Comparing generalized linear models with machine-learning algorithms in risk premium modelling

Mikael Biskop, 34671
Master's Thesis
Faculty of Sciences and Engineering
Mathematics and Statistics

December 6, 2018

Contents

Chapter 1

Introduction

Artificial intelligence (AI) is revolutionizing the modern society. Whether it is the voice assistance on smart phones, product suggestions in online shopping or predicted maintenance of engines, the modern human being is being affected by intelligent machines. Machine-learning is an essential part of AI, which allows computers to handle new situations via observation and self-training. Machine-learning has also found a path into the insurance industry, where it can, for example, be used to estimate the risk premium for customers, i.e. the expected cost of the customers.

For other ways of estimating the risk premium, the insurance companies have relied on generalized linear models (GLMs), which are extensions of linear models. The GLMs are recognized as the industry standard method for pricing. According to Anderson et al. (2007, p. 4), they are commonly used in Scandinavia and in other countries in Europe. The GLMs are also gaining popularity in eastern European countries.

The purpose of this thesis is to explain the theory behind machine-learning and GLMs, as well as to compare them in risk premium pricing. To be able to tie the theory of machine-learning and GLMs to the insurance industry, the thesis starts in chapter 2 with basic facts from the probability theory. After the probability theory has been introduced, the insurance industry is briefly explained from a mathematical point of view. Chapter 2 also introduces the exponential family of distributions, which is an essential part of the GLMs.

Chapters 3 and 4 cover the theory behind the GLMs and machine-learning. Chapter 5 introduces two different machine-learning algorithms which are then used for the simulations in chapter 6. Since the area is wide and the

thesis is limited, the thesis consists only of the most important elements of the theory. Regardless, the aim is to provide a thorough review of the theory needed to obtain the results.

The contents of the thesis is mainly based on four different sources of literature. The underlying mathematics for the insurance industry is retrieved from *Practical risk theory for actuaries*, Daykin et al. (1996). The theory for the GLM is mainly obtained from *A practitioner's guide to generalized linear models*, Anderson et al. (2007) and from *An introduction to generalized linear models - second edition*, Dobson (2002). The chapters regarding machine-learning are based on *An introduction to statistical learning with application in R*, James et al. (2013). The last mentioned book has a homepage with programming laboratories which helped the coding.

Programming is required to be able to handle the methods presented in the thesis, especially the machine-learning algorithms. Since the data analyzed in the thesis are from an insurance company and not public, the programs in chapter 6 are illustrated using built-in data sets. A part of the source codes have also been saved in the Appendix, so that the user is able to execute them. The software used in the thesis is called *R*, which is a common, free and open source software for statistical computing.

The aim of the thesis is to provide an understanding of the most important contents of the different subjects reviewed, as well as an idea of what the advantages and disadvantages of the different methods are. According to James et al. (2013, p. 29), "no one method dominates all others over all possible data sets". As a large benefit, I was able to use real insurance data for the simulations. This allowed patterns and results to be explained, however, all of it cannot be published. In using a real data set, some treatment of the data set was required. This was a big benefit from a learning point of view, and the treatment of the data is shortly explained in chapter 6. Chapter 7 presents conclusions. The issues of the modelling process with the different methods, the benefits, observations and the results of the models are all summarized in the last chapter. I hope the reader finds this thesis interesting and perceptive. A big thank you to my supervisor at the insurance company, whose name will not be published.

Chapter 2

Insurance industry and some underlying mathematics

2.1 Basic model of the insurance industry

An insurer′s financial operations consist of a series of cash inflows and outflows. Earned premiums with interest and income from investments increase the surplus of assets, while payments of claims and operating costs form the expenses of the insurer. In both life and non-life insurance, the insurance companies provide their customers with compensation for financial losses caused by different adverse events. Such an event could be a fire causing damage to a house, where the insurer cover the cost of the repair works. In a life insurance, the death of the insured might result in compensation to surviving family members in the form of pension payments.

To be able to provide compensation of these incidents, the insurer receives payments in the form of insurance premiums from the insured. From the insurer′s point of view, the premium income should cover the contingent cost of these incidents and operating expenses. The premium payments are therefore received in advance, before the covered events possibly happen. Sometimes these events may not occur at all during the contract period. This results in the insurer having funds in the balance sheet temporarily, which can be invested profitably to generate investment returns.

Positive investments returns can be used to increase the insurer′s profit margin, by offsetting the cost of incurred claims or the insurer′s operational costs, such as wages or rents etc. The risk of the insurer not being able to

meet its obligations should remain on an acceptable level, which means that the assets of the insured should at all times be sufficient to cover liabilities with a high probability.

2.2 Basic facts from the probability theory

Definition 2.1.

(i) The function F with $D_f = \mathbb{R}$ is a ***cumulative distribution function*** *if*

(a) F is non-decreasing and right continuous,

(b) $lim_{x\to-\infty}F(x) = 0$ and $lim_{x\to\infty}F(x) = 1$.

(ii) The function f with $D_f = R$ is called a ***probability density function****, if it is non-negative, integrable and*

$$\int_{-\infty}^{+\infty} f(x)dx = 1.$$

Definition 2.2. *Let Ω be a given non-empty set. The family $\mathcal{F}$ of Ω:s subsets is said to be a σ-algebra if*

(i) $\varnothing \in \mathcal{F}$,

(ii) $A \in \mathcal{F} \Rightarrow A^C \in \mathcal{F}$,

(iii) $A_i \in \mathcal{F},\ i = 1, 2, \ldots \Rightarrow \bigcap_{i=1}^{\infty} A_i \in \mathcal{F}$.

Definition 2.3. *A probability measure* $\mathbf{P}$ *in* $(\Omega, \mathcal{F})$ *is a real-valued function*

$$\mathbf{P} : \mathcal{F} \to \mathbb{R}$$

such that

(i) $\mathbf{P}(\Omega) = 1$,

(ii) $\mathbf{P}(A) \geqslant 0 \quad \forall A \in \mathcal{F}$,

(iii) $\mathbf{P}(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mathbf{P}(A_i)$, *where $A_i \in \mathcal{F}$ and $A_i \cap A_j = \varnothing, i \neq j$.*

Definition 2.4. *Let $(\Omega, \mathcal{F}, \mathbf{P})$ be a probability space. The function $X : \Omega \to \mathbb{R}$ is a random variable if for every $x \in \mathbb{R}$ it holds*

$$\{\omega : X(\omega) \leqslant x\} \in \mathcal{F}.$$

Definition 2.5. *Let X be a random variable. The function*

$$F_X(x) = \mathbf{P}(X \leqslant x), \quad x \in \mathbb{R},$$

is called a cumulative distribution function (CDF) of X.

A random variable that can take on at most a countable number of possible values is said to be a *discrete* random variable. A random variable whose set of possible values is uncountable is called a *continuous* random variable.

Definition 2.6.

(i) Random variable X is called discrete if it attains only countable different values $a_1, a_2, \ldots$. The distribution function X is given by

$$F_X(x) = \sum_{a_i \leq x} \mathbf{P}(x = a_i).$$

(ii) Random variable X is called continuous if there exist a density function f_X such that

$$F_X(x) = \int_{-\infty}^{x} f_X(t)dt.$$

Theorem 2.7. *Assume f is continuous on the interval $[a, b]$ and define*

$$F(x) = \int_a^x f(t)dt, \quad x \in [a, b].$$

Then F is continuous on $[a, b]$ and $F'(x) = f(x), \quad \forall x \in (a, b)$.

Remark 2.8. *If the variable X has the density f which is continuous in x, it follows from 2.7 that*

$$F'_X(x) = f(x).$$

Definition 2.9.

(i) Let X be a discrete random variable taking values in $\{a_1, a_2, \dots\}$ with the cumulative distribution function F_X. It is said that X has the expectation $\mathbf{E}[X]$ if

$$\sum_{i=1}^{\infty} |a_i| b_i < \infty,$$

where $b_i := \mathbf{P}(X = a_i) = F_X(a_i) - F_X(a_i-)$. In this case,

$$\mathbf{E}[X] = \sum_{i=1}^{\infty} a_i b_i.$$

(ii) Let X be a continuous random variable with the density function f_X. $\mathbf{E}[X]$ exist if and only if

$$\int_{-\infty}^{\infty} |x| f_X(x) dx < \infty,$$

and in this case

$$\mathbf{E}[X] = \int_{-\infty}^{\infty} x f_X(x) dx.$$

Theorem 2.10.

(i) Let X be a discrete random variable. Let g be a real-valued function and assume the expected value of $Y := g(X)$ exist. Then

$$\mathbf{E}[Y] = \sum_{i=1}^{\infty} g(a_i) \mathbf{P}(X = a_i).$$

(ii) Let X be a continuous random variable and let g be a Borel measurable function. Assume the expectation of $g(X)$ exist. Then

$$\mathbf{E}[g(X)] = \int_{\infty}^{\infty} g(x) f_X(x) dx.$$

Definition 2.11. *Let X be a random variable such that $\mathbf{E}[X]$ exist. If $\mathbf{E}[(X - \mathbf{E}[X])^2] < \infty$, it is said that the variance of X exist and it is given by*

$$\mathbf{Var}(X) = \mathbf{E}\Big[[X - \mathbf{E}[X]]^2\Big].$$

The number $\sqrt{\mathbf{Var}(X)}$ is called the standard deviation of X.

2.3 Number of claims

According to Daykin et al. (1994, p. 30), both the number of claims and the size of each claim can be viewed as random variables. The final model is therefore constructed accordingly. In this section the focus is put on the number of claims N, whereas the individual claim size is analyzed in section 2.4.

The behaviour of the discrete random variable N, which represents the number of claims in a specific time period, can be described by its probability distribution. The probability that exact k claims occur in a given time period is

$$\mathbf{P}(N = k), \quad k = 0, 1, 2, \dots \ .$$

For insurance claims, it is impossible to forecast the exact time between the events or the exact total number of claims. It can, however, be assumed that the claims occur in a specific manner.

1. The number of claims occur independently of each other, in any two disjoint time intervals.
2. Only one claim may arise from the same event.
3. The probability for a claim occuring in a specific time point is equal to zero.

In that case, according to Daykin et al. (1994, p.32), the number of claims in any given time period is Poisson-distributed.

Definition 2.12. *The discrete random variable N, taking values in the set of nonnegative integers, is said to be Poisson distributed with the parameter $\lambda > 0$, if for $k = 0, 1, 2, \dots$*

$$\mathbf{P}(N = k) = e^{-\lambda}\frac{\lambda^k}{k!}, \quad \textit{where} \quad \lambda > 0.$$

The expectation and the variance are given by

$$\mathbf{E}[N] = \lambda \quad \text{and} \quad \mathbf{Var}[N] = \lambda.$$

For a more precise description, let $N(t)$ be defined as the number of claims occuring during a time period $[0, t]$. Then, by following Daykin et al. (1994), we assume that $N(t)$ has the following properties:

(i) $N(t) \geq 0$,

(ii) $N(t)$ is integer valued,

(iii) If $s < t$, then $N(s) \leq N(t)$,

(iv) For $s < t$, $N(t) - N(s)$ equals the number of events that have occurred in the interval $(s, t]$.

(v) The number of claims $N(t)$ possesses *independent increments*. In other words, for any disjoint time intervals, the number of claims that occur are independent. For example, the number of claims that have occurred by time $[0, t]$ that is $N(t)$, is independent of the number of claims occuring in the time period $(t, t + s]$ that is $N(t + s) - N(t)$ for all $s > 0$.

(vi) At most one claim may occur at a given time point t, i.e. the probability that more than one claim occurs at the same time is zero.

(vii) The number of claims $N(t)$ possesses *stationary increments*. This means that the number of claims occuring during the interval $(t_1, t_2]$ that is $N(t_2) - N(t_1)$, has the same distribution as the number of claims in the interval $(t_1 + s, t_2 + s]$ that is $N(t_2 + s) - N(t_1 + s)$, for all $t_1 < t_2$ and $s > 0$.

A stochastic process having properties $(i) - (viii)$ is called a Poisson process. The mathematical definition is as follows:

Definition 2.13. *A stochastic process $\{N(t), t \geq 0\}$ is said to be a Poisson process having rate $\lambda > 0$, if:*

(i) $N(0) = 0$,

(ii) the process has independent increments,

(iii) the number of events in any interval of length t is Poisson distributed with mean λt, that is, for all $s, t > 0$,

$$\mathbf{P}(N(t + s) - N(s) = n) = e^{-\lambda t} \frac{(\lambda t)^n}{n!}, \quad n = 0, 1, 2, \ldots \ .$$

Note that condition (*iii*) in Definition 2.13 implies that a Poisson process has stationary increments. The parameter λ is called the rate of the process. The expected value of the Poisson process X at time t is (cf. (2.1))

$$\mathbf{E}[N(t)] = \lambda t. \tag{2.1}$$

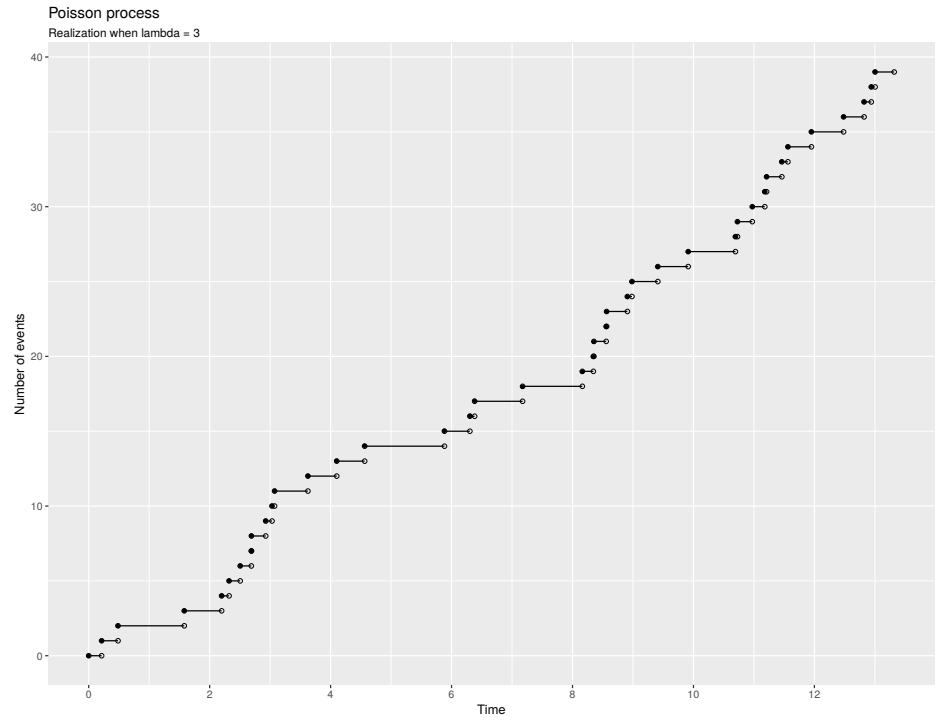


Figure 2.1: Realization of a Poisson process when $\lambda = 3$. The source code is found in appendix A.1.

For a Poisson process, let X_n denote the time between the $(n-1)$st and the nth event. The sequence $\{X_n, n \geq 1\}$ is called the *sequence of the interarrival times*.

Theorem 2.14. *The interarrival times $X_n, n = 1, 2, \ldots$ are independent identically distributed exponential random variables having the mean $1/\lambda$.*

Proof. Note that the event $\{X_1 > t\}$ takes place if and only if no events in the Poisson process occur in the interval $[0, t]$. Thus

$\mathbf{P}(X_1 > t) = \mathbf{P}(N(t) = 0) = e^{-\lambda t}$.

For X_2, by using the arguments of independent and stationary increments, we have

$\mathbf{P}(X_2 > t + s | X_1 = s) = \mathbf{P}(\text{ No events in } (s, t+s] | \text{ One event in } (0, s))$

$= \mathbf{P}(\text{ No events in } (0, t)) = \mathbf{P}(N(t) = 0) = e^{-\lambda t}$.

Using the same reasoning,

$\mathbf{P}(X_n \leq t) = 1 - \mathbf{P}(X_n > t) = 1 - \mathbf{P}(X_n > t + s | X_{n-1} = s)$

$= 1 - \mathbf{P}(N(t) = 0) = 1 - e^{-\lambda t} \sim Exp(\lambda)$.

Therefore, $X_n, n = 1, 2, \ldots,$ are independent exponentially distributed random variables with parameter λ. Hence, $\mathbf{E}[X_n] = \frac{1}{\lambda}$ as claimed.

□

Theorem 2.15. *Assume $N(t)$ is a Poisson process and that exactly one event has happened in the interval $(0, t]$. Then, $X_1 \sim U(0, t)$.*

Proof. For $s \leq t$ we have

$$\mathbf{P}(X_1 < s | N(t) = 1) = \frac{\mathbf{P}(X_1 < s, N(t) = 1)}{\mathbf{P}(N(t) = 1)} = \frac{\mathbf{P}(1 \text{ event in} [0, s), 0 \text{ events in} [s, t))}{\mathbf{P}(N(t) = 1)}$$

$$= \frac{\mathbf{P}(1 \text{ event in} [0, s)) \mathbf{P}(0 \text{ events in} [s, t))}{\mathbf{P}(N(t) = 1)} = \frac{[e^{-\lambda s} \lambda s][e^{-\lambda (t-s)}]}{e^{-\lambda t} \lambda t} = \frac{s}{t},$$

completing the proof.

□

The assumption of independent increments for the number of claims in the insurance business is only approximately true. There are background factors, which results in that the number of claims during disjoint time intervals are, in fact, correlated. For example, the number of claims for car insurances correlates with the season of the year.

2.4 Amount of claims

The claim amount is the sum which the insurer has to pay in the occurence of an event i.e. accident. The sum of these individual claims establish the aggregate claim amount, which is one of the key concerns for an insurance company. The aggregate claim amount is considered in a specific time interval, which usually is a calendar year.

To set up a model for the aggregate claim amount, a doubly stochastic modell is constructed. Let $\{N(t), t \geq 0\}$ be a Poisson process representing the amount of claims for a certain time period, and let Z_i be the amount size of the i:th claim during the time period. The family $\{Z_i, i \geq 1\}$ is assumed to be independent and identically distributed random variables, that are independent of $\{N(t), t \geq 0\}$.

When these requirements are met, the aggregate claim amount $\{X(t), t \geq 0\}$ is a stochastic process called a *compound Poisson process*, where

$$X(t) = \begin{cases} \sum_{i=1}^{N(t)} Z_i, & \text{if } N(t) > 0 \\ \\ 0, & \text{if } N(t) = 0. \end{cases} \tag{2.2}$$

The separation of the aggregate claim amount into the number of claims and individual claims is due to a practical reason. The insurer usually has a wide range of data for the individual claims, and the number of claims can be estimated using the Poisson process. The assumption that the individual claims Z_i are identically distributed may, however, be unrealistic. For example, inflation may change the distribution of the individual claims. The derivation of the claim size distribution is beyond the scope of this section. In this thesis, however, using statistical models and machine-learning algorithms, we will estimate the amount of claims for different customers using observations from insurance data.

The aim is to find an expression for the probability distribution of the aggregate claim amount $\{X(t), t \geq 0\}$, in terms of the claim number probabilities and the distribution of the claim size. The event $\{X(t) \leq x\}$ can occur in the following alternative ways:

$$
\begin{aligned}
N(t) = 0 &\implies X(t) = 0 \\
N(t) = 1 &\implies X(t) = Z_1 \\
N(t) = 2 &\implies X(t) = Z_1 + Z_2 \\
&\text{etc.}
\end{aligned}
$$

By assuming that $\{Z_i, i \geq 1\}$ are independent of $\{N(t), t \geq 0\}$, the probability distribution of $X(t)$ can now be written as

$$
\begin{aligned}
F_X(x) = \mathbf{P}(X(t) \leq x) &= \sum_{k=0}^{\infty} \mathbf{P}(X(t) \leq x | N(t) = k) \mathbf{P}(N = k) \\
&= \sum_{k=0}^{\infty} \mathbf{P}(Z_1 + Z_2 + \ldots Z_k \leq x) \mathbf{P}(N = k).
\end{aligned}
$$

The risk premium is the premium the insurance company requires to cover the expected cost of claims. The risk premium $\mathbf{E}[X(t)], t \geq 0$ is based on the following theorem.

Theorem 2.16. *Let $N(t), t \geq 0$ be a Poisson process with the rate λ and let $Z_i, i = 1, 2, \ldots$ be independent identically distributed random variables independent of $N(t)$. Assume $\mathbf{E}[Z_i] = \mu$. Then*

$$
\mathbf{E}[X(t)] = \mathbf{E}[\sum_{i=1}^{N(t)} Z_i] = \mu \lambda t.
$$

Proof. Let $t > 0$ and consider

$$
\begin{aligned}
\mathbf{E}[X(t)] &= \mathbf{E}[\sum_{i=1}^{N(t)} Z_i] \\
&= \sum_{n=1}^{\infty} \mathbf{E}[\sum_{i=1}^{N(t)} Z_i | N(t) = n] \mathbf{P}(N(t) = n) \\
&= \sum_{n=1}^{\infty} \mathbf{E}[\sum_{i=1}^{n} Z_i | N(t) = n] \mathbf{P}(N(t) = n),
\end{aligned}
$$

since $N(t)$ is independent of Z_i. Now we proceed as follows:

$$\begin{aligned}
\mathbf{E}[X(t)] &= \sum_{n=1}^{\infty} \mathbf{E}[\sum_{i=1}^{n} Z_i]\mathbf{P}(N(t)=n) \\
&= \sum_{n=1}^{\infty} (\sum_{i=1}^{n} \mathbf{E}[Z_i])\mathbf{P}(N(t)=n) \\
&= \sum_{n=1}^{\infty} (\sum_{i=1}^{n} \mu)\mathbf{P}(N(t)=n) \\
&= \sum_{n=1}^{\infty} n\mu\mathbf{P}(N(t)=n) \\
&= \mu\mathbf{E}[N(t)] \\
&= \mu\lambda t.
\end{aligned}$$

□

The variance of the compound Poisson process is given in the following theorem.

Theorem 2.17. *Let $N(t), t \geq 0$ be a Poisson process with the rate λ and let $Z_i, i = 1, 2, \ldots$ be independent identically distributed random variables independent of $N(t)$. Assume $\mathbf{E}[Z_i] = \mu$ and $\mathbf{Var}[Z_i] = \sigma^2$. Then*

$$\mathbf{Var}[X(t)] = \mathbf{Var}[\sum_{i=1}^{N(t)} Z_i] = \lambda t(\sigma^2 + \mu^2).$$

Proof. Let $t > 0$, and consider first

$$\begin{aligned}
\mathbf{E}[X(t)^2] &= \mathbf{E}[(\sum_{i=1}^{N(t)} Z_i)^2] \\
&= \sum_{n=1}^{\infty} \mathbf{E}[(\sum_{i=1}^{N(t)} Z_i)^2 | N(t)=n]\mathbf{P}(N(t)=n) \\
&= \sum_{n=1}^{\infty} \mathbf{E}[(\sum_{i=1}^{n} Z_i)^2 | N(t)=n]\mathbf{P}(N(t)=n),
\end{aligned}$$

where the independence of $N(t), t \geq 0$ and $\{Z_1, Z_2, \dots\}$ is used. Now,

$$\begin{aligned}
\mathbf{E}[X(t)^2] &= \sum_{n=1}^{\infty} \mathbf{E}[(\sum_{i=1}^{n} Z_i)^2]\mathbf{P}(N(t) = n) \\
&= \sum_{n=1}^{\infty} [\mathbf{Var}[\sum_{i=1}^{n} Z_i] + (\mathbf{E}[\sum_{i=1}^{n} Z_i])^2]\mathbf{P}(N(t) = n) \\
&= \sum_{n=1}^{\infty} [n\sigma^2 + (n\mu)^2]\mathbf{P}(N(t) = n) \\
&= \sigma^2 \mathbf{E}[N(t)] + \mu^2 \mathbf{E}[N(t)^2].
\end{aligned}$$

Since $(\mathbf{E}[X(t)])^2 = (\mathbf{E}[N(t)]\mathbf{E}[Z])^2$, we obtain

$$\begin{aligned}
\mathbf{Var}[X(t)] &= \sigma^2 \mathbf{E}[N(t)] + \mu^2 (\mathbf{E}[N(t)^2] - \mathbf{E}[N(t)]^2) \\
&= \sigma^2 \mathbf{E}[N(t)] + \mu^2 \mathbf{Var}[N(t)] \\
&= \lambda t(\sigma^2 + \mu^2).
\end{aligned}$$

□

2.5 Exponential family of distributions

The exponential family is a class of distributions sharing the same density form. It includes the normal, Poisson, inverse Gaussian, binomial, exponential and other well-known distributions, see, e.g., Anderson et al. (2007, p. 13). The GLMs assume that the response variables share a distribution that belongs to the exponential family. Hence, the exponential family of distributions is an essential part of the GLMs.

Definition 2.18. *Let Y be a random variable, which may be discrete or continuous, whose probability distribution depends on a parameter θ. The distribution belongs to the exponential family of distributions if its density function can be written in the form*

$$f(y;\theta) = exp\Big\{a(y)b(\theta) + c(\theta) + d(y)\Big\}, \quad y \in \mathbb{R} \tag{2.3}$$

where a,b,c and d are known functions.

If $a(y) = y$, the distribution is said to be in the *canonical form* and $b(\theta)$ is called the *natural parameter* of the distribution. If there are other parameters in addition to θ they are regarded as *nuisance parameters* forming parts of the functions a, b, c and d.

Example 2.19. *We show that the Poisson distribution belongs to the exponential family. For this consider*

$$\begin{aligned} f(y;\theta) &= \frac{e^{-\theta}\theta^y}{y!}, \quad y = 0, 1, 2, \dots \\ &= e^{(ln(\theta^y))}e^{-\theta}e^{-ln(y!)} \\ &= exp\big\{yln(\theta) - \theta - ln(y!)\big\}. \end{aligned}$$

The Poisson distribution, is hence in the exponential family. The functions a, b, c and d in Definition (2.18) are in this case the following:

$$\begin{cases} a(y) = y \\ b(\theta) = ln\theta \\ c(\theta) = -\theta \\ d(y) = -ln(y!), \quad y = 0, 1, \dots \end{cases}$$

In particular, the Poisson distribution is in the canonical form.

Example 2.20. *For the normal distribution $N(\mu, \sigma^2)$, the parameter σ^2 is regarded as a nuisance parameter. The probability density function can, consequently, be written as:*

$$\begin{aligned} f(y;\mu) &= \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} exp\bigg\{ -\frac{1}{2\sigma^2}(y-\mu)^2 \bigg\} \\ &= exp\bigg\{ ln(2\pi\sigma^2)^{\frac{1}{2}} \bigg\} exp\bigg\{ \frac{1}{2\sigma^2}(-y^2 + 2y\mu - \mu^2) \bigg\} \\ &= exp\bigg\{ y\frac{\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2} - \frac{1}{2}ln(2\pi\sigma^2) - \frac{y^2}{2\sigma^2} \bigg\}. \end{aligned}$$

The normal distribution is therefore also in the exponential family, and in the canonical form. The functions a, b, c and d are the following:

$$\begin{cases} a(y) = y \\ b(\mu) = \frac{\mu}{\sigma^2} \\ c(\mu) = -\frac{\mu^2}{2\sigma^2} - \frac{1}{2} ln(2\pi\sigma^2) \\ d(y) = -\frac{y^2}{2\sigma^2}. \end{cases}$$

2.5.1 Properties of distributions in the exponential family

In this section we will derive expressions for the expected value and variance of $a(Y)$, where the distribution of the random variable Y, is assumed to belong to the exponential family. Results that apply for any probability density function will be used to find the expressions. Let $f(y, \theta)$ be a density of a distribution function belonging to the exponential family. Since

$$\int_{-\infty}^{\infty} f(y;\theta)dy = 1, \tag{2.4}$$

we obtain by differentiating both sides of (2.4) with respect to θ, and by reversing the order of the integration and differentiation,

$$\int \frac{\partial}{\partial\theta} f(y;\theta)dy = 0. \tag{2.5}$$

By differentiating equation (2.5) again

$$\int \frac{\partial^2}{\partial\theta^2} f(y;\theta)dy = 0. \tag{2.6}$$

Since f is in the exponential family, we have by differentiating (2.3)

$$\frac{\partial f(y;\theta)}{\partial\theta} = \Big(a(y)b'(\theta) + c'(\theta)\Big) f(y;\theta). \tag{2.7}$$

Integrating (2.7) and implementing the result from equation (2.5) yield

$$\begin{aligned} &\int \Big(a(y)b'(\theta) + c'(\theta)\Big) f(y;\theta)dy = \int \frac{\partial}{\partial\theta} f(y;\theta)dy = 0 \\ &\iff \int a(y)b'(\theta)f(y;\theta)dy + \int c'(\theta)f(y;\theta)dy = 0. \end{aligned} \tag{2.8}$$

Using Theorem 2.10(ii) and equation (2.4), we obtain

$$\int a(y)b'(\theta)f(y;\theta)dy = b'(\theta)\mathbf{E}[a(Y)],$$
$$\int c'(\theta)f(y;\theta)dy = c'(\theta)\int f(y;\theta)dy = c'(\theta).$$

Rearranging the terms in equation (2.8) yield the result:

$$\mathbf{E}[a(Y)] = -\frac{c'(\theta)}{b'(\theta)}. \tag{2.9}$$

A similar procedure can be used to obtain the variance of $a(Y)$. For this, notice that

$$\frac{\partial^2}{\partial\theta^2}f(y;\theta) = \Big(a(y)b''(\theta)+c''(\theta)\Big)f(y;\theta)+\Big(a(y)b'(\theta)+c'(\theta)\Big)^2 f(y;\theta). \tag{2.10}$$

By rewriting the second term on the right hand side of (2.10), and using (2.9), we get

$$\begin{aligned}
&\Big(a(y)b'(\theta)+c'(\theta)\Big)^2 f(y;\theta)\\
&\qquad = b'(\theta)^2\left(a(y)^2 + 2a(y)\frac{c'(\theta)}{b'(\theta)} + \left(\frac{c'(\theta)}{b'(\theta)}\right)^2\right)f(y;\theta)\\
&\qquad = b'(\theta)^2\Big(a(y)^2 - 2a(y)\mathbf{E}[a(Y)] + \mathbf{E}[a(Y)]^2\Big)f(y;\theta)\\
&\qquad = b'(\theta)^2\Big(a(y)-\mathbf{E}[a(Y)]\Big)^2 f(y;\theta).
\end{aligned}$$

Thus, by integrating (2.10), and by using (2.6), the following is obtained:

$$\begin{aligned}
\int \frac{\partial^2}{\partial\theta^2}f(y;\theta)dy &= \int a(y)b''(\theta)f(y;\theta)dy + \int c''(\theta)f(y;\theta)dy\\
&\quad + \int b'(\theta)^2\Big(a(y)-\mathbf{E}[a(Y)]\Big)^2 f(y;\theta)dy\\
&= b''(\theta)\mathbf{E}[a(Y)] + c''(\theta) + b'(\theta)^2\mathbf{Var}[a(Y)] = 0,
\end{aligned}$$

where it is applied that

$$\int \Big(a(y) - \mathbf{E}[a(Y)]\Big)^2 f(y;\theta)dy = \mathbf{Var}[a(Y)].$$

Rearranging the terms, and using (2.9), the expression of the variance is obtained as:

$$\begin{aligned}\mathbf{Var}[a(Y)] &= \frac{-b''(\theta)\mathbf{E}[a(Y)] - c''(\theta)}{b'(\theta)^2} \\ &= \frac{b''(\theta)c'(\theta)}{b'(\theta)^3} - \frac{c''(\theta)}{b'(\theta)^2} \\ &= \frac{b''(\theta)c'(\theta) - b'(\theta)c''(\theta)}{b'(\theta)^3}.\end{aligned}$$

Example 2.21. *The results are used to verify the expectation and variance for the Poisson and normal distribution.*

For $Y \sim Poisson(\theta)$, we include the results from example 2.19.

$$\begin{aligned}\mathbf{E}[Y] = \mathbf{E}[a(Y)] &= -\frac{c'(\theta)}{b'(\theta)} = -\frac{-1}{1/\theta} = \theta. \\ \mathbf{Var}[Y] = \mathbf{Var}[a(Y)] &= \frac{b''(\theta)c'(\theta) - c''(\theta)b'(\theta)}{b'(\theta)^3} = \frac{-\frac{-1}{\theta^2}}{\frac{1}{\theta^3}} = \theta.\end{aligned}$$

Consequently, for the normal distribution $N(\mu, \sigma^2)$, we use the results from example 2.20.

$$\begin{aligned}\mathbf{E}[Y] = \mathbf{E}[a(Y)] &= -\frac{c'(\mu)}{b'(\mu)} = -\frac{-\frac{\mu}{\sigma^2}}{\frac{1}{\sigma^2}} = \mu. \\ \mathbf{Var}[Y] = \mathbf{Var}[a(Y)] &= \frac{b''(\mu)c'(\mu) - c''(\mu)b'(\mu)}{b'(\mu)^3} = \frac{0 - \frac{-1}{\sigma^2}\frac{1}{\sigma^2}}{\left(\frac{1}{\sigma^2}\right)^3} = \sigma^2.\end{aligned}$$

Chapter 3

Generalized linear models

3.1 Background of generalized linear models

Generalized linear models (GLMs), according to Anderson et al. (2007, p. 4), are today widely regarded as a standard statistical technique for pricing insurances in the European Union market. The primary use of GLMs in the insurance business is in insurance pricing and underwriting, although there has been an increased use for marketing analysis. In insurance pricing, the GLMs are used for estimating the risk premium and for studying the insurance data.

In the past, actuaries have entrusted one-way analyses for pricing of insurances. A one-way analysis may summarize the insurance frequency or loss-ratio for claims, but it does not take into consideration the affect of multiple variables. By the loss ratio is meant the ratio of incurred losses to earned premium. These analyses can be distorted by correlations between the rating factors. For example, a one-way analysis of the age of a car will probably show high claim frequency for old cars, but this may result from the fact that older cars are generally driven by more high-risk younger drivers.

One-way analysis also does not consider how interactions between the factors affect the claims. These interactions exist because of correlation between the variables. The affect of one variable varies depending on the value of another variable. Multivariate methods, such as GLMs, adjust for the correlations between the variables and allow investigation into the interaction affects between the variables.

3.2 Notations

The terms *response*, *outcome* and *dependent variable* are used for measurements that are free to vary with respect to other variables called *explanatory variables*, *predictor variables* or *independent variables*. In this thesis, the terms response variable and predictor variable are preferred.

Let n denote the number of the observations in a sample. Every observation consists of values on p variables. The notation x_{ij} will therefore represent the value of the jth variable for the ith observation, where $i = 1, 2 \ldots n$ and $j = 1, 2 \ldots p$. We let $\mathbf{X}$ denote an $n \times p$ matrix whose (i, j)th element is x_{ij}. $\mathbf{X}$ is called the *design matrix* and has the form

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{np} \end{pmatrix}.$$

The rows of the design matrix are written as $x_1, x_2, \ldots x_n$. Here each $x_i, i = 1, \ldots, n$ is a vector of length p. The columns of $\mathbf{X}$ are written as $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_p}$. Each is a vector of length n. To summarize:

$$x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix}, \quad \mathbf{x_j} = \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{nj} \end{pmatrix}.$$

Using this notation, the matrix $\mathbf{X}$ can be written as

$$\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_p}).$$

For the response variables, we use $y_i, i = 1, \ldots, n$ to denote the ith observation. The observations $y_1, y_2, \ldots, y_n$ are regarded as realizations of the random variables $Y_1, Y_2, \ldots, Y_n$. We let Y represent a column vector with components corresponding to the random variables $Y_1, Y_2, \ldots, Y_n$ and let y represent a column vector with components corresponding to the observed values $y_1, y_2, \ldots, y_n$. Thus,

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \quad , \quad Y = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix}.$$

In this thesis, the symbol $\hat{}$ is used for estimates. The column vector $\boldsymbol{\beta}$ denotes a vector of parameters, whose components $\beta_0, \beta_1, \ldots \beta_p$ are usually to be determined by different methods. β_0 is called the *intercept* and $\beta_1, \beta_2, \ldots, \beta_p$ are called the *slope* of the model. Hence:

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_p \end{pmatrix}$$

Estimating function f

Suppose that a set of values for a quantitative response variable $Y = (Y_1, \ldots, Y_n)^T$ and p different predictors variables $X_1, X_2, \ldots, X_p$ are observed. The models discussed in this thesis assume that there is some relationship between $Y = (Y_1, \ldots, Y_n)^T$ and $X = (X_1, X_2, \ldots, X_p)$. The relationship is expressed according to James et.al (2013, p. 16) as

$$Y = f(X) + \varepsilon,$$

where $f : R^p \mapsto R$, and ε is an error term with mean 0 and variance $\sigma^2 > 0$ assumed to be independent of X.

The term ε, whose mean is zero and variance $\sigma^2 > 0$, is an *error term* which is independent of X. In this thesis, different linear and non-linear methods for estimating the unknown function f are introduced. These types of methods are either parametric or non-parametric. The non-parametric methods are discussed in chapter 4 and in this chapter the focus is on the parametric methods. The parametric methods first make an assumption of the form of f, and once a model has been selected, the parameters $\beta_0, \beta_1, \ldots, \beta_p$ are estimated.

A common assumption is that f is linear:

$$f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p. \tag{3.1}$$

For the parameter estimation in the linear model, the aim is to find values for the parameters $\beta_0, \beta_1, \ldots, \beta_p$ which best explain the observed data.

3.3 The structure of a generalized linear model

A generalized linear model has three components:

(i) Response variables $Y_1, \ldots, Y_n$ which are assumed to belong to the same distribution from the exponential family, but could have, e.g., different means.

(ii) A set of explanatory variables and parameters $\boldsymbol{\beta}$

$$\mathbf{X} = \begin{pmatrix} x_{11} & \ldots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \ldots & x_{np} \end{pmatrix}, \ \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_p \end{pmatrix}.$$

(iii) A differentiable and monotonic *link function* g such that

$$g(\mu_i) = x_i^T \boldsymbol{\beta}, \quad \text{where} \quad \mu_i = \mathbf{E}[Y_i].$$

Therefore the GLM expresses the relationship between the expected value of the response variables and the predictor variables as:

$$\mathbf{E}[Y_i] = g^{-1}\Big(x_i^T \beta\Big).$$

Model formulation

The models described involve response variables $Y = (Y_1, \ldots, Y_n)^T$ and usually several predictor variables. The model formulation consists of two components:

1) Probability distribution of the response variable Y. For the GLMs, the probability distributions all belong to the exponential family of distributions. The exponential family of distributions includes e.g. normal, binomial, Poisson and other common distributions. The exponential family of distributions was introduced in section 2.5.

2) Equation linking the expected value of Y with a linear combination of the predictor variables. The equation is of the general form

$$g[E(Y)] = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p.$$

The right hand side of equation 3.2 is called the *linear component*.

The link function g must satisfy the condition that it is differentiable and monotonic. Various link functions are commonly used, depending on the assumed distribution of the response variable Y. Some typical choices for a link function include:

Notation	$g(x)$	$g(x)^{-1}$
Identity link	x	x
Log link	$\ln(x)$	e^x
Logit link	$\ln(x/(1-x))$	$e^x/(1+e^x)$
Reciprocal link	$1/x$	$1/x$

According to Dobson (2002, p. 35), equation (3.2) can be written in matrix notation as

$$g(\mathbf{E}[Y]) = \mathbf{X}\boldsymbol{\beta},$$

where $Y = (Y_1, \ldots, Y_n)^T$ is a vector of responses and

$$g[E(Y)] = \begin{pmatrix} g[E(Y_1)] \\ \vdots \\ g[E(Y_n)] \end{pmatrix}$$

denotes a vector of functions for the terms $\mathbf{E}[Y_i], i = 1, \ldots, n$.

3.4 Examples with various GLMs

Suppose a private car insurance has two categorical rating variables, territory (urban or rural) and gender (male or female). Suppose we have access to the following observations for the average claim severities.

Gender	**Urban**	**Rural**
Male	800	500
Female	400	200

The vector of observations, the design matrix, and the vector of parameters are as follows

$$\mathbf{y} = \begin{pmatrix} \text{Male Urban} \\ \text{Male Rural} \\ \text{Female Urban} \\ \text{Female Rural} \end{pmatrix} = \begin{pmatrix} 800 \\ 500 \\ 400 \\ 200 \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad \text{and } \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix}.$$

Note that a different design matrix was chosen here than in Anderson et al. (2007, p. 10). Further ahead in the thesis, outputs from statistical software using GLMs are analyzed. The statistical software shows a specific value of the intercept β_0. To be able to obtain the same results as in statistical software, the first column of the design row are only constants. The observations of the second column is 1, if the observation is represented by a male. Correspondingly, the observation of the third column is 1, if the observation is represented by a customer in an urban area.

3.4.1 Solving with the method of least squares

The least squares estimation is thoroughly introduced in section 3.5.1. The four observations above can be expressed as the system of equations:

$$\begin{aligned} y_1 &= 800 = \beta_0 + \beta_1 + \beta_2 + \varepsilon_1 \\ y_2 &= 500 = \beta_0 + \beta_1 + \varepsilon_2 \\ y_3 &= 400 = \beta_0 + \beta_2 + \varepsilon_3 \\ y_4 &= 200 = \beta_0 + \varepsilon_4. \end{aligned}$$

Consequently, the residual sum of squares are obtained from

$$\begin{aligned} RSS =& \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \varepsilon_4^2 \\ =& (800 - \beta_0 - \beta_1 - \beta_2)^2 + (500 - \beta_0 - \beta_1)^2 \\ &+ (400 - \beta_0 - \beta_2)^2 - (200 - \beta_0)^2. \end{aligned}$$

The parameters β_0, β_1 and β_2 are determined by taking derivatives with respect to each parameter:

$$\frac{\partial RSS}{\partial \beta_0} = 0 \Rightarrow 2\beta_0 + \beta_1 + \beta_2 = 950$$
$$\frac{\partial RSS}{\partial \beta_1} = 0 \Rightarrow 2\beta_0 + 2\beta_1 + \beta_2 = 1300$$
$$\frac{\partial RSS}{\partial \beta_2} = 0 \Rightarrow 2\beta_0 + \beta_1 + 2\beta_2 = 1200.$$

The solution to the equation system is given by the values:

$$\begin{cases} \beta_0 = 175 \\ \beta_1 = 350 \\ \beta_2 = 250 \end{cases}$$

Gender	Urban	Rural
Male	$\beta_0+\beta_1 + \beta_2$	$\beta_0+\beta_1$
Female	$\beta_0 + \beta_2$	β_0

Gender	Urban	Rural
Male	775	525
Female	425	175

3.4.2 Normal error structure with an identity link function

The classical linear model assumes a normal error structure, and an identity link function $g(x) = x$. The predicted values take the form

$$\mathbf{E}[\mathbf{y}] = g^{-1}(\mathbf{X}\boldsymbol{\beta}) = \begin{pmatrix} g^{-1}(\beta_0 + \beta_1 + \beta_2) \\ g^{-1}(\beta_0 + \beta_1) \\ g^{-1}(\beta_0 + \beta_2) \\ g^{-1}(\beta_0) \end{pmatrix} = \begin{pmatrix} \beta_0 + \beta_1 + \beta_2 \\ \beta_0 + \beta_1 \\ \beta_0 + \beta_2 \\ \beta_0 \end{pmatrix}$$

The likelihood function is presented thoroughly in section 3.5.1. The likelihood function of the normal distribution is

$$L(\mu, \sigma^2; y_1, \ldots, y_n) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left\{ -\sum_{i=1}^{n} \frac{(y_i - \mu_i)^2}{2\sigma^2} \right\},$$

where $\mu = (\mu_1, \ldots, \mu_n)$. The corresponding log-likelihood function, is

$$l(\mu, \sigma^2; y_1, \ldots, y_n) = -\frac{n}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mu_i)^2.$$

Due to the identity link function, $\mu_i = \sum_{j=1}^{p} x_{ij}\beta_{j-1}$. Note that p has the values $p = 1, 2, 3$. Therefore, the log-likelihood function becomes

$$l(\mu, \sigma^2; y_1, \ldots, y_n) = -\frac{n}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_{j-1})^2.$$

By inserting the values from the vector of observations and design matrix:

$$\begin{aligned} l(\mu, \sigma^2; y_1, \ldots, y_n) = &-\frac{n}{2}\ln(2\pi\sigma^2) \\ &- \frac{(800 - \beta_0 - \beta_1 - \beta_2)^2}{2\sigma^2} - \frac{(500 - \beta_0 - \beta_1)^2}{2\sigma^2} \\ &- \frac{(400 - \beta_0 - \beta_2)^2}{2\sigma^2} - \frac{(200 - \beta_0)^2}{2\sigma^2}. \end{aligned}$$

To maximize the log-likelihood function, we take the derivative with respect to each parameter. The resulting system of equations is as follows:

$$\begin{aligned} \frac{\partial l}{\partial \beta_0} &= 0 \Rightarrow 2\beta_0 + \beta_1 + \beta_2 = 950 \\ \frac{\partial l}{\partial \beta_1} &= 0 \Rightarrow 2\beta_0 + 2\beta_1 + \beta_2 = 1300 \\ \frac{\partial l}{\partial \beta_2} &= 0 \Rightarrow 2\beta_0 + \beta_1 + 2\beta_2 = 1200. \end{aligned}$$

$$\begin{cases} \beta_0 = 175 \\ \beta_1 = 350 \\ \beta_2 = 250 \end{cases}$$

Gender	**Urban**	**Rural**
Male	$\beta_0+\beta_1 + \beta_2$	$\beta_0+\beta_1$
Female	$\beta_0 + \beta_2$	β_0

Gender	**Urban**	**Rural**
Male	775	525
Female	425	175

Consequently, it can be observed that these equations are identical to those derived from example 3.4.1. To obtain a goodness of fit, the null- and residual deviance is calculated. These will be more thoroughly explained in section 3.5.2. The null deviance and the residual deviance is obtained from the following, where the mean of the observations is $\mu = 475$:

Observations	**Null deviance**	**Estimate**	**Residual deviance**
800	$(800-475)^2 = 105625$	775	$(800-775)^2 = 625$
500	$(500-475)^2 = 625$	525	$(500-525)^2 = 625$
400	$(400-475)^2 = 5625$	425	$(400-425)^2 = 625$
200	$(200-475)^2 = 75625$	175	$(200-175)^2 = 625$

The total null deviance and residual deviance is obtained from the sum of the respective deviance. Therefore the null deviance for this model is 187500 and the residual deviance is 2500. This means that the full model, including all the parameters, explaines the data better and is hence preferred. However, the predictor variables are not significant in this model, according to the R output in Appendix A.2. In Appendix A.2, the same example is demonstrated by using the software R.

3.4.3 Poisson error structure with a logarithm link function

By modelling with the Poisson model, using a logarithm link function, the predicted values are given by

$$\mathbf{E}[\mathbf{y}] = g^{-1}(\mathbf{X}\boldsymbol{\beta}) = \begin{pmatrix} g^{-1}(\beta_0+\beta_1+\beta_2) \\ g^{-1}(\beta_0+\beta_1) \\ g^{-1}(\beta_0+\beta_2) \\ g^{-1}(\beta_0) \end{pmatrix} = \begin{pmatrix} e^{\beta_0+\beta_1+\beta_2} \\ e^{\beta_0+\beta_1} \\ e^{\beta_0+\beta_2} \\ e^{\beta_0} \end{pmatrix}.$$

The likelihood function of the Poisson distribution, whose parameter space is $\theta = \{\mu | \mu \in R, \mu > 0\}$, is the following:

$$L(\mu; y_1, \ldots, y_n) = \prod_{i=1}^{n} f_{\mu_i}(y_i)$$
$$= e^{-\mu_1} \cdot \frac{\mu_1^{y_1}}{y_1!} \cdot \ldots \cdot e^{-\mu_1} \cdot \frac{\mu_1^{y_n}}{y_n!}.$$

The corresponding log-likelihood function is

$$l(\mu; y_1, \ldots, y_n) = \sum_{i=1}^{n} \ln f_{\mu_i}(y_i)$$
$$= \sum_{i=1}^{n} \big(- \mu_i + y_i \ln(\mu_i) - \ln(y_i!)\big).$$

Due to the logarithm link function, $\mu_i = exp\{\sum_{j=1}^{p} x_{ij}\beta_{j-1}\}$. The log-likelihood function becomes:

$$l(\mu; y_1, \ldots, y_n) = \sum_{i=1}^{n} \big(- exp\{\sum_{j=1}^{p} x_{ij}\beta_{j-1}\} + y_i \sum_{j=1}^{p} x_{ij}\beta_{j-1} - \ln(y_i!)\big). \quad (3.2)$$

For the observations in this example, (3.2) becomes

$$\begin{aligned} l(\mu; y_1, \ldots, y_n) = {} & y_1(\beta_0 + \beta_1 + \beta_2) - e^{\beta_0+\beta_1+\beta_2} - \ln(y_1!) \\ & + y_2(\beta_0 + \beta_1) - e^{\beta_0+\beta_1} - \ln(y_2!) \\ & + y_3(\beta_0 + \beta_2) - e^{\beta_0+\beta_2} - \ln(y_3!) \\ & + y_4(\beta_0) - e^{\beta_0} - \ln(y_4!). \end{aligned}$$

By taking the derivative with respect to each parameter, the log-likelihood function is maximized. The resulting system of equations becomes as follows:

$$\frac{\partial l}{\partial \beta_0} = 0 \Rightarrow e^{\beta_0+\beta_1+\beta_2} + e^{\beta_0+\beta_1} + e^{\beta_0+\beta_2} + e^{\beta_0} = 1900$$
$$\frac{\partial l}{\partial \beta_1} = 0 \Rightarrow e^{\beta_0+\beta_1+\beta_2} + e^{\beta_0+\beta_1} = 1300$$
$$\frac{\partial l}{\partial \beta_2} = 0 \Rightarrow e^{\beta_0+\beta_1+\beta_2} + e^{\beta_0+\beta_2} = 1200.$$

By using the Newton-Raphson method, the solution to the system of equations is:

$$\begin{cases} \beta_0 = 5.39840\ldots \\ \beta_1 = 0.77319\ldots \\ \beta_2 = 0.53900\ldots \end{cases}$$

The predicted values of the observations are, thus, as follows:

Gender	**Urban**	**Rural**
Male	$e^{\beta_0+\beta_1+\beta_2}$	$e^{\beta_0+\beta_1}$
Female	$e^{\beta_0+\beta_2}$	e^{β_0}

Gender	**Urban**	**Rural**
Male	821,05	478,95
Female	378,95	221,05

The mean of the observations is $\mu = 475$. For the Poisson distribution, the null deviance and residual deviance is calculated by the following:

Observations	**Null deviance**
800	$2(800(\ln(800/475) - (800 - 475)) = 184,074$
500	$2(500(\ln(500/475) - (500 - 475)) = 1,294$
400	$2(400(\ln(400/475) - (400 - 475)) = 12,52$
200	$2(200(\ln(200/475) - (200 - 475)) = 204,002$

Estimate	**Residual deviance**
821,05	$2(800(\ln(800/821,05) - (800 - 821,05)) = 0,544$
478,95	$2(500(\ln(500/478,95) - (500 - 478,95)) = 0,912$
378,95	$2(400(\ln(400/378,95) - (400 - 378,95)) = 1,148$
221,05	$2(200(\ln(200/221,05) - (200 - 221,05)) = 2,072$

The null deviance for this model is 401.889, and the residual deviance is $4,677$. The full model again explains the data better, however, according to the R output for this model, the predictor variables are significant. Note also that the AIC of the R output 42.219 is less than the AIC of the Normal error structure, which was 45.103. This indicates that the model with a Poisson distribution is as a whole a better fit of the data. The R output for this model is demonstrated in Appendix A.3.

3.5 Model fitting of a GLM

The purpose of the GLMs is to express the relationship between the response variables $Y_1, \ldots, Y_n$ and a number of predictor variables. The model-fitting process involves three steps, which are to be introduced thoroughly in the next sections:

1) Model formulation - The GLM is specified with an assumed probability distribution for the response variable and an equation linking the response and the predictor variables.

2) Estimation of the parameters of the model

3) Suitability of the model - Checking how well the model fits or summarizes the data. Calculating confidence intervals and testing hypotheses about the parameters

3.5.1 Parameter estimation

The most commonly used methods for estimation are the *method of least squares* and *maximum likelihood estimation*. When solving for large data sets the parameters are estimated by numerical techniques.

Least squares estimation

At the beginning of the nineteenth century, Carl Friedrich Gauss and Adrien-Marie Legendre published papers on the method of least squares, which implemented the earliest form of what is now known as *linear regression*. Linear regression is a method for predicting quantitative values for the response variable Y. If we have only one predictor variable, the approach is called *simple linear regression*. Consequently, the approach is called *multiple linear regression* if we have access to more than one predictor variable.

In the case of linear regression, the parameters $\beta_j, j = 0, \ldots, p$, are unknown and must be estimated. The method assumes an approximately linear relationship between the response variable Y and the predictor variables X. Given estimates $\hat{\beta}_0, \ldots, \hat{\beta}_p$, predictions can be made using the formula

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots + \hat{\beta}_p x_p.$$

Let $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \ldots + \hat{\beta}_p x_{ip}$ be the predicted value for y_i based on the ith value of X. The *residual*, denoted by $\varepsilon_i = y_i - \hat{y}_i, i = 1, \ldots, n$, is then the difference between the ith observed response value and the ith predicted response value by the model. We define the *residual sum of squares* (RSS) by:

$$\begin{aligned} RSS &= \varepsilon_1^2 + \ldots + \varepsilon_n^2 \\ &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \ldots - \hat{\beta}_p x_{ip})^2. \end{aligned}$$

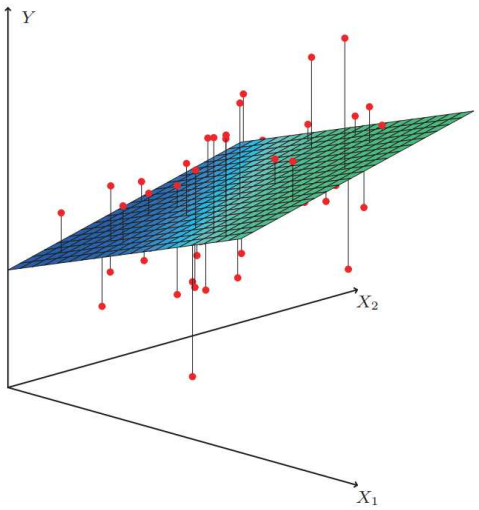


Figure 3.1: In a three-dimensional setting with two predictor variables, the regression line becomes a plane. The plane is chosen to minimize the RSS, i.e. the sum of the squared vertical distances between each observation and the plane. The picture is obtained from James et al. (2013, p. 73).

The method of least squares consists of finding the parameters $\beta_j, j = 0, \ldots, p$, that minimizes the RSS. The parameters $\beta_j, j = 0, \ldots, p$, are usually obtained by differentiating the RSS with respect to each element $\beta_j, j = 0, \ldots, p$, of the column vector $\boldsymbol{\beta}$ and finding $\boldsymbol{\beta}$ by the simultaneous equations

$$\frac{\partial RSS}{\partial \beta_j} = 0, \quad j = 0, 1, \ldots, p.$$

It is then necessary to check that the solutions correspond to the minimum of the RSS. The method of least squares is demonstrated in practice in example 3.4.1 later in this chapter.

Maximum likelihood estimation

The principle of maximum likelihood is relatively straightforward. Let $Y_1, \ldots, Y_n$ denote a random sample from a distribution whose probability density function is $f_\theta(y)$. The maximum likelihood method attempts to find the parameters which maximize the function

$$L(\theta; Y_1, \ldots, Y_n) = \prod_{i=1}^{n} f_\theta(Y_i).$$

Example 3.1. *Maximum likelihood estimation of the parameters of the normal distribution.*

Let $Y_1, \ldots, Y_n$ be a random sample from $N(\mu, \sigma^2)$ distribution. The parameter space is $\theta = \{(\mu, \sigma^2) | \mu \in R, \sigma^2 > 0\}$. We will now find the maximum points of the function

$$\begin{aligned}(\mu, \sigma^2) \mapsto L(\mu, \sigma^2; Y_1, \ldots, Y_n) &= \prod_{i=1}^{n} \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) \exp\left\{ -\frac{(Y_i - \mu)^2}{2\sigma^2} \right\} \\ &= \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right)^n \exp\left\{ -\sum_{i=1}^{n} \frac{(Y_i - \mu)^2}{2\sigma^2} \right\}.\end{aligned}$$

Maximizing the likelihood function is equivalent to maximizing the log-likelihood function:

$$\begin{aligned}l(\mu, \sigma^2; Y_1, \ldots, Y_n) : &= \ln L(\mu, \sigma^2; Y_1, \ldots, Y_n) \\ &= \ln\left[(2\pi\sigma^2)^{-\frac{n}{2}} \right] + \ln\left[\exp\left\{ -\sum_{i=1}^{n} \frac{(Y_i - \mu)^2}{2\sigma^2} \right\} \right] \\ &= -\frac{n}{2} \ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (Y_i - \mu)^2.\end{aligned}$$

To maximize l we take the derivatives with respect to μ and σ^2 and set each of them to zero. The resulting system of equations is:

$$\begin{aligned}
\frac{\partial}{\partial \mu} l(\mu, \sigma^2; Y_1, \ldots, Y_n) &= \frac{1}{\sigma^2} \sum_{i=1}^{n} (Y_i - \mu) = 0 \\
&\Rightarrow \mu = \frac{1}{n} \sum_{i=1}^{n} Y_i =: \hat{Y}. \\
\frac{\partial}{\partial \sigma^2} l(\mu, \sigma^2; Y_1, \ldots, Y_n) &= -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4} \sum_{i=1}^{n} (Y_i - \mu)^2 = 0 \\
&\Rightarrow \sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \mu)^2 \\
&\Rightarrow \sigma^2 := \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y})^2.
\end{aligned}$$

These values are therefore possible maximum points of the function. By Hesse's matrix test, one can conclude that the values are in fact maximum points. For normal distribution, the parameter estimates are therefore the mean and the variance of the sample.

Solving for large data sets using numerical techniques

In section 3.4, simple examples using GLMs are demonstrated for understanding the mechanics involved in solving a GLM. In practice, there are thousands of observations in the data sets for the insurance industry. Therefore, it is not practical to find values of β which maximize the likelihood and, instead, iterative numerical techniques are used. The numerical techniques seek to optimize likelihood by seeking the values of $\boldsymbol{\beta}$ which set the first differentials of the log-likelihood to zero. This is done by using an iterative process, for example, *Newton-Raphson iteration* which uses the formula:

$$\beta_{n+1} = \beta_n - \boldsymbol{H}^{-1} s,$$

where β_n is the nth iterative estimate of the vector of the parameter estimates $\boldsymbol{\beta}$. The vector with the first derivatives of the log-likelihood is denoted by s and $\boldsymbol{H}$ is a $(p \times p)$ matrix containing the second derivatives of the

log-likelihood. This is a generalized form of the one-dimensional Newton-Raphson equation

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)},$$

which seeks to find a solution to $f'(x) = 0$.

3.5.2 Suitability of the model

The true relationship is generally not known for real data. How well the model fits the data is analyzed using *deviance* tests and standard errors. Using statistical software these values are obtained directly from the GLM fit, regardless of how many parameters there are in the model.

The measure of the goodness of fit can be formed in various ways, but we will primarly focus on the concept of deviance. The deviance, also called the *log likelihood (ratio) statistic* is

$$D = 2[l(\theta_{max}; y_1, \ldots, y_n) - l(\hat{\theta}, y_1, \ldots, y_n)],$$

according to Dobson (2002, p. 76). The deviance is hence different depending on the distribution. According to McCullagh and Nelder (1983 p. 34), for some common distributions in the exponential family, the deviance takes the following forms:

Distribution	**Deviance function**
Normal	$\sum(y_i - \mu_i)^2$
Poisson	$2\sum(y_i \ln(y_i/\mu_i) - (y_i - \mu_i))$
Gamma	$2\sum(-\ln(y_i/\mu_i) + (y_i - \mu_i)/\mu_i$
Inverse Gaussian	$\sum(y_i - \mu_i)^2/(\mu_i^2 y_i)$

Note that the summation is over $i = 1, \ldots, n$.

Example 3.2. *Deviance for a normal linear model. Consider the model*

$$\mathbf{E}[Y_i] = \mu_i = \boldsymbol{x}_i^T \beta; \quad Y_i \sim N(\mu_i, \sigma^2), \quad i = 1, \ldots n,$$

where the Y_i:s are independent. The log-likelihood function of the normal distribution was derived in section (3.4.2), and is the following:

$$l(\mu, \sigma^2; y_1, \ldots, y_n) = -\frac{n}{2} ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu_i)^2.$$

Assume that the model parameters has been obtained, and every parameter was used in the model. To obtain the maximum value of the log-likelihood function, we differentiate it with respect to each μ_i*. By solving the equations, we obtain* $\hat{\mu}_i = y_i$*. The maximum value of the log-likelihood, of the saturated model, is therefore:*

$$l(\mu_{max}, \sigma^2; y_1, \ldots, y_n) = -\frac{n}{2} ln(2\pi\sigma^2).$$

The log-likelihood function using the estimated $\hat{\mu}_i$*:s is*

$$l(\hat{\mu}_i, \sigma^2; y_1, \ldots, y_n) = -\frac{n}{2} ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \hat{\mu}_i)^2.$$

The deviance of the saturated normal linear model is hence:

$$\begin{aligned} D &= 2[l(\mu_{max}, \sigma^2; y_1, \ldots, y_n) - l(\hat{\mu}, \sigma^2, y_1, \ldots, y_n)] \\ &= \frac{1}{\sigma^2} \sum_{i=1}^{n} (y_i - \hat{\mu}_i)^2. \end{aligned}$$

This is also called the residual deviance, i.e. we measure the deviance using all the parameters in the model. In the special case where there is only one parameter $\mathbf{E}[Y_i] = \mu$ *for all* i*, we obtain the null deviance. The estimate of* μ *becomes* $\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} y_i = \bar{y}$*, and therefore:*

$$D = \frac{1}{\sigma^2} \sum_{i=1}^{n} (y_i - \bar{y}).$$

In general, a higher number of deviance indicates a worse fit of the model. The programming language *R* reports the null deviance and the residual deviance. The null deviance shows how well the response variable is predicted by a model that only includes the intercept, i.e. the μ is the mean of the observations. For the residual deviance, μ is the estimate for each observation.

The software R also reports a value of the *Akaike Information Criteria* (AIC), which is based on the deviance. The definition of the AIC will not be included in this thesis, but a smaller value of the AIC indicates a better fit of the model. For the purpose of demonstration, an output from R using a Poisson distributed GLM with a logarithm link function is seen.

```
Deviance Residuals:
1         2         3         4
-0.7379   0.9550    1.0717   -1.4394

Coefficients:
             Estimate  Std.Error z value  Pr(>|z|)
(Intercept)  5.39840    0.05068  106.51   <2e-16 ***
variable1    0.77319    0.04935   15.67   <2e-16 ***
variable2    0.53900    0.04756   11.33   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Dispersion parameter for poisson family taken to be 1

Null deviance: 401.889  on 3  degrees of freedom
Residual deviance:   4.677  on 1  degrees of freedom
AIC: 42.219

Number of Fisher Scoring iterations: 3
```

In this model there are three parameters β_0, β_1 and β_2. The parameters β_1 and β_2 are the predictor variables in the data named *variable1* and *variable2*. R has estimated values $\beta_0 = 5,40..., \beta_1 = 0,77...$ and $\beta_2 = 0,54...$ for the different parameters, and calculated the standard error of the respective parameters. The software has also computed the z-value and the significance level according to each parameter. The residual deviance is smaller than the null deviance. This indicates that the model, included with all the parameters, results in a better fit for the data set, than including only one parameter. Note that the observations used for the model are the same as in example 3.4.3.

The Fisher Scoring iterations indicates how many iterations the software needed to perform the Newtons method to obtain the fit. The standard errors in the output can be used to compute confidence intervals. Confidence intervals are a range of values, defined so, that there is a specified probability

that the value of a true parameter lies within it. For example a 95% confidence interval is a range of values such that with 95% probability, the range will contain the true unknown value of the parameter.

Standard errors can also be used to perform *hypothesis tests* on the parameters. The most common hypothesis test involves testing the *null-hypothesis* of

> $H_0 : \beta_1 = 0$ there is no relationship between X_1 and $Y = \beta_0 + \beta_1 X_1 + \varepsilon$ against,
>
> $H_a : \beta_1 \neq 0$ there is some relationship,

since if $\beta_1 = 0$ the model reduces to $Y = \beta_0 + \varepsilon$ and X_1 is not associated to Y.

3.6 Risk premium modelling with GLM

The response variables for the frequency and severity, of the insurance claims, follow different distributions. Therefore, when modelling the risk premium, it can be useful to split the risk premium into frequency and severity. This provides a better understanding of the way in which different predictor variables affect the cost of claims. This results also in that identification and removal of random effects are more easily done. Conclusively, the underlying models need to be combined to generate the final loss model, fromwhere the risk premium is obtained.

Claim frequencies follow a Poisson process. Therefore the frequency models are usually fitted with an assumed Poisson error structure and a logarithmic link function. By assuming a log link function, the condition of the fitted frequencies being positive is guaranteed. For the insurance data, one observation may be related to one month and another may be related to one year. There is more information in the observation relating to the longer exposure period. Therefore in the case of claim frequencies the prior weights are typically set to be the exposure of each observation.

Severity models are usually fitted with an assumed gamma error structure and a logarithmic link function. The use of the log link functions again ensures the positivity of the fitted models. By the use of a gamma function, we have a constant coefficient of variance. Therefore the standard errors around the fitted values are proportional to the fitted values.

As a summary, when modelling the frequency and severity models, the response variables becomes

$$\text{Claim frequency} = \frac{\text{claim count}}{\text{exposure}}$$

$$\text{Claim severity} = \frac{\text{loss}}{\text{claim count}}$$

The different models are then combined to obtain the final model, whose response variable is the estimated risk premium.

$$\text{Risk premium} = \text{frequency} \times \text{severity} = \frac{\text{loss}}{\text{exposure}}$$

Chapter 4

Machine-Learning

4.1 Background of machine Learning

Artificial intelligence (AI) is a field of computer science that has been studied for decades, however, AI is still one of the most elusive subjects. AI is a broader concept of machines being able to carry out tasks, which normally requires human intelligence. It operates today in almost every way we use computers, and ranges from machines processing languages, to algorithms used for playing board games. AI has it′s roots from the 1950s when Alan Turing created the Turing Test, as a measure of machine intelligence. The task of the computer was to convince a human being into believing the computer was a human. Another important milestone for AI was in 1997, when the IBM's Deep Blue challenged, and defeated, the world chess champion.

Machine-learning is a subset field of AI, which is the result of two important breakthroughs. In the late 1950s rather than teaching computers how to carry out different tasks, the idea of the computers being able to teach themselves broke through. This was realized by Arthur Samuel in 1959, he thought that teaching computers to play games was very useful for developing tactics, appropriate to general problems. The second breakthrough was due to the emergence of the internet. This resulted in a huge increase in the amount of digital information being generated. Engineers now realized that rather than teaching computers, it would be more efficient to plug them to the internet to give them information.

Therefore, machine-learning is now a current application of AI, based around the idea that we give machines access to data, and let them learn

for themselves. Today, the machine-learning algorithms enable computers to communicate with humans. Sophisticated search machines, algorithms that predicts suitable products for customers, and cars that drive by themselves are present products of machine-learning. Machine-learning can be further divided into supervised and unsupervised learning, depending on the aim of the problem. Many machine-learning algorithms can also be used for regression or classification, and we will see that they may work too well in some cases.

4.2 Supervised and unsupervised learning

As was mentioned, machine-learning algorithms can be classified into two categories, supervised and unsupervised learning, depending on the goal of the algorithm. Supervised learning is used when the data set has access to a response variable, that has to be predicted. In unsupervised learning, the algorithm instead seeks to find hidden patterns in the data.

In supervised learning we have for each observation $i = 1, \ldots, n$, access to a response variable Y_i, for every predictor measurement X_i. If the aim of the method is prediction, the learning algorithm analyzes the available data, and produces a inferred function, with the aim of accurately predicting future observations. The available data is referred to as training data, while the future observations are referred to as test data. The success of the predictive algorithm is evaluated, by comparing the estimated predictions with the true values within the test data. If the aim is to have a better understanding of the relationship between the response and the predictors, the machine-learning task is called inference.

Unsupervised learning describes the situation where for every $Y_i, i = 1, \ldots, n$, we observe a vector of measurements $(x_{i1}, \ldots, x_{ip})$, but not the response variable Y_i itself. Unsupervised learning seeks to understand the relationship between the observations, and possibly find hidden structures within the data. It is not possible to fit a regression model, for example, in this setting, since there is no response variable to predict. Therefore, in unsupervised learning, there is no evaluation of the accuracy of the structure, that is output by the algorithm. This is one way to distinguish unsupervised learning from supervised learning.

4.3 Regression and classification problems

Supervised learning algorithms can further be divided into classification and regression algorithms. For a classification problem, the goal is to predict a discrete class label for the response variable, to a new unseen observation. For a classification problem, the set of possible values of the response variable, belongs to $k \in \mathbb{Z}$ different classes. In a regression problem, the response variable is a quantitative variable, i.e. a continuous variable.

The distinction between regression and classification algorithms is generally not so clear. Some algorithms may belong to both fields and solve both regression and classification problems. Whether the predictor variables are qualitative or quantitative, is less important. Both machine-learning algorithms discussed in this thesis are capable of being applied, regardless of the predictor type, given that any qualitative predictor is properly coded before the analysis.

4.4 Measuring quality of fit

There is no single machine-learning algorithm that works best on every possible data set. One algorithm may outperform other algorithms on a particular data set, but other algorithms may work better on a different data set. In order to assess the performance of a machine-learning algorithm, for a given data set, the extent to which the predictions are close to the true values of the response variable needs to be evaluated.

In the regression setting, according to James et al. (2013, p. 29), the most commonly used measure for prediction error is the mean squared error (MSE), which is given by

$$MSE = \frac{1}{n}\sum_{i=1}^{\infty}(y_i - \hat{f}(x_i))^2. \tag{4.1}$$

where $\hat{f}(x_i)$ is the prediction of the ith observation. The MSE in (4.1) is computed using the training data that was used to fit the model, which is referred to as the training MSE. However, we are generally not interested in whether $\hat{f}(x_i) \sim y_i$, for the different training observations $i = 1, \ldots, n$. Instead, we want to know if $\hat{f}(x_{n+1}) \sim y_{n+1}$, where (x_{n+1}, y_{n+1}) is a previously

unseen test observation not used to train the model. We want to find the model that gives the lowest test MSE as opposed to the lowest training MSE.

Therefore, if we have access to a large number of test observations, we can compute the mean squared prediction error of these test observations (x_i, y_i) where $i = n+1, n+2, \ldots$, and select the model for which the average of the test MSE is as small as possible. Unfortunately, in real-life prediction problems, the test sets are usually not available. In this case it might be a sensible approach to select the model that generates the lowest training error. However, there is no guarantee that the model, with the lowest training MSE, will also have the lowest test MSE.

Figure (4.1) illustrates this phenomenon. By modelling the training observations using three different models, with different flexibilities, the training MSE was calculated. Not surprisingly, the model with the highest flexibility resulted in the lowest training MSE. However, when the models estimated previously unseen test data, the model with the highest flexibility resulted in a high test MSE. The blue line is from a linear regression fit, which is relatively inflexible. The red and green curves are two machine-learning model fits with different level of flexibilities. As can be seen, the green curve follow the observations well, and it is the most flexible model. Therefore the green curve will result in a low training MSE.

On the bottom-graphs, the training and test MSEs are displayed as functions of the flexibility. The horizontal line on the right graph represents the irreducible error, $Var(\varepsilon)$, which will be presented in the next section. The training MSE declines monotonically as the flexibility increases. The true f is non-linear, hence, the linear regression fit is not flexible enough to estimate f well. The test MSE also declines as the flexibility increases, but at some point the test MSE start to increase, and form a U-shape. When a method has a small training MSE, but a large test MSE, the method is said to **overfit** the data. In this case, the machine-learning method is working too well to find the patterns in the training data.

When overfitting has occurred, the model may be picking up patterns that are caused by random chance, rather than following the true properties, of the unknown function f that it strives to model. The patterns found in the training data may not exist in the test data, therefore, the test MSE is large. Whether or not overfitting has occurred, we expect the training MSE to be smaller than the test MSE, since the aim of the models is to minimize the training MSE.

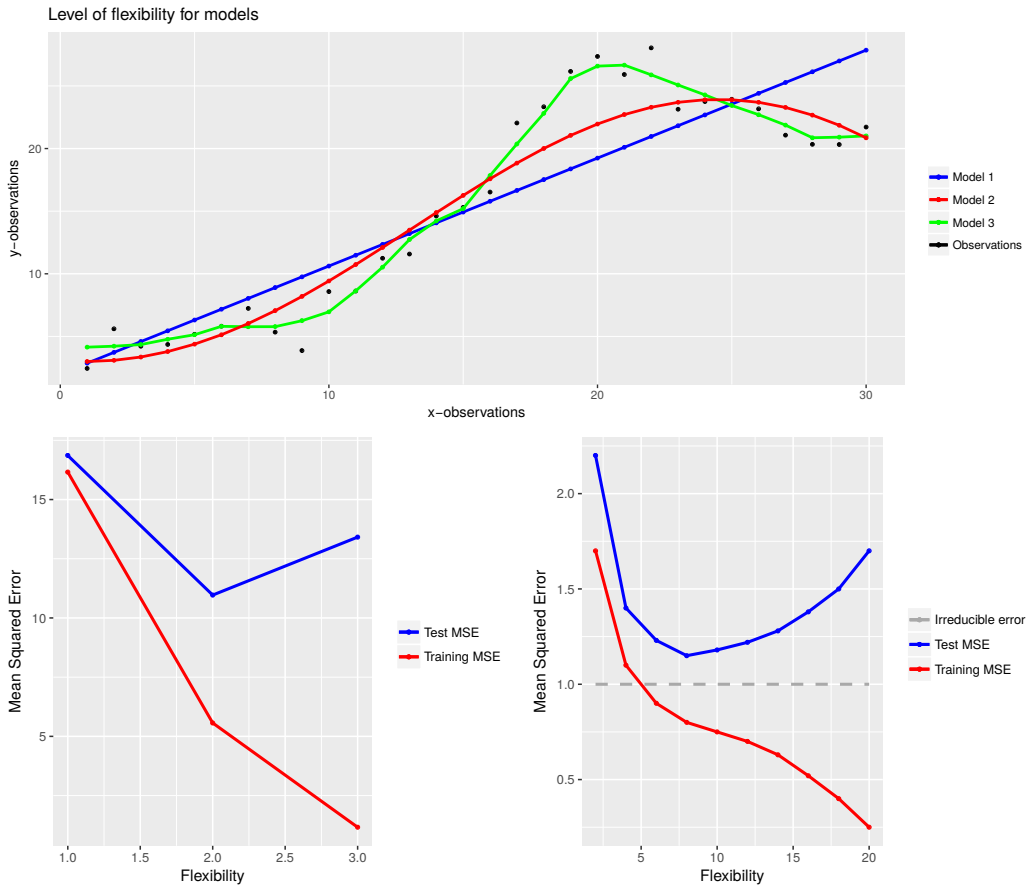


Figure 4.1: Top: Estimates of model fits are generated from a linear regression (blue) and two machine-learning methods, with different flexibilities (red and green curves). Bottom-left: Training MSE and test MSE of the respective models. Bottom-right: By using more models with different flexibilitites, the same graph can look like the following, according to James et al. (2013, p. 31). The source code to the graphs are left out of the thesis.

4.5 Bias-variance trade-off

The U-shape in the test MSE curve, is the result of two properties in the machine-learning methods. It is, according to James et al. (2013, p. 33), possible to show that the expected test MSE, for a given test value x_{n+1}, can

always be set up by the sum of three quantities:

$$\mathbf{E}[(y_{n+1} - \hat{f}(x_{n+1}))^2] = \mathrm{Var}[\hat{f}(x_{n+1})] + [\mathrm{Bias}(\hat{f}(x_{n+1}))]^2 + \mathrm{Var}(\epsilon).$$

In order to minimize the expected test error, we need no select a machine-learning model that achieves both low variance and low bias, which are both non-negative. Therefore, the expected test MSE can never lie below $Var(\varepsilon)$, which is the so-called irreducible error term. The irreducible error term rises from the fact that Y is also a function of ε, which by definition cannot be predicted using X.

Ideally, the estimate for f should not vary too much between different training sets. The term variance refers to the amount by which the estimate of f would change, if we estimated it using different sets. Bias refers to the systematic error in the estimation. For example, linear regression assumes a linear relationship between the predictor and response variables, linear regression will therefore result in high bias. Real-life problems rarely have a truly linear relationship. As a general rule, when the flexibility of the model increases, the variance increases and the bias decreases.

4.6 Resampling methods

Resampling methods consists of drawing repeated samples from the original data set. The methods yield samples from the data observations and refitts the model of interest on each sample, in order to acquire information about the fitted model. In this chapter two of the most commonly used resampling methods are introduced, these are called *cross-validation* and *bootstrap*. These methods are important tools in the model assessment, i.e. the process of evaluating the performance of a model. The theory is developed for the purpose of introducing a method called k-fold cross validation. The k-fold cross-validation is a significant method for this thesis to minimize the test error. In the end of this section the bootstrap will be introduced, which is used to provide a measure of the accuracy of a parameter estimate.

4.6.1 Cross-validation

Cross-validation (CV) is used to estimate the test error for a given statistical method. The method enables the user, therefore, to select the best model with the most suitable parameters. The validation set approach is a simple

alternative for this task. It involves splitting the observations into two parts, a training set and validation set. The model is then fitted on the training set, and the fitted model is used to predict the response values of the observations in the validation set. The resulting validation set error rate provides an estimate of the test error. However, the validation set can be highly variable, depending on precisely which observations are included in the respective set.

Therefore, one can use a method called Leave-one-out cross-validation (LOOCV). LOOCV involves splitting the observations into two parts, but instead a single observation (x_1, y_1) is used as the validation set, and the remaining observations $\{(x_2, y_2) \dots (x_n, y_n)\}$ form the training set. The machine-learning method is then fitted on the training set and a prediction $\hat{y_1}$ is made for the held out observation. This process is repeated by selecting the observation (x_2, y_2) as the validation set, and fitting the method on the new training set $\{(x_1, y_1), (x_3, y_3), \dots, (x_n, y_n)\}$.

By repeating this procedure n times, the respective test errors $MSE_i = (y_i - \hat{y_i})^2, i = 1 \dots n$ are obtained. The LOOCV estimate of the test MSE of the machine-learning method, is then the average of the n test error estimates:

$$\mathrm{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \mathrm{MSE}_i.$$

The disadvantage of the LOOCV is that it can be highly variable, since it is based on a single observation. LOOCV is also computationally expensive, since it has to repeat the procedure n times. Therefore, an alternative to the LOOCV is k-fold CV. This approach involves dividing the set of observations into k approximately equally sized groups. The same procedure is then repeated as in LOOCV, however, each fold at a time is held out. Each fold is in turn considered as the validation set, and the remaining $k - 1$ folds as the training set.

$$\mathrm{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \mathrm{MSE}_i.$$

The LOOCV is therefore a special case of the k-fold CV, where $k = n$.

Bias-variance trade-off for k-fold cross validation

Using k-fold CV, where $k < n$, is not only an advantage from a computational point of view, the estimate of the test error becomes more accurate. This has to do with the bias-variance trade-off. It was mentioned that the validation set approach can be highly variable, and depends on the choice of the validation set. However, LOOCV is less variable, since each training set contains $n - 1$ observations. Each training set contains therefore almost as many observations as in the full data set. Performing k-fold CV, will therefore lead to an intermediate level of bias and variance.

Nonetheless, the choice of k affects the variance of the test error. When performing LOOCV, the output of the test error rates are highly correlated with each other. This is due to the fact that the outputs of the fitted n models, are trained on an almost identical set of observations. When performing k-fold CV, where $k < n$, the overlap between the different training sets are smaller, which leads to less correlated outputs of the model. This reduces the variance of the test error. According to James et al. (2013, p. 184), the most common choices for k in k-fold CV is using $k = 5$ or $k = 10$. These particular values have been shown empirically to yield test error estimates that does not have high bias, or high variance.

4.6.2 The Bootstrap

The bootstrap comes in as a useful tool for quantifying the uncertainty of the model estimates. The bootstrap is a flexible and powerful statistical tool that involves drawing random samples with replacement of the original data set. These random samples are called *bootstrap data sets*, and are, therefore, of the same size as the original data set. As a result some observations may appear more than once in a given bootstrap data set, and some observations not at all. The procedure is repeated B times for some large value of B, often 100 or 1000 times, in order to generate B different bootstrap data sets. For each of these data sets the mean can be computed, and this will generate a histogram of the bootstrapped means. This provides an estimate of the shape of the distribution of the mean, fromwhere the variability of the estimates can be calculated.

Chapter 5

Machine Learning Algorithms

5.1 K-nearest neighbours

The K-nearest neighbours (KNN), where K is a positive integer, is among the simplest of all supervised machine-learning algorithms. The KNN can be used for both classification and regression problems. Despite the simplicity of the algorithm, KNN has been successful in a large number of classification problems. Handwritten digits, satellite image scenes and EKG patterns have all been developed using the KNN. Here we will introduce how it works for classification problems, followed by how it works in the regression setting.

Supposing we have m different classes, the KNN works by first identifying the K points in the training data that are closest to the test observation x_{n+1}. Let the K points selected belong to the set N. The algorithm then estimates the conditional probability for class $j, j = 1, \ldots, m$, as a fraction of observations in N, whose response variable equals class j:

$$\mathbf{P}(Y = j | X = x_{n+1}) = \frac{1}{K} \sum_{i=1}^{K} I(y_i = j), \quad y_i \in N.$$

The test observation x_{n+1}, is then classified to the class which has the highest probability of the value j.

In figure (5.1), a two dimensional binary classification problem is illustrated. A test observation that is labelled as a cross is to be classified by the KNN. For K=3, the algorithm identifies the three nearest training observations of the test observation, shown as a circle. By comparing the conditional probabilities, the test observations is in that environment more likely to be-

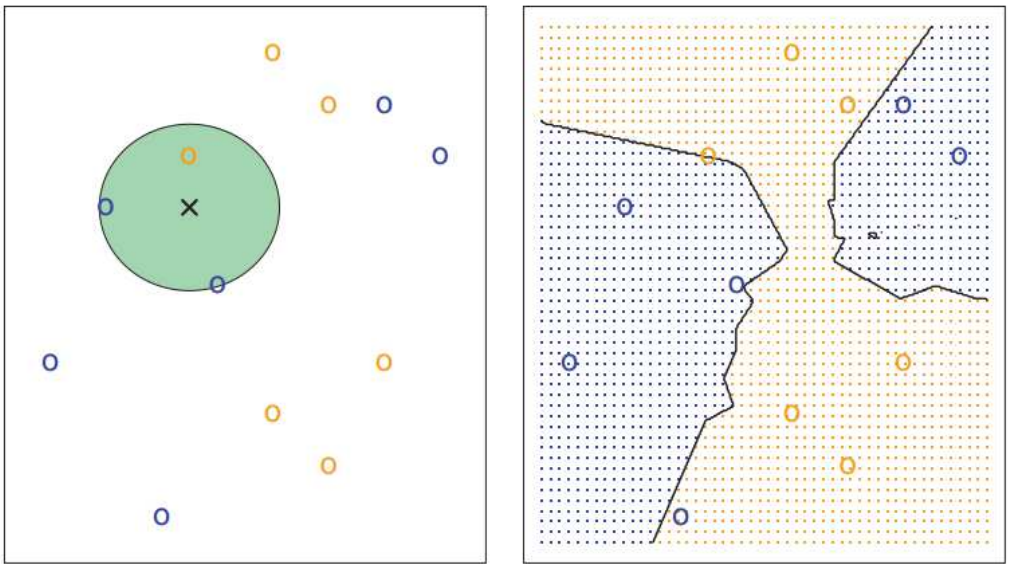


Figure 5.1: Left: The test observation is shown as a black cross. Right: The decision boundary by selecting K=3 is shown. Since the value of K is low and the training data set is small, the model is prone to being an overfit to the data. These pictures are obtained from James et al. (2013, p. 40)

long to the blue class. Therefore, the algorithm predicts the test observation x_{n+1} to belong to the blue class. In the right-hand picture in figure (5.1), the black line indicates the decision boundary of the model. After every possible set of observations have been classified by the algorithm, a test observation that belongs to the center orange region, will be assigned to the orange class and vice versa.

The result of the KNN algorithm depends on the choice of K. When K=1, the method will be overly flexible, and may find patterns in the data, that are caused by the irreducible error. According to James et al. (2013, p. 40), this corresponds to a model with a low bias and a high variance. As K grows, the flexibility of the method decreases, resulting in a lower variance with a higher bias. The optimal value of K will depend, therefore, on the bias-variance trade-off, and the underlying observations.

KNN for regression is closely related to the KNN classifier. Instead of comparing the conditional probabilities, the KNN for regression takes the average of the K-neighbours, and assigns the value of the test observation.

To be more precise, for a given value of K, and a test observation x_{n+1}, the KNN measures the distances to the K nearest neighbours. If N is the set of the neighbours, the KNN estimates $\hat{f}(x_{n+1})$, using the average of N.

$$\hat{f}(x_{n+1}) = \frac{1}{K} \sum_{i=1}^{K} y_i, \quad y_i \in N.$$

The distance function is an important tool of the identification of the K neighbours, and is chosen by the user. The most natural choice of the distance function, for numeric attributes, is the Euclidean distance measure:

$$d(x, x') = \sqrt{\sum_{i=1}^{n} (x_i - x_i')^2}.$$

The KNN algorithm can also use other distance functions. For example, the Minkowski distance with a pre-defined number m, might result in a better model.

$$d(x, x') = (\sum_{i=1}^{n} |x_i - x_i'|^m)^{1/m}.$$

If m=1, the distance function becomes the Manhattan distance. Typically, the Euclidean or Manhattan distances are chosen as the distance measures, for the KNN algorithm.

The KNN may result in good predictions, if the number of variables p in the observations is small. However, if p is large, one has to take into consideration the fact that a given test observation may have no nearby observations in its environment. Therefore, when the algorithm is identifying the K nearest neighbours, the distance within the set of neighbours can be very large. According to James et al. (2013, p. 108), this is called the *curse of dimensionality*. It arises from the fact that when p increases, the space becomes high-dimensional. Therefore, the distances between the observations grows, which will lead to a poor KNN fit.

5.2 Decision Tree

In the following chapters we will introduce supervised machine-learning algorithms, that have a different approach than KNN, called *tree-based methods*.

These methods involve splitting the predictor space into a number of simple regions. In order to make a prediction for a test observation, tree-based methods typically uses the mean or mode of the training observations in the region to where it belongs. Therefore, when the KNN uses a neighbourhood of K neighbours for a test observation, tree-based methods, as a first step, segments the training observations into similar groups.

Tree-based methods are useful since the information structure can be presented graphically by a tree, with branches and leaves. Therefore, these types of approaches are called *decision tree* methods. Decision trees can be used for both classification and regression problems. Here we will exclude the presentation of the classification setting, and focus on the regression setting. The process of building a regression tree involves two steps:

1. The set of all possible values for $X_1, X_2, \ldots, X_p$ i.e the predictor space, is divided into J distinct and non-overlapping regions $R_1, R_2, \ldots, R_J$.

2. For every test observation that belongs to region R_j, we make the same prediction. This is the mean of the response values for the training observations in R_j.

In step one, the algorithm divides the predictor space into high-dimensional boxes. The goal is to find the regions $R_1, R_2, \ldots, R_J$, that minimize the RSS of the model. The RSS is given by

$$\text{RSS} = \sum_{j=1}^{J} \sum_{i:x_i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

where $\hat{y}_{R_j}$ is the mean response of the training observations in the jth box. Due to computational limitations, it is not possible to consider every possible partition of the prediction space into the J boxes. For this reason, tree-based methods uses an top-down approach called *recursive binary splitting*.

Recursive binary splitting involves, according to James et al. (2013, p. 307), selecting the predictor X_j and the cut point s, such that splitting the predictor space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the greatest possible reduction in the RSS. In greater detail, for any j and s, we define the pair of half-planes

$$R_1(j, s) = \{X|X_j < s\} \text{ and } R_2(j, s) = \{X|X_j \geq s\},$$

and seek the value of j and s that minimize the quantity

$$\sum_{i:\ x_i \epsilon R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\ x_i \epsilon R_2(j,s)} (y_i - \hat{y}_{R_2})^2.$$

When the predictors and cut points have been identified, the process is repeated. The method then searches the best predictor and cut point in order to split the data further, to minimize the RSS within the resulting regions. Therefore, instead of splitting the entire predictor space, the method splits one of the two previously identified regions. The process is then continued until a stopping criterion is reached, which is to be defined by the user. When the process is completed and the regions $R_1, R_2, \ldots, R_J$ have been created, the algorithm predicts the value for a given test observation. The predicted value of the test observation is the mean of the training observations, in the region to which the test observation belongs.

Since the number of branches and leaves are increasing, the resulting decision tree might become too complex. This affects the prediction accuracy and might produce an overfitting model. On the other hand, a smaller tree with fewer regions will lead to a lower variance, with an increased bias. James et al. (2013, p. 308), argues that to find the optimal complexity of the tree, one alternative is to build a very large tree T_0, and *prune* it back in order to obtain a *subtree*. The goal is to select a subtree that leads to the lowest test-error rate using cross validation. However, due to computational limits, a smaller set of subtrees should be favoured. A method for achieving this is called *cost complexity pruning*. The method considers a sequence of trees, indexed by a tuning parameter α. For each value of α, there corresponds a subtree $T \subset T_0$, so that

$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \epsilon R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|, \tag{5.1}$$

is as small as possible. $|T|$ refers to the number of terminal nodes in the tree T, and R_m is the subset of the predictor space. The tuning parameter α, therefore, controls the complexity of the tree and eschews the model from overfitting. When α=0, equation (5.1) measures the training error and T=T_0. As α increases, (5.1) will tend to be minimized for a smaller subtree. To summarize, the algorithm of building a regression three is as follows, by James et al. (2013, p. 309):

1. Construct a large tree using recursive binary splitting on the training data. The process stops when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree, in order to obtain a sequence of best subtrees, with varying α.

3. Use k-fold cross validation to chose α. For each step $k = 1, \ldots, K$:

(a) Repeat steps 1 and 2 on all but the kth fold of the training data.

(b) Evaluate the mean squared prediction error on the data in the left-out fold, as a function of α.

(c) Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from step 2, that corresponds to the chosen value of α.

5.3 Random Forests

Decision trees entails useful properties. In particular, they can be displayed graphically by a hierarchical diagram. The result of the algorithm is therefore easily interpreted and easily explained. At the same time, the decision trees suffer from a major deficiency. Decision trees have a high variance, and, therefore, a low predictive power. In fact many other machine-learning algorithms are more accurate in predicting test data than decision trees. To solve this problem, the predictive performance of the method can be substantially improved by aggregating different decision trees. These methods include *bagging* and *random forest*.

According to Hastie et al. (2009, p. 587), the essential idea in bagging is to average many decision trees, and hence reduce the variance. Since different training sets are not accessible, the bootstrap, discussed in section 4.6.2, is used to take B single samples from the original training data set. By training B different models, using each training set $b = 1, \ldots, B$, the resulting predictions can be averaged for a new unseen test observation x_{n+1}. Therefore, the prediction of the test observations x_{n+1} is obtained from:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x).$$

The resulting model of bagging cannot be graphically displayed, and is hence more difficult to interpret. An overall summary of the importance of each predictor can be obtained however, using the RSS for bagging regression trees. The total amount of the decrease of the RSS, due to the splits over a given predictor, can be recorded and averaged over all B trees. A large value will indicate an important predictor. This can be plotted by the software used in this thesis and is known as variable importance.

Random forest is a machine-learning algorithm that works in a manner similar to bagging, with an additional improvement. As in bagging, a number of decision trees are built on bootstrapped training samples. For each split, according to James et al. (2013, p. 320), the random forest algorithm instead uses a random sample of $m < p$ predictors as split candidates. The split is then allowed to only use one of the m predictors. For every split, a new set of m predictors are chosen. The value m is typically chosen to be $m \approx \sqrt{p}$. The coice of m affects therefore the outcome of the model. If the predictor variables are correlated, the model might become better for a small value of m. If $m = p$, the random forest algorithm reduces to bagging.

The random forest algorithm is not allowed to consider a majority of the predictors at the different splits. The benefits of this has to do with controlling the variance in the different training samples. Suppose there would be a strong predictor variable in the data set, then most of the trees would use this strong predictor variable in the top split. Random forest overcome this problem by forcing each split to consider only a subset of the p predictor variables. As in James et al. (2013, p. 320): "We can think of this process as decorrelating the trees, thereby making the average of the resulting trees less variable and hence more reliable".

Chapter 6

Simulations and results

The purpose of this thesis is to analyze how the GLM models the risk premium for real insurance data compared to the machine-learning methods. By removing a certain part of the data and thereby predicting it, the main goals are the following:

1) Estimate future values by the different models as accurately as possible, and measure the results.

2) Obtain different trends from the models, and study the results.

The data obtained is from a insurance company whose name and type of observations is unpublished. The data consists of observations from a period of six years. By modelling the five first years with the different models, the models then estimates the risk premium of the sixth year. The trends in the data is analyzed from the models that are optimized, using the observations from the first five years.

6.1 Simulations

6.1.1 The data and the treatment

For the data obtained, the type of insurance consisted of different subsets of insurances types. Therefore, it was possible to analyze how different predictor variables affected each type of insurance. The number of observations, and the number of predictor variables will, however, be unpublished. Before the modelling could begin, treatment of the data was necessary.

When using real data, potential problems may arise even before the modelling can begin. One example of the problems are *outliers* in the data. An outlier is a observation whose value is far from the value predicted by the model, and differ significantly from the other observations. These outliers can in insurance data, for example, be large claims, deviating from the other claims. To solve the problem of large claims, a decision was made to reduce the large claims to a certain value. In A.4 it is shown how this is done using the software R, where the large claims are reduced to 100000 euro.

Another problem that arouse from the data was observations containing too specific values. For example, the birthdates of the customers were too specific in the data and needed to be clustered into explicit age clusters. The benefit of clustering, is that the output of the model is more easily interpreted. The clustering of the data was done manually, and the different clusters were decided in advance. An example is shown in A.5, where the age of the customers is clustered into age groups of four years. Another set of issues in the data was negative claims, empty observations, columns with the same values etc. These issues needed to be dealt with separately.

For each of the different brands of insurances in the data set, every observation had a value for the amount of claims and the sum of the claims. In addition, the data had a specific exposure for each observation. As was mentioned in section 3.6, to model the claim frequency for a certain insurance brand, the response variable was modified to be the claim count/exposure. To model the severity, the response variable was modified to the total cost of claims / amount of claims.

6.1.2 The GLMs and Machine Learning algorithms in statistical software

Using the statistical software R, generalized linear models are fit using the $glm()$ function. In R, the form of the GLM function is:

$$glm(\text{formula}, family = \text{familytype}(link = \text{linkfunction}), data = \text{datafile})$$

where the formula, familytype, linkfunction and datafile is specified by the user. The formula defines to the program which predictor variables are to be included in the GLM, and can, for example, be $Y = X_1 + X_2 + X_3$, if the variables are named accordingly in the data set.

To be able to use the K-Nearest neighbors algorithm in R, a package was acquired to be downloaded. For this thesis the package *kknn* was used. The user defines the values of K and the value of the distance function, which are represented here by k and d. The form of the KNN function *train.kknn* using the mentioned package is:

$$train.kknn(\text{formula}, K = \text{k}, distance = \text{d}, data = \text{datafile})$$

A certain package is also required to be able to use the random forest algorithm in R, in this thesis the package *randomForest* was used. The user defines the number of trees and the number of variables to use for each split, which are represented by t and m. The form of the *randomForest* function using the package is the following:

$$randomForest(\text{formula}, ntree = \text{t}, mtry = \text{m}, data = \text{datafile})$$

The GLMs, K-Nearest neighbours- and random forest algorithms do have more inputs in the software R. These are, however, left out of the thesis.

6.1.3 Cross-validation in R

To avoid over- or underfitting, the machine-learning algorithms needs to be optimized using resampling methods, as was discussed in section 4.6. In this thesis this was done manually using cross-validation with 5 folds. Since the amount of data was large, k was chosen to be 5 instead of 10 due to computational advantages. By choosing different sequences of the parameters in the machine-learning algorithms, the average test MSE was measured and saved for each combination. To demonstrate the procedure, the data set *Boston* will be used, which is a built-in data set in the software R. By using the principle on the Boston data set the output of the program was the following, where the source code is found in Appendix A.7:

```
[1] "TestMSE,ntree=50,mtry=2:17.4657208145429"
[1] "TestMSE,ntree=50,mtry=4:16.6089961698928"
[1] "TestMSE,ntree=50,mtry=6:17.3133815070617"
[1] "TestMSE,ntree=100,mtry=2:16.6915167334573"
[1] "TestMSE,ntree=100,mtry=4:16.3203167477015"
[1] "TestMSE,ntree=100,mtry=6:17.190588433958"
```

```
[1] "TestMSE,ntree=150,mtry=2:16.754537109479"
[1] "TestMSE,ntree=150,mtry=4:16.344288927649"
[1] "TestMSE,ntree=150,mtry=6:17.1475543021365"
[1] "TestMSE,ntree=200,mtry=2:16.8390160490675"
[1] "TestMSE,ntree=200,mtry=4:16.3058607531488"
[1] "TestMSE,ntree=200,mtry=6:17.1438815888588"
[1] "Optimal_testMSE,ntree=200,mtry=4:16.3058607531488"
```

Information about the Boston data set is obtained from Appendix A.6. By programming that the model would automatically use the combination of parameters that resulted in the lowest average test MSE, the models would optimize themselves. Below is an example code where a random forest algorithm is optimized by the coded program:

```
require(randomForest)
mse = function(error){
        mean(error^2)
}
k_fold_value=5
t_value_seq=seq(200,250,50)
m_value_seq=seq(6,12,3)
list_t=list()
list_m=list()
list_mse=list()

yourData=datafile[sample(nrow(datafile)),]
folds = cut(seq(1,nrow(yourData)),breaks=k_fold_value,
        labels=FALSE)

ticker=1
for (t in t_value_seq){
  for (m in m_value_seq){
    cv.error=rep(0,k_fold_value)
    for(i in 1:k_fold_value){
      testIndexes = which(folds==i,arr.ind=TRUE)
      trainData = yourData[-testIndexes, ]
      testData = yourData[testIndexes, ]
      training.rf=randomForest(Y=X1+X2+X3,data=trainData,
          ntree=t, mtry=m)
      testPred=predict(training.rf,newdata=testData)
```

```
        testData$testPred=testPred
        error=testData$Risk-testPred
        testMSE=mse(error)
        cv.error[i]=testMSE
        }
      testMSE=sum(cv.error)/k_fold_value
      list_t[[ticker]]=(t)
      list_m[[ticker]]=(m)
      list_mse[[ticker]]=(testMSE)
      ticker=ticker+1
      }
}
Model=randomForest(Y=X1+X2+X3,data=datafile,
ntree=list_t[[which.min(list_mse)]],
mtry=list_m[[which.min(list_mse)]])
```

The program begins by sampling the data set, dividing the sampled data set into the number of folds, defined by the user. The specific parameters of the model are obtained from the values in the sequences *t_value_seq* and *m_value_seq*. The program then uses the first combination of parameters leaving out the last fold of the data set, and predicts the values of the last fold. The errors in the prediction are saved in an array called *cv.error*. By repeating the process with all the folds, the average test MSE is calculated using the mean of the values in *cv.error*. The results are then saved in a list called *list_mse*.

The parameters used to obtain the values are continuously saved in the lists *list_t* and *list_m*. The program then repeats the process with all the different combination of parameters. When the process is done, the program searches for the index in the list *list_mse* that resulted in the lowest average test MSE using the *which_min* function. It then obtains the objects from the *list_t* and *list_m*, with the same indexes used for the best result. A final model is then constructed using the parameters that resulted in the lowest average test MSE.

6.1.4 The different models used

As was mentioned in section 3.5.2, software as *R* shows a statistical output of the goodness of fit of the GLM. This output includes information about the significance level of each predictor variable. In practice when modelling

with the GLMs, many models are tested before the final model is obtained. All the predictor variables in the data are included in the first model. Thereafter, according to Anderson et al. (2007, p. 52), the variables that are not significant are dropped one by one, until a final model is obtained. The final model thus consists only of significant predictor variables.

Therefore, to be able to compare the different models, two different sets of GLMs and machine-learning models were used. One set of models included only the significant variables, according to the GLM outputs. The other set of models included all the variables. The aim was to find out how much the models deviated from each other using all available predictor variables, compared to using only the significant predictor variables. To summarize the procedure, with all variables as well as with the significant variables, the resulting models were the following:

	Frequency	**Severity**
Significant variables	GLM Poisson with log link KNN RF	GLM Gamma with log link KNN RF
All variables	GLM Poisson with log link KNN RF	GLM Gamma with log link KNN RF

For the insurance company's data set, two different type of insurances were available. For both insurance types, observations of the number of claims and total amount of claims were accessible in the data set. The aim was to model frequency and severity separately. To obtain the risk premium, as was discussed in section 3.6, the frequency and severity estimates of the models were multiplied. The models constructed using machine-learning algorithms were optimized using cross-validation. The optimized values for K, *distance*, *ntrees* and *mtry* of the K-Nearest neighbors- and random forest models will not be published.

6.2 Predictions and finding the trends

By leaving out the last year of the data set, it was possible to measure how well the models estimated the known future values. The MSE was used to

measure the error of the prediction, where the response variable was the risk premium. For the aim of being more practical, the sum of all claims, the mean claim amount and average severity were estimated. To predict values on a certain data set, the function *prediction* is used in *R*. The command is the same for every model used in the thesis as the following:

$$predict(\text{model}, newdata = \text{datafile})$$

When using the GLMs, software as *R* is able to show a statistical output of the GLM fit to the data. The output can be studied to obtain information about the different predictor variables. From this output the user can obtain trends in the data, and receive information about the discrepancy within the predictor variables. This enables the user, for example, to study the claim frequency for a certain type of insurance, depending on the customers residence area. A thorough analysis of the GLM output is left out of the thesis.

When using the machine-learning algorithms, however, a similar summary of the model is not available. Using the random forest algorithm, nonetheless, it is possible to obtain which predictor variables are the most important for a fitted model. This is done using the built-in function $varImpPlot()$, and the importance of the variable is measured with the unit *IncNodePurity*. The IncNodePurity is, according to James et al. (2013, p. 330), a measure of the total decrease in the training RSS that results from splits over a variable, averaged over all trees. The theoretical background for this output is also left out of the thesis.

To be able to compare the different models, and for the aim of understanding how they modeled the training data, the models were examined in a similar manner. When the models had been fitted on the training data, they were made to predict the response value on the same data it was trained on. By categorizing the observations in the data, the model estimates of the response variable were studied for each category, within a certain predictor variable.

To demonstrate the procedure, the built-in data set *Auto* will be used, which is introduced in Appendix A.6. The aim is to model the miles per gallon *mpg* using the variables *cylinders*, *horsepower*, *weight*, *acceleration* and *year* of the car. In the data set are five different values of the cylinders variable, 3,4,5,6 and 8. Suppose we are interested in how much the number of cylinders affects the miles per gallon according to the models. After modelling

the different models on the data set using the variable *mpg* as a response variable, the estimated values on were divided into different tables, according to the number of cylinders. The estimates of the *mpg* for each value of the cylinder were studied, and the mean and confidence interval was obtained.

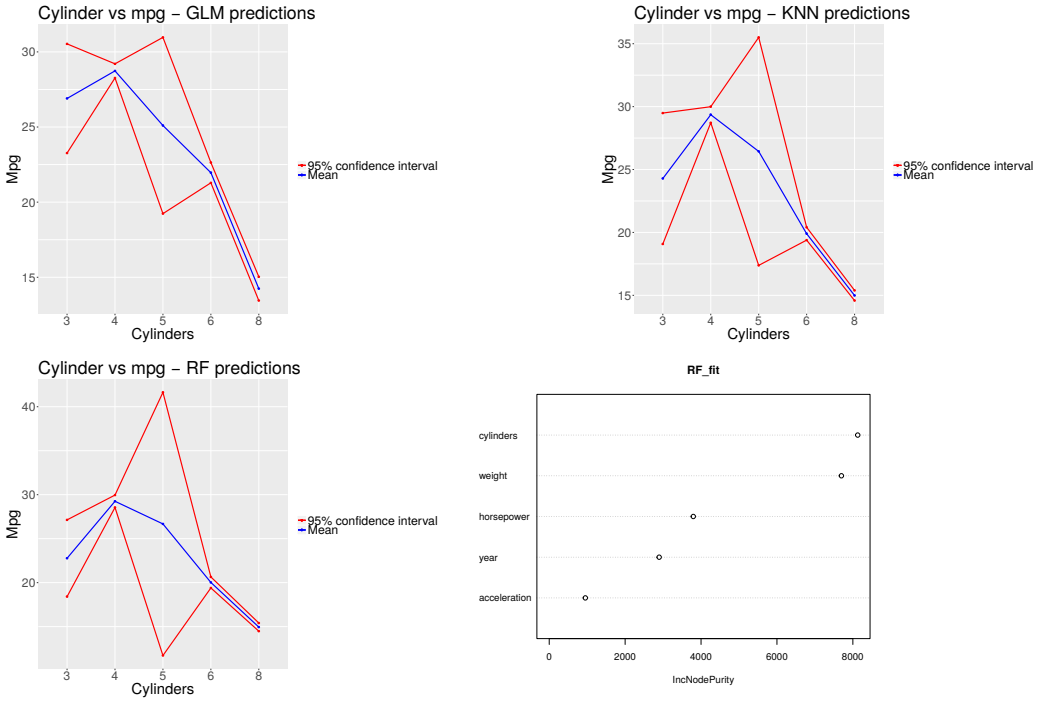


Figure 6.1: The mean and confidence interval of the estimates of each *cylinders* value are shown. From the bottom-right graph, the output of the variable importance function using the random forest algorithm is shown. The importance of the variable is measured with the unit *IncNodePurity*, which is a built-in unit for the random forest algorithm. According to the output, the variables *cylinders* and *weight* are the most important variables for estimating the *mpg* of the car.

From the different graphs, the user receive information about how discrepancy within the *cylinder* variable affects the *mpg* variable, within the model estimates. The source code is found in Appendix A.8. The same principle was used on the insurance data set, whose results is obtained from the next section.

6.3 Results

Predictions

For both insurance types in the data set, each model predicted the claim frequency and average claim severity of the last year in the data set. The predictions of the total claim cost, MSE, number of claims and average claim severity can be obtained from the following table:

Model	Total claims	MSE predictions	Type 1: Claims	Type 1: Aver. sev.	Type 2: Claims	Type 2: Aver. sev.
True values	1624682.1	0	1051.00	2491.58	578.00	689.30
GLM sign. var.	752226.8	13525074	232.60	2695.97	154.89	807.87
KNN sign. var.	1009971.2	14763660	436.87	2027.19	150.23	827.75
RF sign. var.	1121655.4	13113843	416.49	2496.72	113.83	718.50
GLM all var.	751426.6	13507334	234.65	2638.23	153.14	864.37
KNN all var.	1185227.7	13185386	421.90	2611.64	108.36	769.53
RF all var.	1559681.1	13221856	492.06	2936.52	137.89	832.00

Note that the values in the table are rounded. The total claim cost of the last year in the data set was 73% larger than the average of the previous years. Hence the estimates of the models were generally lower than the true values. For both insurance types, the models estimated the average claim severity with better results than the average claim count. This can be explained due to the fact that the observations are integers of the claim numbers, which generally contains many zero-values.

To be able to compare the estimated values further, the estimated total cost of the last year were studied for different groups in the data. For example, the estimates of the total cost of each age group in the data set is presented. We will also demonstrate the estimates of total cost for customers having different sums insured. The exact sums insured will not be published. However, different levels will be shown where a higher number indicates a higher sum insured of the customer.

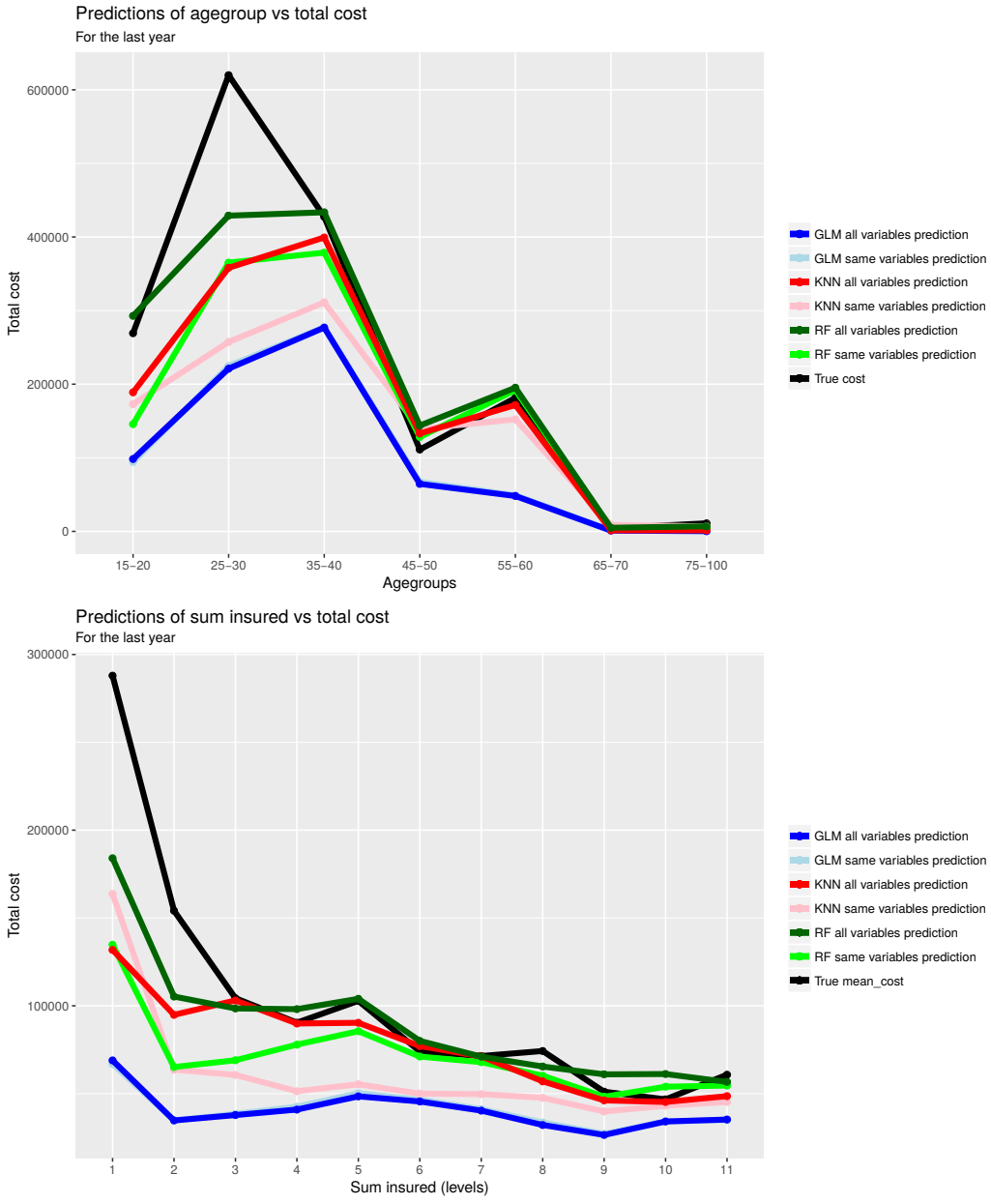
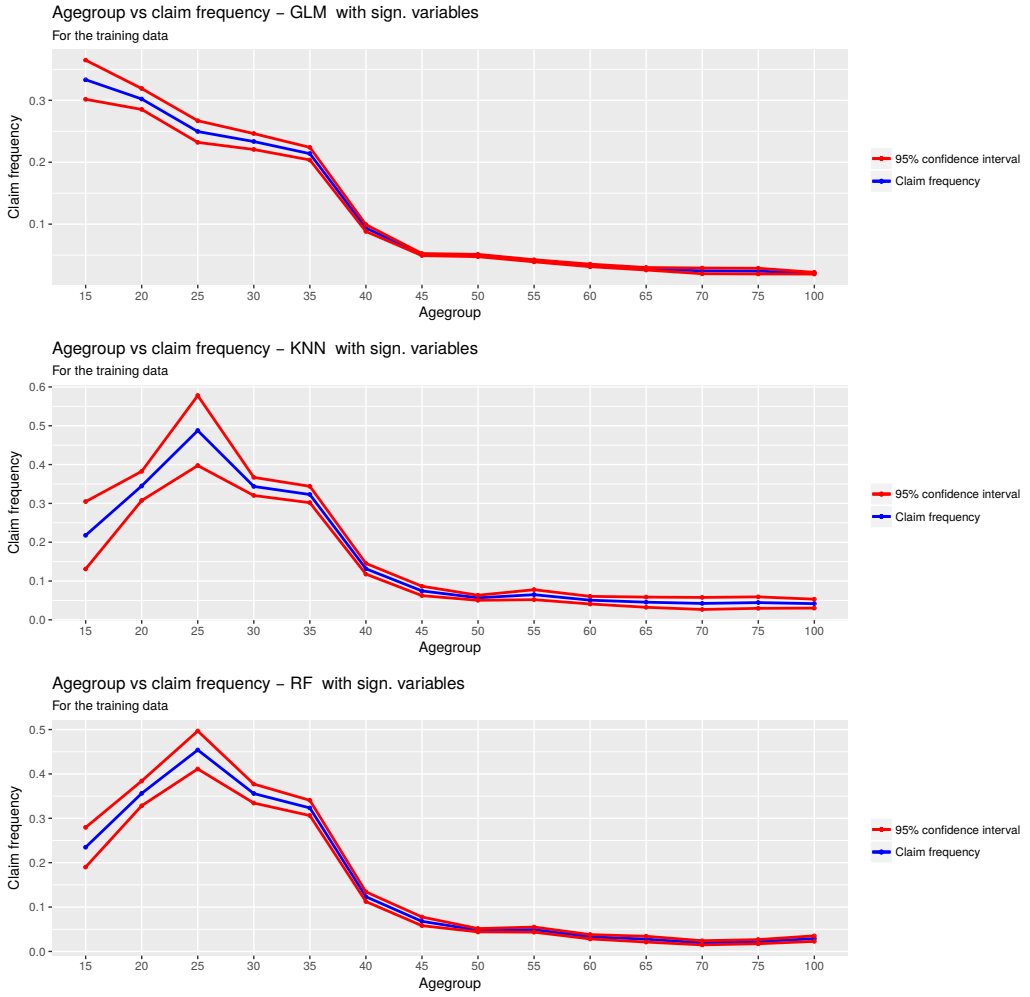


Figure 6.2: The model estimates of total cost vs different ages groups and different sums insured are shown

The trends in the data set

To be able to compare how the different models modeled the training data, the procedure presented in section 6.2 was used. The following is the frequency of a certain insurance claim according to the age of the customer. For this particular data set, the frequency of the unpublished insurance types are generally higher for younger customers.



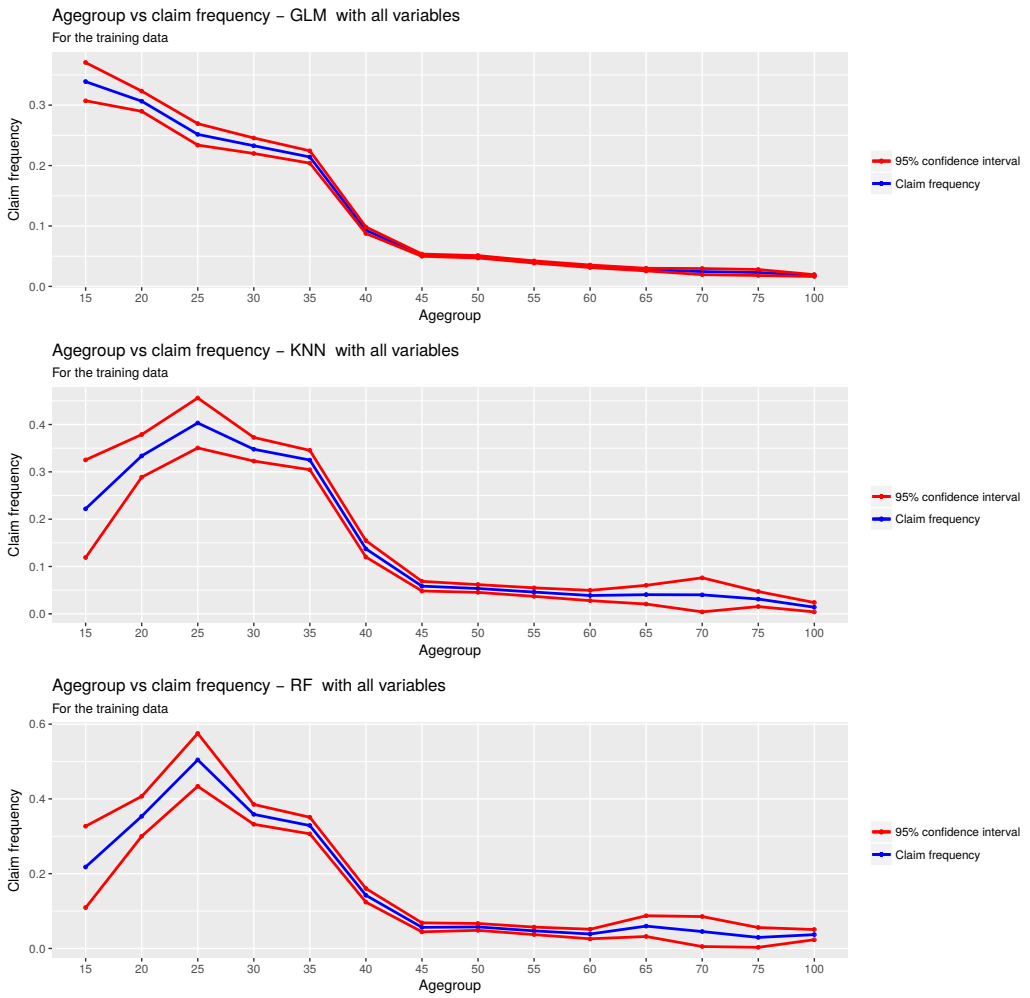


Figure 6.3: The procedure discussed in section 6.2 is used to compare how the age of the customer affects the claim frequency according to the models

Chapter 7

Conclusions

7.1 The modelling process

The time it takes for the GLMs, KNN and RF to model the data has so far been left out of the discussion. The time for the modelling differs significantly between the models and depends on the amount of data and the parameters in the machine-learning models. To demonstrate the difference, the time was measured for the GLM, KNN and RF to model different data sets where the number of observations was increasing. The parameters in the models were constant. It can be seen in figure 7.1, that the time for the modelling is increasing exponentially for the machine-learning algorithms. Since cross-validation is required when modelling with the machine-learning algorithms, the modelling process could, as a whole, last hours using a large data set.

When the machine-learning algorithms are used for modelling data, the user is required to test a different set of parameters using cross-validation. As could be seen in section 6.1.3, a major amount of coding is required to optimize the models. The best combination of the parameters is then used for the final model. When the number of parameters included and the amount of data are increasing, computational issues may arise and the time required becomes substantial. This can be solved by only choosing a subset of the predictor variables in the model. This procedure may, however, affect the results, and the result depends on the choice of predictor variables.

As was mentioned in chapter 5.3, the random forest algorithm models bootstrapped data sets from the training data, from where the final model is obtained as a mean of the models. This implies that over- or underfitting

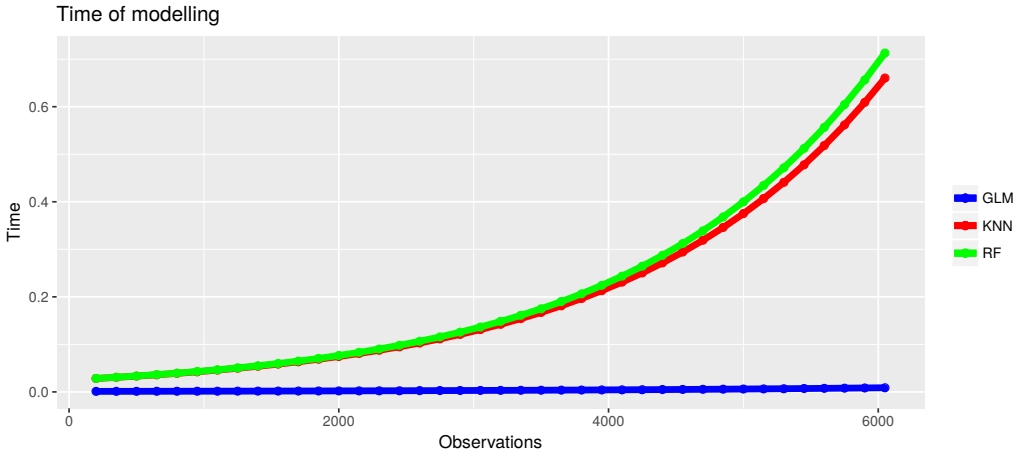


Figure 7.1: The time for modelling was measured using a random forest, k-nearest neighbor and a GLM, where the parameters in the models were constant and the number of observations were increasing. The measured times for each model were then fitted using a GLM to obtain smoother lines. The source code is left out of the thesis.

is avoided by the model itself, before using the cross-validation. The K-nearest neighbors algorithm does not work in a similar manner, and is not a mean of different models. The cross-validation is therefore an important aspect when using the KNN-algorithm, so that the user receives information about the sensitivity of changes in the parameters within the model. For the modelling process, it was noticed that the value for K in the K-nearest neighbors algorithm does not affect the cross-validation result as much as the *distance*-parameter. This suggests that the user is required to try several combinations with varying values for the *distance*-parameter.

GLMs, however, does not require the user to test a set of different parameters. Contrarily, when modelling with the GLM, the user is required to understand the data, so that the appropriate distribution of the exponential family and link functions are selected. When the distribution and link functions are chosen, the modelling with GLMs in statistical software does not require additional programming. If the user would be unfamiliar to the data set, cross-validation could in theory be used on GLMs; the user would then test a set of distributions and link functions for finding the best model.

7.2 Comparing the models

According to the table-values and different graphs from section 6.3, it seems that the machine-learning models are more suitable than the GLM for predicting future values for the particular data set used. This can be explained from the fact that the GLM was possibly an under- or overfit for the data, or, in other words, that the family type or link function was not appropriate for the data set.

Since the total claim cost for the last year in the data set was larger than the previous years, all the models underestimated the values. The random forest algorithm using all the variables resulted in the best estimate, while both GLMs resulted in substantial underestimates. For both types of insurances, all the models estimated a considerably lower number of claims. This might have to do with the fact that the training data contained customers with no claims at all, which might affect the model. Another explanation is that the data for the last year had a higher number of claims than the average in the training data.

The model estimates of the average severity, however, was significantly closer to the true data. This might have to do with the fact that the models used training data where the number of claims were non-zero. This results in that the models were able to model the data more accurately, since it used training data with less zero-value observations.

The model estimates in figure 6.2 shows that the random forest algorithm using all predictor variables is closest to the true values. Except for the GLMs, there was a considerable variability in the model estimates. The GLM estimates using only the significant variables does not differ almost at all from GLM using all the predictor variables. It can be seen that the machine-learning model estimates differ when they only use a subset of the predictor variables. The data set for the last year seems to contain patterns that no model could estimate. The total cost for the age groups 25-30 and the level 1 sum insured had a greater total cost than any of the model estimates.

According to figure 6.3, the GLM predictions have less variability than the machine-learning model estimates, which is shown by the confidence intervals. This suggests that the GLMs have a higher bias and less variance, discussed in section 4.5, which implies that both GLMs are underfits to the data. This also explains why the GLM predictions deviated from the other model predictions on the data set for the last year. By comparing the results in figure 6.3, it can be seen that the estimates of GLM using only significant

variables and GLM using all variables does not differ. Rather, the results are almost identical. This cannot be said about the machine-learning models where the estimates had a greater variability.

The biggest difference between the models in figure 6.3 is that the machine-learning models estimate a lower frequency for the younger customers than the GLM estimates. This might be a result of the GLM models being underfits to the data set. The machine-learning models are also estimating broader results for the age groups 65-75, which is not the case for the GLM models.

From the variable importance plot obtained from the random forest algorithm, it can be seen which variables are the most important for the frequency or severity for both insurance brands. This information is valuable from an insurer′s point of view. Unfortunately, for the data used, the variable importance plot from the random forest algorithm will not be published.

7.3 Summary

These results suggest that the models derived from machine-learning algorithms are able to model the data set with less bias and more variance than the GLMs. Thus, the predictions of the machine-learning models are also more accurate for the last year in the data set. The results also shows that when all the predictor variables are included in the machine-learning models, the estimates of the models have more variability. Using all the variables, the machine-learning algorithms are able to find certain patterns in the data set. This cannot be said about the GLMs, whose results are almost identical when only using the significant predictor variables in the modelling process. It is unfortunately impractical to include all variables in the machine-learning algorithms, since the time required for the modelling becomes too long.

It was concluded that for this particular data set, the random forest algorithm using all the variables was the most accurate when predicting future values. The total number of claims and average severity of claims to insurance type 1 was closest to the true values. Figure 6.2 shows that for the age groups and sums insured, the random forest algorithm using all variables was able to predict values that were closest to the true values for almost every group.

As was discussed in section 3.5.2, when using GLMs in statistical software, the user obtains an output of the fitted model. From the output the user can receive information about the coefficients in the model, measures for the

goodness of fit, such as deviance and AIC, and a thorough overview about the model for understanding the training data. A complete introduction of the GLM output was left out of the thesis, but the user is able to obtain information that is valuable from an insurer′s point of view. By including factors in the model, the user can, for example, find which residence areas result in the highest and lowest frequency or number of claims, directly from the model output.

The same output cannot be obtained from the machine-learning algorithms. This is due to the fact that the theory behind the machine-learning algorithms is different from the theory of GLMs. Using the random forest algorithm, however, the user obtain information about which predictor variables are the most important for estimating the response variable, as was discussed in sections 5.3 and 6.2. Using the KNN algorithm, however, a built-in function for understanding the training data is not obtainable.

Hence, to understand the training data, the GLM would be a suitable option for the user. If the aim is predicting future values, the results in section 6.3 suggest that the machine-learning algorithms are more suitable. The user is also not required to understand the data in advance when using the machine-learning algorithms and, by finding the best parameters, the machine-learning models will optimize themselves. No particular method is, however, regarded as the best one, as was mentioned by James et al. (2013 p.29): No method dominates all others over all possible data sets. For a particular data set, one specific method may work best, but some other method may work better for a similar but different data set.

Chapter 8

Svenskspråkig sammanfattning

Målet med denna avhandling är att introducera den centrala teorin inom generaliserade linjära modeller (GLM) och maskininlärning, samt påvisa hur man använder dessa för att modellera riskpremien inom försäkringsbranschen. Till modelleringen används två kända algoritmer ur maskininlärning. Modellerna jämförs genom att för- och nackdelarna med de olika modellerna diskuteras och analyseras.

Den första helheten handlar om att få en noggrannare inblick i försäkringsmatematiken. För att kunna presentera försäkringsmatematiken, börjar avhandlingen med de centrala begreppen ur sannolikhetsläran. Sannolikhetslärans teori används som en byggsten till de olika helheterna, och är en väsentlig del speciellt inom försäkringsmatematiken. Den första helheten kulmineras i hur man modellerar antalet skador och storleken på skadorna ur en matematisk synvinkel.

De två följande helheterna utvecklar teorin inom GLM samt maskininlärning. Bakgrunden till GLM, modelleringsproceduren och den exponentiella familjen av fördelningar tas upp och teorin sammanknyts med räkneexempel. För att få en inblick i maskininlärning, presenteras den bakomliggande teorin i kapitel 4. I kapitel 5 presenteras de två algoritmerna som slutligen tas med i avhandlingen.

Den fjärde helheten handlar om simulationer och hur man åstadkommer dem med hjälp av programmering. För att kunna jämföra de olika modellerna används försäkringsdata vars innehåll, tid och storlek inte kan publiceras i denna avhandling. Försäkringsdata består av två olika försäkringstyper, där varje observation innehåller flera beroende variabler. För att kunna jämföra hur metoderna fungerar, används olika uppsättningar av beroende variabler

för modelleringen. Vissa metoder använder sig av alla beroende variabler medan vissa använder sig enbart av de signifikanta variablerna. En hel del programmering behövs för modelleringen, och exempelprogrammen hittas som bilaga.

Till sist jämförs modellerna sinsemellan. Jämförelserna baseras på att man låter modellerna estimera framtida värden, där de framtida värdena är kända. Dessutom analyseras hur de olika metoderna har modellerat data i fråga. Resultaten presenteras i form av tabeller och grafer. Här diskuteras varför resultaten ser ut som de gör, hur de olika metoderna skiljer sig åt samt vilka metoder som modellerade data bäst. Till sist diskuteras för- och nackdelar med de olika metoderna, och slutligen avslutas avhandlingen med analyser och slutsatser i det sista kapitlet.

Materialet som undersöks i denna avhandling hör till ett brett område. I och med att avhandlingen är begränsad försöker jag få med och utveckla den nödvändigaste teorin. Till själva undersökningen används en enda uppsättning av data och slutresultatet baseras på hur metoderna modellerade denna datauppsättning. För att förbättra analysen kunde flera olika uppsättningar av data samt flera observationer tagits med i undersökningen. Från början var det tänkt att flera algoritmer inom maskininlärning skulle innefattas i avhandlingen, men dessa rymdes inte med. För fortsatt läsning rekommenderar jag att man läser om logistisk regression (logistic regression), stödvektormaskin (support vector machine) samt neuronnät (neural networks). Teorin till dessa hittas bland annat i Hastie et al. (2009).

Appendix A

R codes for chapters 2-5

A.1 Realization of a Poisson process with R

```
require(ggplot2)
values=data.frame(matrix(ncol=4,nrow=40))
colnames(values)=c("x","y","xend","yend")
values[1,1]=0
values[,2]=c(0:39)
values[,3]=cumsum(rexp(40,rate=3))
values[,4]=c(0:39)
for(i in 2:40){
        values[i,1]=values[i-1,3]
}

p = ggplot(values, aes(x=x, y=y, xend=xend, yend=yend)) +
        geom_point() +
        geom_point(aes(x=xend, y=y), shape=1) +
        geom_segment() +
        ggtitle("Poisson process", subtitle="Realization when lambda = 3") +
        xlab("Time") + ylab("Number of events") +
        theme(legend.title=element_blank()) +
        expand_limits(x=c(0,12)) +
        scale_x_continuous(breaks = seq(0,15,2)) +
        scale_y_continuous(breaks = seq(0,40,5))
plot(p)
```

A.2 Normal error structure with an identity link function with R

For the following examples A.2 and A.3, the data has been manually generated to the software. The data is generated by the following source code:

```
UrbanRural_table=data.frame(matrix(ncol=3,nrow=4))
colnames(UrbanRural_table)=c("Y","gender","territory")
UrbanRural_table[,1]=c(800,500,400,200)
UrbanRural_table[,2]=c(1,1,0,0)
UrbanRural_table[,3]=c(1,0,1,0)
```

Observation	Y	Gender	Territory
1	800	1	1
2	500	1	0
3	400	0	1
4	200	0	0

The table in the R-session, named UrbanRural_table

```
model=glm(Y~gender+territory, data=UrbanRural_table,
          family=gaussian)
summary(model)

Call:
glm(formula = Y ~ gender + territory, family =
gaussian, data = UrbanRural_table)

Deviance Residuals:
1     2     3     4
25   -25   -25    25

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    175.0       43.3   4.041   0.1544
gender         350.0       50.0   7.000   0.0903 .
territory      250.0       50.0   5.000   0.1257
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Dispersion parameter for gaussian family taken to be 2500

Null deviance: 187500  on 3  degrees of freedom
Residual deviance:   2500  on 1  degrees of freedom
AIC: 45.103

Number of Fisher Scoring iterations: 2
```

A.3 Poisson error structure with a logarithm link function

```
model=glm(Y~gender+territory, data=UrbanRural_table,
          family=poisson)
summary(model)

Call:
glm(formula = Y ~ gender + territory, family = poisson
, data = UrbanRural_table)

Deviance Residuals:
1         2         3         4
-0.7379    0.9550    1.0717   -1.4394

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.39840    0.05068  106.51   <2e-16 ***
gender       0.77319    0.04935   15.67   <2e-16 ***
territory    0.53900    0.04756   11.33   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Dispersion parameter for poisson family taken to be 1
```

```
Null deviance: 401.889  on 3  degrees of freedom
Residual deviance:   4.677  on 1  degrees of freedom
AIC: 42.219

Number of Fisher Scoring iterations: 3
```

A.4 Reduction of large claims

```
for (i in 1:nrow(data)){
        if ( data$sum_of_claims[i]>100000 ){
        data$sum_of_claims[i]=100000
        }
}
```

A.5 Clustering of data

```
for (i in 1:nrow(data)){
        format=format(data$birthdate[i], '%Y')
        data$age[i]=format
}
ages=unique(data$age)
ages=sort(ages)
birthyear= c("1915-1925","1926-1929","1930-1933","1934-1937","1938-1941","1942-1945",
"1946-1949","1950-1953","1954-1957","1958-1961","1962-1965","1966-1969","1970-1973",
"1974-1977","1978-1981","1982-1985","1986-1989","1990-1993","1994-1997","1998-2000")
for (i in 1:nrow(data)){
        if( data$age[i]==ages[1] | data$age[i]==ages[2] | data$age[i]==ages[3] |
        data$age[i]==ages[4] | data$age[i]==ages[5] | data$age[i]==ages[6] |
        data$age[i]==ages[7] | data$age[i]==ages[8] | data$age[i]==ages[9] )
                {data$customer_agecluster[i]=birthyear[1]
        }
        if( data$age[i]==ages[10] | data$age[i]==ages[11] | data$age[i]==ages[12] |
        data$age[i]==ages[13] )
                {data$customer_agecluster[i]=birthyear[2]
        }
        .
        .
        .
        if( data$age[i]==ages[78] | data$age[i]==ages[79] | data$age[i]==ages[80] |
        data$age[i]==ages[81])
                {data$customer_agecluster[i]=birthyear[19]
        }
        if( data$age[i]==ages[82] | data$age[i]==ages[83] | data$age[i]==ages[84] )
                {data$customer_agecluster[i]=birthyear[20]
        }
}
data$birthdate=NULL
data$age=NULL
data=data[,c(1,2,3,17,4,5,6,7,8,9,10,11,12,13,14,15,16)]
rm(list=setdiff(ls(),"data"))
```

A.6 The Auto and Boston data sets

The Auto data set is found in the package "ISLR", which is a package created by the authors of *An introduction to statistical learning with application in R*. Miles per gallon, horsepower, weight and other information is obtained for different vehicles. The data set has 392 rows and 14 columns.

The Boston data set is found in the library "MASS". The data set contains housing values and other information about Boston suburbs. It has

506 rows and 14 columns. The response variable used in the thesis is named *medv*, which is the median value of owner-occupied homes in $1000s.

A.7 Cross validation using the Boston data sets

```
library(MASS)
require(randomForest)
set.seed(101)

mse = function(error){
        mean(error^2)
}

k_fold_value=5
a_value_seq=seq(50,200,50)
b_value_seq=seq(2,6,2)

list_a=list()
list_b=list()
list_mse=list()

yourData=Boston[sample(nrow(Boston)),]
folds = cut(seq(1,nrow(yourData)),breaks=k_fold_value,labels=FALSE)

ticker=1
for (a in a_value_seq){
        for (b in b_value_seq){
                cv.error=rep(0,k_fold_value)
                for(i in 1:k_fold_value){
                        testIndexes = which(folds==i,arr.ind=TRUE)
                        trainData = yourData[-testIndexes, ]
                        testData = yourData[testIndexes, ]
                        training.rf=randomForest(medv~crim+indus+chas+nox+tax+lstat,
                        data=trainData, ntree=a, mtry=b)
                        testPred=predict(training.rf,newdata=testData)
                        testData$testPred=testPred
                        error=testData$medv-testPred
                        testMSE=mse(error)
                        cv.error[i]=testMSE
                }
                testMSE=sum(cv.error)/k_fold_value
                print(paste0("TestMSE,ntree=",a,",mtry=",b,":",testMSE))
                list_a[[ticker]]=(a)
                list_b[[ticker]]=(b)
                list_mse[[ticker]]=(testMSE)
                ticker=ticker+1
        }
}
print(paste0("Optimal_testMSE,ntree=",list_a[[which.min(list_mse)]],",
mtry=",list_b[[which.min(list_mse)]],":",list_mse[[which.min(list_mse)]]))
```

A.8 Affect of number of cylinders on miles per gallon

The models were fitted with the following models:

- GLM using a normal distribution with identity link function
- KNN using $K = 20$ and $distance = 1.5$

- Random forest using $ntree = 270$ and $mtry = 3$

The code for the graph using the package *ggplot2* is demonstrated only for the GLM model.

```
library(ISLR)
require(ggplot2)

cylinder_vs_mpg=data.frame(matrix(ncol=17,nrow=5))
colnames(cylinder_vs_mpg)=c("Cylinder", "True_mean", "True_down","True_up",
"True_relativity","GLM_pred_mean","GLM_pred_down","GLM_pred_up","GLM_relativity",
"KNN_pred_mean", "KNN_pred_down","KNN_pred_up","KNN_relativity","RF_pred_mean",
"RF_pred_down","RF_pred_up","RF_relativity")
cylinder_values=unique(Auto$cylinders)
cylinder_values=sort(cylinder_values)
cylinder_vs_mpg[,1]=cylinder_values

GLM_predict = predict(GLM_fit,newdata=Auto)
Auto$GLM_predict = GLM_predict
for (i in 1:5){
cylinder_vs_mpg[i,6]=with(Auto[,c(2,10)][Auto[,c(2,10)]$cylinders==cylinder_values+
[i],],mean(GLM_predict))
cylinder_vs_mpg[i,7]=t.test(Auto[Auto$cylinders==cylinder_values[i],][10])$conf.int[1]
cylinder_vs_mpg[i,8]=t.test(Auto[Auto$cylinders==cylinder_values[i],][10])$conf.int[2]
}
for (i in 1:5){
cylinder_vs_mpg[i,9]=cylinder_vs_mpg[i,6]/cylinder_vs_mpg[1,6]
}

KNN_predict = predict(KNN_fit,newdata=Auto)
Auto$KNN_predict = KNN_predict
for (i in 1:5){
cylinder_vs_mpg[i,10]=with(Auto[,c(2,11)][Auto[,c(2,11)]$cylinders==cylinder_values+
[i],],mean(KNN_predict))
cylinder_vs_mpg[i,11]=t.test(Auto[Auto$cylinders==cylinder_values[i],][11])$conf.int[1]
cylinder_vs_mpg[i,12]=t.test(Auto[Auto$cylinders==cylinder_values[i],][11])$conf.int[2]
}
for (i in 1:5){
cylinder_vs_mpg[i,13]=cylinder_vs_mpg[i,10]/cylinder_vs_mpg[1,10]
}

RF_predict = predict(RF_fit,newdata=Auto)
Auto$RF_predict = RF_predict
for (i in 1:5){
cylinder_vs_mpg[i,14]=with(Auto[,c(2,12)][Auto[,c(2,12)]$cylinders==cylinder_values+
[i],],mean(RF_predict))
cylinder_vs_mpg[i,15]=t.test(Auto[Auto$cylinders==cylinder_values[i],][12])$conf.int[1]
cylinder_vs_mpg[i,16]=t.test(Auto[Auto$cylinders==cylinder_values[i],][12])$conf.int[2]
}
for (i in 1:5){
cylinder_vs_mpg[i,17]=cylinder_vs_mpg[i,14]/cylinder_vs_mpg[1,14]
}


g2=ggplot(cylinder_vs_mpg, aes(x=factor(Cylinder))) +
geom_point(aes(y=GLM_pred_mean,colour="Mean"),group=1,size=1) +
geom_line(aes(y=GLM_pred_mean,colour="Mean"),group=1,size=1) +
geom_point(aes(y=GLM_pred_down,colour="95% confidence interval"),group=2,size=1) +
geom_line(aes(y=GLM_pred_down,colour="95% confidence interval"),group=2,size=1) +
geom_point(aes(y=GLM_pred_up,colour="95% confidence interval"),group=3,size=1) +
geom_line(aes(y=GLM_pred_up,colour="95% confidence interval"),group=3,size=1) +
ggtitle("Cylinder vs mpg - GLM predictions") + xlab("Cylinders") + ylab("Mpg") +
theme(text = element_text(size=27), legend.title=element_blank()) +
scale_colour_manual(values=c("red","blue"))
plot(g2)
ggsave("Cylinder_vs_mpg_GLM.pdf",width=11,height=8.5)
```

Bibliography

[1] Anderson, D.; Feldblum, S.; Modlin, C.; Schirmacher, D.; Schirmacher, E.; Thandi, N.: *A Practitioner's Guide to Generalized Linear Models*, CAS Study Note 2007

[2] Boland, P.J.: *Statistical and probabilistic methods in actuarial science*, Chapman & Hall/CRC 2007

[3] Daykin, C.D.; Pentikäinen, T.; Pesonen, M.: *Practical risk theory for actuaries*, Chapman & Hall 1996

[4] Dobson, A.: *An introduction to generalized linear models - second edition*, Chapman & Hall/CRC 2002

[5] Hastie, T.; Tibshirani, R.; Friedman, J.: *The elements of statistical learning - second edition*, Springer 2009

[6] James, G.; Witten, D.; Hastie, T.; Tibshirani, R.: *An introduction to statistical learning with application in R*, Springer 2013

[7] McCullagh, P.; Nelder, J.A.: *Generalized linear model*, Chapman & Hall 1983

[8] Parvinen, K.: *Riskiteoria*, Matematiikan laitos, Turun Yliopisto 2009

[9] Prakasa Rao, B.L.S.: *A first course in probability and statistics*, World Scientific Publishing Co. Pte. Ltd, 2009

[11] Ross, S.M.: *Stochastic processes*, John Wiley & Sons, Inc 1983

[10] Ross, S.M.: *Introduction to probability models - 9th edition*, Elsevier 2007

[12] Salminen, P.: *Sannolikhetslära*, Matematik och statistik, Åbo Akademi 2010