
### ia\_band1 x iy\_band1



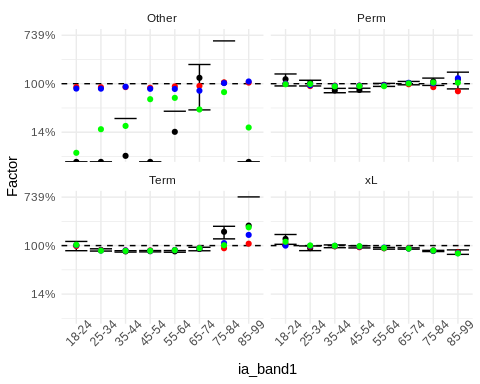
### gender x uw



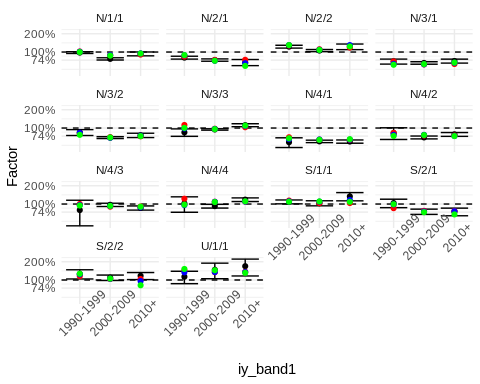
### face\_amount\_band x insurance\_plan



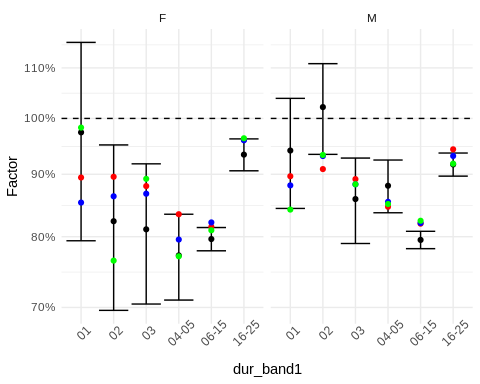
### ia\_band1 x insurance\_plan



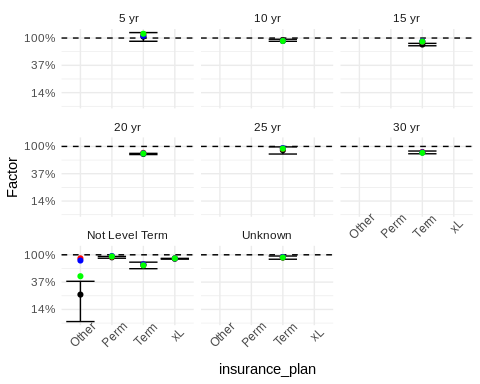
### iy\_band1 x uw



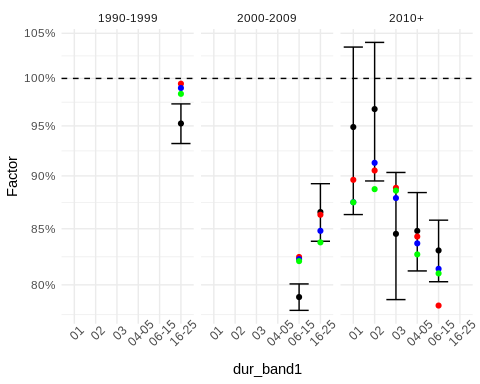
### dur\_band1 x gender



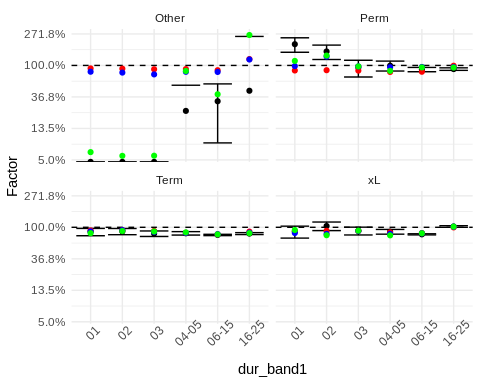
### insurance\_plan x ltp



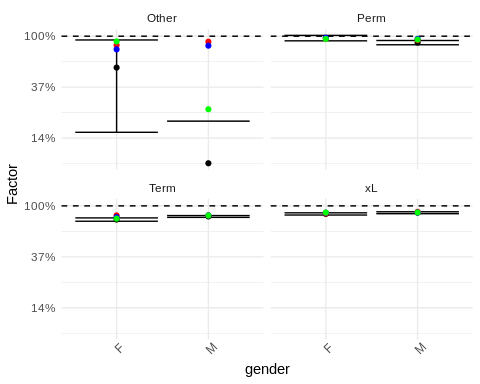
### dur\_band1 x iy\_band1



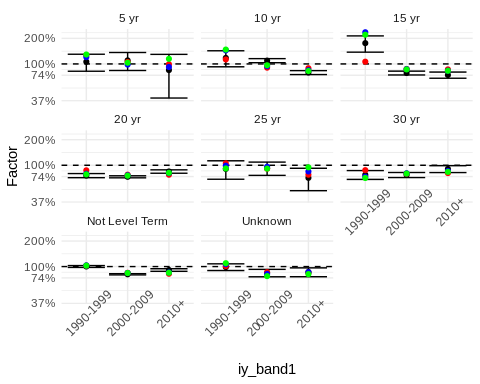
### dur\_band1 x insurance\_plan



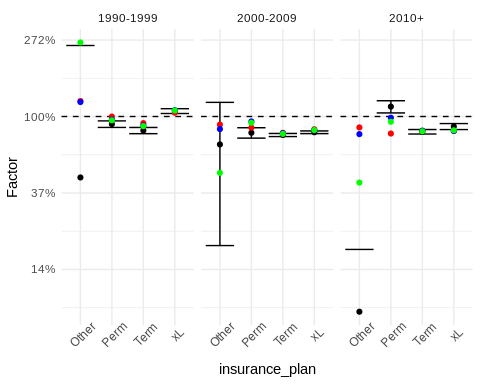
### gender x insurance\_plan



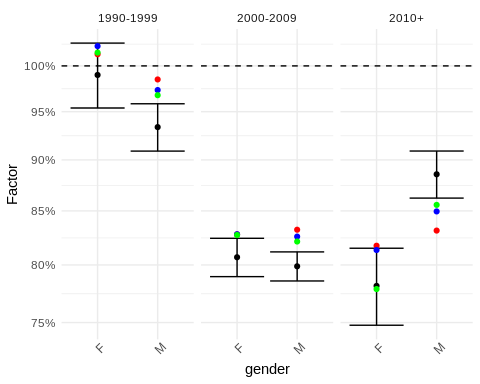
### iy\_band1 x ltp



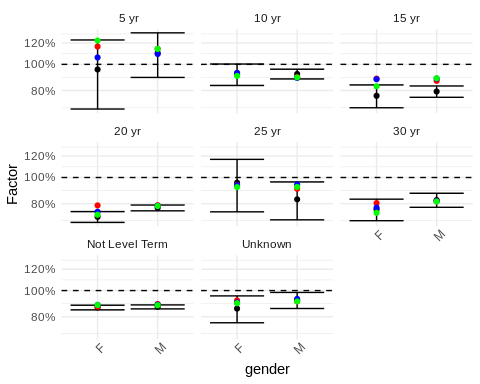
### insurance\_plan x iy\_band1



### gender x iy\_band1



### gender x ltp



# Mortality Differences by Product Use Case

## Observations from the Raw Data

When assessing differences by product, it is not hard to find challenges when looking at the raw, unadjusted data.

One example is that there is virtually no 4-class non-smoker exposure in Perm, while there is significant exposure in Term. This implies that there is a potential issue of identifiability in interactions between insurance plan and underwriting due to the imbalance in exposures. This manifests as an apparent instability in calibrations.

For example, the marginal difference between the 4-class and 2-class non-smokers in the data is 82.42% (78.2%/94.9%), while the marginal difference between Term and Perm is 86.5% (83.4%/96.4%).

ds[,  
 c("Smoker\_Status","NClasses","Class"):=tstrsplit(uw,"/")]  
  
ds[Smoker\_Status=="N",.(A\_2015VBT=sum(get(resp\_var))/sum(get(resp\_offset))),  
 by=.(NClasses)][order(NClasses)] %>%  
 flextable() %>%  
 set\_header\_labels(NClasses="No. of Pref. Classes",  
 A\_2015VBT="A/2015VBT") %>%  
 set\_formatter(values = function(x) {  
 if(is.numeric(x))  
 sprintf( "%.1f%%", x\*100 )  
 else  
 x  
 }  
 )

| No. of Pref. Classes | A/2015VBT |
| --- | --- |
| 1 | 97.2% |
| 2 | 94.9% |
| 3 | 81.4% |
| 4 | 78.2% |

ds[,.(A\_2015VBT=sum(get(resp\_var))/sum(get(resp\_offset))),by=.(insurance\_plan)] %>%  
 flextable() %>%  
 set\_header\_labels(insurance\_plan="Insurance Plan",  
 A\_2015VBT="A/2015VBT") %>%  
 set\_formatter(values = function(x) {  
 if(is.numeric(x))  
 sprintf( "%.1f%%", x\*100 )  
 else  
 x  
 }  
 )

| Insurance Plan | A/2015VBT |
| --- | --- |
| Term | 83.4% |
| xL | 89.2% |
| Perm | 96.4% |
| Other | 84.1% |

By way of comparison, the GLM calibrates 86.8% for Term versus Perm, and the weighted average factors for 4-class systems from the GLM model is 79.3% versus 95.7% for 2-class, for a ratio of 82.8%. The main effects GLM is therefore asserting that both conditions are associated with lower mortality.

For the elastic net GLM, the situation is complicated. All in, there are 52 factors which mention insurance plan, and assessing when perm and term differ is challenging on a bare reading of the factor table.

reformatCoefs(cvfit, pred\_cols) %>%  
 filter(Coef != 0) %>%  
 select(Feature1Name,  
 Feature1Level,  
 Feature2Name,  
 Feature2Level,  
 Coef) %>%  
 mutate(Coef=exp(Coef)) %>%  
 filter(  
 Feature1Name == "insurance\_plan" | Feature2Name == "insurance\_plan"  
 ) %>%  
 select(Feature2Name, Feature2Level,  
 Feature1Name, Feature1Level,  
 Coef) %>%  
 arrange(Feature2Name, Feature2Level,  
 Feature1Name, Feature1Level) %>%  
 flextable() %>%  
 set\_header\_labels(Coef="Factor") %>%  
 set\_formatter(values = function(x) {  
 if(is.numeric(x))  
 sprintf( "%.1f%%", x\*100 )  
 else  
 x  
 }  
 )

For the LightGBM model, there is arguably no interesting mean difference for between Perm and Term.

data.table(  
 insurance\_plan=train[shp\_int\_subset,insurance\_plan],  
 shap=shp$S[,which(names(shp$X)=="insurance\_plan")] + shp$baseline,  
 response=train[shp\_int\_subset,get(resp\_var)],  
 offset=train[shp\_int\_subset,get(resp\_offset)]  
) %>%  
 group\_by(insurance\_plan) %>%  
 summarize(M\_T=sum(exp(shap)\*offset)/sum(offset)) %>%  
 flextable() %>%  
 set\_header\_labels(insurance\_plan="Insurance Plan",  
 M\_T="Model / 2015VBT") %>%  
 set\_formatter(values = function(x) {  
 if(is.numeric(x))  
 sprintf( "%.1f%%", x\*100 )  
 else  
 x  
 }  
 )

| Insurance Plan | Model / 2015VBT |
| --- | --- |
| Other | 67.0% |
| Perm | 91.3% |
| Term | 93.3% |
| xL | 102.9% |

For class system, the LightGBM model is illustrating a substantial reduction in mean mortality for the 4-class systems relative to 2-class systems.

shaps.uw <- data.table(  
 uw=train[shp\_int\_subset,uw],  
 shap=shp$S[,"uw"] + shp$baseline,  
 response=train[shp\_int\_subset,get(resp\_var)],  
 offset=train[shp\_int\_subset,get(resp\_offset)]  
)   
  
shaps.uw[,  
 c("Smoker\_Status","NClasses","Class"):=tstrsplit(uw,"/")]  
  
shaps.uw %>%  
 group\_by(NClasses) %>%  
 summarize(M\_T=sum(exp(shap)\*offset)/sum(offset)) %>%  
 flextable() %>%  
 set\_header\_labels(NClasses="No. of Pref. Classes",  
 M\_T="Model / 2015VBT") %>%  
 set\_formatter(values = function(x) {  
 if(is.numeric(x))  
 sprintf( "%.1f%%", x\*100 )  
 else  
 x  
 }  
 )

| No. of Pref. Classes | Model / 2015VBT |
| --- | --- |
| 1 | 104.4% |
| 2 | 100.9% |
| 3 | 101.0% |
| 4 | 86.2% |

All of this strongly suggests the need for more sophisticated analysis.

## Observations from the GLM

One question of interest to actuaries is why different products have different mortality outcomes. Many things could contribute to the difference, such as UW practice, anti-selection risk level, market segment, etc., and generally, it is hard to quantify their impact. With the GLM model and relevant analysis, we have a possible solution.

Let us revisit the table output with insurance plan as the predictor of interest.

## function that calls individual predictor, e.g., insurance plans.   
mainF(df = ds,  
 model = modelGLM,  
 rf = "insurance\_plan",  
 resp = resp\_var,  
 offset = resp\_offset) %>%  
 flextable() %>%  
 set\_header\_labels("rowname" = "") %>%  
 set\_formatter(values = ~ if(is.numeric(.)) sprintf("%.1f%%", . \* 100) else .) %>%  
 set\_caption(caption = paste0("Weighted Average GLM Factors for Variable: insurance\_plan"))

Weighted Average GLM Factors for Variable: insurance\_plan

|  | Other | Perm | Term | xL |
| --- | --- | --- | --- | --- |
| amount\_2015vbt | 0.8411 | 0.9638 | 0.8340 | 0.8923 |
| Factor: insurance\_plan | 1.0000 | 0.8749 | 0.7596 | 0.9590 |
| Ave Fac: dur\_band1 | 0.9306 | 0.9098 | 0.9102 | 0.9024 |
| Ave Fac: face\_amount\_band | 0.7300 | 0.7913 | 0.7339 | 0.7366 |
| Ave Fac: gender | 1.0067 | 1.0070 | 1.0082 | 1.0060 |
| Ave Fac: ia\_band1 | 0.9124 | 0.9220 | 0.9325 | 0.8870 |
| Ave Fac: iy\_band1 | 0.8452 | 0.9105 | 0.8636 | 0.8741 |
| Ave Fac: ltp | 0.6733 | 0.6733 | 0.8455 | 0.6733 |
| Ave Fac: uw | 0.9790 | 1.0127 | 0.9051 | 1.0025 |

For illustration, let us select Perm and Term for pairwise comparison. By A/15VBT, Perm (96.4%) seems to have better mortality than Term (83.4%). Is this due to “product differences”?

mainF(df = ds,   
 model = modelGLM,   
 rf = "insurance\_plan",   
 resp = resp\_var,   
 offset = resp\_offset) %>%  
 select(rowname, Perm, Term) %>%  
 flextable() %>%  
 set\_header\_labels("rowname" = "") %>%  
 set\_formatter(values = ~ if(is.numeric(.)) sprintf("%.1f%%", . \* 100) else .) %>%  
 set\_caption(caption = paste0("Weighted Average GLM Factors for Variable: insurance\_plan in (Perm, Term)"))

Weighted Average GLM Factors for Variable: insurance\_plan in (Perm, Term)

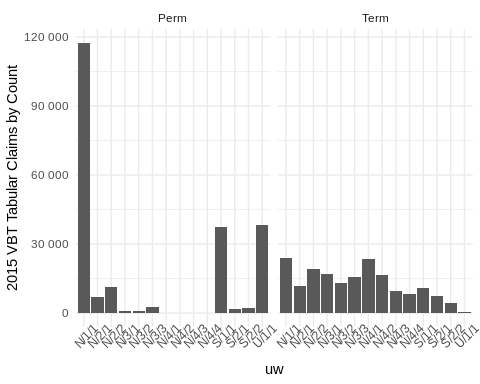
|  | Perm | Term |
| --- | --- | --- |
| amount\_2015vbt | 0.9638 | 0.8340 |
| Factor: insurance\_plan | 0.8749 | 0.7596 |
| Ave Fac: dur\_band1 | 0.9098 | 0.9102 |
| Ave Fac: face\_amount\_band | 0.7913 | 0.7339 |
| Ave Fac: gender | 1.0070 | 1.0082 |
| Ave Fac: ia\_band1 | 0.9220 | 0.9325 |
| Ave Fac: iy\_band1 | 0.9105 | 0.8636 |
| Ave Fac: ltp | 0.6733 | 0.8455 |
| Ave Fac: uw | 1.0127 | 0.9051 |

Thanks to the GLM model, we can work with a multiplicative formula for prediction. And, with this elegant structure, we can parse out the impact of each individual predictor and make comparisons. The relative impact is represented by the rate of change.

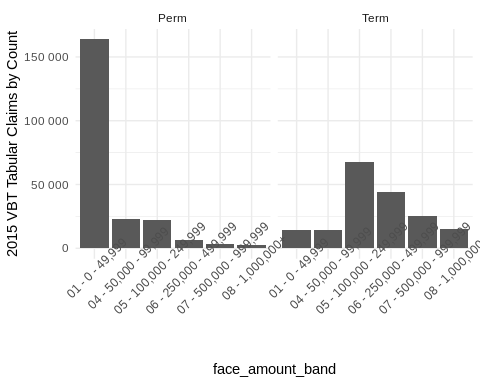
Of these, the movement in uw is the most influential, with ratio 111.9% for Perm over Term. This means that, if all the other predictors are controlled, the average uw factor on a risk-adjusted basis will make Perm mortality prediction approximately 11.9% higher than that of Term. Other influential drivers from this analysis include face amount band, issue year band, and level term period. This may suggest, if actuaries/modelers want to build a simpler model yet still capture essential impact to mortality outcome, they may consider including at least those predictors in the GLM model.

One should nonetheless look to residuals and distributions to ensure that valuable interactions are not being lost. Part of what we are seeing has to do with the different distributions between Perm and Term of uw and face\_amount\_band. Perm tends to favor 1- and 2-class risk class systems, while Term tends to favor 3- and 4-class systems. Perm also tends to favor lower face amounts, while Term favors higher face amounts.

## create histogram of 2015 VBT Tabular Claims by Count and UW  
ds %>%  
 filter(insurance\_plan %in% c("Perm","Term")) %>%  
 group\_by(uw,  
 insurance\_plan) %>%  
 summarize(  
 AM\_policy=sum(policy\_actual)/sum(predictions\_glm),  
 policy\_2015vbt=sum(policy\_2015vbt)) %>%  
 as.data.table() %>%  
 ggplot(aes(x = uw)) +  
 geom\_bar(aes(y = policy\_2015vbt), stat = "identity") +  
 facet\_wrap(facets = vars(insurance\_plan)) +  
 scale\_y\_continuous(labels = scales::number, name = "2015 VBT Tabular Claims by Count") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 45))

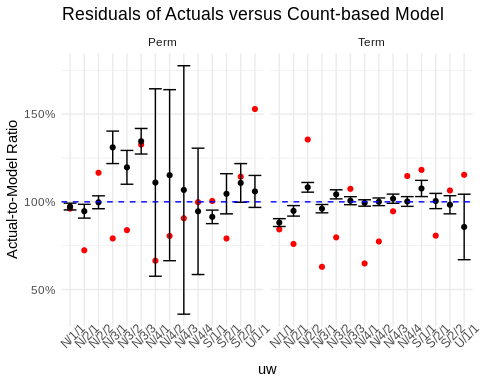


## create histogram of 2015 VBT Tabular Claims by Count and Face Amount  
ds %>%  
 filter(insurance\_plan %in% c("Perm","Term")) %>%  
 group\_by(face\_amount\_band,  
 insurance\_plan) %>%  
 summarize(  
 AM\_policy=sum(policy\_actual)/sum(predictions\_glm),  
 policy\_2015vbt=sum(policy\_2015vbt)) %>%  
 as.data.table() %>%  
 ggplot(aes(x = face\_amount\_band)) +  
 geom\_bar(aes(y = policy\_2015vbt), stat = "identity") +  
 facet\_wrap(facets = vars(insurance\_plan)) +  
 scale\_y\_continuous(labels = scales::number, name = "2015 VBT Tabular Claims by Count") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 45))

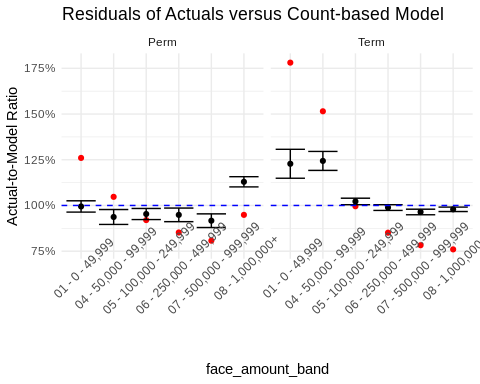


In light of what we see for distribution, it is unsurprising that the model fits poorly for the Term subset for smaller face amounts and 1- and 2-class systems, while the model fits the Perm subset poorly for 3- and 4-class systems. Since Perm tends to dominate the lower face amounts, model fit is not nearly as poor there as for the Term subset.

## Create graph of Residuals of Actuals versus Count-based Model by UW  
  
ds %>%  
 filter(insurance\_plan %in% c("Perm","Term")) %>%  
 group\_by(uw,  
 insurance\_plan) %>%  
 summarize(  
 AM\_policy=sum(amount\_actual)/sum(predictions\_glm),  
 policy\_actual=sum(amount\_actual),  
 predictions\_glm=sum(predictions\_glm),  
 amount\_2015vbt=sum(amount\_2015vbt)) %>%  
 as.data.table() %>%  
 ggplot(aes(x = uw)) +  
 geom\_point(aes(y = policy\_actual / amount\_2015vbt), color = "red") +  
 geom\_point(aes(y = AM\_policy, group = 1)) +  
 geom\_errorbar(aes(ymin = AM\_policy - 1.96 \* sqrt(glm\_disp / predictions\_glm), ymax = AM\_policy + 1.96 \* sqrt(glm\_disp / predictions\_glm))) +  
 geom\_hline(yintercept = 1, color = "blue", linetype = 2) +  
 facet\_wrap(facets = vars(insurance\_plan)) +  
 scale\_y\_continuous(labels = scales::percent, name = "Actual-to-Model Ratio") +  
 ggtitle("Residuals of Actuals versus Count-based Model") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 45))



## Create graph of Residuals of Actuals versus Count-based Model by Face Amount  
  
ds %>%  
 filter(insurance\_plan %in% c("Perm","Term")) %>%  
 group\_by(face\_amount\_band,  
 insurance\_plan) %>%  
 summarize(  
 AM\_policy=sum(amount\_actual)/sum(predictions\_glm),  
 policy\_actual=sum(amount\_actual),  
 predictions\_glm=sum(predictions\_glm),  
 amount\_2015vbt=sum(amount\_2015vbt)) %>%  
 as.data.table() %>%  
 ggplot(aes(x = face\_amount\_band)) +  
 geom\_point(aes(y = policy\_actual / amount\_2015vbt), color = "red") +  
 geom\_point(aes(y = AM\_policy, group = 1)) +  
 geom\_errorbar(aes(ymin = AM\_policy - 1.96 \* sqrt(glm\_disp / predictions\_glm), ymax = AM\_policy + 1.96 \* sqrt(glm\_disp / predictions\_glm))) +  
 geom\_hline(yintercept = 1, color = "blue", linetype = 2) +  
 facet\_wrap(facets = vars(insurance\_plan)) +  
 scale\_y\_continuous(labels = scales::percent, name = "Actual-to-Model Ratio") +  
 ggtitle("Residuals of Actuals versus Count-based Model") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 45))



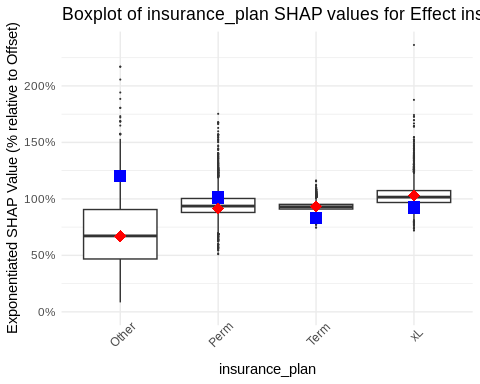
Fitting a main-effects GLM on a dataset is a frequently used first step when modeling any dataset. Analyzing residuals from this model and assessing parameter variability by subgroup can reveal useful patterns for further analysis. It is often the case that interactions of effects are present. While a useful starting point, main-effects models cannot capture such interactions effectively. It is therefore necessary to turn to richer models and approaches.

## Observations from the Gradient Boosted Decision Tree

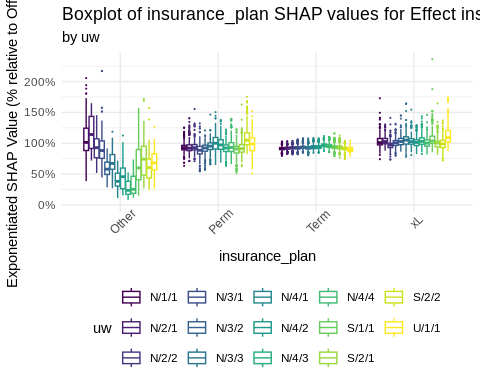
Below are box plots of SHAP values for insurance\_plan by the other variables.

imp.int2["insurance\_plan" == Feature1 | "insurance\_plan" == Feature2] %>%   
 head(nPlotTopInteractions) %>%  
 select(Feature1,Feature2) %>%  
 pivot\_longer(cols=c(Feature1,Feature2),  
 names\_to=NULL,  
 values\_to="Feature") %>%  
 distinct() %>%  
 filter(Feature != "insurance\_plan") ->   
 int.vars  
  
plist <- ilec\_shap\_plot(  
 shp,  
 "insurance\_plan",  
 setdiff(pred\_cols,"insurance\_plan"),  
 resp\_var = resp\_var,  
 resp\_offset = resp\_offset,  
 train.data = train[shp\_int\_subset]  
 )  
  
  
plist %>%  
 iwalk(~ {  
 cat('### ',.y,'\n\n')  
 print(.x)  
 cat('\n\n')  
 } )

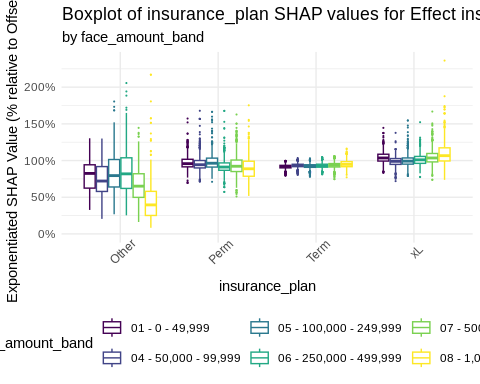
### main effect: insurance\_plan



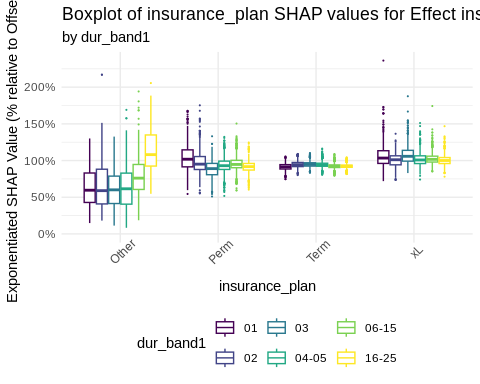
### insurance\_plan x uw



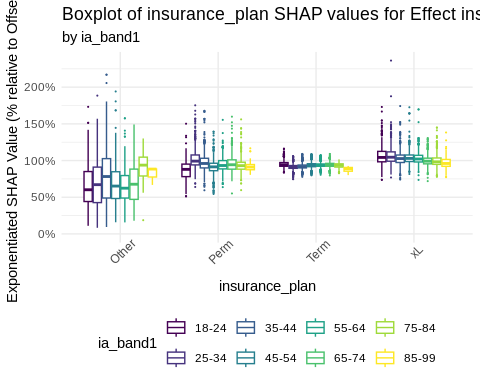
### insurance\_plan x face\_amount\_band



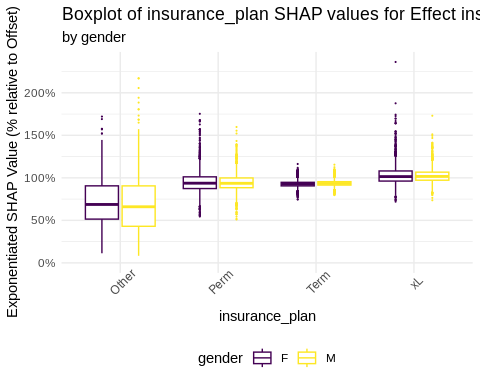
### insurance\_plan x dur\_band1



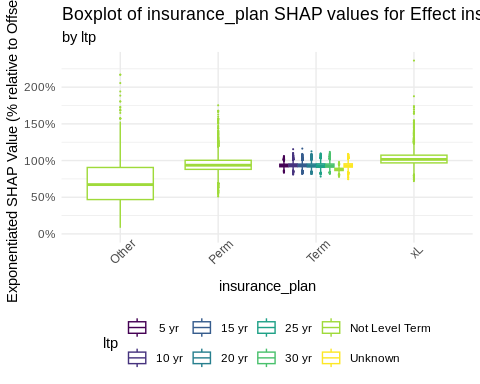
### insurance\_plan x ia\_band1



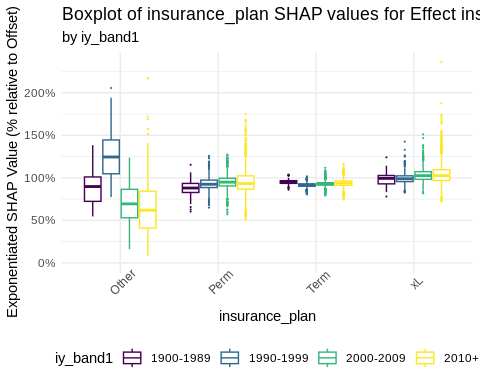
### insurance\_plan x gender



### insurance\_plan x ltp



### insurance\_plan x iy\_band1



We have noted the interactions with insurance plan from the LightGBM SHAP values as follows:

1. Main Effect: Perm and Term shap distributions are qualitatively similar, with the xL class higher.
2. Interaction with underwriting:
   1. Substantial interaction with the “Other” category
   2. For Term, some evidence of higher mortality for N/4/3 and N/4/4
3. Interaction with face amount band:
   1. No obvious interactions with Perm and Term
   2. Weak evidence for interaction with xL, based on U-shaped pattern in boxplots
4. Interaction with duration
   1. Weak evidence for elevated mortality in early durations for Perm
   2. Weak evidence for opposite in early durations for Term
   3. Face amounts 1 million and higher for “Other” are plainly different from lower face amount “Other”
5. Interaction with issue age
   1. Evidence for different issue age slope (relative to 2015VBT) for xL based on downward trend in boxplots
   2. Weak evidence for slight upward issue age slope (relative to 2015VBT) for Term based on trend in boxplots
6. Interaction with gender: no obvious interaction
7. Interaction with level term period:
   1. Obviously, no interactions outside of Term
   2. Within Term, “Not Level Term” has lower mortality than the other level term types
8. Interaction with issue year band:
   1. Since 1990, there is evidence of an upward trend in mortality for all categories outside of “Other”.

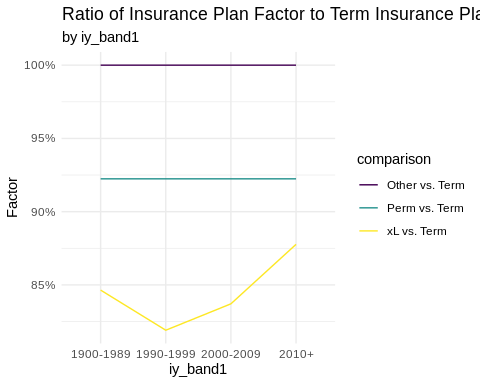
## Contrasts of Interactions with Insurance Plan from the Elastic Net Model

The elastic net model encodes interesting interactions of insurance plan with other predictor variables. Graphing the contrast between insurance plan types can reveal patterns which are difficult to see when looking at the bare coefficients or tables of factors.

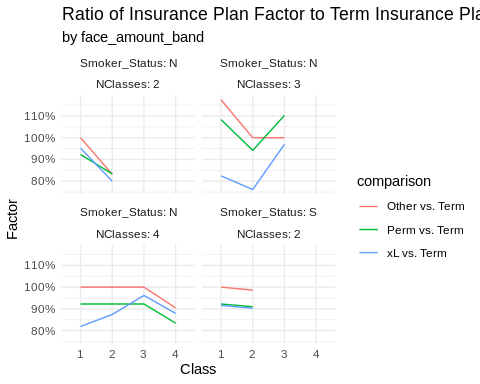
To do so, we can gather the table of factors which include insurance plan and compute the ratio of the factors versus the factors for term. For example, if the marginal factor for Perm males is 99%, and the factor for Term males is 90%, then the ratio is 110%. These contrasts can then be plotted for both males and females across the plan comparisons, as can be seen in the following graphs.

## Combine the list of glmnet interactions into a data.table and filter relevant interactions  
data.table(do.call(rbind, glmnet.int.list)) %>%  
 filter((Feature1Name == "insurance\_plan" | Feature2Name == "insurance\_plan") &  
 (Feature1Name != "ltp" & Feature2Name != "ltp")) ->  
 ints.with.plan  
  
## Extract interaction feature names related to "insurance\_plan"  
c(ints.with.plan[Feature1Name == "insurance\_plan", Feature2Name],  
 ints.with.plan[Feature2Name == "insurance\_plan", Feature1Name]) ->  
 ints.with.plan  
  
## Map over each interaction feature  
ints.with.plan %>%  
 map(  
 .f = \(x) {  
 symx <- sym(x) # Convert feature name to symbol  
  
 ## Create a table of coefficients for the interaction with "insurance\_plan"  
 tableCVNetCoefs(train.grid,  
 c(x, "insurance\_plan"),  
 "Factor",  
 pred\_cols,  
 levellist = list("dur\_band1" = "04-05")) %>%  
 data.table() ->  
 tblFacts  
  
 ## Normalize factors by "Term" and pivot longer for ggplot  
 cbind(  
 tblFacts[, 1],  
 tblFacts[, lapply(.SD, "/", Term), .SDcols = colnames(tblFacts)[-1]]  
 ) %>%  
 pivot\_longer(  
 cols = colnames(tblFacts)[-1],  
 values\_to = "Factor",  
 names\_to = "comparison"  
 ) %>%  
 filter(comparison != 'Term') %>%  
 mutate(comparison = paste0(comparison, " vs. Term")) %>%  
 data.table() ->  
 dftmp  
  
 ## Special handling if the feature is "uw"  
 if (x == "uw") {  
 dftmp %>%  
 separate(  
 col = uw,  
 into = c("Smoker\_Status", "NClasses", "Class"),  
 sep = "/",  
 remove = FALSE  
 ) ->  
 dftmp  
  
 ## Filter out single class and plot  
 dftmp %>%  
 filter(NClasses != 1) %>%  
 ggplot(aes(x = Class, y = Factor)) +  
 geom\_line(aes(group = comparison, color = comparison)) +  
 scale\_y\_continuous(labels = scales::percent) +  
 facet\_wrap(Smoker\_Status ~ NClasses, labeller = label\_both) +  
 theme\_minimal() +  
 ggtitle(  
 "Ratio of Insurance Plan Factor to Term Insurance Plan Factor",  
 subtitle = paste0("by ", ints.with.plan[3])  
 ) -> p  
 } else {  
 ## General plot for other features  
 dftmp %>%  
 ggplot(aes(x = !!symx, y = Factor)) +  
 geom\_line(aes(group = comparison, color = comparison)) +  
 scale\_y\_continuous(labels = scales::percent) +  
 theme\_minimal() +  
 ggtitle(  
 "Ratio of Insurance Plan Factor to Term Insurance Plan Factor",  
 subtitle = paste0("by ", x)  
 ) +  
 theme(axis.text.x = element\_text(angle = ifelse(x == "face\_amount\_band", 45, 0))) +  
 scale\_color\_viridis\_d() -> p  
 }  
  
 p # Return the plot  
 }  
 ) %>%  
 purrr::set\_names(ints.with.plan) %>%  
 iwalk(~ {  
 cat('### ', .y, '\n\n')  
 print(.x)  
 cat('\n\n')  
 })

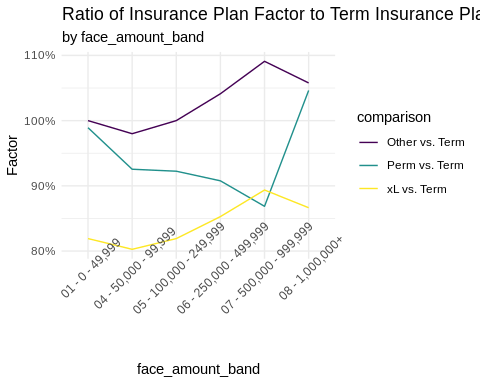
### iy\_band1



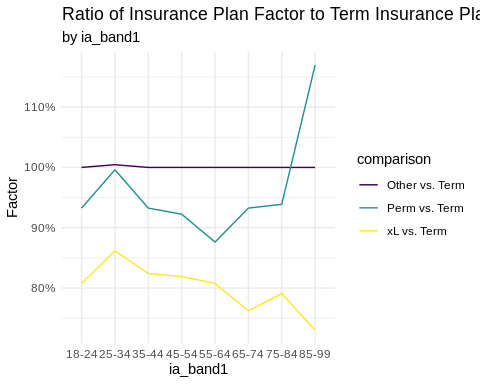
### uw



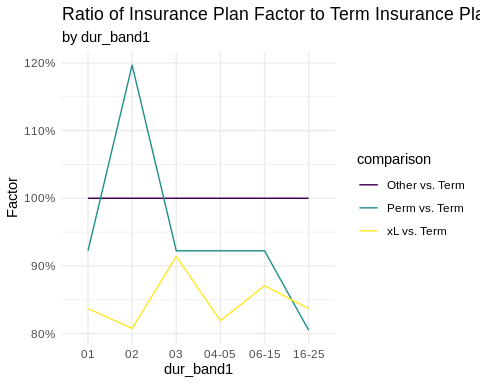
### face\_amount\_band



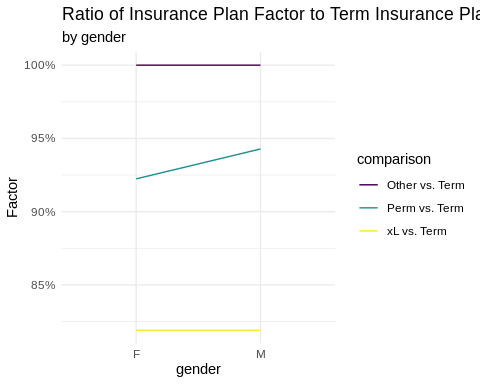
### ia\_band1



### dur\_band1



### gender

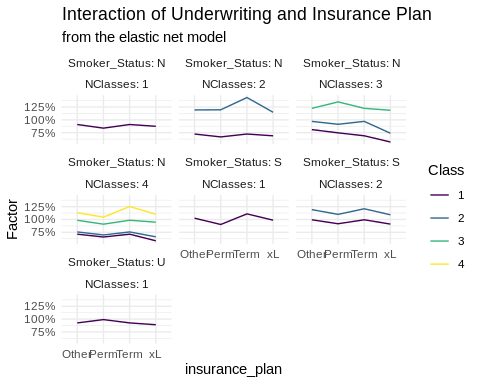


* It appears that the gap between UL/VL/ULSG/VLSG and term has been narrowing with increasing issue year.
* xL and Other tend to have a wider spread of factors for face amount than term, while perm has a narrower spread of face amount factors than term.
* Perm and xL tend to have flatter slope than Term by issue age, except above issue age 65. Above issue age 65, the slope of Perm and xL diverge.
* Perm tends to have higher duration 2 experience than others.
* The gender differential for males is narrower for Perm than for term.

A different view helps illustrate the interactions of underwriting and insurance plan. It is easier to see in this view that

* The residual standard class of a 2-class non-smoker system for Term is much higher than the others.
* The spread for Term and xL in the 4-class non-smoker system is wider than for Perm and Other.

## Create graph of interaction of Underwriting and Insurance Plan  
  
tableCVNetCoefs(train.grid,  
 c("uw", "insurance\_plan"),  
 "Factor",  
 pred\_cols,  
 levellist = list("dur\_band1" = "04-05")) %>%  
 pivot\_longer(cols = c("Perm", "Term", "Other", "xL"), values\_to = "Factor", names\_to = "insurance\_plan") %>%  
 separate(col = uw, into = c("Smoker\_Status", "NClasses", "Class"), sep = "/", remove = FALSE) %>%  
 data.table() -> tblFacts  
  
  
tblFacts %>%  
 ggplot(aes(x = insurance\_plan, y = Factor)) +  
 geom\_line(aes(group = Class, color = Class)) +  
 facet\_wrap(Smoker\_Status ~ NClasses, labeller = label\_both) +  
 theme\_minimal() +  
 scale\_y\_continuous(labels = scales::percent) +  
 scale\_color\_viridis\_d() +  
 ggtitle(  
 "Interaction of Underwriting and Insurance Plan",  
 subtitle = "from the elastic net model"  
 )



# Summary

In this analysis, we explored the application of a predictive modeling framework within actuarial experience studies, focusing on mortality differences by product type in the ILEC dataset. Our analysis revealed several key insights and mortality differentials that can be useful to understand drivers of mortality. This framework and the findings demonstrated the use and considerations required of predictive modeling and underscore its value in making informed actuarial decisions.

# Acknowledgements

## Working Group Members

This framework is the result of the tireless efforts of members of the Individual Life Experience Committee, including

* Philip Adams, FSA (chair)
* Cynthia Edwalds, FSA
* Brian Holland, FSA
* Ed Hui, FSA
* Michael Niemerg, FSA
* Haofeng Yu, FSA

## Society of Actuaries

The authors would like to thank the staff of the Society of Actuaries for their help throughout this project. Many thanks to Korrel Crawford and Pete Miller.

# Appendices

## Computational Requirements

## view session info for documentation purposes  
sessionInfo()

## R version 4.4.1 (2024-06-14)  
## Platform: x86\_64-pc-linux-gnu  
## Running under: Ubuntu 22.04.4 LTS  
##   
## Matrix products: default  
## BLAS: /usr/lib/x86\_64-linux-gnu/openblas-pthread/libblas.so.3   
## LAPACK: /usr/lib/x86\_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.20.so; LAPACK version 3.10.0  
##   
## locale:  
## [1] C  
##   
## time zone: Etc/UTC  
## tzcode source: system (glibc)  
##   
## attached base packages:  
## [1] parallel stats graphics grDevices utils datasets methods   
## [8] base   
##   
## other attached packages:  
## [1] flexlsx\_0.2.1 openxlsx2\_1.8 MatrixModels\_0.5-3 patchwork\_1.2.0   
## [5] shapviz\_0.9.3 here\_1.0.1 arrow\_16.1.0 ftExtra\_0.6.4   
## [9] flextable\_0.9.6 dtplyr\_1.3.1 magrittr\_2.0.3 lubridate\_1.9.3   
## [13] forcats\_1.0.0 stringr\_1.5.1 purrr\_1.0.2 readr\_2.1.5   
## [17] tibble\_3.2.1 tidyverse\_2.0.0 doParallel\_1.0.17 iterators\_1.0.14   
## [21] foreach\_1.5.2 tidyr\_1.3.1 ggplot2\_3.5.1 EIX\_1.2.0   
## [25] dplyr\_1.1.4 glmnet\_4.1-8 Matrix\_1.6-5 lmtest\_0.9-40   
## [29] zoo\_1.8-12 data.table\_1.15.4 lightgbm\_4.4.0 pre\_1.0.7   
##   
## loaded via a namespace (and not attached):  
## [1] libcoin\_1.0-10 mycor\_0.1.1 RColorBrewer\_1.1-3   
## [4] rstudioapi\_0.16.0 jsonlite\_1.8.8 shape\_1.4.6.1   
## [7] farver\_2.1.2 DALEX\_2.4.3 rmarkdown\_2.27   
## [10] ragg\_1.3.2 vctrs\_0.6.5 askpass\_1.2.0   
## [13] iBreakDown\_2.1.2 htmltools\_0.5.8.1 plotrix\_3.8-4   
## [16] curl\_5.2.1 broom\_1.0.6 xgboost\_1.7.7.1   
## [19] Formula\_1.2-5 sjmisc\_2.8.10 ppcor\_1.1   
## [22] htmlwidgets\_1.6.4 plyr\_1.8.9 uuid\_1.2-0   
## [25] mime\_0.12 lifecycle\_1.0.4 pkgconfig\_2.0.3   
## [28] sjlabelled\_1.2.0 R6\_2.5.1 fastmap\_1.2.0   
## [31] plotmo\_3.6.3 shiny\_1.8.1.1 digest\_0.6.35   
## [34] colorspace\_2.1-0 rprojroot\_2.0.4 textshaping\_0.4.0   
## [37] labeling\_0.4.3 fansi\_1.0.6 urltools\_1.7.3   
## [40] timechange\_0.3.0 mgcv\_1.9-1 compiler\_4.4.1   
## [43] bit64\_4.0.5 fontquiver\_0.2.1 withr\_3.0.0   
## [46] backports\_1.5.0 highr\_0.11 MASS\_7.3-60   
## [49] openssl\_2.2.0 gfonts\_0.2.0 tools\_4.4.1   
## [52] zip\_2.3.1 httpuv\_1.6.15 glue\_1.7.0   
## [55] nlme\_3.1-165 promises\_1.3.0 inum\_1.0-5   
## [58] grid\_4.4.1 reshape2\_1.4.4 generics\_0.1.3   
## [61] gtable\_0.3.5 tzdb\_0.4.0 hms\_1.1.3   
## [64] xml2\_1.3.6 utf8\_1.2.4 ggrepel\_0.9.5   
## [67] pillar\_1.9.0 ggiraphExtra\_0.3.0 partykit\_1.2-20   
## [70] later\_1.3.2 splines\_4.4.1 lattice\_0.22-5   
## [73] survival\_3.7-0 bit\_4.0.5 tidyselect\_1.2.1   
## [76] fontLiberation\_0.1.0 knitr\_1.47 fontBitstreamVera\_0.1.1  
## [79] crul\_1.4.2 xfun\_0.45 stringi\_1.8.4   
## [82] yaml\_2.3.8 evaluate\_0.24.0 codetools\_0.2-19   
## [85] httpcode\_0.3.0 officer\_0.6.6 gdtools\_0.3.7   
## [88] cli\_3.6.2 rpart\_4.1.23 xtable\_1.8-4   
## [91] systemfonts\_1.1.0 munsell\_0.5.1 Rcpp\_1.0.12   
## [94] triebeard\_0.4.1 assertthat\_0.2.1 viridisLite\_0.4.2   
## [97] mvtnorm\_1.2-5 ggiraph\_0.8.10 scales\_1.3.0   
## [100] earth\_5.3.3 insight\_0.20.1 crayon\_1.5.2   
## [103] rlang\_1.1.4

# References

Corporation, Microsoft, and Steve Weston. 2022. *doParallel: Foreach Parallel Adaptor for the ’Parallel’ Package*. <https://CRAN.R-project.org/package=doParallel>.

Dowle, Matt, and Arun Srinivasan. 2023. *Data.table: Extension of ‘Data.frame‘*. <https://CRAN.R-project.org/package=data.table>.

Fokkema, Marjolein. 2020. “Fitting Prediction Rule Ensembles with R Package pre.” *Journal of Statistical Software* 92 (12): 1–30. <https://doi.org/10.18637/jss.v092.i12>.

Gohel, David, and Panagiotis Skintzos. 2023. *Flextable: Functions for Tabular Reporting*. <https://CRAN.R-project.org/package=flextable>.

Maksymiuk, Szymon, Ewelina Karbowiak, and Przemyslaw Biecek. 2021. *EIX: Explain Interactions in ’XGBoost’*. <https://CRAN.R-project.org/package=EIX>.

Mayer, Michael. 2023. *Shapviz: SHAP Visualizations*. <https://CRAN.R-project.org/package=shapviz>.

Müller, Kirill. 2020. *Here: A Simpler Way to Find Your Files*. <https://CRAN.R-project.org/package=here>.

Pedersen, Thomas Lin. 2023. *Patchwork: The Composer of Plots*. <https://CRAN.R-project.org/package=patchwork>.

R Core Team. 2023. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. <https://www.R-project.org/>.

Richardson, Neal, Ian Cook, Nic Crane, Dewey Dunnington, Romain François, Jonathan Keane, Dragoş Moldovan-Grünfeld, Jeroen Ooms, and Apache Arrow. 2023. *Arrow: Integration to ’Apache’ ’Arrow’*. <https://CRAN.R-project.org/package=arrow>.

Shi, Yu, Guolin Ke, Damien Soukhavong, James Lamb, Qi Meng, Thomas Finley, Taifeng Wang, et al. 2023. *Lightgbm: Light Gradient Boosting Machine*. <https://CRAN.R-project.org/package=lightgbm>.

Tay, J. Kenneth, Balasubramanian Narasimhan, and Trevor Hastie. 2023. “Elastic Net Regularization Paths for All Generalized Linear Models.” *Journal of Statistical Software* 106 (1): 1–31. <https://doi.org/10.18637/jss.v106.i01>.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. <https://ggplot2.tidyverse.org>.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D’Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. “Welcome to the tidyverse.” *Journal of Open Source Software* 4 (43): 1686. <https://doi.org/10.21105/joss.01686>.

Wickham, Hadley, Maximilian Girlich, Mark Fairbanks, and Ryan Dickerson. 2023. *Dtplyr: Data Table Back-End for ’Dplyr’*. <https://CRAN.R-project.org/package=dtplyr>.

Zeileis, Achim, and Torsten Hothorn. 2002. “Diagnostic Checking in Regression Relationships.” *R News* 2 (3): 7–10. <https://CRAN.R-project.org/doc/Rnews/>.