KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Mathematical Statistics

Second cycle 30 credits

Analysis and Use of Telemetry Data for Car Insurance Premiums

MAX BERG WAHLSTRÖM
ANTON HAGELBERG

Stockholm, Sweden 2023

Abstract

Paydrive is a pioneer in the Swedish auto insurance market. Being able to influence your insurance premium through your driving is a concept that is still in its early stages. Throughout this thesis, an attempt to consolidate the vast amounts of data gathered while driving with neural networks has been made, together with comparisons to the currently existing generalized linear models. In the end, a full analysis of the data yielded four distinct groupings of customer behavior but because of how the data is structured the results from the modeling became sub-optimal. Insurance data is typically very skewed and zero-heavy due to the absence of accidents. The original research question is whether it is possible to use two neural networks, calculating the probability of an accident, r, and the size of a potential claim, s respectively. These two factors could be multiplied to determine a final insurance premium as $c = r \cdot s$.

Using statistical standards and tools such as the Gini-coefficient, R^2 values, MSE, and MAE the models were evaluated both individually and pairwise. However, previous research in the field shows there haven't been big enough advancements in this area yet. This thesis comes to the same conclusion that due to the volatile nature of neural networks and the skewness of the data, it is incredibly difficult to get good results. Future work in the field could result in fairer prices for customers on their insurance premiums.

Analys och Användning av Telemetridata för Bilförsäkringspremier

A. Hagelberg, M.Wahlström

May 2023

Abstrakt

Paydrive är pionjärer i den Svenska fordonsförsäkringsmarknaden. Att kunna påverka hur mycket du betalar genom din körstil är ett koncept som ännu är i sin barndom. Genom denna avhandling har ett försök att konsolidera de enorma mängderna insamlad data med neurala nätverk gjorts, tillsammans med en jämförelse mot de redan existerande generaliserade linjära modellerna. Fyra distinkta kundgrupperingar kunde hittas i datan efter en fullskalig analys men på grund av hur datan är strukturerad producerades endast icke-optimala modeller. Försäkringsdata är alltid väldigt noll-fylld och skev mot noll då kunder oftast inte råkar ut för olyckor. Den ursprungliga forskningsfrågan är huruvida det är möjligt att med hjälp av två neurala nätverk, beräkna sannolikheten för en olycka, r, och storleken av en skada, s. Dessa två faktorer kan sedan multipliceras ihop för att bestämma en slutgiltig försäkringspremie som $c = r \cdot s$.

Genom statisiska standarder och verktyg som Gini-koefficienten, R^2 värden, MSE och MAE utvärderades modellerna individuellt och parvis. Dessvärre visar våra resultat vad föregående forskning redan visat på, det saknas resurser och verktyg för att effektivt kombinera neurala nätverk med telemetrisk data. Framtida arbete inom området kan komma att leda till rättvisare priser för kunder vad gäller försäkringspremier.

Acknowledgements

First of all, we would like to present our gratitude to Paydrive AB as a whole for allowing us to learn more about the exciting world of insurance and develop our skills in mathematics further while providing an exciting and inviting place of work. The whole team at Paydrive AB has been very helpful in our day-to-day task, and have been a constant source of support, with a special thanks to Devina Choi for her help with cloud solutions and infrastructure.

Furthermore, we would like to personally thank Andreas Broström, Ossian Hättestrand, and Sepehr Zolfeghari at Paydrive for their incredible help and support throughout this thesis. As is the case with many theses, there have been hurdles and hiccups in the process that have been easily solved through their support and knowledge.

We would also like to extend our deepest thanks and gratitude to our supervisor at KTH, Martina Scolamiero, for her wonderful support and vast knowledge. Large amounts of data have been analyzed during this thesis and Martina's deep knowledge of data anlysis have been incredibly helpful in interpreting and visualizing the data.

Also special thanks to Pippi[1].

[1] Pippi: The office dog who has been a constant source of moral support to us.

Contents

1 Introduction

Paydrive AB is a Swedish auto insurance brokerage firm based in Stockholm. Their goal is to provide their customers with insurance premiums that relate to how good of a driver you are. Whilst other insurance companies base their premiums on more traditional parameters such as the drivers age, what car you drive (horsepower, weight, make etc.) and postal code, Paydrive also incorporate what is known as telemetry data. In practice this means that a small device is installed in your car which reports on your position, speed, how aggressively you break or accelerate as well as if you're speeding. This allows Paydrive to generate monthly costs based on two different parameters, one being customer data containing demographic parameters, customer data, and historical claim data on the customer and the other one being telemetry data that senses how you drive. This leads to good and safe drivers having lower premium costs with Paydrive than they would with a different insurance company.

1.1 Background

Mathematical and statistical methods for determining the risk of insuring an individual or an object have been integral in the insurance industry for decades, as an accurate assessment of risk is necessary for the operation of an insurance company.

Traditionally insurance providers have relied on data provided by the customer such as historical claims data, demographic data, and more. However, in recent years the technology of connected devices has allowed insurers within the auto insurance industry to collect data directly from a device in the car. The type of data gathered from devices such as this is called telemetry data and provides a range of information about how and when the car is driven, which can help the insurer deliver more fair and transparent insurance.

The use of telemetry data along with modern mathematical and statistical methods for prediction opens up an opportunity for fairer pricing and more accurate risk assessment of the customer, making it a highly relevant topic of research for insurers.

Previous work in the sector has been centered around generalized linear models [1] and did not explore the concept of combining neural networks with telemetry data. We hope that by extending this work and using previous results as a baseline we will be able to provide an even more fair price point for consumers based on historical driving patterns. This raises some valid issues that need to be dealt with outlined by [2], as it would appear that neural networks tend to be too volatile to be fully utilized in conjunction with telemetry data. We try to overcome this problem by employing statistical methods for lowering the variance and other problems that may arise.

1.2 Problem statement

The goal of the thesis is to predict the expected claim amount, c, of a customer each month based on data from the previous month as well as the previous three months. Let r be the risk of a customer having a claim each month and s the severity of the claims. The equation for determining the expected claim amount is then given by

$$c = r \cdot s. \tag{1}$$

As these parameters are not known the goal of the thesis is to train two models approximating r and s as very complex functions relating the data to a single parameter. The approximations can be used to determine the expected claim amount of a customer, which in turn can be used to set a premium that is to be paid by the customer. This thesis is limited to the approximation of r and s, with the premium set based on these being a topic for discussion within the company.

1.3 Outline

In section 2 relevant theory and mathematical background for the models and methods used in the thesis are presented. Next in section 3 the method of transforming the data from raw telemetry into features appropriate for training models are presented along with an analysis of the data needed to facilitate model selection and choices within the models such as hyperparameters in the case of neural networks and link-functions in the case of GLM:s. Once the analysis of the data is completed the method of training the models is presented in section 4. Once the model has been implemented in practice and have been trained the results are presented in section 5. Lastly, a discussion regarding the results as well as choices made in the theses as a whole along with suggestions for future work are presented in section 6.

2 Theory

In this section relevant theory for the thesis is presented to facilitate the methods used with the goal of analysing the data set as well as predicting the expected claim of a customer each month. We also set the stage for the reader to have some mathematical understanding of concepts which are useful when understanding certain aspects of this thesis.

Correlation

Correlation, or dependence in statistical theory is any statistical relationship between two random variables. However we typically relate correlation to the linear relationship between variables. There are dozens of ways to measure correlation, one of the more popular being Spearman's rank correlation coefficient. Given two sets of real-valued numbers, $X = (x_1, x_2, ..., x_k)$ and $Y = (y_1, y_2, ..., y_k)$ each set is given a ranking from 1 to k according to their size separately, denoted as $\text{rank}(x_i)$ and $\text{rank}(y_i)$. With this definition, let:

$$r = \text{rank}(x_i) - \text{rank}(y_i), \quad i = 1, 2, ..., n \tag{2}$$

$$R := \sum_{i=1}^{k} r_i^2 \tag{3}$$

the correlation coefficient is then calculated as:

$$r_s := 1 - \frac{6R}{n(n^2 - 1)} \tag{4}$$

for $-1 \leq r_s \leq 1$

2.1 Generalized linear models

Before introducing generalized linear models, a simple general linear regression model is introduced to define important basic concepts that are then easily transferred to the more complex generalized linear model.

Linear regression
Given data $\{y_i, x_{1,i}, x_{2,i}, ...x_{k,i}\}_{i=1}^{n}$ we assume that the relationship between the dependent variable y and the predictors x is linear. The relationship can thus be modeled with a random error ε as

$$y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + ... + \beta_k x_{k,i} + \varepsilon_i = \mathbf{x}_i^T \boldsymbol{\beta} + \varepsilon_i. \tag{5}$$

Utilising matrix notations, the equation for the whole data set $i = 1...n$ can be rewritten as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{6}$$

The parameters $\boldsymbol{\beta}$ are not known and are approximated the least-squares method. Let $y_i \approx \boldsymbol{x_i}^T \beta$. Then the least square method is given by

$$\hat{\boldsymbol{\beta}} = \underset{\hat{\boldsymbol{\beta}}}{\text{argmin}} \sum_{i=1}^{n} L(y_i, \boldsymbol{x}_i^T \boldsymbol{\beta}), \tag{7}$$

where the loss function L is given by

$$L(y_i, \boldsymbol{x}_i^T \boldsymbol{\beta}) = (\boldsymbol{x}_i^T \boldsymbol{\beta} - y_i)^2 \tag{8}$$

for each i and by,

$$L(\boldsymbol{Y}, \boldsymbol{X}\boldsymbol{\beta}) = ||\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{Y}||^2 \tag{9}$$

in the matrix notation. The optimal parameters $\hat{\boldsymbol{\beta}}$ are then given by

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}. \tag{10}$$

Generalized linear models

Generalized linear models (GLM) are a class of models in which the response variable $\mathbf{Y}$, which we want to model is assumed to be from an exponential family with mean μ. As the name suggests, a GLM is a generalized version of several regression models and effectively combines models such as linear regression and logistic regression, etc.

The GLM contains three different components. The first component is the systematic component, which is a linear predictor. The systematic component is given by:

$$\eta = \mathbf{X}\beta \tag{11}$$

which can be rewritten on the form

$$\eta_i = \beta_0 + \beta_1 x_{1,i} + ..\beta_k x_{k,1}. \tag{12}$$

Just as in the linear regression model, this models the dependent variable as a function of the predictors and parameters.

The generalized linear model also contain a random component, which is given by a distribution for $\mathbf{Y}$ from the exponential family, which has the probability mass/density function given by

$$f_x(y) = e^{\frac{yx - b(x)}{a(\phi)} + c(y,\phi)}, \tag{13}$$

where a,b and c are arbitrary functions and ϕ a scale parameter.
The last component of the GLM is the link function, $g(\mu) = \mathbf{X}\beta$, which specifies the relation between the random and systematic components and fulfills

$$E(\mathbf{Y}|\mathbf{X}) = \mu = g^{-1}(\eta), \tag{14}$$

and

$$g^{-1}(\eta) = g^{-1}(g(\mu)) = \mu. \tag{15}$$

The link function thereby relates the mean of the response variable $\boldsymbol{Y}$ to the linear predictor. The link function is chosen depending on the structure and characteristics of the data. For example, if the response is normally distributed, the link function would simply be $g(\mu) = \mu$ and we would therefore get a general linear regression model. With these three components, the GLM has the flexibility that there are no assumptions of normality on the dependent variable $\mathbf{Y}$ and there is no assumption of linearity between the response variable and the explanatory variables, due to the link function. Some typical link functions with their associated distributions are as follows:

Distribution	Link Name	Link function
Normal	Identity	$X\beta = \mu$
Exponential	Negative inverse	$X\beta = -\mu^{-1}$
Inverse Gaussian	Inverse squared	$X\beta = \mu^{-2}$
Poisson	Log	$X\beta = \ln \mu$
Bernoulli	Logit	$X\beta = \ln \frac{\mu}{1-\mu}$

2.2 Neural networks

2.2.1 Structure

The concept of a neural network is as the name suggest, based on the human brain and the way information passes through it. An artificial neural network is composed of nodes and links between them as shown in figure 1 below, where all nodes are connected through these links.

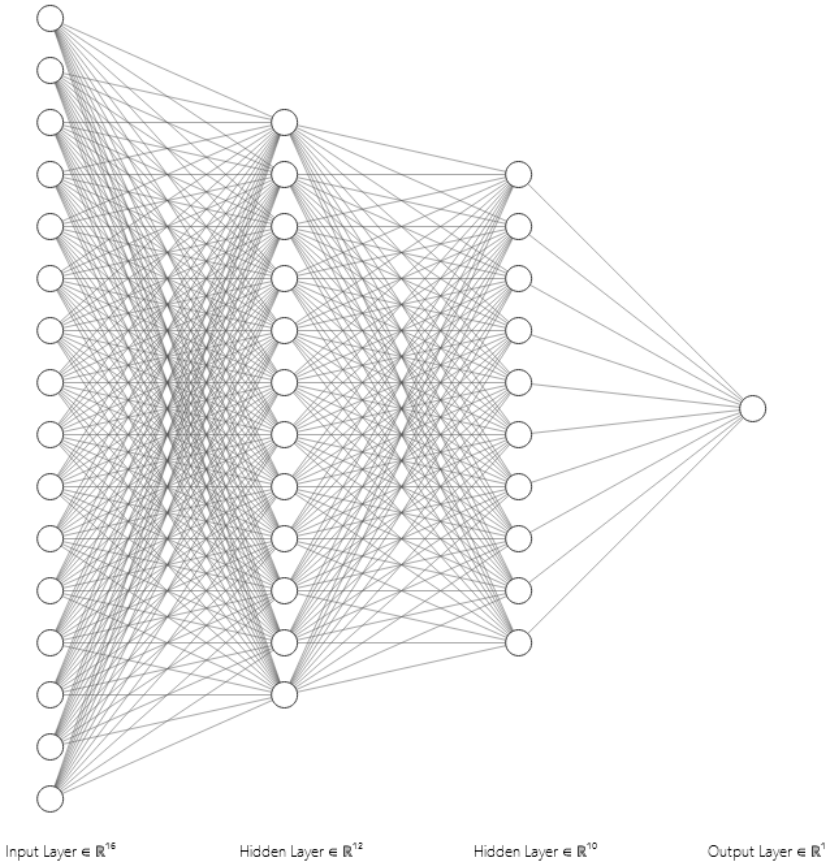


Figure 1: Example neural network

All of the links in the neural network have a weight associated with them. As each individual node can be connected to multiple previous nodes, the weight associated with the links can intuitively be seen as a way of weighting the importance of information passed from previous nodes to the current node. In addition to the weights, another parameter of the network is the activation function. Each node has an activation function, which determines what information passes forward from the nodes. Similar to the weights, this activation function can be seen as a way to determine if the information passed from previous nodes to the current node is going to be passed further

forward in the network. Each node also has a bias parameter associated with it, which can be seen as a parameter that gives a certain importance, or bias, to the node itself.

Because of the structure of the network, we can input some information to the first layer of the network, called the input layer, through the hidden layers until we reach the last layer called the output layer. The weights, biases, and activation functions will determine what the output layer produces. By changing these weights by various algorithms, the network can be trained iteratively by changing these weights until the desired output is produced. Several algorithms for training the network exist, such as,

- **Gradient Descent:** This is the most commonly used algorithm for training neural networks. The basic idea behind this algorithm is to iteratively adjust the weights and biases of the network in the direction of the steepest descent of the loss function. The loss function measures the difference between the predicted output and the actual output of the network for a given input.
- **Backpropagation:** This algorithm is used to calculate the gradients of the loss function with respect to the weights and biases of the network. These gradients are then used to update the weights and biases in the direction of minimizing the loss function. Backpropagation is a type of gradient descent algorithm.
- **Adam:** This is an adaptive learning rate optimization algorithm that is designed to work well with stochastic gradient descent. It uses a moving average of the gradient and the squared gradient to adaptively adjust the learning rate for each weight and bias.
- **RMSProp:** This is another adaptive learning rate optimization algorithm that uses a moving average of the squared gradients to adaptively adjust the learning rate for each weight and bias. It is similar to Adam but does not use the moving average of the gradient.

In a more mathematically rigorous definition, let l denote the l:th layer of the network. Let a node in the l:th layer be denoted by j and a node in the previous layer, $l-1$, be denoted by k. The weight between a node in the current layer and a node in the previous layer can then be denoted by $\omega^l_{j,k}$. In addition to the weights, each node also has a bias associated with it, denoted by b^l_j. We let the output of a node be denoted by a^l_j which gives the following expression for the output of a node based on the output from all previous nodes:

$$a^l_j = \sigma(\sum_k \omega^l_{j,k} a^{l-1}_k + b^l_j), \tag{16}$$

where $\sigma(x)$ is the activation function. The parameters of the network are illustrated in figure x below.

The expression from equation 16 can be rewritten in a more compact matrix form by introducing $z^l = \omega^l a^{l-1} + b^l$ where ω^l is a vector containing all the weights of the nodes in the $l : th$ layer, a^{l-1} is a vector containing all of the outputs from the previous node and b^l is a vector containing all the biases. The output of a node can then be written as $a^l = \sigma(z_l)$.

Another important concept of a neural network is the loss function, which is used as a quantitative measure of the performance of the network. If we let $x^{in} \in \mathbb{R}^m$ be the input to the network, and

$x^{out} \in \mathbb{R}^n$ the output of the network, the loss function can be defined as

$$L(x^{out}, x) = \sum_{i=1}^{n} (x_i^{out} - x_i)^2. \tag{17}$$

where $x \in \mathbb{R}^n$ is the desired output of the network. The loss function as defined above is the mean squared error function, but other suitable loss functions exist and can be chosen by the constructor of the network. There are dozens of different loss functions, all specific to different types of data. This project has attempted to work with primarily one of these:

- **Binary Cross-Entropy**: Binary cross-entropy, also called logarithmic loss or log loss, is a metric used to evaluate how well a model is performing on a binary classification task. It measures the degree to which the model's predicted probability distribution deviates from the true probability distribution of the labels. Low cross-entropy loss values indicate that the model is accurately predicting the labels, and correspondingly, the model's accuracy is high. The formula for binary cross-entropy is given by taking the negative logarithm of the predicted likelihood and multiplying it by -1. Binary cross-entropy is an excellent tool to deal with highly biased data since it penalizes false positives and false negatives severely.
- **Poisson loss** is a type of loss function that measures the distance between the predicted and actual values granted that the response variable can be modeled after a Poisson distribution.
- **Gamma loss**, similarly to Poisson loss measures the distance between predicted values and actual values granted that the response variable is modelled after the gamma distribution.

2.2.2 Backpropagation algorithm

For the training of a neural network, a popular method is the backpropagation (BP) algorithm. It is a supervised learning method which means it requires labeled input-output pairs to train the network. The goal of the algorithm is to optimize the weights in the model in order to minimize the difference between our expected output and the predicted output. It consists of two general steps;

- Forward propagation: In this step, the input is passed through the network to produce an output. The input is multiplied by the weights of each connection in the network and passed through an activation function to produce an output. This process is repeated for each layer in the network.
- Backward propagation: In this step, the error between the predicted output and the true output is calculated and used to update the weights of the network. The error is propagated back through the network, starting from the output layer and working backward toward the input layer. The weights are updated using an optimization algorithm, such as gradient descent.

Algorithm 1 Backwards propagation algorithm

Set K
Set Threshold
for $k \leftarrow 1$ to K **do**
  $a^k = g^k(W^k a^{k-1} + b^k)$
  $\delta^k = (a^k - y)g'^k z^k$ {determine the error at the output layer}
end for
IDX= $\{K-1, K-2, K-3, ..., 2, 1\}$
for k in IDX **do**
  $\delta^k = (W^{k+1})^T \delta^{k+1} g'^k z^k$ {determine the error at each layer}
  $b^k = b^k - \alpha\delta^k$ {update the bias}
  $\epsilon = \delta^k$
  if $\epsilon \geq$ Threshold **then**
    break;
  end if
end for

2.2.3 Stochastic Gradient Descent

Often we want to equip the backward propagation algorithm with some kind of optimization tool, one very popular such tool is the Stochastic Gradient Descent (and regular Gradient Descent, not discussed in this thesis). Stochastic Gradient Descent (SGD) is an iterative optimization algorithm used in machine learning and deep learning to minimize the loss function of a neural network. The basic principle of SGD is to update the model parameters (weights and biases) in the opposite direction of the gradient of the loss function with respect to the parameters. The algorithm starts with randomly initialized parameters and then iteratively performs the following steps for each batch of training samples:

- Compute the gradient of the loss function with respect to the parameters for the current batch.
- Update the parameters in the direction of the negative gradient by taking a small step in that direction, multiplied by a learning rate hyperparameter.
- Repeat until convergence or a predefined number of iterations is reached.

The learning rate determines the size of the step taken in each iteration and is a crucial hyperparameter that affects the performance of the algorithm. If the learning rate is too small, the convergence will be slow, and if it is too large, the algorithm may overshoot the minimum of the loss function and diverge.

SGD is called "stochastic" because it updates the parameters using only a subset of the training data (i.e., a batch) at each iteration. This makes it more computationally efficient than traditional gradient descent, which computes the gradient using the entire training set at each iteration. However, the stochastic nature of SGD means that the algorithm may converge to a sub optimal solution or get stuck in a local minimum of the loss function. To overcome this, various techniques such as momentum, adaptive learning rate, and regularization are used in practice.

2.3 Estimation

Let $\{y_i\}_{i=1}^n$ be given data. In this section, several alternatives for approximating different quantities are presented

Least squares estimators
Let $L(y_i, \theta)$ be a loss function that determines the squared difference between an actual observation y_i and an approximation $\hat{y}_i(\theta)$. For a least square estimator we seek to find the best choice of the parameter θ, such that the loss function is minimized across all

$$\hat{\theta} = \underset{\hat{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \theta) = \underset{\hat{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \hat{y}_i(\theta))^2 \tag{18}$$

Maximum likelihood estimator
For the maximum likelihood estimator, we want to maximize the likelihood-function, which is the probability of measuring the response y_i given some underlying parameter θ. The likelihood-function is given by

$$L(y_i, \theta) = P(y_i|\theta) \tag{19}$$

The maximization problem is then given by

$$\underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} L(y_i, \theta) \tag{20}$$

To simplify we often look at the log-likelihood, as the logarithm of a monotonically increasing function has its maximum at the same place as the original location. Furthermore, multiplicative constants (additive if log-likelihood) not depending on θ can be removed, as the only changes the magnitude of the maximum and not the location.

Maximum a-posteriori estimator
For maximum a-posteriori estimator, we have some prior distribution of the parameter θ, $P(\theta)$. For the MAP-estimator we want to maximize the posterior probability of θ given the observations and the prior distribution. The maximization problem is stated as follows:

$$\underset{\theta}{\operatorname{argmax}} P(\theta|y_i) \tag{21}$$

Utilizing Bayes rule and removing multiplicative constants the maximization problem is given by

$$\underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} P(y_i|\theta)P(\theta) \tag{22}$$

2.4 Dimensionality reduction

Linear principal component analysis [3] is based on finding a lower dimensional linear subspace of the feature space occupied by the data, that captures the most important components of the data, as implied by the name.

The first step in the principal component analysis is the normalization of the data. This will allow each feature to contribute to the analysis equally, and not let features with significantly higher mean dominate.

Let X be a data set containing n samples with m features each, structured as

$$X = \{x_i^j\}_{i,j=1}^{n,m}, \tag{23}$$

where $x_{i,j}$ is a datapoint containing the i:th sample of the j:th feature. Each feature j in the data set has a corresponding mean of μ_j and a standard deviation of σ_j. The normalization is done as follows for each sample i of the feature j:

$$x_i^j = \frac{x_i^j - \mu_j}{\sigma_j}. \tag{24}$$

The normalized datapoints are structured in a new data set as

$$X^{Normalized} = \{x_i^j\}_{i,j=1}^{n,m}. \tag{25}$$

The second step in the principal component analysis is to determine the correlation matrix. The correlation between the variables is determined by Spearman's rank correlation coefficient as presented in section 2. The data is considered to be a realization of random variables and thus the correlation between two random variables i and j in the feature set are denoted by $r_{i,j}$. The correlation matrix is structured in the following manner:

$$C = \{r_{i,j}\}_{i,j=1}^{n}. \tag{26}$$

Once that the correlation matrix has been determined, the next step is to determine the principal components. This is done by determining the eigenvectors and corresponding eigenvalues of the correlation matrix, i.e the vectors v fulfilling

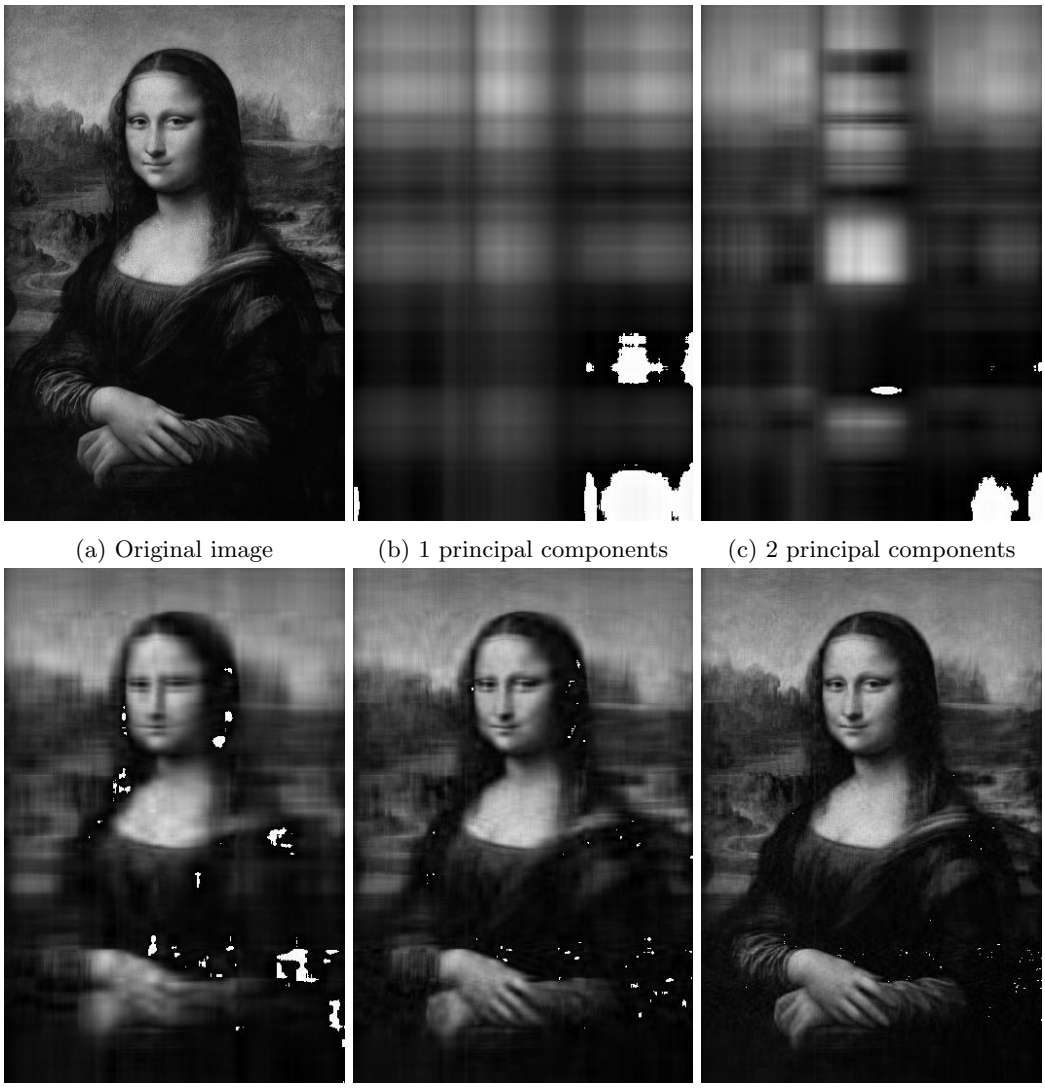
$$(C - \lambda I)v = 0, \tag{27}$$

where λ is the corresponding eigenvalues. The eigenvectors of the correlation matrix correspond to the axes with the greatest variance, and thus the greatest information. The corresponding eigenvalues give a measure of the amount of variance in each axis given by the eigenvectors. Thus, by ranking the eigenvectors after the eigenvalues in decreasing order we will find the axes that contain the most information. The principal components mentioned are simply these axes. The The last step will be to determine which of the components we want to keep and create a new feature set F of size $\mathbb{R}^{k\times n}$ from this where k are the k:th first components, and recast the original data to the axes provided by the new feature vector using

$$X^{PCA} = F^T X^{Normalized}, \tag{28}$$

where X^{PCA} is in $\mathbb{R}^{k\times n}$ and $X^{Normalized}$ is in $\mathbb{R}^{n\times m}$. By utilizing this approach of Linear PCA we have thus reduced the data set into a k-dimensional subspace while retaining much of the information.

To better exemplify PCA, an example of the PCA algorithm applied on a greyscale image of size 530×350 is shown below. Each pixel in the image was assigned an intensity value. Each pixel is then seen as a sample, and each possible intensity value is seen as a feature. This will yield a matrix X of size $(530 \times 350) \times (530 \times 350)$. The PCA algoirithm as defined above was then applied to this matrix for different numbers of principle components and converted back into an image, as can be seen in figure 2 below.



(a) Original image (b) 1 principal components (c) 2 principal components

(d) 10 principal components (e) 20 principal components (f) 50 principal components

Figure 2: Original image and five different reconstructions.

As we can see from the figure above, even with only one principal component we can see some resemblance of the original picture, which increases as the number of principal components increases.

Kernel PCA

The method for kernel PCA [4] is similar to the linear variant, but does not make the assumption that the features can be recast into a linear lower dimensional subspace, which allows for a nonlinear subspace which is often more suitable as many real-world data sets are inherently nonlinear. The approach is to first cast the data into a higher-dimensional space and then extract the principal components of that space. The principle of kernel PCA relies on the notion of mapping the points to a n-dimensional space with some arbitrary function ϕ. By doing these mappings for all datapoints $\{x_i\}_{i=1}^n$ we can create a hyperplane separating these points. The mapping will create linearly independent vectors, so the method of determining the eigendecomposition explicitly is

not viable. Instead, a kernel will be utilized. By using a kernel we will not have to explicitly determine the mapping of ϕ. We assume that we have a kernel that is on the form of

$$k(x, y) = (\phi(x), \phi(y)) = \phi(x)^T \phi(y). \tag{29}$$

Several types of kernels exist, but for the purposes of explanation and illustration let the kernel be a Gaussian kernel defined by

$$k(x, y) = \exp(\frac{-||x - y||^2}{2\sigma^2}), \tag{30}$$

where x and y are two points. By applying this kernel the kernel trick is utilized, whereby we never explicitly determine the eigendecomposition of the correlation matrix and the principal components, but instead directly determine the projection of the components onto the feature space. As with the linear variant, the first step is to normalize the data set yielding a normalized dataset $X^{Normalized}$. In the second step we let

$$K = k(x, y) = (\phi(x), \phi(y)) = \phi(x)^T \phi(y) \tag{31}$$

be a $\mathbb{R}^{n \times n}$ matrix representing the inner dot product space. We then centralize the kernel matrix, which is synonymous with normalizing the matrix, utilizing

$$K' = K - \mathbf{1}_n K - K\mathbf{1}_n - \mathbf{1}_n K \mathbf{1}_n, \tag{32}$$

wher $\mathbf{1}_n$ is $\mathbb{R}^{n \times n}$ a matrix where all elements have the value $1/n$.

We will then determine the eigendecomposition of this matrix. As before, we create a new feature set F from the eigenvectors of size k where k are the number of dimensions of the reduced data. We then recast the original data onto the new space using

$$X^{KPCA} = F^T X^{Normalized}, \tag{33}$$

reducing the data into a k-dimensional space.

2.5 Clustering

When doing exploratory data analysis another powerful tool is clustering algorithms. What they allow us to do is visualize data in a topological framework, using certain metrics, features and similarities between data points to group these together. Since we have a lot of data with many interesting features we will be utilizing the K-means clustering algorithm [5] as well as hierarchical clustering.

Given a set of observations, $(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, ..., \mathbf{x_i}$ where each observation is itself a d-dimensional real valued vector we partition the n observations into k sets (where $k \leq n$) $\mathbf{S} = \{S_1, S_2, S_3, ..., S_k\}$ in order to minimize the variance. Expressed formally the goal here is to find

$$\underset{S}{argmin} \sum_{i=1}^{k} \sum_{x \in S_i} \|\mathbf{x} - \mu_\mathbf{i}\|^2 = \underset{S}{argmin} \sum_{i=1}^{k} \|S_i\| \text{ Var } (S_i) \tag{34}$$

where μ_i is the so called cluster centroid (mean). The algorithm then follows

Algorithm 2 K-Means Clustering

Input : $X = x_1, x_2, ..., x_i$, the data set to be clustered
Input : k, the number of clusters
Output: $S = S_1, S_2, ..., S_k$, the set of clusters

Initialize: Randomly choose k data points as initial centroids, $c_1, c_2, ..., c_k$
repeat
  Cluster Assignment: Assign each data point x_i to the nearest centroid s_j $C_j = \{x_i : ||x_i - c_j||^2 \leq ||x_i - c_l||^2 \forall l, 1 \leq l \leq k\}$

  Centroid Update: Update each centroid c_j as the mean of its assigned data points $c_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$
until *convergence*;
return C

2.6 Hierarchical Clustering

Another type of clustering algorithm is Hierarchical clustering which is a much simpler approach. Hierarchical algorithms can be either **agglomerative** (bottom-up) or **divisive** (top-down)[5]. Agglomerative algorithms start out by treating each individual data point as it's own cluster and then successively merging them into larger clusters based on some criteria. Divisive algorithms work as they sound, by taking the entire data set as a starting cluster, successively splitting and partitioning it into smaller clusters. In either case we utilize what is known as a distance matrix where the distance between each point respectively is calculated using some formulae, either the Manhattan distance function which is simply

$$d = \sum_{i=1}^{n} |x_i - y_i| \tag{35}$$

or by the Euclidean distance function

$$d = \sqrt{\sum_{j=1}^{n} (x_j - y_j)^2} \tag{36}$$

But how do we actually decide which clusters to merge/split? Typically it comes down to the calculated distance being used as a divisor depending on if we want the maximum distance, minimum distance or the mean distance. We can also look at other interesting results from the clustering, such as variance increase, probabilities of candidate clusters being spawned from the same distribution etc.

2.7 Resampling & Transformation Techniques

Resampling techniques are often applied in situations in which the data set is heavily skewed towards one class. Data representing a customers insurance claims is often heavily skewed towards zero, with only a few occurances of a claim, as most customers are not involved in an accident. This could potentially lead to worse results when applying predictive models as discussed in [6]. One technique for combating this is random oversampling [7]. In it's essence the technique is rather simple, and relies on duplication of instances of the minority class, with the amount of duplicates adjustable to fit the problem at hand and as such does not warrant any further explanation in this section.

Yeo-Johnson transformation
When working with any type of machine learning it is common practice to attempt to normalize the data to have univariate results. One such technique is the Yeo-Johnson transformation [8] which is an extension of the Box-Cox method. The basisc idea is that we have data which has some heteroscedasticity, that is we have data where the variance is unequal which we want to get rid of.

$$\psi(y,\lambda) = \begin{cases} \dfrac{(y+1)^{\lambda}-1}{\lambda} & y \geq 0 \text{ and } \lambda \neq 0, \\ \log(y+1) & y \geq 0 \text{ and } \lambda = 0, \\ -\dfrac{(-y+1)^{2-\lambda}-1}{2-\lambda} & y < 0 \text{ and } \lambda \neq 2, \\ -\log(-y+1) & y < 0, \lambda = 2. \end{cases}$$

Figure 3: Yeo-Johnson technique

2.8 Evaluation Metrics

2.8.1 Classification Metrics

Confusion Matrix
The confusion matrix is a table which summarizes the amount of true positives, false positives, true negatives and false negatives. It creates a system of evaluation based solely on the results of a model which makes it a prime candidate for most classification problems. The table looks like:

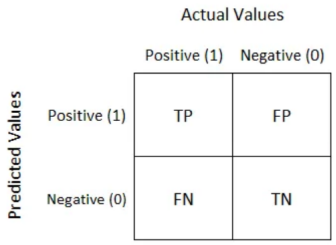


Figure 4: Confusion Matrix

From this table all the following classification metrics can be calculated.

Accuracy
The most straight-forward and easily interpretable metric for evaluating a classification model is the classification accuracy. The accuracy is in essence the fraction of correctly classified samples to the total amount of samples

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}. \tag{37}$$

Gini-coefficient
As will be apparent in chapter 3 our data is zero-heavy, as is often the case with insurance claim data. This will make the prediction accuracy somewhat less of a good measure. Let y be some binary target variable containing some amount of samples, with each sample being 0 or 1. If 90%

of the samples of the target variable is zero and we create a model that always predicts 0 the prediction accuracy would also be 90%. This might sound like a good result, but in the case with very zero heavy data the classification for the remaining 10% is more important, and we would expect to see a model with over 90% accuracy if the training has been successful. One way to better evaluate models related to zero-heavy data is the Gini-coefficient, not to be confused with the impurity index of the same name often used when evaluating tree-based models.
The Gini-coefficient can be determined as follows:

$$Gini = 2AUC - 1, \tag{38}$$

where AUC is the area under ROC curve, which in turn is a graph plotting the true positive rate against the false positive rate. The Gini-coefficient takes values between 0 and 1, with 0 representing a model that performs as well as a random choice while 1 would represent a perfect model.

F_1-score
The F-score or F_1-score is a metric that is used with binary classification and it is a measure of the accuracy of a test. It is defined as the harmonic mean of the precision and recall:

$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{39}$$

where TP is the total amount of true positives, FP is the number of false positives, and FN is the number of false negatives.

ROC - Curve
A ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at different threshold values. The TPR is the proportion of true positive predictions (i.e., cases where the model predicted a positive outcome and the actual outcome was positive) out of all positive cases, while the FPR is the proportion of false positive predictions (i.e., cases where the model predicted a positive outcome but the actual outcome was negative) out of all negative cases.
The ROC curve is created by varying the classification threshold of the model from 0 to 1 and calculating the corresponding TPR and FPR values. The area under the ROC curve (AUC) is a measure of the overall performance of the model. A model with an AUC of 1 indicates perfect classification performance, while a model with an AUC of 0.5 indicates random guessing.

2.8.2 Regression Metrics

The metrics presented for classification purposes are not suitable for regression, as there exist one correct number but the model could in theory provide any number in a continuous range. Due to this, other metrics are used to evaluate the models performance. Let y be some true target variable containing n samples and $\hat{y}$ be the predicted target variable with n samples.

Mean Absolute Error
The mean absolute error provides an intuitive measure of the error in the prediction and can be seen as the average distance between predictions and target variables and is defined as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{40}$$

where y_i and $\hat{y}_i$ is the i:th samples of the true and predicted value respectively.

Mean Square Error
The mean squared error provides is a similar metric to the mean absolute error and is related

to the distance between true and predicted values, but penalizes larger deviations from the true value.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{41}$$

R-Squared

R-squared (R^2) is a measurement of how well the model fits the data taking values in the range $[0, 1]$ with a higher value representing a better fit. The formula for determining R^2 is as follows:

$$R^2 = 1 - \frac{SSres}{SStot} \tag{42}$$

where $SSres = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ and $SStot = \sum_{i=1}^{n}(y_i - \bar{y})^2$ where $\bar{y}$ is the mean of the true values.

3 Data Analysis

3.1 Data set

The data set can be considered to consist of two parts, telemetry data, and customer data. The customer data consists of what can be seen as traditional insurance data such as the driver's age, the make of the car, etc. The telemetry data consists of measurements taken by a device installed in the OBD port of the customer's car. The raw data from the device is collected multiple times per minute. Through feature engineering, the data is aggregated into a monthly form, which is then combined with the customer data.

3.1.1 Feature Engineering

In this section, we aim to improve the construction of certain features in the data set to make them more suitable for implementation using the raw telemetry data provided by the device. Specifically, we need to structure the data so that the model can use past data to determine the new premium, which is calculated every three or 12 months. There are several options for structuring the data, such as using all previous data or selecting a subset of months when setting the premium. In the method for feature engineering, the data is aggregated on a monthly basis in such a manner as to easily facilitate further aggregation into data spanning several months.
It is important to note that these features represent risk factors that the model considers when determining the final premium. Therefore, the selection and construction of features become critical in ensuring that the model is explainable, especially when customers want to know why their calculated driving score was a particular value. Overall, this section focuses on optimizing the data structure to enhance the model's interpretability and transparency.

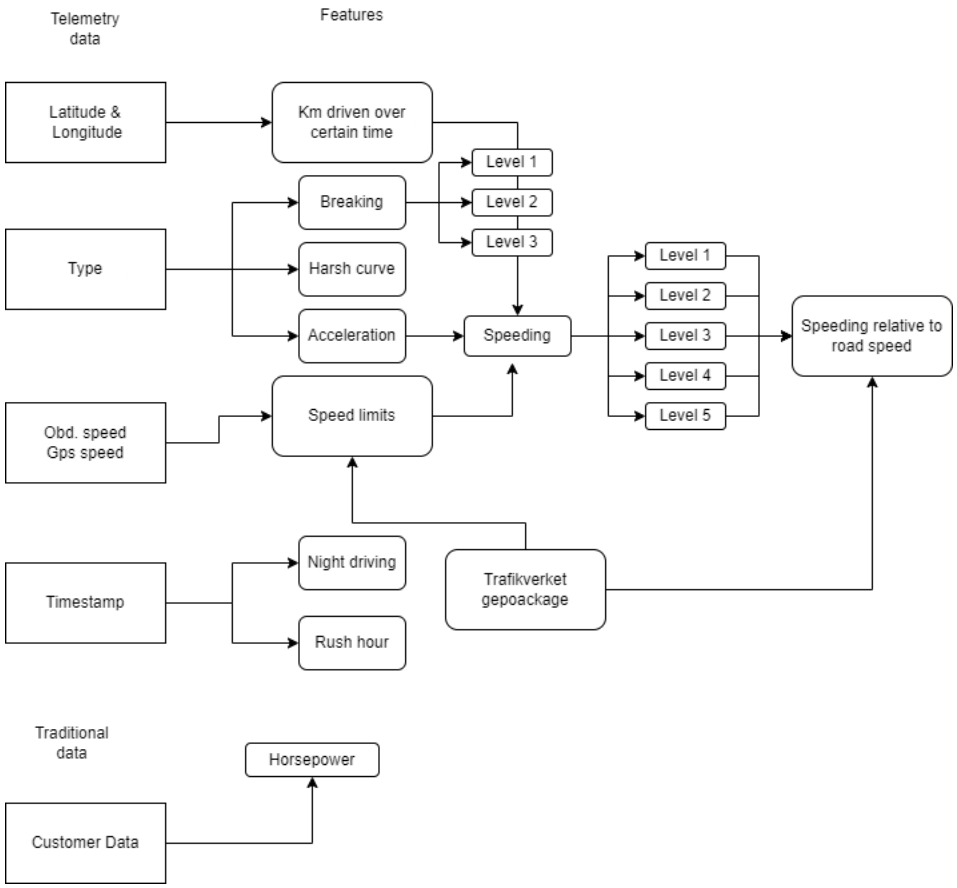


Figure 5: Feature relationships (Illustrative)

Figure 5 provides an illustrative overview of the current data workflow. Telemetry data and traditional customer data are ingested into a cloud platform on the left side of the diagram. Using the current GPS coordinates and Trafikverkets geo-package, we can determine the road the customer has been driving on and the speed limit on that road with sub-1 meter precision. We combine this information with the OBD speed and GPS speed to create new features that indicate how much a customer exceeds the legal speed limit. Additionally, we can determine the level of speeding, which indicates a more or less risky driving behavior, by comparing the customer's speed with the speed limit. For example, driving 60 km/h on a road with a 30 km/h speed limit is probably riskier than driving 140 km/h on a road with a 110 km/h speed limit.

The feature named 'Type' is the most interesting data point. The device that is plugged directly into the car senses g-forces and directions, which allows it to automatically label certain events as 'Breaking', 'Harsh Curve', or 'Acceleration'. Previous iterations of the model have shown that the 'Breaking' type is one of the most influential factors when calculating a customer's premium. However, we believe that the current breaking levels are not optimal, and therefore we have adjusted them to account for a different range of g-forces. In similarity to the brake levels, new values and ranges for acceleration and sharp turn event levels have been set.

In addition to this, using the timestamp we also classify the point as night driving or peak-hour

driving if the customer is driving at certain set time intervals.

3.1.2 Data wrangling and cleaning

A major challenge when working with telemetry data is the raw amount of data that needs to be processed. At the time of writing a year of data consists of roughly 2 billion data points, all of which must be taken into consideration and handled properly. Going from raw data ingestion to finished trip tables for every customer is thus the biggest challenge of this thesis from a purely computational viewpoint.

3.1.3 Aggregated data set

The aggregated data set consists of the following variables:

Data types		
Name:	Type:	Explanation
id	int	Id of the customer
nightdriving	float	Fraction of driving during night events during month and total events
peakhour	float	Fraction of driving during peak-hour events during month and total events
breaklevel	float	Fraction of different braking-severity levels events during month and total events
turnlevel	float	Fraction of different turn-severity levels events
acclevel	float	Fraction of different acceleration-severity levels events
speedinglevels	float	Fraction of different speeding-severity levels at different speed-limit roads
modelyear	float	Which year is the car model from
directimport	bool	Is the car imported
make	string	The make of the car
effect	float	The effect of the car
weight	float	Weight of the car
type	string	Type of car (sedan, combi, etc.)
postcode	float	Postcode of the customer
distributor	float	Sales channel
fueltype	float	Type of fuel that the car uses
driverage	float	Age of the driver
driverlicenseyear	float	Age of driver when license obtained
carage	float	Age of the car
timeowned	float	Time that the custoemr has owned the car
korkortsar	float	Time that the customer has had a driver license
Response variables		
claimnum	int	Amount of claims for the customer that month
claimcost	float	Cost of the claims that the customer has incurred that month

Table 1: Variables represented in the data set

The variables of a level type are categorical, i.e. they represent level 0, level 1, etc. In the data-set, each level is represented as a separate column. The number of columns in the final data set is 67 columns.

In the results shown below data have been aggregated based on one month of customer data. For confidentiality reasons, the name of the variables and columns in the following plots are not presented to the viewer, and are instead referred to as v_1, v_2,... and 1,2,... in no particular order.

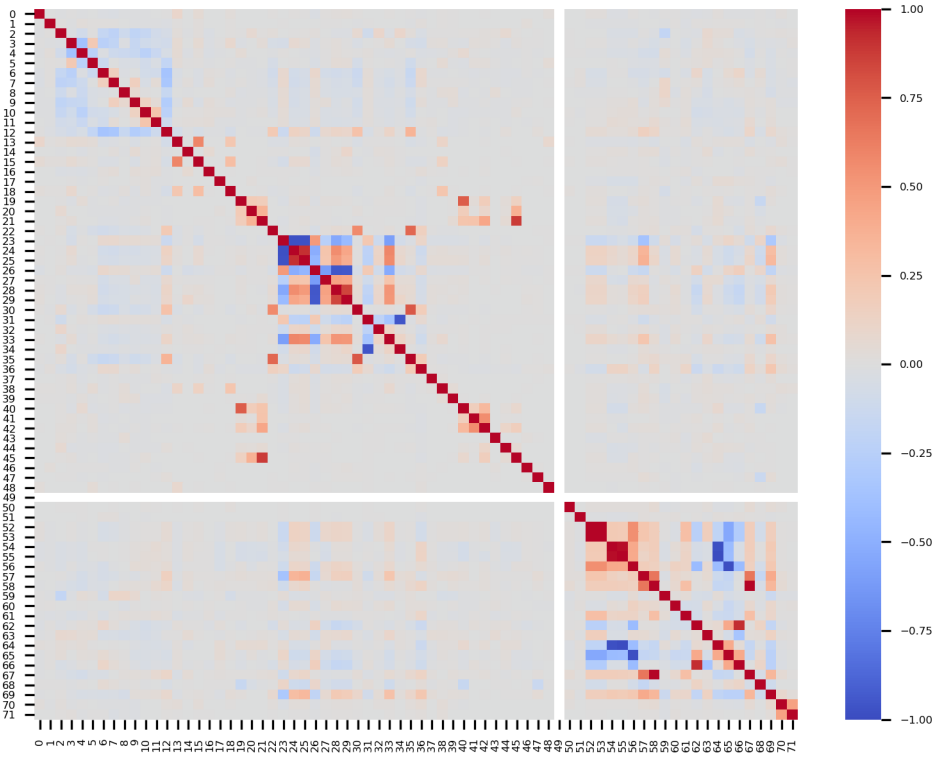


Figure 6: Correlation heat map of variables

A plot of the correlation between all the columns can be seen above in figure 6 where the x and y labels represent the indices of the columns. In this figure, we can see some strongly correlated behavior for some of the columns such as columns 2-11 being somewhat inversely correlated while columns 23-33 and 52-68 are both presenting correlated and inversely correlated behavior. In addition to this, some separate columns far from each other also present a correlated behavior, such as column 22 being highly correlated with column 45.

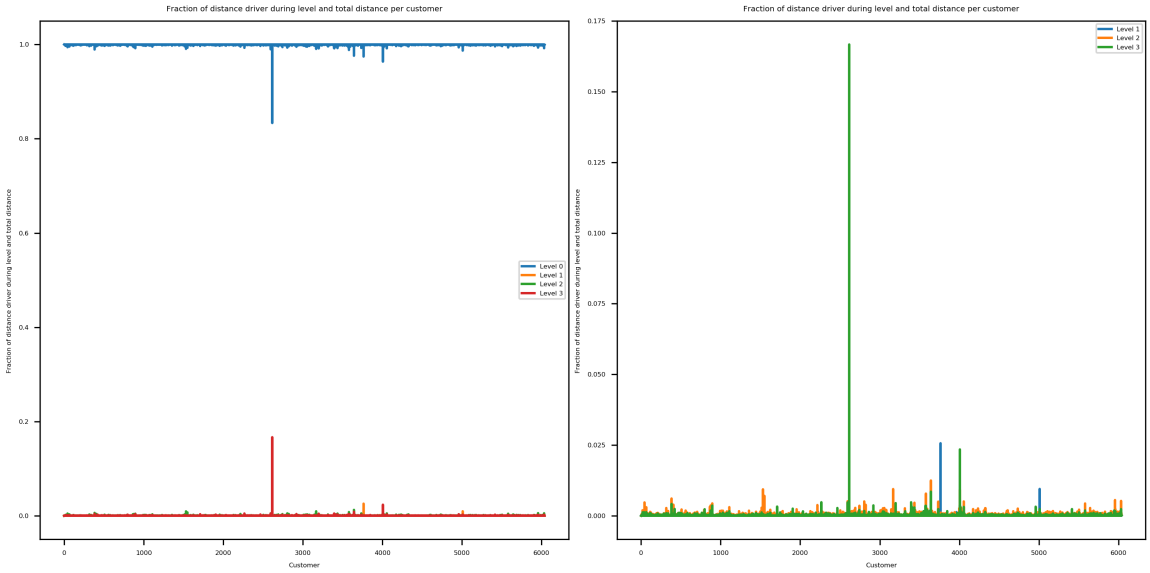


Figure 7: Distribution of v_1

In figure 7 we can see that the largest values of variable v_1 is found at level 0 which suggest that the variable is not well distributed, and rather largely skewed towards level 0. There are some notable cases where for some customers the distance driven during fulfillment of level 0 is very large. We see that one customer has driven a significant amount while fulfilling the criteria for level 3.

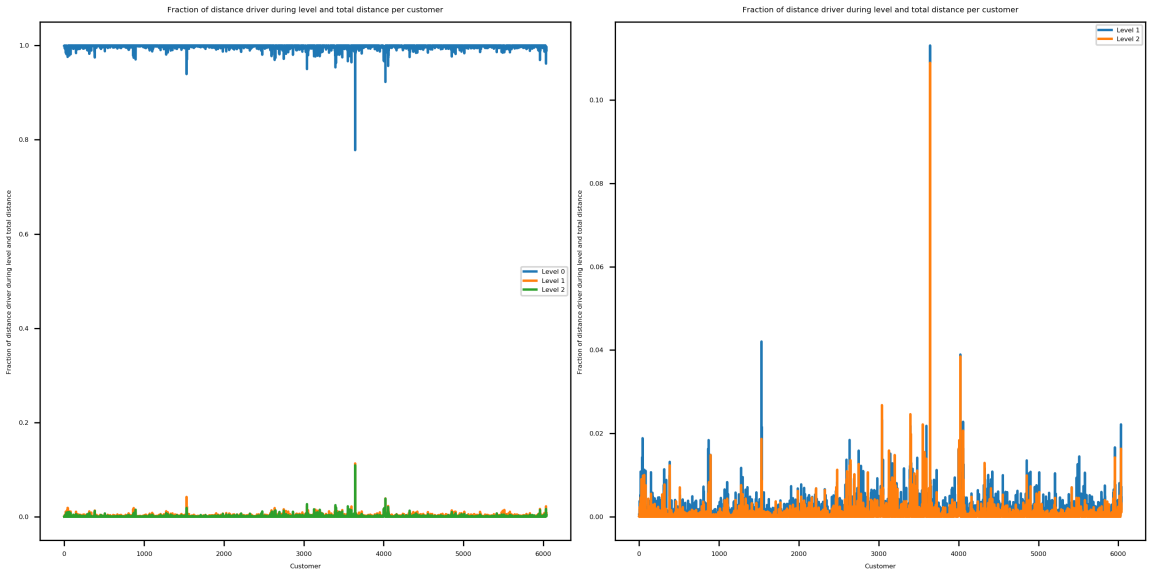


Figure 8: Distribution of v_2

From figure 8 above showing the distribution of variable v_2 we can see that for level 0 the behavior

is very much the same as for variable v_1 with some outliers as well as generally large values. However, when disregarding level 0 and analyzing the plots of the other levels we see that the distribution of values per customer is smoother with a more even distribution than for variable v_1 in figure 7.

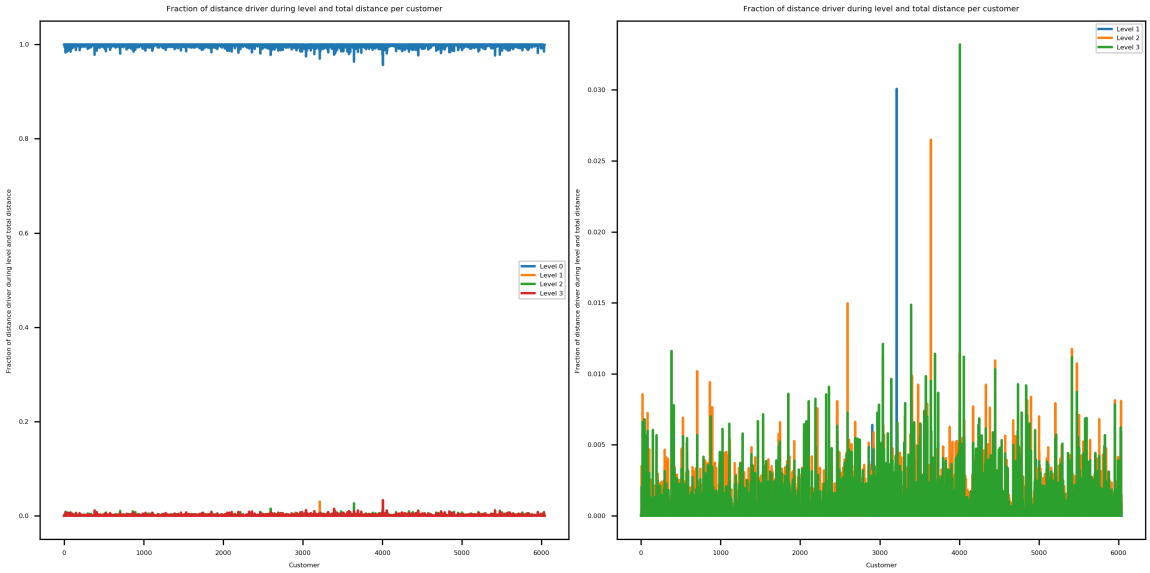


Figure 9: Distribution of v_3

The distribution of variable v_3 as seen in figure 9 is similar to variable v_1 with generally large values for level 0 that contains some large outliers, as well as a large outlier of level 2 that can be seen when disregarding from level 0.

For the combined level of speeding and road speed limit the levels are presented in no inherent order, i.e. level 0 does not necessarily represent the lowest level.

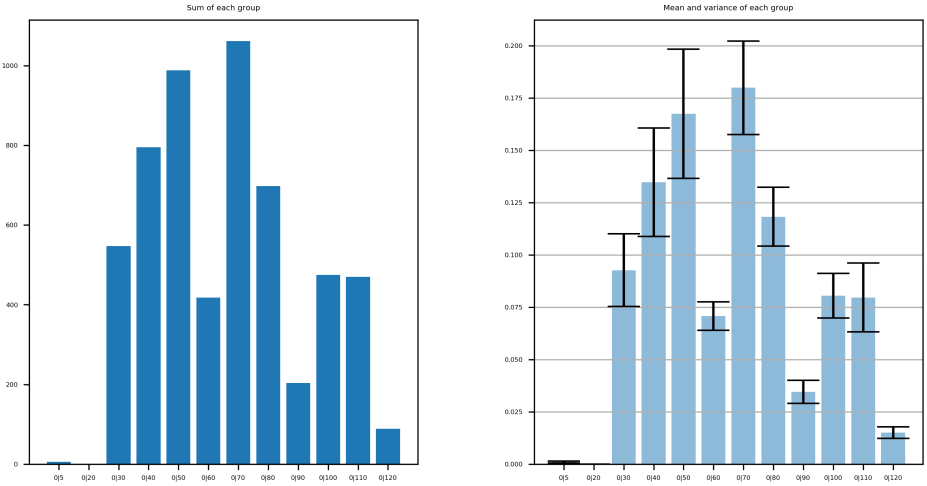


Figure 10: Speeding 0

In figure 10 we can see that the distribution is somewhat similar to the normal distribution, with speeding of level 0 not being common at very low or high speed-limit roads.

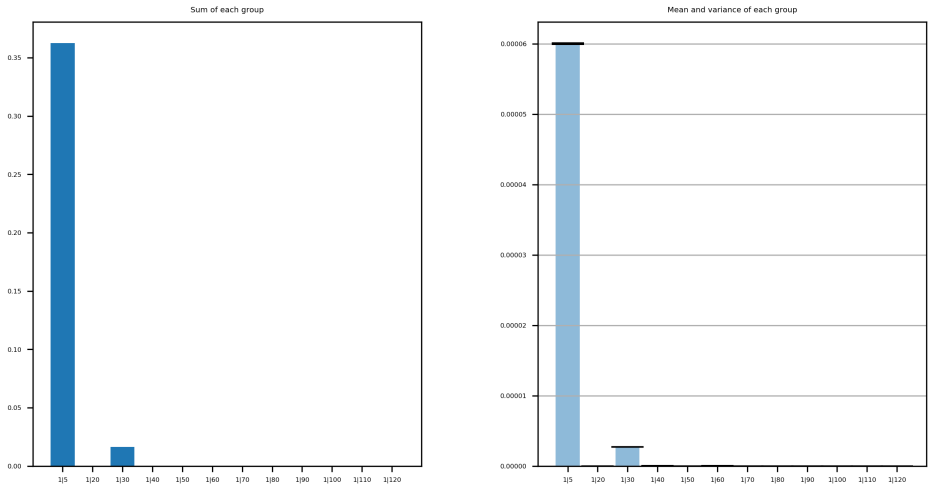


Figure 11: Speeding 1

From figure 11 we can see that speeding of level 1 almost always occurs at roads where the speed limit is $\leq$ 30 km/h.

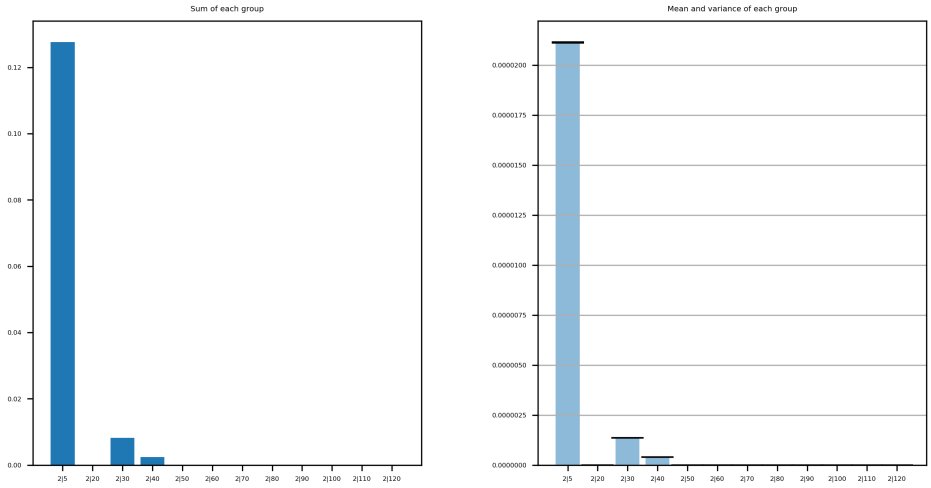


Figure 12: Speeding 2

When looking at figure 12 we can see that speeding of level 2 is very similar to the behavior of speeding of level 1 presented in figure 11, in that most of the speeding occurs at low-speed roads where the speed limit is $\leq$ 40km/h.

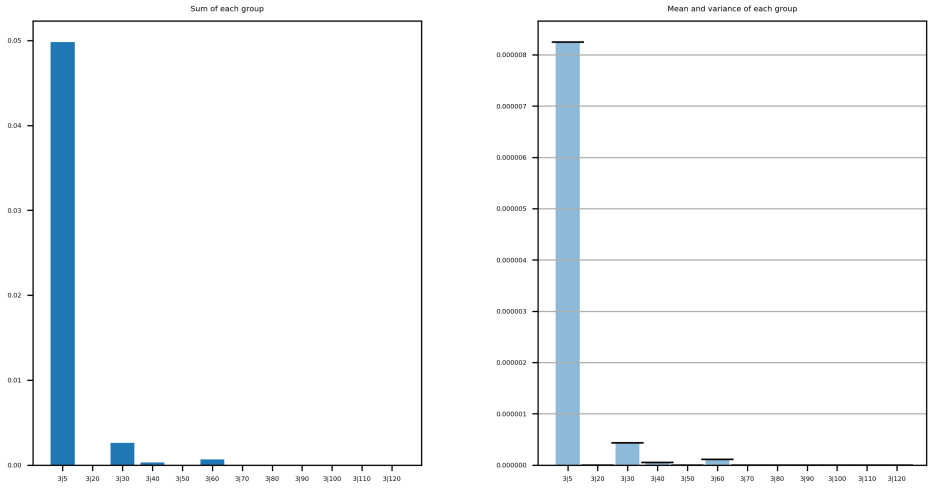


Figure 13: Speeding 3

The speeding in level 3 presented in figure 13 shows a similar behavior to the speeding levels presented in figure 11 and 12 with a shift towards speeding more at mid range speed-limits, but with no apparent speeding over 60km/h.

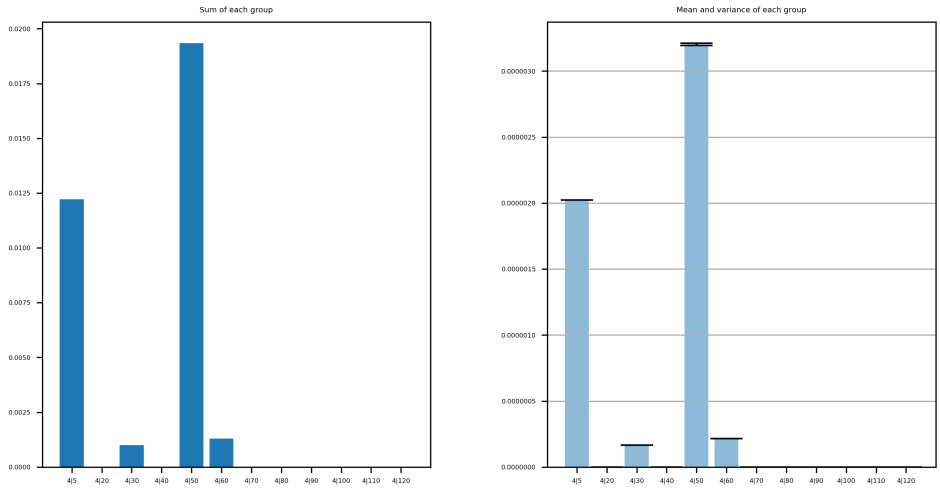


Figure 14: Speeding 4

In figure 14 we see significant speeding of level 4 at low speed roads as well as mid speed roads.

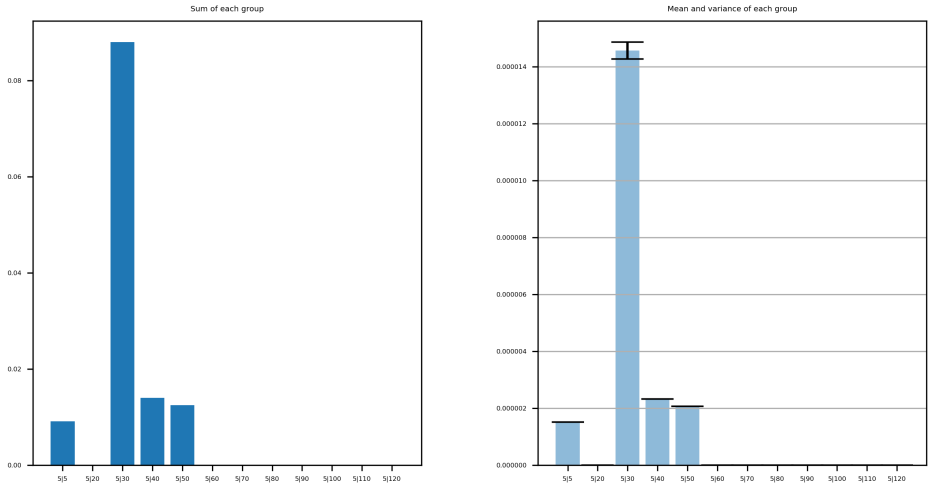


Figure 15: Speeding 5

In figure 15 we see that speeding of level 5 occurs at a significant level for low to mid-speed roads, with roads with a speed limit of 20km/h being the exception.

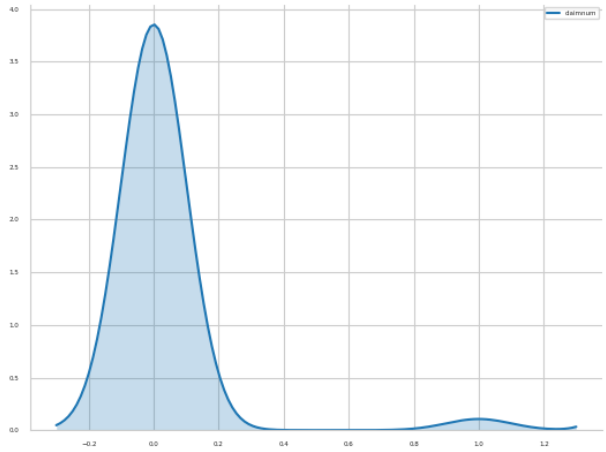


Figure 16: Number of claims modeled as a Poisson distribution

Moving on to analysis of the response variables, number of claims, and cost of claims, we can see that in figure 16 that most customers are not involved in an accident during the months. Standard practice in the insurance business is to model the number of claims as a Poisson distribution (see appendix 7.2.5). since we naturally have a lot of mass around 0 since accidents or claims are the exceptions rather than the norm. It is also worth noting that no customer has had two or more accidents during the month.

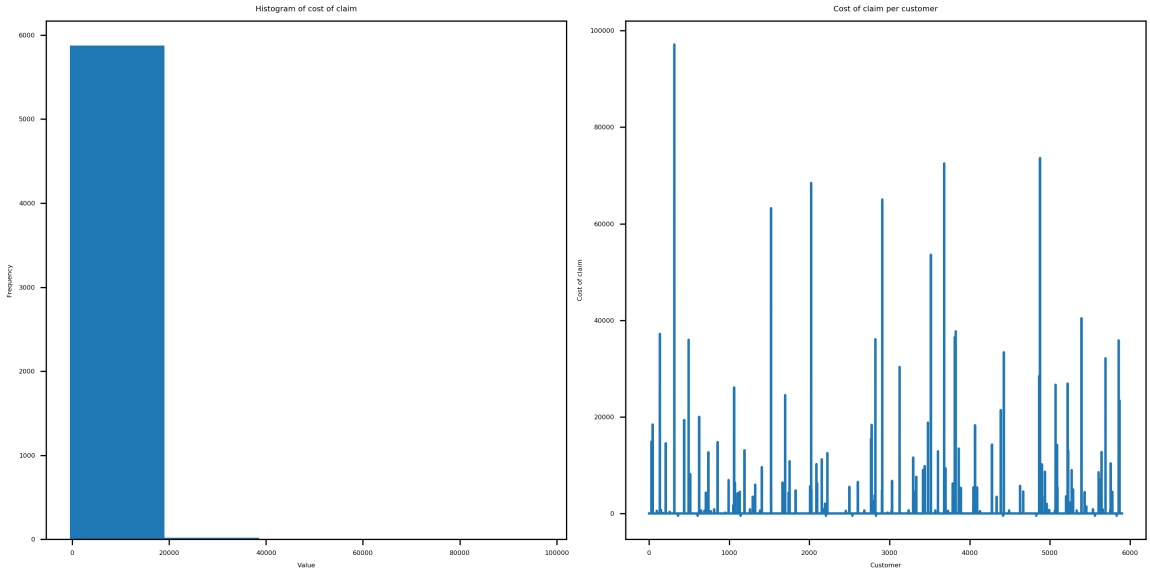


Figure 17: Cost of claims

In figure 17 both a histogram of the cost of claims and the distribution of the cost of claims per customer is presented. From the figure, we can see that most customers have no claim cost, which is to be expected as most customers don't file a claim, as shown previously in figure 16. We can also see that most of the incurred claim costs are low.

To illustrate the distribution more clearly, customers that don't incur a claim cost have been excluded in figure 18 below.

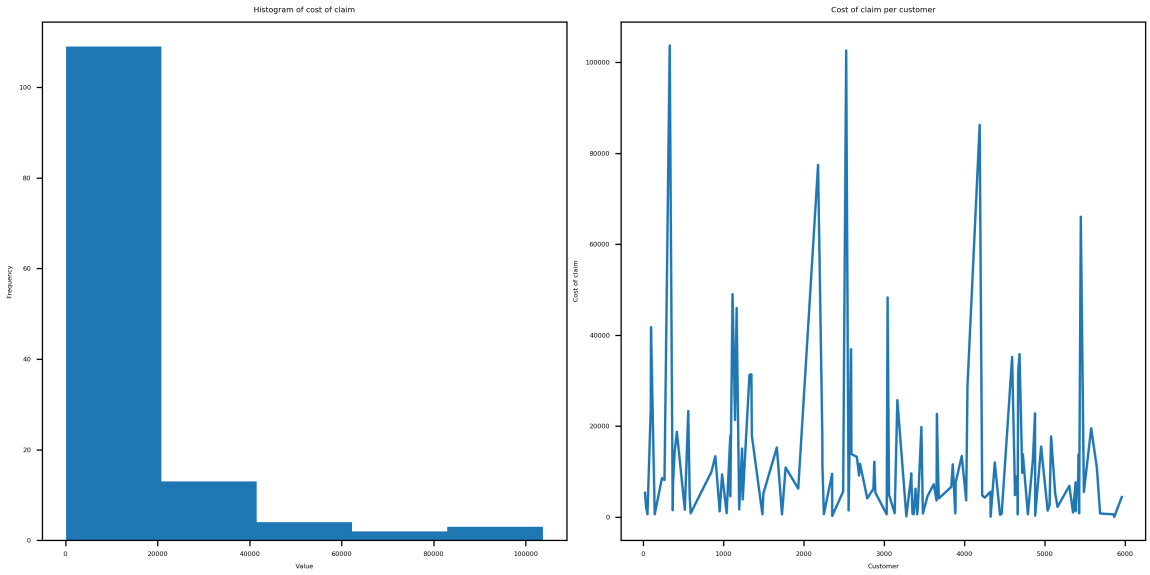


Figure 18: Cost of claims filtered

As can be seen from the filtered cost of claims in figure 18 the distribution of claim cost is still skewed towards most customers incurring a low claim cost and with this, we see that the cost of the claims can be modeled as a gamma distribution (see appendix 7.2.6).
Using the methods presented in section 2.4,2.5 and 2.6 clustering and dimensionality reduction is performed on the data set. The results are presented in section 5.

4 Method

The objective of the model is to accurately predict the expected claim cost, c that a customer will have each month. To this end, both the risk of a claim,r, along with the severity of a claim s will need to be determined. Two separate sub-models are created, working in conjunction with each other to determine the expected claim cost. The first sub-model determines the risk of a claim for a customer, and the second sub-model produces an expected claim severity. The final model then determines the expected claim cost by by multiplying the risk of a claim by the expected severity of a claim.

$$c = rs. \tag{43}$$

Regulations regarding machine learning within the insurance industry restrict the use of so-called 'black-box' methods for determining pricing. Neural networks fall under the category of 'black-box' methods and can therefore not be used to determine the expected claim cost explicitly. However, a neural network can be used as the sub-model to determine the claim risk, and another network as a sub-model to determine the claim severity. This possibly circumvents the issue by creating two new variables, expected claim risk, and expected claim severity, with two 'black-box' methods that are used in determining the expected cost, but that don't determine it explicitly. However, this will be discussed in more depth in the discussion.

4.1 Data set

The premium of each customer is set each month. The underlying data which the model uses to determine the expected claim cost is aggregated monthly. Each monthly data set can then be aggregated further depending on how many months that are to be taken into consideration. For example, the model could use the previous month, the previous three months, etc. Based on the scope of the analysis, we limit our analysis to one and 6 months.

To set a premium based on telemetry data as well as customer data for previous months, we let X be the data set containing the gathered telemetry data along with customer data. The data set either contains one or 6 months. The data set consist of data gathered from roughly 6000 customers. For the response variable, let y_r be the data set containing the corresponding to the claim risk and y_s the severity of the claims.

The data set is further divided into training and testing data with a split of 70/30 as this is a commonly used split.

4.2 GLM

The first sub-model for determining the expected claim risk for the customer is a generalized linear model, based on the theory set in section 2.1. For the implementation of the GLM, let the data sets X and y_r be the realization of the stochastic variables $\mathbf{Y}$ and $\mathbf{X}$ which are the response variable and the data respectively.

As discussed in the theory section 2.1 there exist variants of GLM:s. These variants have been studied in a similar setting in [1]. Based on the similarity of the data sets in the theses, the variant of choice for this paper is a standard GLM, as there are no significant differences between this model and the variants, and thus the simpler model is chosen.

As we can see from the data analysis, the distribution of the response variable is heavily skewed toward low values as most customers are not involved in accidents. We investigate two different link functions, log and inverse power for the response variable which is estimated as Poisson, and for the gamma-distributed response variable respectively.

The second sub-model is similar in its construction, with the realization of the stochastic response variable $\mathbf{Y}$ instead being set as y_s. The realization of the predictor variable $\mathbf{X}$ will thus be set as X.

The variables created by these two sub-models are then used to determine the expected claim cost to determine the premium of the customer.

4.3 Neural Network

The other model used for determining the premium of a customer is two neural networks. There exist many types of neural networks suitable for different purposes. Due to our data not being linearly separable and being incredibly biased, we need to utilize hidden layers. In this specific setting, three hidden layers have been set.

The first sub-model is a neural network determining the risk of a claim. And as the response of the risk prediction network, we will be using a sigmoid function for the activation function of the output layer. This leads to instead of getting a binary response we generate a likelihood from $0 \rightarrow 1$ which encodes a probability. Since the target variable on which the network is trained is a binary output the response of the network is thus the probability of having a claim. The activations in the hidden layers are set as RelU functions.

As input to the network we will set our data set x and the dimension of the input layer of the network will thus be 33 nodes, with a second hidden layer equipped with 17 nodes and the third at 17 nodes, as we have 67 features. As the output layer is meant to predict the claim risk y_r, it has the dimension of our response data set which for this model is $\mathbb{R}^{1\times n}$.

For the choice of an optimizer we use the Adam optimizer as proposed by [9]. Adam is similar to the stochastic gradient descent as presented in section 2.2. Without going into large detail in this section, Adam uses an adaptive learning rate and momentum optimization which in general leads to better results in cases where SGD is traditionally applied.

Once the optimizer is set we train the network iteratively using the data set along with the optimizer, creating a model which will predict the risk of claims.

The second sub-model is a neural network in the same way as for the first sub-model, with the difference being that instead of predicting the risk of a claim the severity of the claim will be predicted.

The structure of the network along with the method of optimization is similar to the case of the risk prediction network, with the difference being that the loss function is set as the MSE-loss and the activation of the last layer is binary.

Another model for risk prediction and claim severity was constructed in the same way as above but with the Random Oversampling algorithm applied to the data in an attempt to redistribute the zero heavy data.

Due to the distribution of the response variables, two other loss functions have been implemented as well. The distribution of the number of claims can be seen as Poisson distributed while the distribution of claim severity has been seen to approximate the gamma distribution. Due to these distributions of the response variables, the Poisson-deviance and gamma-deviance loss functions as defined in section 2.2 were implemented for the risk and severity models respectively.

Additionally, in a way to deal with the skewness of the data resampling techniques and transformations as presented in section 2.7 are performed on the data.

4.4 Evaluation

The models have been trained on 70% of the data and the remaining 30% are to be used for testing. As both the risk prediction and the regression are in essence tasks of regression producing continuous output, metrics like prediction accuracy can not be directly utilized. Instead, the metrics used are Mean Squared Error (MSE), Mean Absolute Error (MAE), and the R^2-value as presented in section 2.8

For further insights into the predictive powers of the model classification metrics are used for the risk prediction by setting e threshold on the output. By setting the response as 1 if the risk is ≥ 0.5 and 0 else, we can look at the rate of true positives, false positives, etc. In addition to this, we implement the Gini-coefficient defined in section 2.8 along with other metrics defined in the same, as the Gini-coefficient is a more suitable metric than the classification accuracy in our case with zero-heavy data, since by setting all output to be zero, we will have an accuracy that is equal to the percentage of zeros in the data.

5 Results

5.1 Exploratory Data Analysis

In this section a more thorough analysis of the data set is performed, using methods of dimensionality reduction and clustering to provide further insight into the data.

5.1.1 Dimensionality reduction

Two types of dimensionality reduction were performed on the data to generate interesting and valuable insights from the data. The dimensionality reduction was performed using a linear PCA method and a kernel PCA method. The PCA method is based on a transformation of the original features into a lower dimensional space using the concept of principal components. The idea behind principal components is that the new components are constructed as a linear combination of the initial features in a manner such that the new components are uncorrelated and contain the information from the original features so that the first component contains the most information, the second the next most, etc. This will allow the data to be transformed into a lower dimensional space while retaining as much information as possible which facilitates visualization of the data.

Linear PCA
Since PCA is a numerical analysis method we first need to get rid of data points that are NaN (not a number) and irrelevant information. This is easily done with pandas after which sklearn procures a very handy tool for PCA as a standalone library. Since our goal here is to introduce some more explainability it only comes naturally to compare the two methods of PCA. We generate a scatter matrix between different sets of components, with two and three components respectively. This along with the variance ratio is helpful in determining how many components are needed to capture

as much data as possible. By analyzing the so-called scree plot or the explained variance ratio in 22 we see that after 3 components there appears to be no computational or analytical gain from using more components which is also reflected in the kernel PCA.

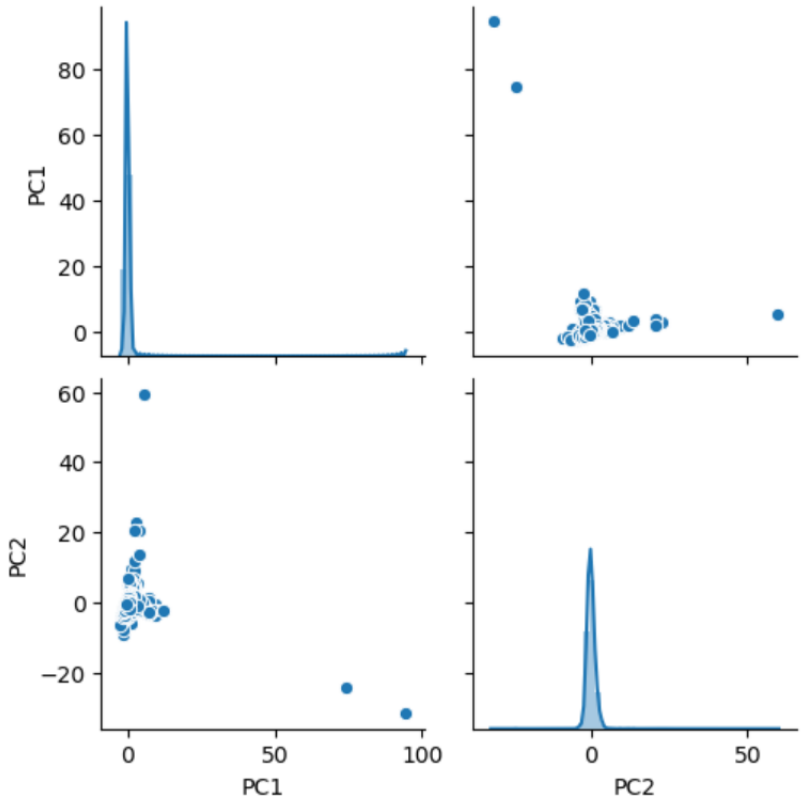


Figure 19: PCA 2 components

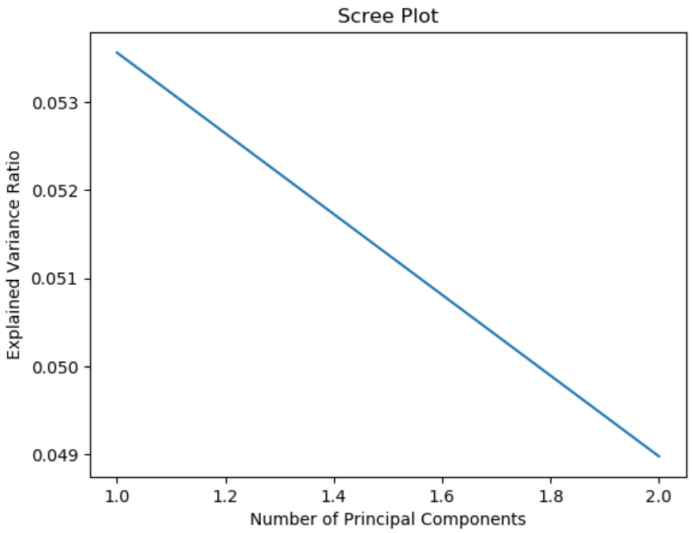


Figure 20: Screeplot, 2 components

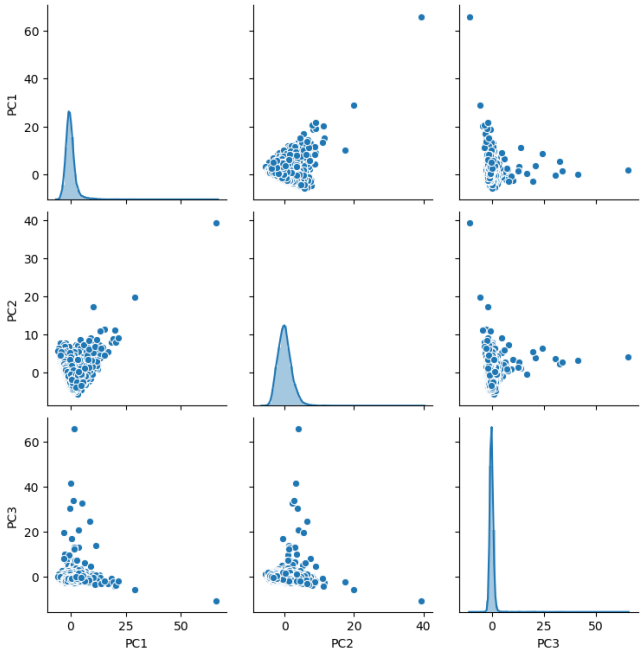


Figure 21: 3 components

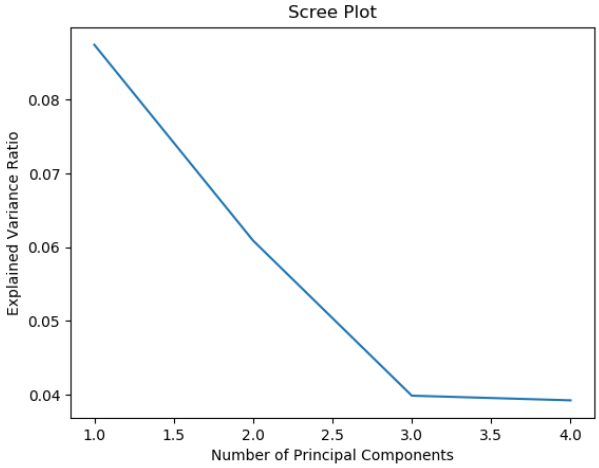


Figure 22: Scree-plot

Kernel PCA

Here we have a wide variety of kernels to choose from when trying to reduce the dimensionality of the data. Since we have a large data set that is very zero-heavy a cosine kernel is thought to generate the best results. What we can tell from this analysis is that there are very distinct groupings in the data and that when applying the cosine kernel component 2 follows a Gamma distribution, (see appendix 7.2.6) with the other components either being Poisson (see appendix 7.2.5) or don't follow any particular distribution very closely.

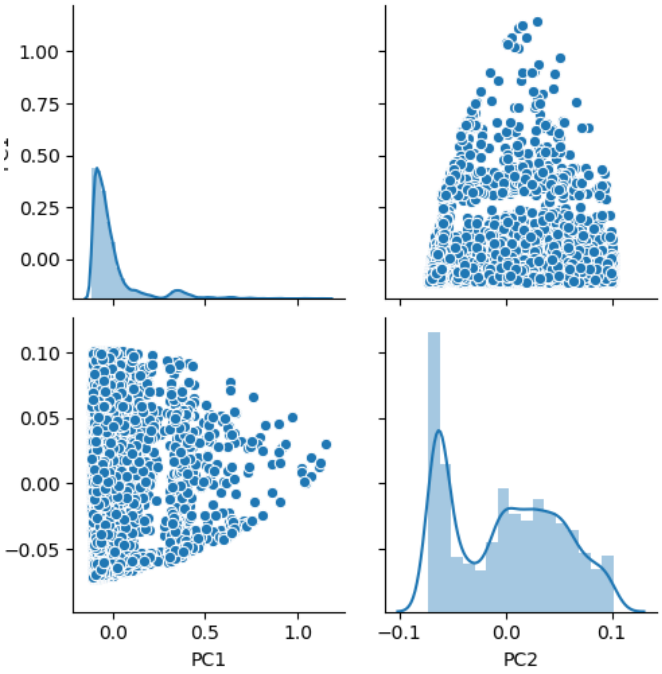


Figure 23: KPCA Cosine, 2 components

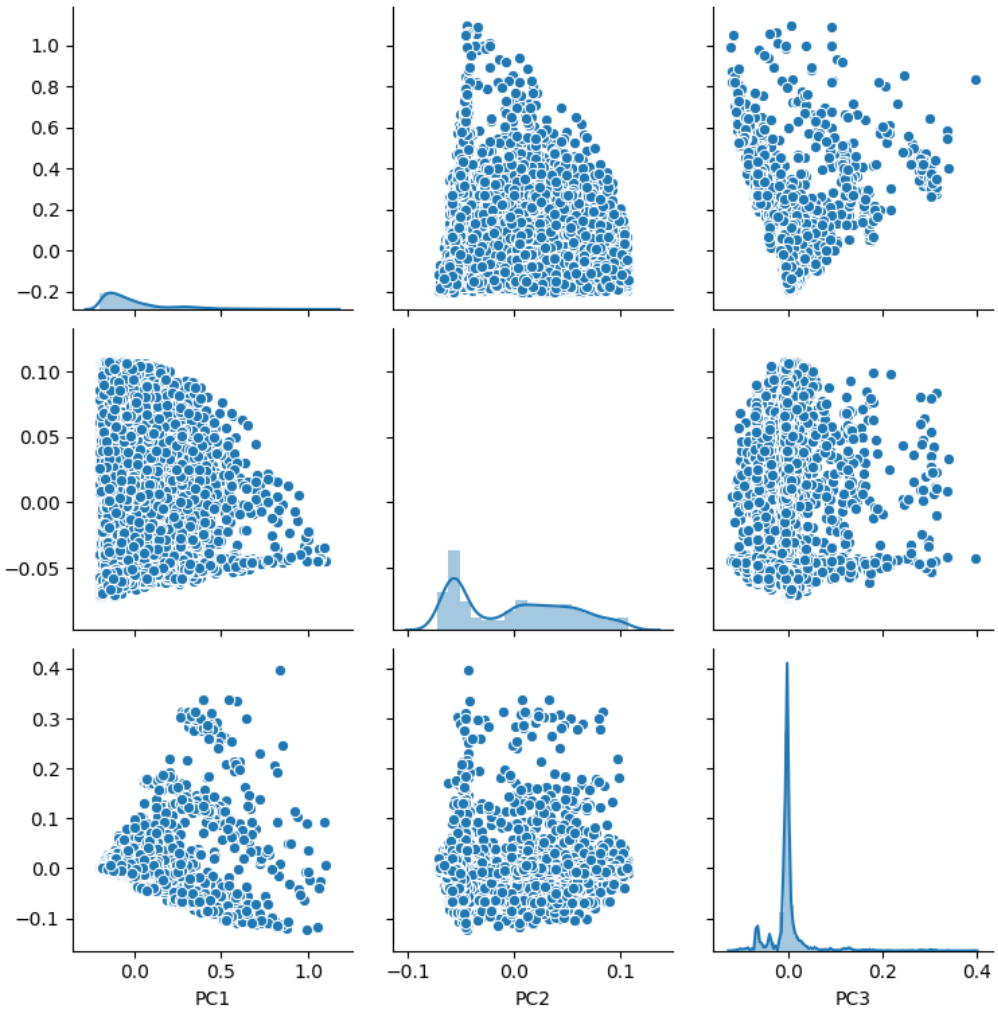


Figure 24: KPCA Cosine, 3 components

5.1.2 Agglomerative Hierarchical Clustering

If we instead assume that each initial data point is its own cluster center, and successively measure the distance between points, generating new clusters we get what is known as a dendrogram. Here the longest vertical line represents the largest distance between two clusters and each horizontal line represents a new cluster split. So what we can tell here is that there seem to be 4 general clusters forming, at level 100. Any lower than this and we see how the clustering algorithm can no longer make out the differences between the data points and no clustering structure can be seen.

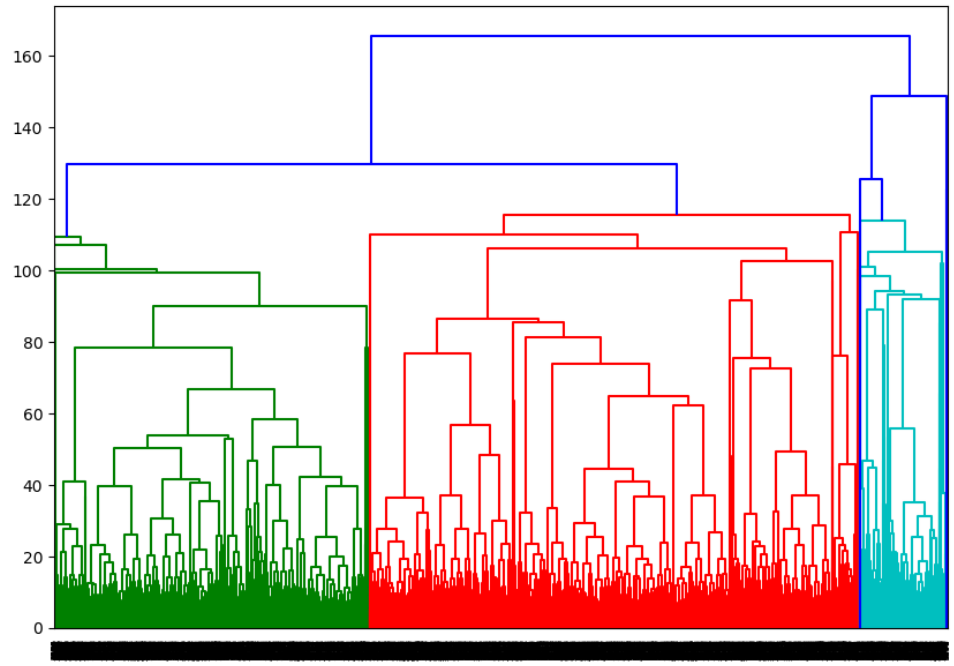


Figure 25: Dendrogram

5.1.3 K-means Clustering

As explained earlier in chapter 2.5 clustering methods come in many different forms. For this thesis, we're choosing to limit ourselves to K-means clustering and agglomerative hierarchical clustering. To perform K-means clustering we first need to gain insight into the data and determine what the optimal number of clusters are. Luckily there are many methods for this, we've chosen to go with the so-called elbow method. This is done by normalizing the data and calculating the WCSS (Within-cluster sum of squares) which is a measure of how tightly the data points in each cluster are packed together. It represents the sum of the square distance between each data point and its cluster center. To that end when trying to find the optimal number of clusters we utilize what is known as the elbow method.

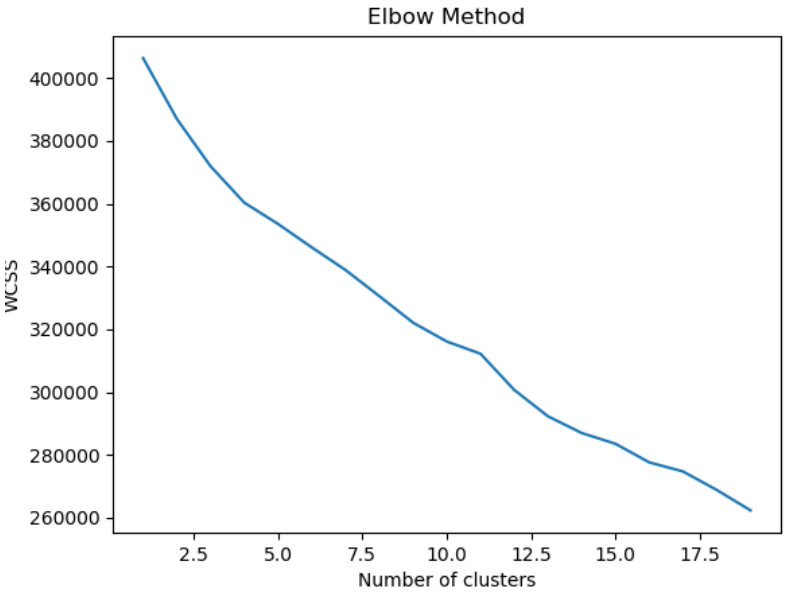


Figure 26: WCSS vs. No. of clusters

Here we're looking for a point on the x-axis (number of clusters) where we see an almost elbow-shaped line which appears to happen around 4 clusters. This idea behind having 4 clusters seems to be supported by the hierarchical clustering results as well, as we can clearly see at least 4 distinct clusters being formed. This yields an interesting cluster grouping when combined with the earlier results from the PCA analysis showing some outliers which can be favorably removed from the dataset as well as the very distinct groupings. Pictured in red are the cluster centroids.

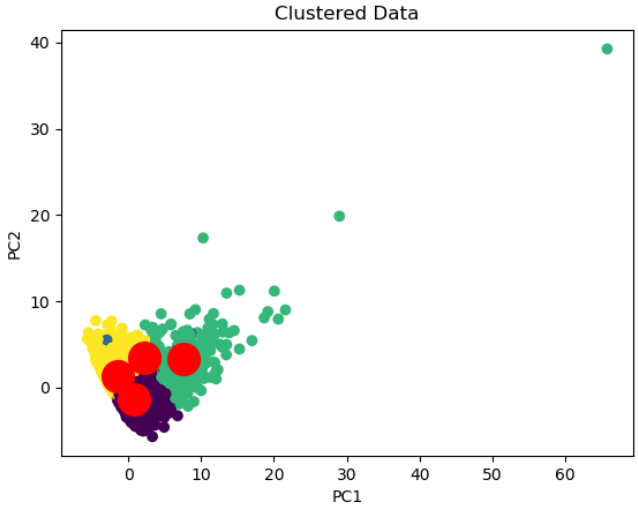


Figure 27: Clustered PCA data

5.2 Model results

In this section the results of the implementation of the models as presented in section 4 are evaluated using the metrics presented in 2.8. The data on which the model was trained was comprised of one month of data.

5.2.1 GLM

Due to the highly correlated nature of some of the variables, some of the variables were excluded from the modeling of the GLM. To set a threshold of which variables are to be included, the Variance Inflation Factor (VIF) has been used, as this gives a measure of the multicollinearity of variables. If too low of a threshold was used the model did not reach an optimal state due to numerical instability when training the model. The final threshold set for the model during training was a VIF of 10, which led to only 11 of the 64 variables being included. When implementing the GLM the data set has not been normalized due to numerical instability, and is instead used as-is.

Risk Model
The first GLM used for risk prediction was fitted with a log-link function with a random component taken from the Poisson distribution. As can be seen from figure 28a the model produces a negative R-squared value, albeit close to zero. This means that the model is, in essence, performing worse than a constant function which always guesses the mean value of the data. The classification

(a) Regression Metrics

MAE:	0.05
MSE:	0.02
R^2:	-0.03

(b) Classification Metrics

Accuracy:	0.975
Precision:	0
Recall:	0
F1-Score:	0
AUC Score:	0.5
Gini-coefficient:	0

Figure 28: Evaluation Metrics

metrics presented in table 28b also suggest that the model performs as well as a random chance model when setting a threshold to 0.5 to transform the regression into classification. This can be confirmed by the fact that the accuracy presented in table 28b is the same as the percentage of zeros in the test data set. The confusion matrix along with the ROC-curve of the model are plotted in figure 29 which further confirms that the model only predicts labels of 0.

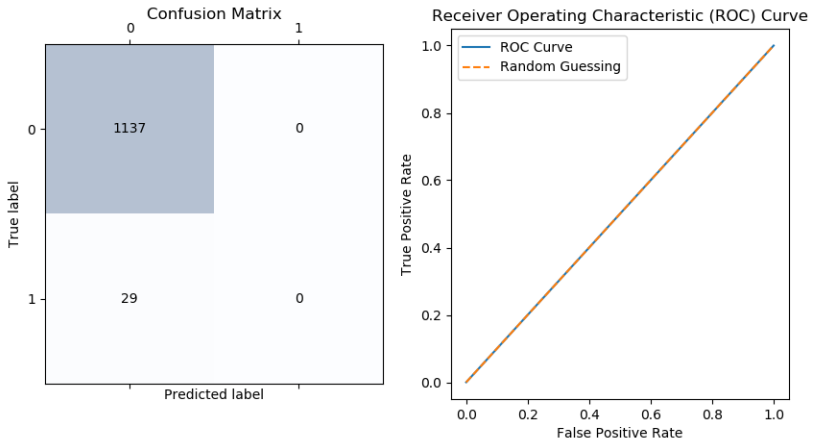


Figure 29: Confusion Matrix and ROC-Curve for the GLM risk model

Claim Cost Model
The GLM used to predict the claim cost of a customer was equipped with a random component from the gamma distribution and the link function as the inverse power function. The results are presented in table 2

MAE:	758.51
MSE:	27117892.13
R^2:	-0.39

Table 2: GLM claim-cost model regression metrics

5.2.2 GLM With Resampling

To potentially improve the results, the data on which the risk-prediction model was trained was resampled to reduce the skewness of the data. The claim cost model was trained on a target transformed via the Yeo-Johnson transformation. The risk model and claim cost model were equipped with the same random components and link functions as in the previous section.

Risk Model
The results of the risk model trained on the resampled data are presented below in figure 30. Table

(a) Regression Metrics

MAE:	0.46
MSE:	0.23
R^2:	-8.49

(b) Classification Metrics

Accuracy:	0.738
Precision:	0.046
Recall:	0.483
F1-Score:	0.084
AUC Score:	0.613
Gini-coefficient:	0.227

Figure 30: Evaluation Metrics

30a shows that the model performs worse than without resampling, with both higher MSE, MAE,

and R-Squared values. In table 30b we can note that the accuracy is lower, while the model no longer only makes label predictions of 0, as can be confirmed from figure 31.

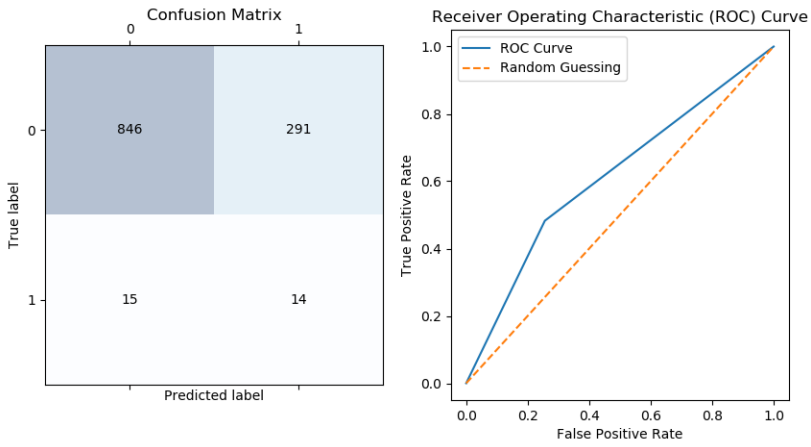


Figure 31: Confusion Matrix and ROC-Curve for GLM risk model with resampling

Claim Cost Model
The GLM that was used for the regression of claim cost was trained on transformed data using the Yeo-Johnson transformation. The evaluation of the model is presented in table 3. We can see that in contrast to the metrics presented in table 2 all of the metrics are lower.

MAE:	354.73
MSE:	19652763.28
R^2:	-0.01

Table 3: GLM claim-cost model regression metrics transformed

5.2.3 Neural Networks

The hindering factor of limitations on correlation was not a factor in the implementation of the neural networks although, as presented in [10], even if the model is trained the training rate could slow down significantly. The data is also normalized.

Risk Model
The first neural network used for risk prediction was equipped with the binary cross-entropy loss function and was run for 100 epochs with a batch size of 2. This produced the results presented in figure 32. From table 32a we can see that the performance is slightly similar to the performance of the risk prediction GLM without resampling. Although the neural network can classify other labels than 0, this leads to a slightly lower accuracy when used as presented in table 32b, which can also be seen from figure 33.

(a) Regression Metrics

MAE:	0.044
MSE:	0.043
R^2:	-0.761

(b) Classification Metrics

Accuracy:	0.956
Precision:	0.077
Recall:	0.069
F1-Score:	0.073
AUC Score:	0.524
Gini-coefficient:	0.048

Figure 32: Evaluation Metrics

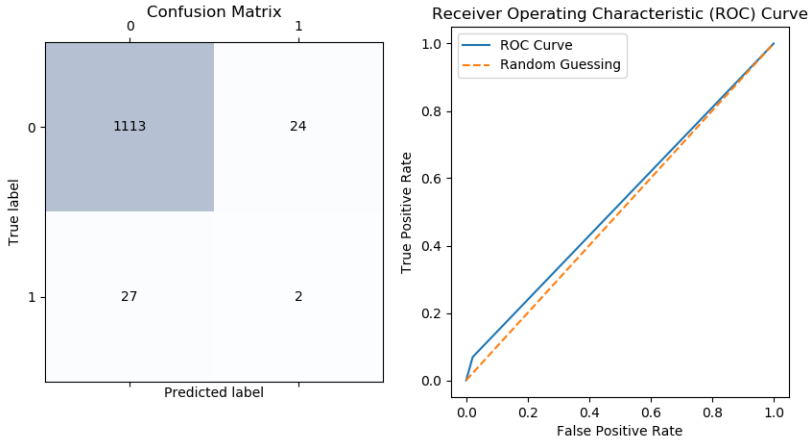


Figure 33: Confusion Matrix and ROC-Curve for risk model

Claim Cost Model

The neural network for claim cost was equipped with the MSE loss function and produced the values presented in table 4 below. The model produces values quite similar to the GLM used for claim cost regression both with and without transformation.

MAE:	776.63
MSE:	21627834.02
R^2:	-0.108

Table 4: Neural Network claim-cost model regression metrics

5.2.4 Neural Networks with Resampling

As the data for the risk model is quite zero-heavy, resampling techniques are applied to potentially improve the performance of the models.

Risk Model

The neural network used for risk prediction was equipped with the binary cross-entropy loss function. The network trained on data resampled using the theory presented in section 2.7 for 100 epochs with a batch size of two. The model was evaluated and yielded the following results as presented in table 34 and 35. As can be seen from figure 34 the model performs slightly better than

the risk model trained in the original data as presented in figure 32 if we look at the regression metrics. However, when analyzing the classification metrics in table 32b and figure 35 the metrics perform somewhat worse in terms of Gini-coefficient but with somewhat higher accuracy.

(a) Regression Metrics

MAE:	0.039
MSE:	0.039
R^2:	-0.603

(b) Classification Metrics

Accuracy:	0.960
Precision:	0
Recall:	0
F1-Score:	0
AUC Score:	0.493
Gini-coefficient:	-0.014

Figure 34: Evaluation Metrics with Resampled data

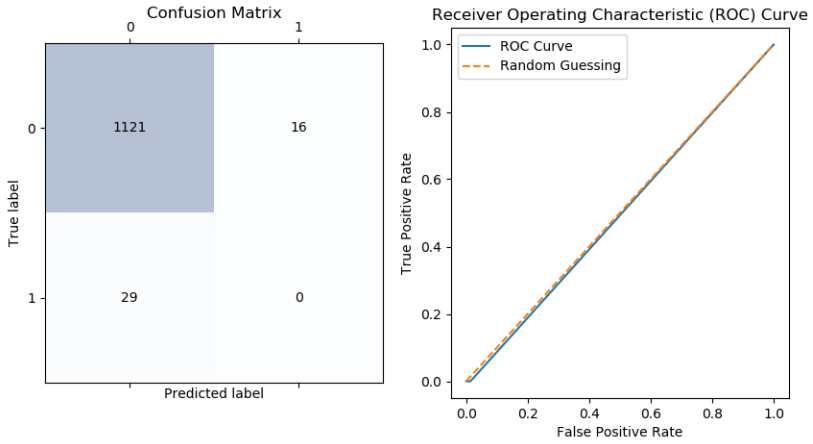


Figure 35: Confusion Matrix and ROC-Curve for risk model trained on resampled data

Claim Cost Model

The network for predicting the claim cost used the MSE loss function and was trained on data transformed using the Yeo-Johnson transformation technique. This produced the following results presented in table 5 which is quite similar but somewhat better than the claim cost model without transformation.

MAE:	354.73
MSE:	19652761.65
R^2:	-0.006

Table 5: Neural Network claim-cost model regression metrics with transformation

5.2.5 Neural Networks with Poisson and gamma loss functions

To improve the results the risk and claim-cost networks were equipped with a Poisson deviance loss function and a Gamma deviance loss respectively.

Risk Model

The neural network used for risk prediction produced the results as presented in figure 36 and 37 when equipped with the Poisson-deviance as loss function on non-resampled data. The results are quite similar to the original risk prediction network both when trained on the original data and the resampled data.

(a) Regression Metrics

MAE:	0.048
MSE:	0.042
R^2:	-0.730

(b) Classification Metrics

Accuracy:	0.956
Precision:	0.077
Recall:	0.069
F1-Score:	0.073
AUC Score:	0.524
Gini-coefficient:	0.048

Figure 36: Evaluation Metrics with Poisson loss

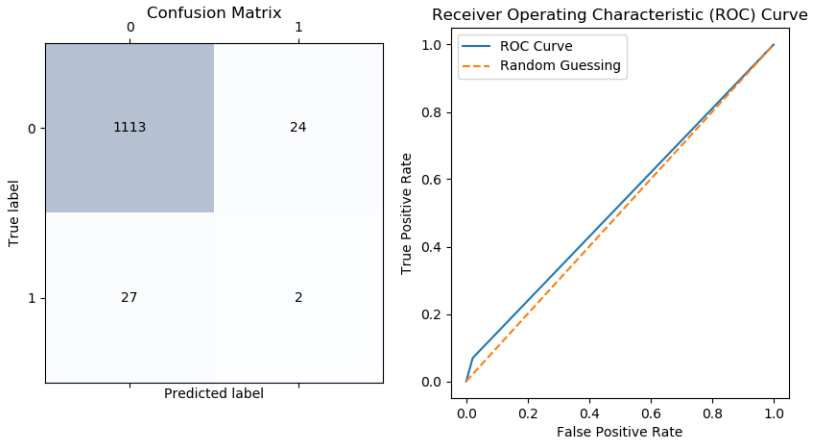


Figure 37: Confusion Matrix and ROC-Curve for risk model trained with Poisson loss

Claim Cost Model

The network for determining claim cost was equipped with the gamma deviance function and produced the results as shown in table 6. The performance of using the Gamma loss function is close to identical as the network trained on the transformed target presented in table 5.

MAE:	355.10
MSE:	19653076.71
R^2:	-0.006

Table 6: Neural Network claim-cost model regression metrics with Gamma-loss

5.2.6 Neural Networks with resampling, transformation, Poisson, and gamma loss functions

The last implementation of the neural networks combines the resampling used for the risk model along with the Poisson deviance loss to potentially improve the model further. For the claim cost network the gamma deviance loss function was used along with a Yeo-Johnson transformation of the data.

Risk Model

The neural network used for risk prediction produced the results as shown in figure 38 and 39 when equipped with the Poisson-deviance as loss function and was trained on the resampled data. The results of the risk netowrk are quite similar to the other versions previously presented.

(a) Regression Metrics

MAE:	0.075
MSE:	0.080
R^2:	-2.074

(b) Classification Metrics

Accuracy:	0.956
Precision:	0.077
Recall:	0.069
F1-Score:	0.073
AUC Score:	0.524
Gini-coefficient:	0.048

Figure 38: Evaluation Metrics with resampled data and Poisson loss

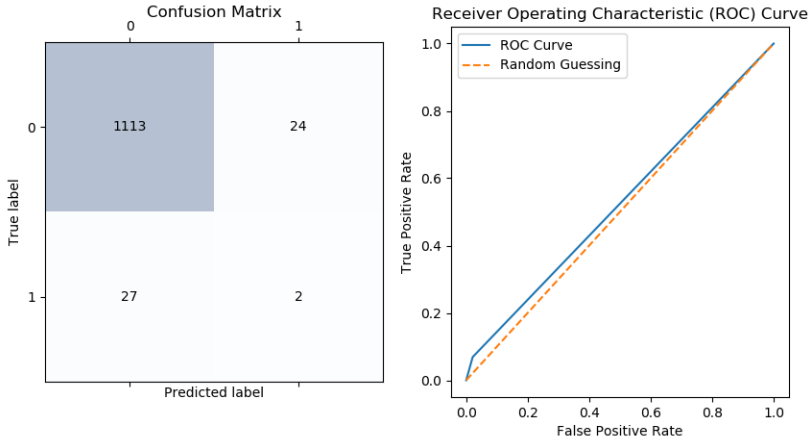


Figure 39: Confusion Matrix and ROC-Curve for risk model trained on resampled data with Poisson Loss

Claim Cost Model

The network for determining claim cost was equipped with the gamma deviance function and used on transformed data and when evaluated produced the following metrics as presented in table 7

MAE:	355.00
MSE:	19652993.30
R^2:	-0.006

Table 7: Neural Network claim-cost model regression metrics with Gamma loss and transformation

6 Discussion

6.1 General findings

In this thesis, we've worked with telemetric data which by nature is quite heavy. What we mean by this is that there's a lot of data and the data isn't necessarily very pretty. A majority of our work was structured around building systems that would allow us to incorporate and create new features which aren't currently being explored. Even before we started working we found through the literature review that telemetric data and neural networks don't generally work [11] because of the high variance. There was however the hope that by different statistical techniques and varying loss functions, we would be able to circumvent this but as is expanded on further under the results chapter of the discussion we did not manage to do this.

6.2 Data

Unfortunately with how we chose to structure the data and create features, we introduced some heavy correlation between features. This becomes an issue when working with Generalized Linear Models (GLM) as is clearly evident in the results chapter, seeing as it becomes impossible to fit a GLM to such correlated data. We also find similar problems with the neural network however for a different reason. Insurance data is inherently skewed and very zero-heavy because a majority of the customers have never nor will ever submit a claim. Such is the nature of the data. It then becomes important to find workarounds;

- Aggregate over a longer time span (6+ months)
- Introduce a base risk for each customer by random sampling techniques
- Oversampling data to simulate an increase in the occurrence of claims

All options come with their respective downsides, a longer time period of analysis means more computational power is required both for the aggregation and for the fitting and training of the model. Introducing a base risk for each customer by sampling techniques generates a dataset that doesn't aptly represent reality, thus introducing unpredictable bias which might skew the data in the other direction. As such it becomes a balancing act where on the one hand you want data that is as close to reality as possible but on the other, you need the data to follow a very particular type of structure. The same applies to resampling data to increase the frequency of claims, namely that the data set doesn't necessarily represent the real world anymore. Optimally one would first try to incorporate all historical data (4+ years) and see how the skewness problem develops, as it is entirely plausible that one would not need to introduce a base risk if more data is present. With longer periods, it is also possible to divide the claim data into quantiles by counting the number of claims and dividing per month, thus getting a claim frequency. This would restructure the regression model into a classification model, which typically is easier to train according to [12].

6.3 Results

The results presented in section 5 show that the models did not perform quite as expected. This could be due to many factors and choices in the implementation of the models as well as being the result of inherent properties in the data set.

The first problem possibly affecting the results of the modeling was the amount of correlation in our data set. The correlation in itself may be a valuable insight to Paydrive, as this allows them to understand customer behavior better, but in the modeling of the risk prediction and claim costs, this could be one of the reasons for the low performance. For example, when implementing the GLM the majority of the features had to be removed from the data set as it otherwise hindered convergence of the optimization algorithm.

For the implementation of the neural networks, one problem was the zero-heavy nature of the data. One resampling method, random oversampling, was utilized to combat this. However, multiple choices of resampling algorithms exist but due to constraints on time and resources other possibly more powerful algorithms could not be explored.

The Poisson and gamma loss were introduced as a way to create a loss function that better represents the data and problem at hand. However, no deeper analysis of Poisson and gamma loss functions were performed to ensure that they were optimally fitted to our data. By resampling and transforming data, the data was possibly no longer Poisson or gamma distributed which could explain the poor result of the combined model.

6.4 Future work

An obvious problem with using Neural Networks for auto insurance premiums is the legality. There are currently laws in place that require explainability of the involved parameters which is to be even further restricted by new EU regulations coming into action during 2023. These need to be examined further and a dialogue with the responsible authorities regarding how to proceed should be initiated.

As for the modeling, one improvement of the risk and claim modeling is a re-scoping of the problem into a classification problem. This could be done by extending the analysis to span over a long enough time period to contain several claims per customer. This would allow each customer to be assigned a claim frequency which could be used to label each customer into a category, which could then be predicted using multi-class classification. This is similar to the approach taken by [1] which showed promising results. A similar approach could be applied to the claim cost modeling which would lead to two multi-class models to be applied either through a GLM or a Neural Network.

The above solution would only transform the problem and the target variables, but the high correlation in the data would still be present. As such, techniques for reducing correlation and models more robust in the face of high correlation could be further explored. One such technique could be to use the resulting components of PCA analysis as the features in the data set.

The existing models could also be developed and investigated further. As mentioned above, techniques for dealing with highly correlated features could be explored, along with a more rigorous exploration of parameters and modeling choices. The results of the GLM in section 5 was constructed using a log link function and a Poisson random component for the risk model and an inverse power link function and gamma random component for the claim cost prediction. Other link functions and potentially other distributions could be explored. The networks used could also be explored further both in terms of architecture and choice of parameters and optimizers. In

this thesis, common choices found to generally produce good results were used, but more specific choices could be made.

7 Appendix

7.1 Basic probability theory

A random variable

A random variable $\mathbf{X}$ is a measurable function $X : \Omega \to \mathbf{E}$ from a sample space $\mathbf{\Omega}$ as a set of possible outcomes to the measurable space $\mathbf{E}$. Then the probability that $\mathbf{X}$ takes on a certain value in the measurable set $S \subseteq \mathbf{E}$ can be defined as

$$P(x \in S) = P(\{\omega \in \mathbf{\Omega} | X(\omega) \in S) \tag{44}$$

The probability space

In probability theory, a probability space is a construct that provides a formal mathematical structure for a random "process" or experiment. A popular such random process is the throwing of a die. Any probability space consists of three underlying structures:

- A sample space, $\mathbf{\Omega}$ which is the set of all possible outcomes,
- An event space, which is a set of events, $\mathcal{F}$ where an event is a set of outcomes in the sample space,
- A probability function, which assigns each event in the event space a probability between 0 and 1.

Probability Mass Function

A probability mass function is the probability distribution of a discrete random variable, i.e. $\mathbf{X}$ and it provides the possible values of said random variable and the probability that $\mathbf{X}$ takes on each value. Formally it is the function $p : \mathbb{R} \to [0, 1]$ as defined by:

$$p_X(x) = P(\mathbf{X} = x) \tag{45}$$

for $-\infty < x < \infty$. Where P is a probability measure. The probabilities associated with all values must hold up to 2 criteria;

- The sum of all probabilities must be 1, i.e $\sum_X p_X(x) = 1$.
- $p_X(x) \geq 0$, $x \in \mathbb{R}$

Probability Density Function

A probability density function is a function that describes the probability of a continuous random variable taking on a given value. The function is always non-negative and the surface area under the curve if graphed must be equal to 1. Given a random variable, **X** with probability density function (PDF), f(X). The probability that **X** falls within a certain range [a,b] is given by:

$$P(a \leq X \leq b) = \int_a^b f(x)dx \tag{46}$$

Expected values

An expected value in probabilistic theory is quite intuitively what you might expect. Consider now a random variable X with a finite list of possible outcomes ordered $\{x_1, x_2, x_3, ..., x_k\}$ associated with each of these possible outcomes is their respective probabilities $\{p_1, p_2, p_3, ..., p_k\}$. The **expected** value of **X** is then defined as:

$$E(X) = x_1p_1 + x_2p_2 + ... + x_kp_k \tag{47}$$

in other words for discrete random variables the expected value is simply calculated as the probability of each item times the value of said item. For continuous random variables the equation changes slightly. If we instead consider a continuous random variable, X which has a related *probability density function* 7.1 given by some function $f \in \mathbb{R}$. The expectation of X is then given by:

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx \tag{48}$$

Conditional probability

Let A and B be events such that $P(B) > 0$, the conditional probability of A given B is defined as:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{49}$$

in other words, the probability that A happens given that B has happened is the relationship between all events in the space $\{A \cap B\}$ and $P(B)$.

Law of total probability

The law of total probability is a theorem which states that in the discrete case, if the set B_n is at least a finite partition of a sample space and each event contained in B_n is measurable, then for any event **A** in the same probability space:

$$P(A) = \sum_n P(A \cap B) \tag{50}$$

where $A \cap B$ is the intersection between the two sets which can be written more explicitly as $P(A \cap B) = P(A|B_n)P(B_n)$.

In the continuous case we need to utilize our definition of a *probability space*. Let $(\Omega, \mathcal{F}, P)$ be such a space and suppose **X** is a random variable with associated distribution function F_X and A an event on the probability space. The law of total probability in the continuous case then states:

$$P(A) = \int_{-\infty}^{\infty} P(A|X = x)dF_X(x) \tag{51}$$

Variance

The variance for any random variable X with an expected value 7.1, defined as μ is given by:

$$\text{Var}(X) = E\left[(X - E[X]^2)\right] = E[X^2] - E[X]^2 \tag{52}$$

and it is the measure of dispersion, how far a set of numbers is spread out from their average value.

Covariance

Covariance is simply the measure of variance between two random variables, X and Y for example. It is defined as:

$$\text{cov}(X, Y) = E\left[(X - E[X])(Y - E[Y])\right] = E[XY] - E[X]E[Y] \tag{53}$$

and can be a useful tool when analyzing linear dependencies between two random variables.

7.2 Probability Distributions

A probability distribution is a function that describes the likelihood of different outcomes for a random variable. It tells us how the probabilities of different outcomes are distributed among the possible values of the random variable.

7.2.1 Gaussian distribution

The Gaussian distribution, or more commonly known as the normal distribution. For our purpose the Gaussian distribution will be used in our Monte Carlo sampler. It's PDF, 7.1 is given by:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{54}$$

when $\mu = 0$ and $\sigma = 1$ we have what's called the standard normal distribution. It's a very useful distribution as the PDF follows a bell curve and the mean and variance are given by:

$$\mathbb{E}(X) = \mu$$

$$\mathbb{V}(X) = \sigma$$

7.2.2 Multivariate Gaussian distribution

Just as for the Gaussian distribution we must also define the multivariate Gaussian distribution for when we're dealing with sequences of random variables or simply multiple pairs. It's PDF is given by:

$$p(\mathbf{x}; \mu, \Sigma) = \frac{1}{|\Sigma|^{\frac{1}{2}}(2\pi)^{\frac{n}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)} \tag{55}$$

where the mean and variance are given by:

$$\mathbb{E}(X) = \mu$$

$$\mathbb{V}(X) = \Sigma$$

7.2.3 Exponential distribution

The exponential distribution is a probability distribution that describes the time between events in a Poisson process, where events occur continuously and independently at a constant average rate. Therefore it comes in handy when modeling for example the rate at which insurance claims happen. It is a continuous probability distribution, defined by the following probability density function (PDF):

$$f(x) = \lambda e^{-\lambda x},\ x \geq 0 \tag{56}$$

with mean and variance as:

$$\mathbb{E}(X) = \tfrac{1}{\lambda}$$

$$\mathbb{V}(X) = \tfrac{1}{\lambda}$$

7.2.4 Bernoulli distribution

With a Bernoulli distribution our aim is to model the outcome of a single experiment where the outcome is binary (0 or 1). It's a discrete probability function and is modeled by it's probability mass function:

$$P(X = x) = p^x(1-p)^{1-x} \tag{57}$$

where p is the probability of a positive outcome and $(1-p)$ is the probability of a negative one. It's mean and variance are given by:

$$\mathbb{E}(X) = p$$

$$\mathbb{V}(X) = p(1-p)$$

Since this distribution allows us to model single experiments, it has many use cases. Particularly in engineering where it can be used to model the probability of failure of a component. It's important to note here that this is for a single experiment. If we start modeling n independent attempts instead we use the Binomial distribution of which the Bernoulli is a special case.

7.2.5 Poisson distribution

The Poisson distribution is a probability distribution that describes the number of events that occur in a fixed interval of time or space, if these events happen independently and at a constant average rate. It is a discrete probability distribution, defined by the following probability mass function (PMF):

$$P(X = x) = (\lambda^x e^{-\lambda})(\lambda!)^{-1} \tag{58}$$

where λ is the average number of events per interval and x is the actual number of events in an interval. It's mean and variance are given by:

$$\mathbb{E}(X) = \lambda$$

$$\mathbb{V}(X) = \lambda$$

due to the Poisson distribution being a one-parameter family of distributions with λ being our only shape parameter it is incredibly easy to use and powerful. It can be used to simulate for example the number of phone calls made to a call-center in a given time interval and model future expected number of claims.

7.2.6 Gamma distribution

The Gamma distribution is a probability distribution that describes the time until the next event in a Poisson process, where events occur continuously and independently at a constant average rate. It is a continuous probability distribution, defined by the following probability density function (PDF):

$$f(x) = \frac{(x^{(k-1)}e^{-x})}{\Gamma(k)\theta^k} \tag{59}$$

where Γ is the Gamma function, θ is the scale parameter and k is the shape parameter. It's mean and variance are calculated as:

$$\mathbb{E}(X) = k\theta$$

$$\mathbb{V}(X) = k\theta^2$$

This is obviously a generalization of the exponential distribution if $k = 1$ and $\theta = \frac{1}{\lambda}$

7.2.7 Weibull distribution

The Weibull distribution is a probability distribution that is often used to model time-to-failure data in reliability engineering and other fields. It is a continuous probability distribution, defined by the following probability density function (PDF):

$$f(x) = \frac{k}{\lambda}(\frac{x}{\lambda})^{k-1}e^{(-\frac{x}{\lambda})^k},\ x \geq 0 \tag{60}$$

with mean and variance calculated as:

$$\mathbb{E}(X) = \lambda\Gamma(1 + \tfrac{1}{k})$$

$$\mathbb{V}(X) = \lambda^2\left[\Gamma(1 + \tfrac{2}{k}) - (\Gamma(1 + \tfrac{1}{k})^2\right]$$

where Γ is the Gamma-function.

References

[1] V. Kröger and R. Nordström, *Expected individual insurance cost based on driving pattern: Machine learning methods using telemetric data*. PhD thesis, 2020. Due to confidentiality, this report is censored.

[2] M. V. Wuthrich, "Bias regularization in neural network models for general insurance pricing," *SSRN*, 2019. Available at SSRN: https://ssrn.com/abstract=3347177 or http://dx.doi.org/10.2139/ssrn.3347177.

[3] H. Abdi and L. J. Williams, "Principal component analysis," *WIREs Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[4] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Artificial Neural Networks—ICANN'97: 7th International Conference Lausanne, Switzerland, October 8–10, 1997 Proceeedings*, pp. 583–588, Springer, 2005.

[5] T. S. Madhulatha, "An overview on clustering methods," *CoRR*, vol. abs/1205.1117, 2012.

[6] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[7] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine learning with oversampling and undersampling techniques: Overview study and experimental results," pp. 243–248, 04 2020.

[8] I. Yeo and R. A. Johnson, "A new family of power transformations to improve normality or symmetry," *Biometrika*, vol. 87, pp. 954–959, 12 2000.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[10] S. r. Halkjæ r and O. Winther, "The effect of correlated input data on the dynamics of learning," in *Advances in Neural Information Processing Systems* (M. Mozer, M. Jordan, and T. Petsche, eds.), vol. 9, MIT Press, 1996.

[11] M. Ayuso, M. Guillen, and J. P. Nielsen, "Improving automobile insurance ratemaking using telematics: incorporating mileage and driver behaviour data," *Transportation*, vol. 46, no. 3, pp. 735–752, 2019.

[12] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econometric Reviews*, vol. 29, no. 5-6, pp. 594–621, 2010.