Pricing in insurance

Intro to insurance pricing (and insurance in general)

A brief history of modern insurance

- First fire insurance company: Hamburger Feuerkasse (1676)
  - Similar companies in London (1680s) after the Great Fire of London (1666)
  - And in Copenhagen after the Great Fire of Copenhagen (1728) creating Kjøbenhavns Brandforsikring (1731)
- Providing protection to insured through own fire brigades
- Many of these companies where mutual companies
  - No listing and profit sharing with policy holders

How does insurance companies make money?

$$\text{Profit} = \text{Premiums} - \text{Claims} - \text{Expenses}$$

To formalize a bit

$$P = \pi - X - e,$$

where P is the profit, π is the premium, X is the random variable representing the claims and e is the expenses. As X (and to a degree e) are random variables, so is P.

So how do we make sure that $P \geq 0$? i.e. how do we determine the premium?

Determine the premium

The naive approach

Lets simplify by not having any expenses, i.e. $e = 0$.

Then we might have have

$$\pi_{\text{naive}} = \mathbb{E}[X]$$

Decomposing the random variable X

Decompose X into frequency and severity

$$X = \sum_{i=0}^{N} S_i$$

where N is the number of claims and S_i is the severity of each claim.

Typical distributions are N being a Poisson distributed random variable and $S_i \sim S$ are Gamma distributed random variable.

$$N \sim \text{Poisson}(\lambda), \quad S \sim \text{Gamma}(\alpha, \beta), \quad \lambda, \alpha, \beta > 0, \quad N \perp\!\!\!\perp S$$

This gives us a premium π of

$$\pi_{naive} = \mathbb{E}[X] = \mathbb{E}[N] \cdot \mathbb{E}[S] = \lambda \cdot \frac{\alpha}{\beta}$$

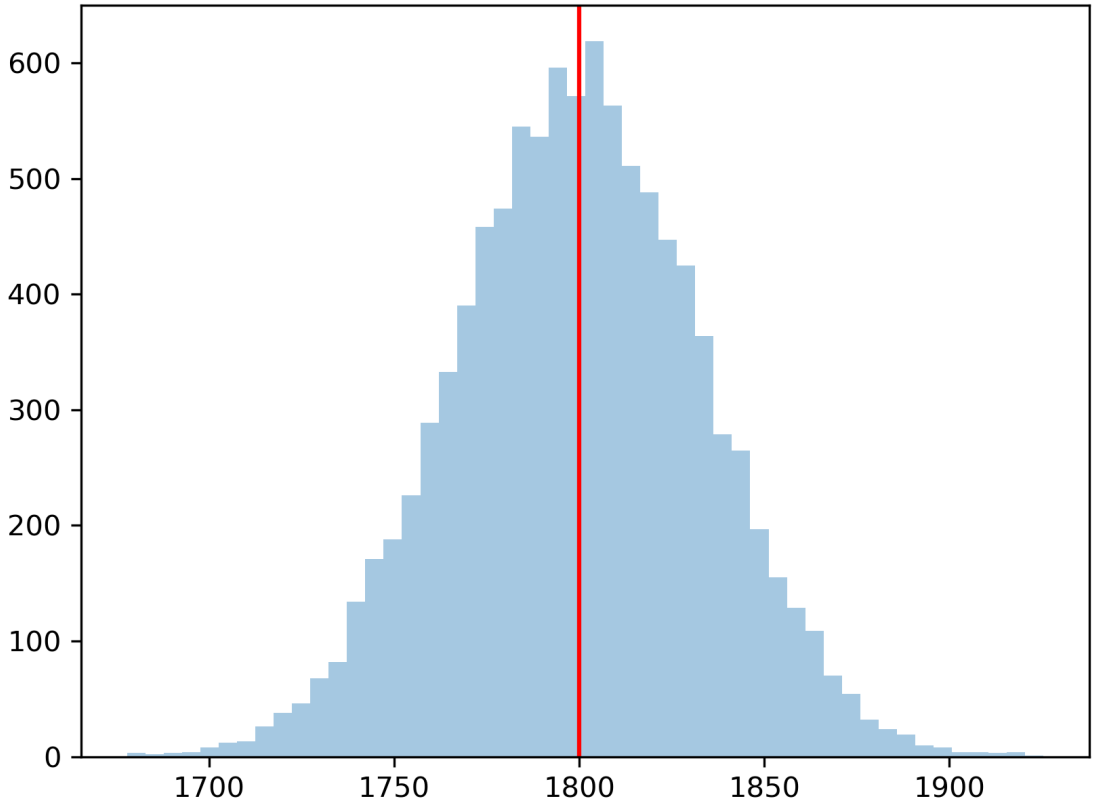
Simulating the naive approach

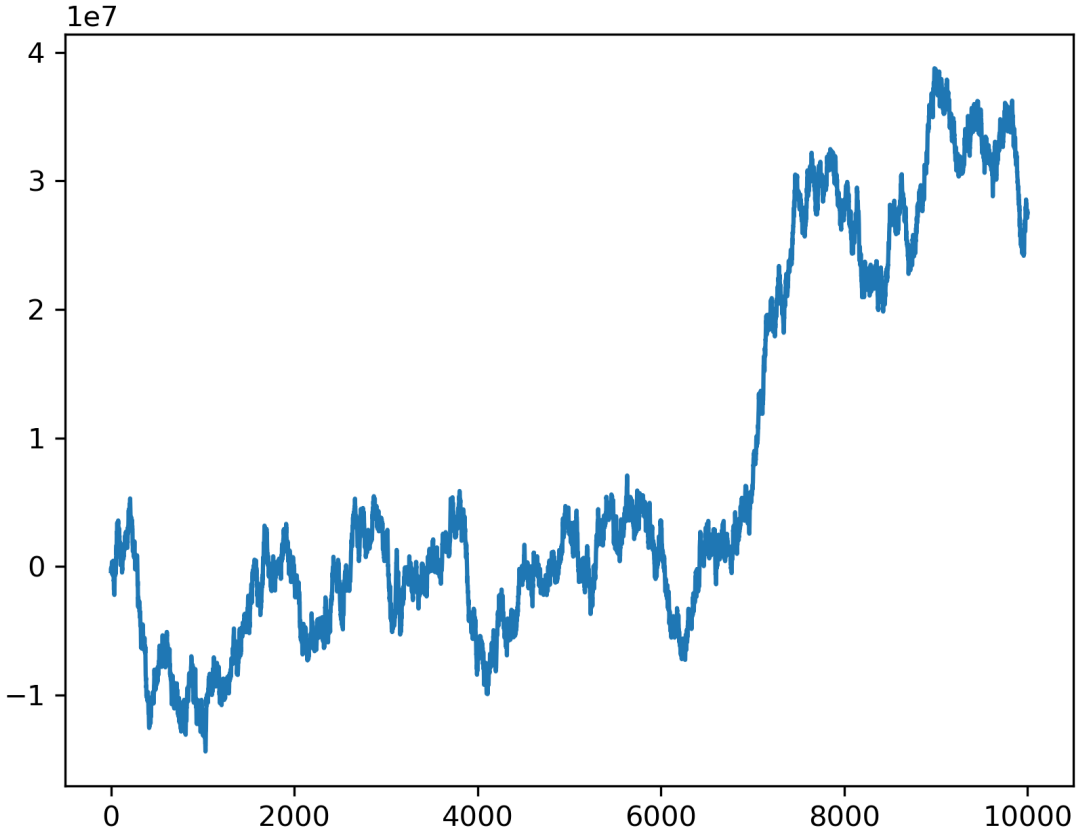
With $\lambda = 0.03$, $\alpha = 20$ and $\beta = 1/300$ we get

$$\pi_{naive} = 0.03 \cdot \frac{20}{1/300} = 1800$$

Lets simulate this in Python

```
n_iter = 10000
n_policy = 10000
sims = []
for i in range(n_iter):
    n_claims = npr.poisson(par_lambda, n_policy)
    severity = npr.gamma(gamma_shape, gamma_scale, n_claims.sum())
    # Compute the average claim expense
    sims.append(severity.sum()/n_policy)
```


600
500
400
300
200
100
0
1700
1750
1800
1850
1900


1e7
4
3
2
1
0
−1
0
2000
4000
6000
8000
10000

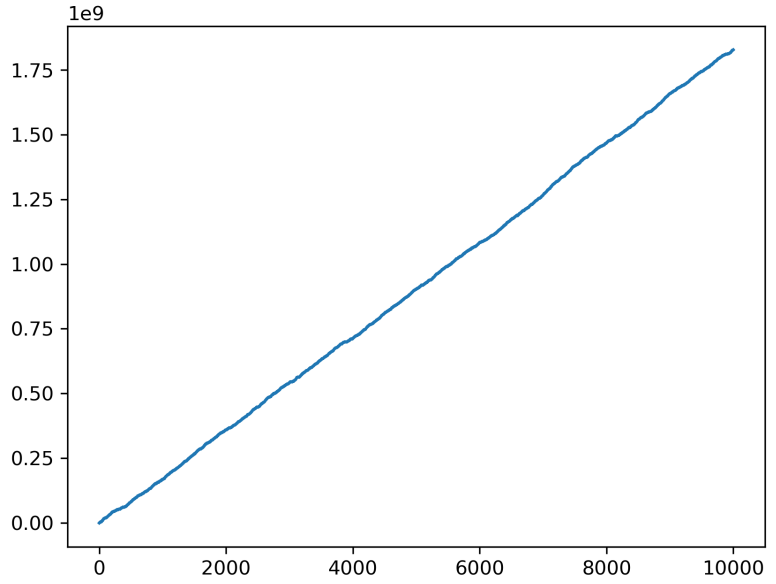
Loading the premium with a safety margin

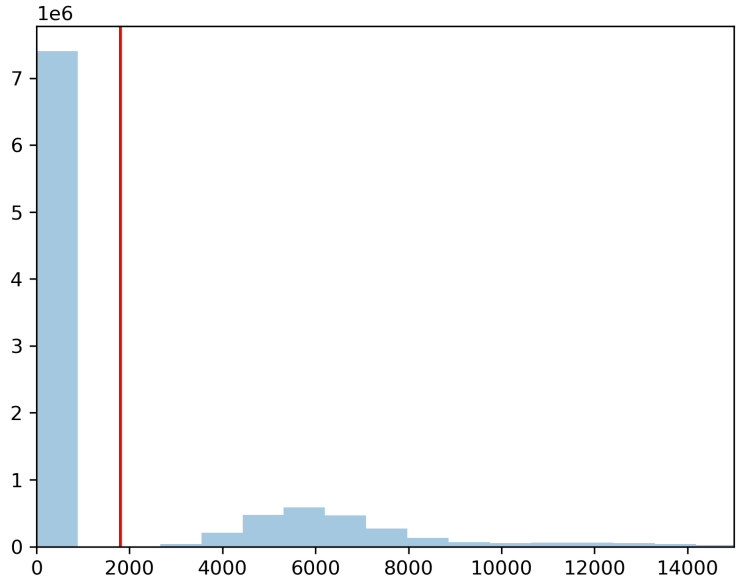
Add 1% safety margin to the premium

But why would the customer pay for this?

Insured are paying a *risk premium* to avoid the risk of a large loss.

So when is an insurance agreement reached?





Utility of insured

The insured is risk averse, i.e. the utility of the insured is concave. This means that the insured is willing to pay a risk premium to avoid the risk of a large loss.
In other words

$$u(w - G) = \mathbb{E}[u(w - X)] \leq u(w - \mathbb{E}[X]) \implies$$
$$w - G \leq w - \mu \implies G \geq \mu$$

as a result from Jensen's inequality, where u is the utility function of the insured, G is the premium willing to pay and μ is the expected value of the loss.

Utility of insurer

The insurer is also risk averse with a utility function u_I and wealth w_I (but closer to $u_I(w_I) = w_I$).
H is the premium charged to the insured and X is the loss.

$$u_I(w_I) = \mathbb{E}[u_I(w_I + H - X)] \leq u_I(w_I + H - \mathbb{E}[X]) \implies$$
$$w_I \leq w_I + H - \mu \implies H \geq \mu$$

As a result when $G \geq H$, the insurance agreement is reached as the insured has a higher utility and the insurer has a higher wealth with unchanged utility.

Another implication is a range of premiums that are acceptable to both parties:

- Having a higher window price and bargaining down to a lower price will a premium $\pi = G$ and an extra profit of $\pi - H$ for the insurer.

Due to pooling of risk (law of large numbers), the insurer is not exposed to the same risk as the insured (lower variance). This means that "for the same quantile risk" the insurer is usually willing to accept a lower premium than the insured is willing to pay.

Insured or insurer have a wrong perception of the risk. As an insurance company here's where pricing enters the picture.

The "art" of pricing the insurance

We don't have the luxury of knowing the true distribution of the loss or having identical policy holders. We have to estimate it from historical data.

Need a model which can account for different policy holders having different risk profiles. This is where GLM (Generalized Linear Models) comes into play.

Linear regression

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

where y is the dependent variable, x_i are the independent variables, β_i are the coefficients and ϵ is the error term. Here error term is assumed to be normally distributed with mean 0 and variance σ^2.

The core idea is to assign a weights β_i to each independent variables x_i representing the age, postal code, etc.. of the policy holder.

But as the example hinted, we don't have a normal distribution of the loss. We have a distribution of the loss which is skewed to the right for the severity and a counting distribution for the number of claims (or a compound distribution). This is where GLM comes into play.

Generalized Linear Models

Generalized Linear Models (GLM) is a generalization of linear regression. The core idea is to transform the dependent variable y using a link function g such that

$$g(\mathbb{E}[Y|X]) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

where g is the link function and $\mathbb{E}[y] = \mu$ is the expected value of Y.

Log link function

The log link function is a common choice in insurance because of it's multiplicative properties. The log link function is defined as

$$g(\mu) = \log(\mu)$$

This means that

$$\log(\mathbb{E}[Y|X]) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p \implies$$
$$\mathbb{E}[Y|X] = \mathrm{e}^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p} = \prod_{i=0}^{p} \mathrm{e}^{\beta_i x_i}$$

with $x_0 = 1$.

Back to the risk

From before we had the decomposition of the loss into frequency and severity

$$X = \sum_{i=0}^{N} S_i = N \cdot S$$

We get $X \sim N \cdot S$ from independence of N and S_i, and $S_i \sim S$.

$$\mathbb{E}[X] = \mathbb{E}[N] \cdot \mathbb{E}[S]$$

Using GLMs with log-link function we get

$$\begin{aligned}\mathbb{E}[X] &= \mathrm{e}^{\beta_0^s+\beta_1^s x_1+\beta_2^s x_2+\cdots+\beta_p^s x_p}\mathrm{e}^{\beta_0^n+\beta_1^n x_1+\beta_2^n x_2+\cdots+\beta_p^n x_p} \\ &= \mathrm{e}^{\beta_0^s+\beta_0^n+(\beta_1^s+\beta_1^n)x_1+(\beta_2^s+\beta_2^s)x_2+\cdots+(\beta_p^s+\beta_p^n)x_p} \\ &= \prod_{i=0}^{p}\mathrm{e}^{(\beta_i^s+\beta_i^n)x_i} = \prod_{i=0}^{p}\mathrm{e}^{\beta_i x_i}\end{aligned}$$

$\mathrm{e}^{\beta_i x_i}$ can be interpreted as the relative risk of the policy holder with the characteristics x_i compared to the base policy holder with $x_i = 0$ or "risk factor".

Why not estimate the compound distribution directly?

The compound distribution with $N \sim \textbf{Poisson}(\lambda)$ and $S \sim \textbf{Gamma}(\alpha, \beta)$ is known as a Tweedie distribution and also an exponential GLM.

Limitations:

- Only works with gamma distributed severity
  - Not always the case, especially for liability insurance and other long-tail insurance
- Fitting can be quite finicky with a lot of parameters
  - Usually we have good signal in data from both frequency, but severity can be quite noisy

General limitations of GLMs

- Assumes linear relationship between the independent variables and the (link transformed) dependent variable
- No interaction between the independent variables

Solutions

- Generalized Additive Models (GAM) and Splines
  - Non-linear relationship between the independent variables and the (link transformed) dependent variable
- Feature engineering
  - Create new features from the existing features
  - e.g. create a feature for the interaction between two features

Why not machine learning?

My thesis was on using machine learning for pricing in insurance. Comparing GLMs with

- (Deep) neural network with tweedie loss function
- Marginal (Deep) neural networks with poisson and gamma loss respectively
- Gradient boosting with tweedie loss function
- Gradient boosting with poisson and gamma loss respectively

TLDR; Machine learning models outperformed GLMs, but at loss of interpretability.

Interpretability

The most important attribute of a model in insurance is interpretability. The model should be able to explain why a policy holder is charged a certain premium to sales, insurance agents, business, regulation (*Right to explanation*) (EIOPA, GDPR, Finanstilsynet), etc..

Explainability vs Interpretability

- Explainability: The model can explain why it made a certain decision
  - Why did the model predict a certain premium?
- Interpretability: The model can be understood by humans
  - If age increases by 1 year, the premium increases by 1.5%

$$\textbf{Interpretability} \implies \textbf{Explainability}$$

From model to rating table/structure

What actually gets implemented?

Take the component $\mathrm{e}^{\beta_i x_i}$. Assume that x_i represents age.

The factor related assume $b_i = 0.01$ and x_i is in years. Then

$$\mathrm{e}^{\beta_i x_i} = \mathrm{e}^{0.01 \cdot 30} = 1.3499$$

But models are not implemented continuously, but in discrete steps. So we need to discretize the continuous variable x_i into discrete steps. This is done by creating a rating table.

Rating table

Using the mid-point of the age interval as base for x_i, we get the rating table.

This has the unfortunate side effect of "errors" in the rating table for ages not in the mid-point of the interval.

Age	Factor
-20	1.22
21-30	1.28
31-40	1.42
41-50	1.57

What happens mathematically?

Originally we optimized the log-likelihood function

$$\mathcal{L}(\beta) = \sum_{i=1}^{n} \log(f(y_i|x_i, \beta))$$

where f is the probability density function of the distribution of the loss.

But the actual implementation used a rating table, so the problem we actually need to optimize is.

$$\mathcal{L}^*(\beta) = \sum_{i=1}^{n} \log(f(y_i|r_i(x_i), \beta))$$

where r_i represents the bucketing function. The optimal β for $\mathcal{L}$ and $\mathcal{L}^*$ are not the same!

How to optimize the rating table?

Within the GLM framework

Solution: Embed the bucketing function r_i into the model, by creating features for each bucket.

Problem: Loosing (local) monotonicity of the model.

Outside the GLM framework

What models are good at bucketing? Decision trees!
What algorithms (can) use decision trees? Gradient boosting!

What I would do my thesis on today

Using restricted gradient boosting algorithm to directly optimize the rating table without loss of interpretability.

Decision trees

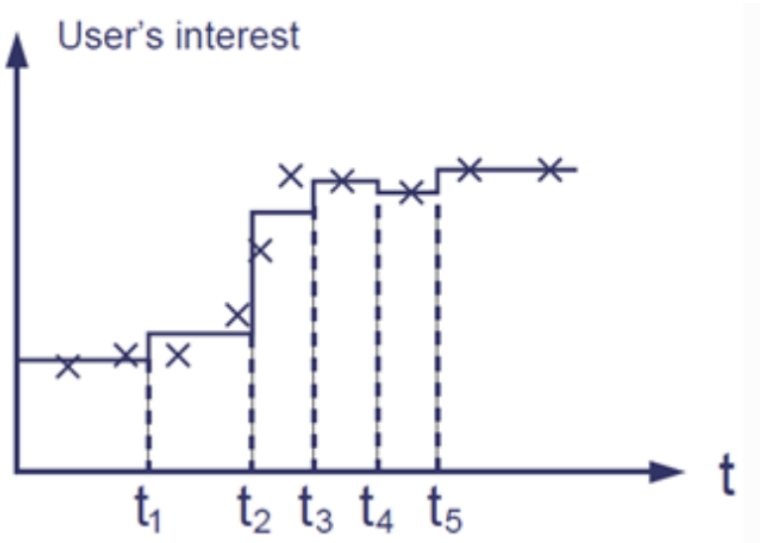
Well actually regression trees..

Gradient Tree Boosting

Ensemble method of trees

$f(x) = \sum_{i=1}^{n} f_i(x)$

where f_i is a tree.



Boosting

Training on residuals. Let y be the dependent variable and f be the model. Then the predictions at step t are $y_i^{(t)}$ with $y_i^{(0)} = 0$.

$$
\begin{aligned}
y_i^{(0)} &= 0 \\
y_i^{(1)} &= f_1(x_i) = y_i^{(0)} + f_1(x_i) \\
y_i^{(2)} &= f_1(x_i) + f_2(x_i) = y_i^{(1)} + f_2(x_i) \\
&\dots \\
y_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = y_i^{(t-1)} + f_t(x_i) \implies y_i^{(t)} - y_i^{(t-1)} = f_t(x_i)
\end{aligned}
$$

At each step are training on the negative gradient of the loss function.

Essentially training on the residuals of the previous step to correct predictions.

How this solves the problem

Gradient boosting frameworks like XGBoost and LightGBM have option for

- Monotonicity constraints
- Restrictions on the number of features used in each tree (limit interactions)
- Common insurance distribution as loss functions (including log-linking)

The log-link transforms $f(x)$ to

$$\log(f(x)) = \sum_{i=1}^{n} f_i(x) \implies$$

$$f(x) = \prod_{i=1}^{n} \exp(f_i(x))$$

For each tree we can extract the splits and use them as the rating table buckets.

Other subjects

- Demand elasticity and optimization of the price
- Risk selection bias
- When is profit realized? Reserving, IBNR, run-off triangles..
- Cross-product risk and pricing
- Fairness and ethics
- Risk of new customers vs existing customers
- Modelling life time value of customers