TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DIPLOMARBEIT

Neural Networks in Insurance

Comparing neural networks with generalized linear models in solving the retention problem

ausgeführt am

Institut für
Stochastik und Wirtschaftsmathematik
TU Wien

unter der Anleitung von

Associate Prof. Dipl.-Ing. Dr.techn. Stefan Gerhold

durch

Victoria Zach, BSc
Matrikelnummer: 01525629

09.November 2020

Kurzfassung

Die immer stärker werdende Konkurrenz in der Versicherungsbranche macht viele Unternehmen auf die Wichtigkeit und Notwendigkeit eines Customer Relationship Management aufmerksam. CRM Systeme werden sowohl dazu verwendet, Neugeschäft zu generieren als auch die Beziehung zu Bestandskunden zu festigen und dadurch die Stornorate zu senken. In der Umsetzung werden vor allem Machine Learning Techniken verwendet, um das Verhalten der Kunden zu analysieren und vorherzusagen. In der vorliegenden Arbeit wird mittels statistischer Methoden und künstlicher Intelligenz versucht, die Stornoerwartung bei Hauhaltsversicherungen vorherzusagen. Vor allem bei langjährigen Verträgen ist es ein Wettbewerbsvorteil einschätzen zu können, wie viele und welche Versicherungsnehmer kündigen werden. Dadurch kann die Prämie dementsprechend angepasst bzw. die Kundenabwanderung aktiv durch gezieltest Marketing gestoppt werden. Für diese Prognose wird einerseits ein verallgemeinertes lineares Modell verwendet, andererseits werden - zu Vergleichszwecken - zwei verschiedene neuronale Netzwerke gebaut. Vergleicht man die Modellgüte, stellt sich heraus, dass in diesem Fall der komplexe Ansatz mit neuronalen Netzwerken nur geringe Verbesserungen in den Ergebnissen liefert.

Abstract

With increasing competition, insurance companies are nowadays especially interested in customer relationship management. These systems are used to attract new customers as well as establish a continous relationship with the existing customers to increase the retention rate. The implementation process often uses machine learning techniques to analyze and predict customer behaviour and therefore the churn rate probability.[27]
In this thesis, the goal is to predict customer churn in household insurance policies using statistical models and artificial intelligence.
Especially with longterm contracts, it is an competitive advantage to know who and how many customers will cancel their policy. With that knowlegde it is possible to adjust the premium and to develop specific marketing strategies to prevent customer retention.
In the prediction process, we will on the one hand use a generalized linear model (GLM) and on the other hand we construct two different neural networks.
Comparing the results e.g. the performance of the models, there is a slight improvent-ment in using more complex models like neural networks.

Acknowledgments

This thesis would not have been possible without the support of many people and institutions. Firstandforemost, I would like to express my gratitude to my supervisor, Associate Prof. Dipl.-Ing. Dr.techn. Stefan Gerhold for his patience and support.
I would like to extend my appreciation to VAV Versicherungs- Aktiengesellschaft for the great cooperation and support.
I am also very grateful for the help of Meyerthole Siems Kohlruss, especially Carina Götzen and Verena Brökelmann regarding the data preparation.

Declaration

This thesis is submitted to the Technical University in Vienna in fulfillment of the requirements of the degree of Master of Science. This thesis represents my own original work towards this research degree and contains no material which has been previously submitted for a degree or diploma at this University or any other institution, except where due acknowledgement is made.

Date

Signature

Contents

1 Introduction

Especially in the insurance sector, customers are the basis of the success. That and because of the rising competition is why many companies are becoming aware of the importance of customer relationship management (CRM). Customer satisfaction and retention is a big challenge, which CRM systems are trying to deal with in using machine learning techniques to analyze and predict the customers behaviour.
These models can help generating new business as well as predict customers who are likely to churn based on demographic, personal and behavioural data. Therefore a customer - orientated marketing strategy can be developed to gain customer retention.[27]
The main objective of Customer Relationship Management is retaining already existing customers, because it is - depending on the business domain - five to twenty times more cost - effective than acquiring new customers.[27]
It includes all activities helping to prevent customer churn and to gain and retain customers loyalty. The churn rate can be predicted using data mining and predictive analytical models.
The literature suggests various categories of prediction techniques such as decision tree-based, Regression analysis, Support Vector Machine, Bayesian algorithm, Instance – based learning, Ensemble learning, Artificial neural network, and Linear Discriminant Analysis. [27]
Many studies compared different statistical and machine learning techniques in order to predict the churn rate.
In [1] Decision trees, Linear Regression and a Neural network are compared. The performance was slightly higher using neural networks.
In this thesis, the hypothesis, that neural networks perform better than common statistical models such as GLM will be tested.
First, Customer Relationship Management will be explained and will create awareness of the retention problem introduced in chapter three. Further, the data structure and possible dependencies are analysed and the concept of the compared models GLM

and Neural Networks is explained .
In the application process, a generalized linear model and two neural networks are computed to predict the churn rate. For the GLM, ten characteristics will be considered to specify the linear predictor. These features are normalized variables that include claim history, contract details and demographic information.
To model the stochasticity in our dataset, the glm() method in R with "family=binomial" and the logit link function will be used.
The model will be applied on 80% of the data, the other 20% will be used to test the prediction.
Computing the Receiver Operating Characteristics(ROC) and the therefore resulting area under the curve(AUC), we can evaluate the performance of the model. The resulting scalar value of 0.65 can then be compared to the neural network´s AUC value.
The structure of the first neural network will be four hidden layers containing six, four, two and one hidden nodes. The second model will contain only three hidden layers with five, three and one hidden nodes.
The algorithm used in both networks will be "rprop+", the resilient backpropagation algorithm and the activation function will be set to logistic. Using the same features as in the GLM modelling, the R-function neuralnet() creates our neural networks with the given structure for the training set.
To see how good the predictions are, the models will be applied on the test set to compare the predicted values with the actual output, e.g. the information if the policy is still active.
The performance of the neural networks can be evaluated using ROC and AUC as in the previous GLM model. Comparing the results, both neural networks performed slightly better (AUC ~ 0.66) than the generalized linear model.

2 Customer Relationship Management (CRM)

Relationship management was first introduced in 1983 by Berry and was developed through various researchers like Mattsson or Rashid. As Ansari and Riasi describe in their study about modelling and evaluating customer loyalty using neural networks "relationship management is about satisfying both firms and customers through identifying, establishing, maintaining, enhancing and terminating relationships with customers". Therefore it is important, that companies introduce communication and interaction processes to make their CRM easier.[6]

2.1 The Business Problem

Especially in the insurance sector, the competition is enormous and without loyal customers it is impossible for companies to survive. That is why customer loyalty is the key to a successful business. Experts even describe customer loyalty as "the marketplace currency of the twenty-first century". Researchers showed that loyalty affects the profitability of a firm and can lead to a reduction in marketing and customer acquisition costs. [6] CRM systems use business intelligence and analytical models to identify the most profitable consumers. Therefore it supports the marketing in creating the most cost-effective relationship with customers. This can lead to a higher customer satisfaction and higher retention rates.
Using personal, demographic and behavioural data, models can predict, who will be likely to churn and who will probably renew the contract. The four main stages of the business - customer relationship are:

- identification
- attraction

- retention
- development

AI and especially advanced machine learning are among the top strategic technologies organisations use to improve or reinvent their business.
There are many insurance business areas, where machine learning can be used. In customer services, machine learning plays already a significant role. A recent survey showed, that customers are happy to get computer-generated insurance advice and the well known chat bots are already used in many organisations to improve the customer service and therefore the customer relationship. [27]

2.2 Customer Relationship Management using Machine Learning

Especially in the insurance industry, data is playing an important role and in the last few years more data is created than the whole human race has ever done. Most insurance companies are overwhelmed of this explosion of data and statistics show that only 10-15 percent of the accessible data is processed. It requires data science techniques to analyse the hidden unstructured data. Because machine learning techniques can be applied on structured as well as semi structured or unstructured data, the predictive accuracy is often higher. It is mostly used to bring order into the data and to understand customer behaviour.[22]

Machine learning is a very common used term, which basically includes every computer program that can "learn" by itself. A question one might ask is why we need machines that have to learn at all. Why not program machines to get the desired performance immediately?
There are several engineering reasons:
There are a lot of new events happening in the world, new knowledge is discovered and the environment can change. Some of that can be captured by machines, but nevertheless redesigning is often necessary.
Humans are often not able to recognise relationships between the input and output variables. Machine learning techniques can help recognise and extract hidden relationships and correlations.[20]

Machine learning usually refers to changes in the systems associated with artificial intelligence (AI). These changes can be enhancements to already existing systems or synthesising a new one for example.[20] The well known spam filter or the product recommendations on the internet, nearly every predictive programm is based on machine learning algorithm.[**Wehle**]

There are numerous possible applications for machine learning techniques in insurance companies: Fraud detection, expense management, litigation or understanding risk apetite and premium leakage for example.[22]

2.2.1 The Terminology of Machine Learning

It can be distinguished between the two major types of machine learning: the *supervised* and the *unsupervised learning*. In the first method, the trainer feeds a network with input data until the output is close enough to the desired output. The output values are therefore given, but often only approximately. An example for this type is the MLP design. In finance it is often used to forecast stock market prices. The unsupervised learning is used to explore important properties of the data for example looking for the difference between good and defaulting costumers in bank data. There, *self organizing features maps (SOFM)* are often used as neural networks.[16] Some methods can not be assigned to one of these types, they are intermediate between the supervised and the unsupervised learning methods. This type is sometimes called the *speed up* method. Here, deductive processes are often involved. [20]

2.2.2 Challenges in Implementing Machine Learning

When adopting machine learning, there are typical challenges insurers have to face:

1. Relevant data
   The data quality is very important when using machine learning techniques. The training dataset should be representative and balanced, many insurance companies struggle to provide right and relevant data.

2. Dataset requirements
   The AI powered systems require a huge amount of data to train the model. To

cover all scenarios, insurers have to provide a lot of documents and transactions, which can be very hard.

3. Difficulty in planning and budgeting
   It is hard to predict the return on investment of machine learning for the company. Funding needs can change during the project and the improvements can only be guessed. Therefore, it is not easy to convince investors to implement machine learning techniques.

4. Data security
   The huge amount of data provided for the machine learning algorithm contain a lot of personal informations,and therefore create a high risk in case of data leaks and security breaches. Many insurers are too afraid of taking this additional risk. [22]

3 The Customer Retention Problem

Customer retention and churn prediction already gained much importance in many business domains like banking, telecommunication or cloud services subscriptions. Customer churn prediction means forecasting customers moving to a competitive organization or company because of better quality of service or benefits for example. The churn rate is a very important indicator, because losing customers with low loss ratios will probably lead to a loss of businesses and therefore a reduce in profit.

To effectively control customer churn, statistical and data mining techniques can be used to construct a customer churn prediction model. The data mining techniques for example can be used to discover relationships or patterns in the data and classify the behaviour through a specific model. To predict potential customers, that are likely to churn, different data mining techniques are proposed, the popular ones are:

- neural networks
- support vector machines
- logistic regression models

[4] Especially AI and advanced machine learning are among the top strategic technologies organisations use to improve or reinvent their business.

3.1 Methology

Many different statistical or machine-learning models can be used to achieve a reduction in the churn rate.
In several data mining studies, researchers suggest using machine learning techniques such as neural networks, especially for non parametric datasets. These techniques

often outperform traditional statistical methods such as linear and quadratic discriminant analysis approaches. [4]
In the study "Machine-Learning Techniques for Customer Retention: A Comparative Study" Sahar F. Sabbeh compares neural networks with logistic regression and decision trees. Within two comparisons, the neural networks always outperform the logistic regression and are slightly better than the decisions trees. Therefore we will focus on the prediction using neural networks and compare them to the results from a linear regression model.[27]

3.2 Dataset

3.2.1 Factors affecting Customer Loyality

Customer loyalty can be seen as a two dimensional concept containing attitudinal and behavioural loyalty. Both dimensions should be considered for proper investigations. Several studies focus on the link between customer satisfaction and loyalty on the one hand and the firm´s profitability on the other hand. Researchers state that customer loyalty depends on customer satisfaction and can have an impact on the firm´s profitability.
Regarding loyalty, we use the most complete definition from Richard L. Oliver´s Whence Customer Loyalty (1999), where he describes it as "deeply held commitment to re-buy or re-patronize a preferred product/service consistently in the future, thereby causing repetitive same-brand or same-brand set purchasing, despite situational influences and marketing efforts having the potential to cause switching behaviour".[6]

To choose significant inputs for our model later, we want to know the factors, that affect customer loyalty. In fact, we will use the software ARIANE to recognise features that influence customer´s decisions to resign.

3.2.2 Input and Output Vectors

In general, there are three main types of values for the components of a input vector:

1. real-valued numbers

2. discrete-valued numbers
3. categorical values

There can be a mixture of these types as well and categorical values might be ordered or unordered.
The factors for the input vectors in the following models will be:

1. sales characteristics:
   - sales channel
   - payment method
2. claim characteristics:
   - amount of previous damage
   - amount of damage-free years
   - information if previous year is claim-free (including claims without any payment)
3. regional characteristics:
   - regional zoning
4. contract characteristics:
   - begin of contract
   - duration of contract
   - deviation from the average premium
   - age of the policyholder
   - insurance segments

To simplify the dataset, we will cluster the attribute values:
begin of contract

- before or at 2010
- after 2010

customer age

- till 30 years
- till 45 years
- till 60 years
- till 75 years
- older than 75 years

duration of the contract

- 5-years contract
- 10-years contract
- other

deviation from the average premium

- higher than average premium
- lower than average premium

information if previous year is claim-free

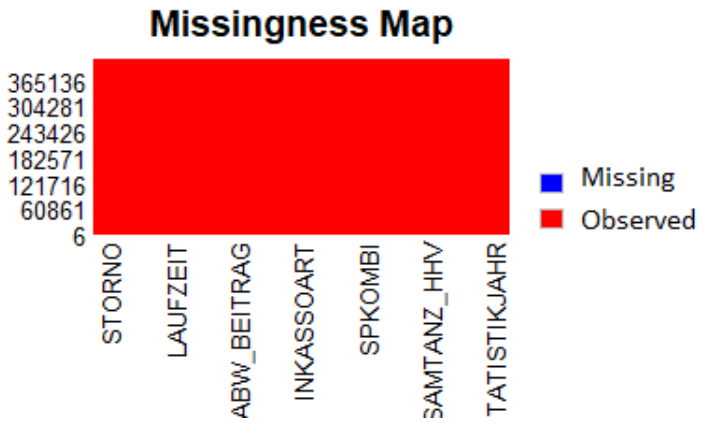
- unknown
- claim without payment in the last year
- claim without payment and claim with payment last year
- claim with payment in the last year
- no claim in the last year

amount of damage-free years

- unknown
- one year damage-free
- two years damage-free
- three years damage-free

- four years damage-free
- five years damage-free
- six years damage-free
- seven years damage-free
- begin of contract in statistic year
- claim in previous year

Before we start modelling, we have to look at the quality of the data. In insurance policies, some attribute values are often missing. Using the function *missmap* in R on the dataset, we get the proportion of known to unknown data [26]:



In the two following models, the output will be zero or one, depending on whether the customer will cancel the contract or stays at the insurance company the next year. In the model, we will just focus on cancellation through the policyholder and not through the insurance company itself. This includes:

1. Policyholder withdraws the contract
2. Policyholder cancels the policy at expiry
3. Policyholder cancels the policy in case of claims
4. Cancelation due to change in ownership
5. other reasons

We will look closer at the data, to find any a priori connection between any of the features and cancellations.

Correlation between the input vectors

Correlation is a measure of dependence that is used to better understand the data and to discover relationships within the dataset. The correlation coefficient can either be positive, neutral or negative. A positive/negative value means the two variables move in the same/opposite direction, a neutral coefficient means there is no relationship at all. There are different methods in measuring the correlation, depending on the structure of relationship:

- **Linear relationship** (Gaussian distribution) In this case, the dependence between two variables can be calculated using *covariance*, an expected value of the product of the difference of the values of a sample and their mean:

  $COV(X,Y) = sum((X - mean(X)) * (Y - mean(Y))) * (1/(n-1))$
  with sample X, Y, mean function $mean()$ and sample size n.
  With this indicator, we can calculate the **Pearson´s Correlation** as the normalization of the covariance by dividing it by the product of the standard deviations of the samples:

  $CoRR(X,Y) = COV(X,Y)/(SD(X) * SD(Y))$

- **Unknown structure of the relationship**

  In the case, where the variables are connected through a non-linear or unknown relationship, the **Spearman´s Correlation** is used to summarize the strength of the dependence of our variables. Instead of making distributional assumptions and applying the covariance function on the sample itself, this indicator is calculated using the relative rank of the values of the sample[8]:

  $CORR(X,Y) = COV(rank(X), rank(Y))/(SD(rank(X)) * SD(rank(Y)))$
  where $rank()$ ranks the variables and $SD()$ calculates the standard deviation of the rank variables.

In our calculations, we will use the second tool to describe the connections within the data due to the unknown distribution of the values.

With the knowledge of the relationship of the input variables, it is possible to improve statistical algorithm as linear regression by removing one of the highly correlated input variables. In our case, there is a strong relationship between SFOZHHV and SFHHV. Therefore we will not use the information if previous year is claim-free (including claims without any payment) (SFHHV) for further analysis and just look at the amount of damage-free years (SFOZHHV).

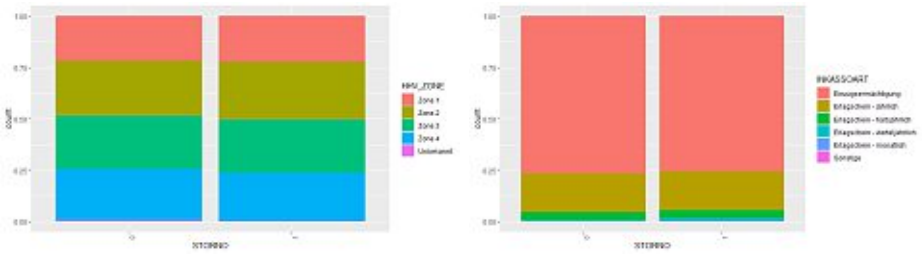
It is also interesting to look at the correlation between the input and the output variables to provide insight in what variables are relevant for developing our model. Looking at the numbers in the correlation matrix, it will not be necessary to consider the sales channel (VERTRIEB_CL) any longer.

Relation between Input variables and customer churn

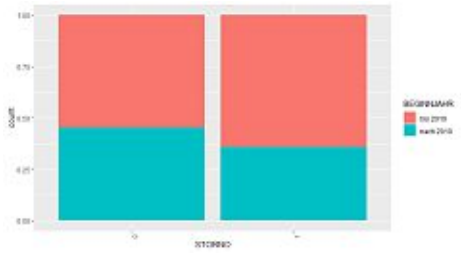
We already know from the correlation matrix, on which variables customer churn is depending. But to see, which customers are more likely to churn, we will look at the distributions of the variables. In the following images, the output zero includes all customers who are still at the company, where one indicates churn.

Relation between customer churn and regional zoning and payment method

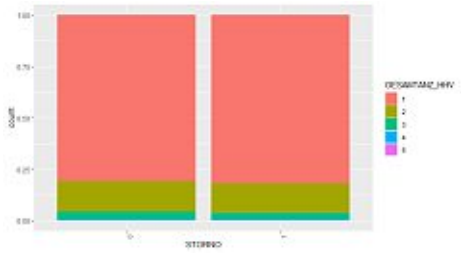
The parameter-value is not significantly different for customer, who cancelled the contract.



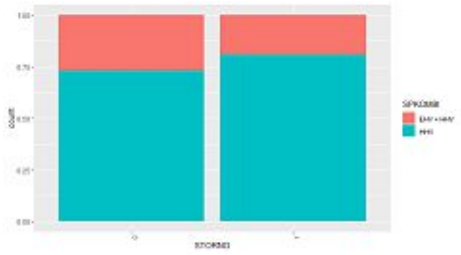
Relation between customer churn and begin of the contract As demonstrated in the plot, mostly contracts, that start before 2010 got canceled.



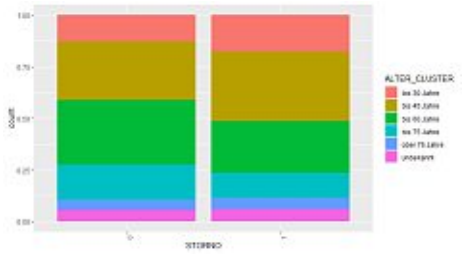
Relation between customer churn and amount of contracts The parameters have almost the same distribution for every level of our churn variable.



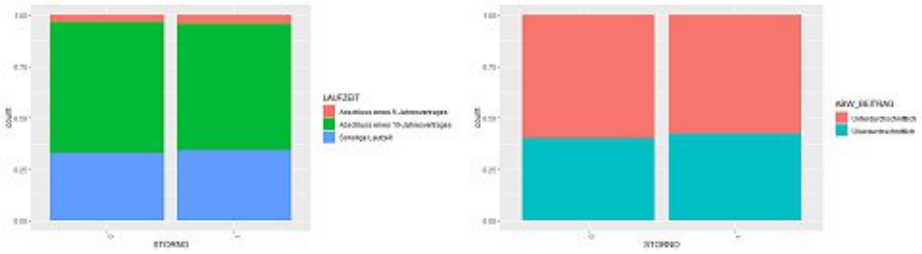
Relation between customer churn and insurance segments These variables are lightly dependent in the way that the portion of customers with only household contracts is higher in the churn dataset.



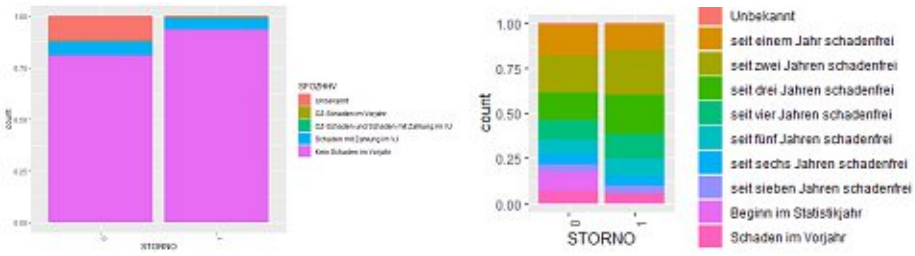
Relation between customer churn and customer age The distribution for the churn dataset looks slightly different then the distribution of the existing portfolio in the way that the percentage of people younger than 45 years is higher.



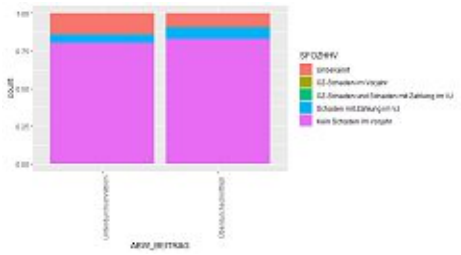
Relation between customer churn, duration of the contract and premium level These plots are showing, that customer churn does only slightly depend on the premium level or the duration of the contract.



Relation between customer churn and claims There is a significant relation between cancellations and the amount of claims in the previous year.



Looking at the distributions, we can see that there are a lot of "good" customers, meaning customers without any claim in the last years who cancel the contract. Our goal therefore is to prevent these kind of churn. Especially if we look at the relationship of the average premium and the claim-free years, it shows that a big portion of these claim free customers are paying a high premium. Therefore, preventing churn in the right way is important because money and customers can be saved.



3.2.3 Training and Testing Set

The training set will be a randomly chosen 80% part of the dataset using a dummy variable:

$$\text{dummy\_sep} < -rbinom(nrow(data_storno), 1, 0.8)$$

For our model to learn best from the data, we will try to get an equally distributed set regarding the churn variable, meaning that we will not use all the data with churn variable one. The dimension of the training set is now instead of 331870 89259. To validate our model, we use part of the dataset as test set and apply our model.

3.3 Retention Modelling using Generalized Linear Models

Using machine learning algorithm, a rudimental one is the linear regression. In general, regression is a method using independent variables to model a target value. This technique is mostly used to forecast or find out relationships between variables. In a linear regression algorithm, there is only one independent variable and the relationship between the independent variables (x) and the dependent variables (y) is linear: $y = k * x + d$.[13] The aim of a linear regression is to find the best values for k and d. To better understand how regression models works, we will first introduce some main properties:

Linear Predictor

The predicted value for the observed value y is obtained by transformating the value resulting from the linear predictor: $\eta = \sum_1^p x_{ij}\beta_j$ with explanatory variables x_j and

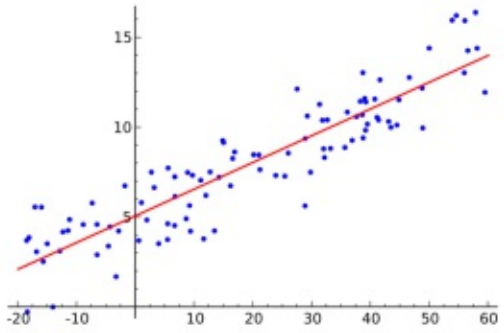


Figure 3.1: Linear Regression
[1]

the unknown parameters β, which will be estimated from the data.
In the generalized linear model, the linear predictor will be evaluated for each response value and compared to the transformed value of y to determine the fit of the model. The transformation process is explained via the link function further below. To get back to the original scale of measurement of the output variable, the reciprocal of the link function is applied.[10]

Cost Function

In order to find the best values for k and d, we convert this problem into a minimization problem, where we want to minimize the error between the predicted and the actual value:

$$J = \frac{1}{n}\sum_{i=1}^{n}(predicted\ value - actual\ value)^2 -> \min$$

This function measures the squared error over all data points divided by the number of data points, therefore named *Mean Squared Error (MSE)*. The equation is equivalent to the assumption that y are realizations of normally distributed variables with mean μ and constant variance.[15] Even if this approach is the most common one, the model can also be fitted using other ways and norms.The error structure is directly defined through the distribution family and can therefore also be poisson, binomial or other.[10]
Knowing this cost function, we will change k and d as long as the error is not small enough. Now the second important concept for linear regression algorithm is used.[13]

[1] https://www.educba.com/data-science-algorithms/

Gradient Descent

Gradient Descent is a method to change the values of k and d in order to reduce the cost function MSE. The starting points are some randomly chosen values for the unknown variables k and d. To reduce the MSE, we change the values iteratively depending on the learning rate.

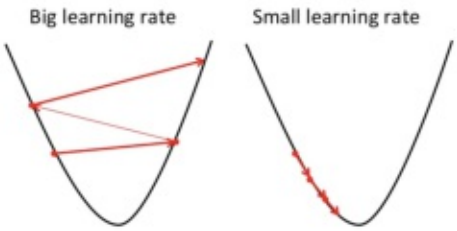


Figure 3.2: Gradient Descent
[2]

The learning rate decides how fast the algorithm converges to the minimum. To update the variables, we neet the partial derivatives from the cost function with respect to k and d. Derivation of the formulas with predicted value:$= k * x_i + d$ and actual value $:= y_i$:

$$\frac{\partial J}{\partial d} = \frac{2}{n}\sum_{i=1}^{n}(k * x_i + d - y_i)$$

$$\frac{\partial J}{\partial k} = \frac{2}{n}\sum_{i=1}^{n}(k * x_i + d - y_i) * x_i$$

Updating the variables using partial derivatives and the learning rate α[13]:

$$k = k + \alpha * \frac{\partial J}{\partial k}$$

$$d = d + \alpha * \frac{\partial J}{\partial d}$$

Although linear regression models are easy to explain, fast to train and can be

[2] https://machinelearning-blog.com/2018/01/24/linear-regression/

applied for small datasets, there are some weaknesses. The underlying assumptions, especially the normality assumption. Normal distributed data are by definition continuous, symmetrical and the data should be defined over the entire number line. As a consequence, discrete or asymmetrical data should not be modelled using linear regression.[14]

To extend classic linear models and overcome these restrictions, we will use **Generalized Linear Models**.

The first generalization allows our random components to be exponential-, normal-, binomial-, poisson-, gamma or inverse gamma distributed. In fact, the distribution can be any of the "exponential family", which includes basically every common probability distribution.[15]
Because now the output variable can be defined in a different interval than a normal distributed variable or even be categorical, we need a link function that allows us writing the output as linear combination of the input variables.

3.3.1 Link Function

The link function $g()$ transforms a real number into a probability number and therefore provides a relationship between the predicted value and the mean of the distribution function. In the simple linear regression model, the link function is just the identity operator. In the generalized linear model, the link function can be any monotonic differentiable - for example a log or power - function.[15]
In our model, we will use the logit-link function:

$$g(\theta) = \log \frac{\theta}{1-\theta} \tag{3.1}$$

with $g(\theta) = \beta_1 * x_1 + \beta_2 * x_2 + \beta_3 * x_3 ..... + \beta_p * x_p$
where the parameters β describe the importance of the characteristics x and θ is the churn probability $P(Y = 1)$.
Therefore if β is greater than zero, the churn probability rises.

In our generalized model, the traditional way of measuring discrepancy (MSE) is not possible anymore. [15]

3.3.2 Discrepancy Measure

It is unlikely that the fitted values will match the values from the data perfectly. A small discrepancy will be tolerated, but since a large discrepancy indicates an inadequate model it will not be tolerated at some point. A measure of the goodness of the model is deviance: [10]

Deviance

Deviance is defined over the difference in log likelihood of the saturated model and the fitted model as:

$$D = -2\phi(l_{mod} - l_{sat})$$

The saturated model is the model, that fits the data perfectly, where the number of parameters is equal to the number of observations. The minimization of the discrepancy measure is here equivalent to the maximization of the likelihood.[10] To evaluate the quality and the performance of the model, there are also other measures such as ROC and AUC, which will be explained and apllied in the application process.

3.3.3 The Application Process

We will use R to construct the GLM with regards to the churn prediction. The considered variables are the relevant (due to correlation) ones from our dataset, e.g. the dataset without the attribute values of the sales channel and the amount of claim-free years. Therefore the formula - the symbolic description of the model to be fitted - for the model is:

```
FORMULA <- STORNO ~ STATISTIKJAHR + GESAMTANZ_HHV + SFOZHHV
+SPKOMBI + INKASSOART + HHV_ZONE + ABW_BEITRAG + ALTER_CLUSTER
+LAUFZEIT + BEGINNJAHR
```

To further specify the model, we set the generic function *family* to describe the link and error function.
The binomial families allow the *Logit* link function. We fit our generalized linear model on our training set *data_train* using the command *glm* in R:

```
model <- glm(formula,data=data_train, family = 'binomial')
```

The output of the model coefficients/weights of the linear function:

```
                                                Estimate
(Intercept)                                     -4.70641
STATISTIKJAHR2013                                0.01960
STATISTIKJAHR2014                                0.02949
STATISTIKJAHR2015                                0.07092
STATISTIKJAHR2016                               -0.22662
STATISTIKJAHR2017                               -0.07192
GESAMTANZ_HHV2                                  -0.15224
GESAMTANZ_HHV3                                  -0.13875
GESAMTANZ_HHV4                                  -0.38851
GESAMTANZ_HHV5                                  -0.05929
SFOZHHVOZ-Schaden im Vorjahr                     3.29984
SFOZHHVOZ-Schaden und Schaden mit Zahlung im VJ -8.17442
SFOZHHVSchaden mit Zahlung im VJ                 2.95903
SFOZHHVKein Schaden im Vorjahr                   3.06863
SPKOMBIHHV                                       0.44656
INKASSOARTErlagschein - jaehrlich                 0.01392
INKASSOARTErlagschein - halbjaehrlich             0.09240
INKASSOARTErlagschein - vierteljaehrlich          0.05007
INKASSOARTErlagschein - monatlich                2.19712
INKASSOARTSonstige                               0.83326
HHV_ZONEZone 2                                  -0.09911
HHV_ZONEZone 3                                  -0.18873
HHV_ZONEZone 4                                  -0.36293
HHVZONEUnbekannt                                -0.37412
ABW_BEITRAGUeberdurchschnittlich                  0.09777
ALTER_CLUSTERbis 45 Jahre                       -0.27007
ALTER_CLUSTERbis 60 Jahre                       -0.69571
ALTER_CLUSTERbis 75 Jahre                       -0.78777
ALTER_CLUSTERueber 75 Jahre                      -0.51541
ALTER_CLUSTERunbekannt                          -0.60275
LAUFZEITAbschluss eines 10-Jahresvertrages      -0.17067
LAUFZEITSonstige Laufzeit                       -0.04003
BEGINNJAHRnach 2010                             -0.24220
```

3.3.4 Results

With the generated weights, we can apply our model on the test set to predict the churn rate, e.g. which customer will cancel the contract.

```
predict <- predict.glm(model,data_test,type ="response")
```

To evaluate the prediction, we create a $2x2$ *confusion matrix*, a table that compares the predicted values with the actual values from our test set. Because the predicted value is any number greater than zero, we have to apply a threshold first to classify the data:

$$Predict = \begin{cases} 0 & \text{if } Predict \leq threshold(0.2) \\ 1 & \text{if} Predict > threshold(0.2) \end{cases}$$

Here any other threshold can be used, in this application 0.2 leads to the best results concerning the prediction of the churn rate:

Storno/Predict	no	yes
no	80171	701
yes	2103	61

Dividing the number of the correct predicted values thorough the number of all predictions, we get the accuracy of our model:

$$accuracytest = \sum(diag(confusionmatrix))/\sum(confusionmatrix) = 0.9662315$$

To evaluate the quality of the generated model, we compute the receiver operating characteristics (ROC) and the therefore resulting area under the curve(AUC).
The receiver operating characteristics graph can be used to visualize the performance of the model and as an technique for selecting classifiers. To compare several models in regards to their performance, it is easier to reduce the ROC graph to a single scalar value: the area under the curve.
This method returns the expected performance for each model as a number between zero and one, realistically higher than 0.5. [12]
Looking at the results of our computed generalized linear model, there is an AUC of 65%.

3.4 Retention Modelling using Neural Networks

Due to the ability of distinguishing policy holders who are likely to terminate their policies from those who tend to renew the contract, neural networks are ideal for customer relationship management, especially for the retention problem. The trained network can be easily applied to unseen data in order to predict the relationship between company and customer.

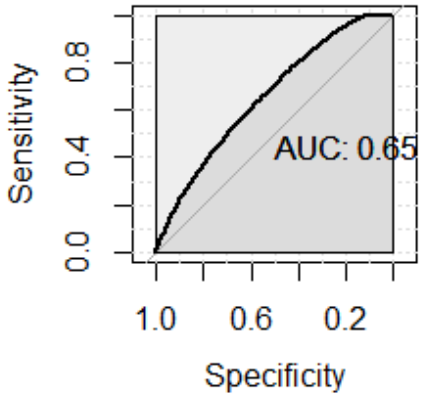


Figure 3.3: ROC GLM

Yeo1, Smith1, Willis1, Brooks2 [2] and many other researchers have already applied neural networks understand the factors affecting a customers decision to churn and have successfully solved the churn prediction problem. [2]

3.4.1 Neural Networks

Artificial neural networks (ANNs) are flexible statistical models containing a learning process. NNs are using artificial intelligence and are designed for forecasting or analysing data. One of many advantages using neural networks is that the do not have any distributional assumptions about the data, but the methods are very data-intensive. They require an immense amount of historical prices to get a well-trained network. Networks can be used for both learning and estimating a pricing formula, especially for derivative securities with highly non-linear pricing formulas.

3.4.2 The Topology of a Neural Network

A neural network typically consists of many different layers like the *input layer*, the *hidden layers* and the *output layer*.

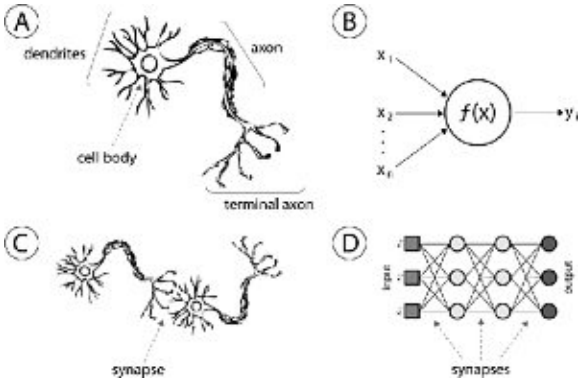


Figure 3.4: Neural Network Topology
[3]

The leftmost layer is the input layer, and the neurons there are called input neurons. The layer at the right side of the network is called output layer, containing the output neuron(s). Between the input and the output layer are the hidden layers. It is possible to have only one or several hidden layers as in the MLP and they are not directly connected to the environment. The more hidden layer the network has, the more complex it is.
The basic ingredient of every layer is a artificial neuron. The model of a neuron mimics the human brain in the way it processes information. It consists of many highly connected elements, the neurons, containing three major components for the artificial intelligence: the dendrites, the cell body and the axon. The dendrites can be seen as the inputs feed into the cell body, our processing stage.The output can be feed into another neuron similar to the axon and his terminals.
What is happening in the information processing stage be explained in three steps:

1. Inputs and weights
   First, when a signal comes in, the inputs will be scaled up or down through particular weights. These weights can be adjusted through the learning process.

2. Adding a bias
   Now, all the adjusted values will be summed up to a single value. An Offset, called the bias is also added to the sum. The bias will be adjusted through the learning phase.

[3] https://qph.fs.quoracdn.net/main-qimg-8a95fc0ca8142508da75bcb90f8b3ff2

3. Activation function
   The result from the second stage will be feed into an activation function and turned into an output signal.[5]

The key characteristic of neural networks is that they have to be trained for a specific application. In the beginning, there are just random weights assigned to each neuron. The data sample, which contains a large number of observations is then divided into two data sets - the training set and the test set - which is used to evaluate the accuracy of a NN. The number of training runs is depending on the complexity of the network and the rate at which the NN learns is very important to not miss any patterns or waste time. After the connection weights between the neurons have adjusted enough - meaning the network learned the patterns in the dataset - the test data is used to evaluate the performance.[16]

Activation Functions

These functions are used to determine the value of the output, to map the output commonly in between -1 and 1 or 0 and 1. There are many types of activation functions.They can be divided into linear and non-linear functions. The non-linear activation functions are most used because it is easier for the model to differentiate between the outputs. We will start with the most basic form: a binary function. A neuron with such an activation function is called *perceptron*.

Perceptron A perceptron is a type of artificial neuron developed in the 1950s and 1960s by a scientist called Frank Rosenblatt. Nowadays there are more common neurons used like the *sigmoid neuron*, but to understand the concept we will first explain how perceptrons work.
The basic mathematical model is that the neuron gets several binary inputs ($x_1, x_2, x_{3,..}$) and generates a single binary output (y). [19]

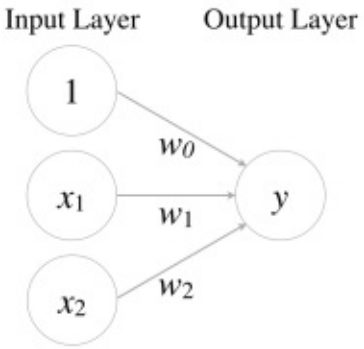


Figure 3.5: Perceptron
[4]

The perceptron in the example has two inputs x_1, x_2 and a bias w_0. To compute the output Rosenblatt introduced *weights* w_1, w_2 representing the importance or significance of the inputs to the output. The output is determined by the weighted sum $\sum_j w_j * x_j$:

$$Output = \begin{cases} 0 & \text{if } \sum_j w_j * x_j \leq threshold \\ 1 & \text{if } \sum_j w_j * x_j > threshold \end{cases}$$

The *threshold* value is a parameter of the neuron. Both weights and the threshold can be changed to get different models of decision-making.[19] The output will be fed into an *activation function* h. Therefore the neuron output a is of the form:

$$a = h(\sum_j w_j * x_j + b)$$

with bias b.

If there is just one neuron to which the inputs are fed into, the perceptron is called *single-layer perceptron*. These neurons are only able to solve linearly separable classification problems, therefore Rosenblatt and Widrow indroduced multilayer perceptrons to overcome these disadvantages. The number of coefficients is depending on the number of 'hidden units'. [21]

The problem with these neurons is, that a small change in the weights of bias of a single perceptron can cause a complete change in the behaviour of the network. But there is a clever way to overcome this problem: a new type of artificial neuron - *the sigmoid neuron*.

[4] https://schwalbe10.github.io/thinkage/2017/01/21/perceptron.html

Sigmoid neuron or logistic activation function This type of neuron is similar to the one above but has some modifications like small changes in weights and bias will only cause a small change in the output. That is very important, because a network of sigmoid neurons is able to learn.
To calculate the output of a sigmoid neuron, the *sigmoid function*

$$\sigma(x) = \frac{1}{1+\exp -z}$$

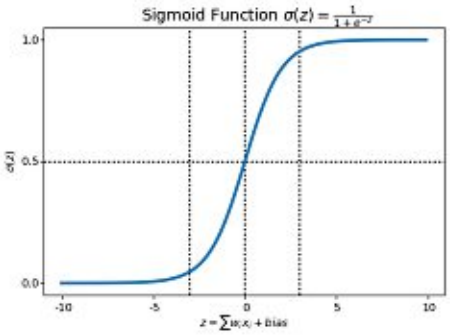


Figure 3.6: Sigmoid Function
[5]

is used.
with weights $\omega_1, \omega_2, ..$, inputs $x_1, x_2, ..$ and bias b the output is computed by:

$$\frac{1}{1+\exp -\sum_j \omega_j x_j - b}$$

The sigmoid function is smooth and therefore the output can take on any value between 0 and 1, not only be 0 or 1 as with perceptrons. But if sigmoid neurons seem completely different, there are some similarities to the perceptrons. If the weighted input z is a large positive number, then th output $\sigma(z) \approx 1$ as it is for perceptrons because $\exp -z \approx 0$. On the other hand if z is a large negative number, $\sigma(z)$ is approximately 0, therefore close to the output of a perceptron. [19]

Another S-shaped activation function is the *tanh*. It is similar to the sigmoid function but better, because negative inputs will be mapped negative. The range is from -1 to 1.

[5] https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learnin

There are a lot of functions, that can be used to determine the output. It would not make sense to state all of them, but one more should be mentioned, because it is the most used in the word right now: *ReLU (Rectified Linear Unit) Activation Function*.

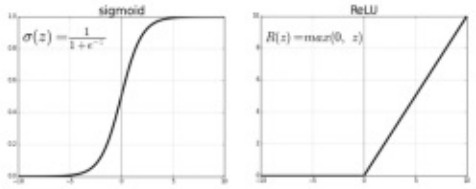


Figure 3.7: ReLU function
[6]

Rectified Linear Unit (ReLU) Activation Function The ReLU function is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero. Comparing to the sigmoid function, there is an important disadvantage. Any negative input will immediately turned to zero, and therefore the model will not be able to train or earn from the data properly. an attempt to solve this problem is the activation function named *Leaky ReLU*.

Types of Neural Networks

Nowadays there are many different types of neural networks, each with its unique strength as you can see in the figure below. Unlike with other algorithm, a neural network improve as it gets bigger and include more data. In this thesis, we will only explain a few important networks.

[6] https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

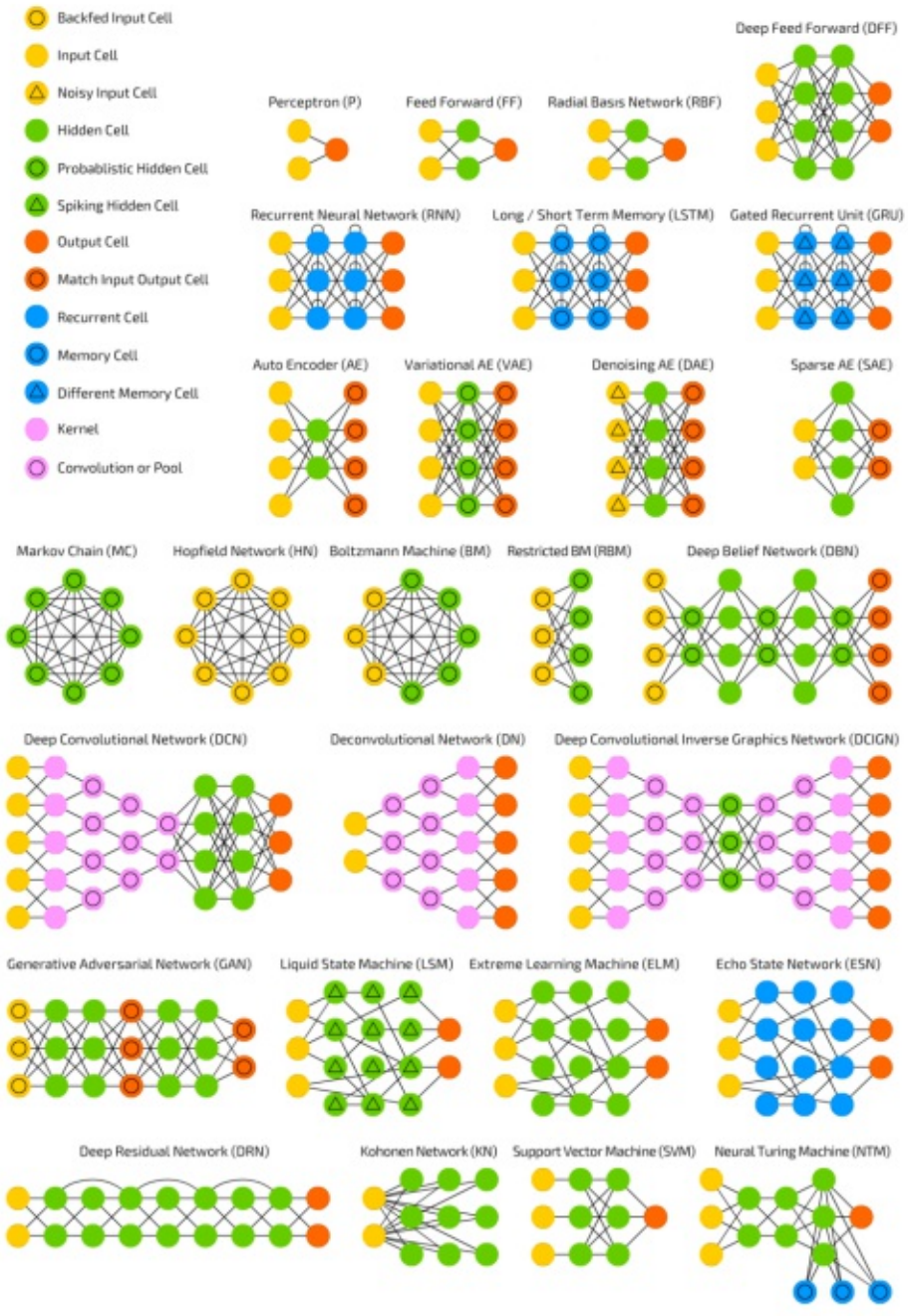


Figure 3.8: Types of Neural Networks
[7]

[7] https://www.asimovinstitute.org/neural-network-zoo/

Feed Forward (FF) Regarding the architecture of neural networks, there are two main categories: *feed forward* and *recurrent* neural networks.
If there is no connection between the output vector towards the inputs, i.e. if there is no 'feedback', then the network is called feed forward.
Characterized through the number of layers, he simplest form of a feed forward network is a single layer NN. If there are more than one hidden layer computed, it is called a *multi layer feed-forward neural network*. [28]

1. **Multilayer Perceptron(MLP)**
   In the MLP the signals within the network are transmitted only in one direction, from input to output using sigmoid neurons. This straightforward type of training a data set is called a feed forward network.[5]
   The trainer feds the data into the input layer(s) where they are processed and sent to the hidden layers. After sending the data to the output layer, the trainer can see the response, which will be compared to the desired output. If the results are not close enough, the desired output will be "backpropagated" through the network. Contrary to the feed-forward approach, now the desired output is the starting point and fed into the neural network at the output layer. The transmission i.e. the readjusting of the weights between the neurons, finishes at the input layer. This backpropagating process will be repeated many times until the actual response approximates the desired response good enough i.e. until the trainer is satisfied. [7]

   Because the backpropagation algorithm is essential for training a network, we will take a closer look at it in the next paragraph.

   Backpropagation Even if backpropagation was already introduced in the 1970s, it was not appreciated until the famous paper of David Rumelhart, Geoffrey and Ronald Williams about learning representations by back-propagating errors. In this paper, the authors discovered, that the backpropagation algorithm works far faster on several neural networks and solves problems, that where unsolvable until then.
   The new learning procedure repeatedly adjusts the weights in the neural networks to minimize the error, a measure of difference between the desired output and the actual output. The aim is to find learning rules to develop an internal

structure to achieve the desired input - output behaviour. If there is a direct connection between input and output vectors, it is relatively easy to adjust the weights so that the difference between the actual and the desired output is minimal. It gets more difficult if there are hidden units, which have to be activated under certain circumstances to achieve the perfect input-output behaviour.
In general, backpropagation is a algorithm who gives us detailed information about how changes in weights and biases changes the behaviour of the network.[11] Explaining backpropagation mathematically, we should begin with the notation:
To denote the weight for the connection from the k^{th} neuron in the $()l-1)^{th}$ hidden layer to the j^{th} neuron in the l^{th} layer, we will use ω^l_{jk}, for example:

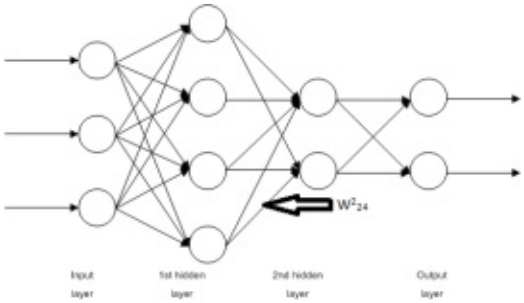


Figure 3.9: Backpropagation
[8]

Furthermore, we will define similar notation for the biases b and activations a: b^l_j defines the bias of the j^{th} neuron in the l^{th} layer and the same for the activation a^l_j:

[8] https://medium.com/@m.treerungroj/ten-machine-learning-algorithms-you-should-know-to-become-a

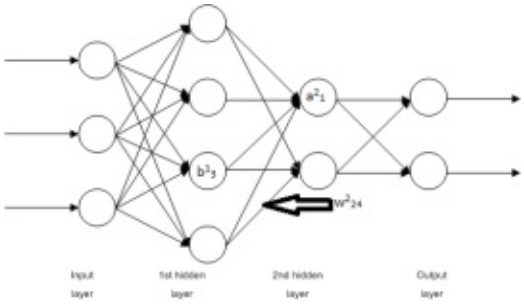


Figure 3.10: Backpropagation

The relation between the activations of different layers is expressed by the equation:

$$a_j^l = \sigma(\sum_k \omega_{jk}^l a_k^{l-1} + b_j^l)$$

We will rewrite this equation in vector or rather matrix form and apply the function σ element-wise. With this expressions, we get a way more global way of thinking about the relations of the activations in different layers. To simplify the notation we will define the weighted input z[19] :

$$z_j^l = \sum_k \omega_{jk}^l a_k^{l-1} + b_j^l$$

The goal of backpropagation is to find weights which ensure that the outputs from the network is the same as the desired output for each input vector. For a finite set of inputs and outputs, the total error of the network performance can be computed by comparing actual output y and desired outputs d:

$$E = \frac{1}{2} \sum_c \sum_j (y_{j,c} - d_{j,c})^2$$

c ...index over cases
j ...index over output units

To minimize the total error of performance E we use *gradient descent*, therefore we have to compute the partial derivatives of E for each weight in the network. There are two phases in computing the derivatives. Above we already discussed

the forward pass, in which the units of each layer are determined by the lower layers using the equations for the weighted input z and activations a. The second part is more difficult. In the backward pass the derivatives will be propagated from the top layer back to the bottom one. [11]

Hutchinson, Lo and Poggio tested the pricing and delta hedging performance of different networks on daily call option prices on S&P 500 futures from 1987-1991 and compared them to the Black Scholes model. Using the coefficient of determination, R^2 as a risk measure the Multilayer perceptron network (MLP) is a better choice compared to the radial basis function (RBF) and the projection pursuit regressions (PPR).Therefore we will focus on the MLP, which is now arguably the most popular form of neural networks.[5]

A more complex algorithm to use instead of standard backpropagation is the **Resilient Backpropagation (Rprop)**. It has two advantages in comparison to the common algorithm: faster training of the neural network and fewer parameters are required to be specified.[18]

2. **Radial Basis Functions (RBF)**
   Another type of learning network are the so called *Radial Basis Functions (RBF)*. At first, they were used to solve interpolation problems, but several researchers extended the RBF Formulation to perform more general approximation tasks. Poggio and Girosi (1990) show that the RBF can be derived from the classical regularization problem ($y = f(\vec{x})$) which then can be seen as the minimization problem of the functional:

$$H(\hat{f}) = \sum_{t=1}^{T}(\|\hat{y}_t - \hat{f}(\vec{x}_t)\|^2 + \lambda\|P\hat{f}(\vec{x})\|^{2)}$$

   with some vector norm $\|$, differential operator P , a given dataset $(\vec{x}_t, y_t)$ and a nonlinear function f(.) representing the conditional expectation of y_t given x_t.
   This type of networks can create any real-valued output, but if we consider some a priori knowledge it is more efficient to use a transfer function to reflect that knowledge. [21]

Recurrent Neural Network In contrast to feed-forward neural networks, there is an existing feedback from the outputs towards the inputs of other neurons or their own inputs. These types of networks are useful in handwriting or speech recognition for example.[28]

3.4.3 Learning Methods

The capability of "learning" distinguishes neural networks from conventional models. There are two main types on how we wish to learn a computational structure (function, system,..):

1. Supervised Learning
2. Unsupervised Learning

To simplify the terminology, we focus on learning a function f.

Supervised Learning

In the case of an already knowing, even if just approximately, function - we talk about supervised learning. We are trying to find a hypothesis h that closely agrees with the function f for the training set. A simple example for this method is curve fitting, where we want to approximate the values of f with another function h.
In our retention problem, we already know the function i.e. the information if the customer cancelled the contract or not - meaning the function is learned through supervised learning.[20]

Unsupervised Learning

In this type of learning, we simply have a training set but no function (desired output) values for them.
The goal of Unsupervised learning is to find and recognize patterns within the data and create natural partitions, mostly through measures of similarity. The simplest way of doing this is to define the distance between the patterns. Applying unsupervised learning is especially meaningful in problems, in which it is desired to invent classifies for categories in the data.
There are also methods intermediate between supervised and unsupervised learning,

for example if there is an existing function of the input vector but we try to find a more efficient one. [20]

3.4.4 Neural Network Applications in Finance and Insurance

ANNs can be applied in many different area in the finance and insurance sector. Using neural networks, it is possible to improve the performance of several existing technologies, especially in classifying data, modeling and forecasting, and signal processing. Classifying data is a widely used application of neural networks, for example in credit approval decisions, targeted marketing, stock picking by ranking the predicted performance, picking winning football teams and also in the automobile industry for testing electric motors or diagnosing problems with the engines. Replacing other statistically based systems, applying neural networks can lead to a performance improvement from 5% to 50%.
In the area of modelling and forecasting, NNs are used for developing relationships between several continuous input variables and one or more outputs. Examples of applications are interest rate or inventory level predictions, develop models for optimization performance or building models for designing electric generators. Technologies used are linear and polynomial regressions as well as auto-regressive moving average (ARMA) and Box-Jenkins. Why neural networks should be used instead can be shown by the research of Lapedes & Farber from the Los Alamos National Laboratories: They found that the neural network technique outperforms all existing techniques for forecasting chaotic time series.
In signal processing, it should also be thought of neural networks, especially back-propagating can improve the results of speech recognition and classifying sounds.

In general, neural networks are a breakthrough in approximating complex mathematical mappings. Therefore there is a wide range of applications that benefit from this breakthrough.[24]
In the next few pages, we will look at some important and actual application in the finance and insurance sector.

Stock Price Prediction

With historical data as input, neural networks can be designed to discern complex patterns and predict future stock prices or identify stocks, that outperform the market. [16] Already in 1993, Lawrence Kryzanowski, Michael Galler and David Wright showed that an artificial neural network can discriminate between stocks with positive return and stocks that provide a negative return. In fact, with a particular learning algorithm, the ANN could correctly classify 72% of the positive/negative returns. Especially because of the ability of relearning relationships automatically and recognizing patterns made it popular for financial applications. Due to the decline in the costs of computing, this method became attractive especially in north America´s investment firms. [17] Nowadays a lot of firms in the financial sector are experimenting with artificial intelligence. Especially multilayer perceptrons is by far the most popular and applied type of neural network in research and practice. It is possible to derive the hedging parameters from the sigmoid function, but it is more reasonable to use two output nodes instead, one for the delta and one for the vega. [3]

Security Trading Systems

"Artificial intelligence is to trading what fire was to the cavemen." - that´s how an industry player described the importance of AI. A well known example for a user of neural networks for security trading systems is Brad Lewis of Fidelity Investments. He outperformed the market by developing and relying on ANNs to pick stocks and manage $ 2 billion of Fidelity funds. While the market went up by 71% since 1988, he registered a return of $99,8\%$ over the same period.
Deere & Co is another successful user of artificial intelligence. They manage part of their pension fund and they have been able to generate a return 3% over the S&P 500 since using neural networks.
Especially on trading stocks of small companies, the technologically advanced trading systems have a huge impact. Nowadays there are a lot of companies using artificial intelligence for smarter trading using an AI platform to recognize complex trading patterns. Kavout uses this technology to recommend daily top stocks, Auquan can help data scientists to solve investment challenges, Epoque for finding and analysing potential trades and many more.[30]

Bond Rating Prediction

A common problem in financial investments is the classification of bonds based on the probability that the company may default. One way to solve this problem is to use artificial intelligence, because they do not need any prior assumptions. Dutta and Shekhar compared the results of neural networks with a multiple linear regression problems using different numbers of hidden layers and financial parameters.The input variables can be for example:

- Liabilities
- Debt ratio
- Sales / Net worth
- Profit / Sales
- Financial strength
- Earnings / Fixed costs
- Past five years revenue growth rate
- Projected next five years revenue growth rate

The output is the bond rating. No matter how many parameters they used, applying neural networks to bond ratings showed that these models outperform linear regression problems.[29] Especially with fewer grouping, the neural networks are more accurate in predicting bond rating changes. Even if the model misclassified a bond, it is of at most one rating class. In contrast the linear regression problems are off several ratings classes sometimes.
It is possible to apply unsupervised or supervised neural networks. If there is enough target (teacher) data, it is usually more effective to use a supervised learning approach. Backpropagation is used in several research studies. [23]

Foreign Exchange Markets

Investing in foreign exchange is a time consuming and complicated process. Subjective judgement is the main cause of risks. Using a machine and algorithm to analyse objective and update the data constantly is the best way to reduce this risk. Humans

are still necessary to build and maintain the software, but a lot of time and energy can be redirected to other important tasks.
Although NASDAQ estimates that over $5 trillion of many banks and financial institutions are traded in forex (foreign exchange) every day, AI is relatively unimportant in this sector compared to in comparison to other sectors of the finance industry.[25] Nevertheless, there are some applications of NNs for the use in currency trading. Olsen & Associates of Switzerland developed fee-based applications for their customers and Refenes and Zaidi showed that with neural networks, it is possible to outperform some forecasts for exchange rate predictions of USD/DEM. [16]

Financial Distress

Neural networks are very successful in forecasting financial distress. According to the authors, they are well suited for the task because of the incompatibility of Multiple Discriminant Analysis [MDA] for recognizing financial distress patterns. More specific, MDA can correctly identify most healthy firms as viable but distressed firms can only be identified as such in less that 70 %. In Comparison, NN can identify both types of firms in more that 80 out of 100.
Although it is recommend to be cautious when using neural networks because of the danger of over-fitting and the much longer training phase.[16]

Other Applications

Falcon Asset uses neural networks to predict inflation rates and T-bills.The nine neural networks are helpful for their bond activities. After fifteen months of use, the systems showed an accuracy of over 90% for the inflation rate predictions and over 60% for the 30-day and 90-day T-bills. ANNs can also help detecting credit card fraud. Already in 1993, Visa developed a neural network system for fraud detection and saved therefore $ 2 million in the first operation year. Just within seconds, these smart solutions allow Visa to look at hundreds of data elements such as location, amount spent or payment method to determine if a transaction is suspicious.
Also in the insurance industry, there exist a lot of different applications. For example, Allianz uses NNs to identify the best product for the customer through analysing data such as trip length, cost or the traveller´s age.
It can be helpful as well in the marketing sector of insurance companies for segmentation of customers or to make special offers to encourage policyholders to renew the

contract. A new way to use smart technology was introduced in China. ATMs there are using face recognition technology for payments. In fact, there is no need for pin numbers and could lead to the end of paying with plastic. [9]

The Application Process

In this section, we will look at the basic application process. Later, we will apply each of these steps on our dataset to illustrate how the application process works.

1. Collect all the necessary data in one place

   Similar to statistically based models, all the possible and potential information has to be brought together in order to generate test and training sets. Often, zip or block-code overlayers are used to get additional information about the individual. Overlayers are used because it is illegal to use zip codes or sex as an input field because of discrimination. This is known as "redlining".[24]

2. Seperate the database into a training and a testing set

   It is important to separate the available data to see if the neural network has memorized the data, especially if the network found of learned something about the relationship between the input and the output variables. Therefore it helps preventing errors and increases our confidence in the performance. The selection of the training and testing set has to be done very carefully because a poorly distribution betweens the different outcomes can produce errors in the predictions. That is because neural networks are lazy. If 95 percent of of the examples are "good", the network will conclude that it is right most of the time and will classify everything as "good". Using backpropagation , the ideal training set should be equally distributed between the possible output variables. The testing set should represent the data as a whole. It is recommended to select the testing set first. One strategy is to pick every Nth example to get a picture of how the neural network will perform overall.
   In general, the more training examples the better the performance of the network. For simple problems, 50 to 100 examples can be enough but for more complex problems a training set of $30,000$ to $40,000$ examples has been adequate in the past.[24]

In our model, we use the same training and testing set as we used in the previous generalized linear model.

3. Transform the data into network-appropriate input variables

The mission in this step is to rescale the variables in a suitable form. Typically neural networks work with inputs in the range -1 to 1 or 0 to 1. The database on the other hand can consist of different types of data like category, free-form text or numeric for example.
If we have a category type feature that can take on four different values (example: martial status divorced, single, widowed, divorced) we can use a "one of N" coding. Each category will be assigned to a separate input in the neural network.Just one of the inputs will be "on" (value 1) at a time, the others will be "off" (value 0).
If the field is numeric, it can easily be rescaled into the range 0 to 1 and used directly as an input.
Mapping free-form text features into a vector of numbers is more complex. The "one of N" coding can be used, but for some features like occupation there would be thousands of different categories with probably few examples of most. A better solution is to divide the categories into groups. Grouping together different occupations can be for example: principal, management, professional, skilled labour, unskilled labour.
[24]

We already explained how to cluster the data appropriate in section 3.2 Dataset. To simplify the data, the implement a normalizing function in R:

```
normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x)))
data_storno <- as.data.frame(lapply(data_storno, normalize))
```

For applying a neural network, we check for non-numeric data and change them to numeric values using the *as.numeric* function.

4. Select the network architecture.

The most popular and almost standard neural network for forecasting, modelling and classifying is backpropagation. When selecting a backpropagating architecture, a series of decisions has to be made relating the amount of hidden units, the activation functions and the learning rates to get the optimal network.

Depending on what the network should learn and the nature of the data, one has to choose the activation function that works best. Sigmodial functions, the output takes on a value between 0 and 1 and works best when the problem should learn an "average" behaviour. If the "deviations" from the average are interesting for the problem, it is better to use hyperbolic tangent activation functions. There, the limits of the outputs are -1 and 1.

Another decision that has to be made is selecting a learning rule. There is the original learning rule developed by Rummelhart: the delta rule. Now there are some extensions as the popular cumulative delta rule and the normalized delta rule. According to Robert Trippi and Efraim Turban, the normalized delta rule works very well in general. The number of presentations over which weight changes are accumulated is known as *Epoch* and has to be set in advance but can be optimized later:

Four steps to adjust the size of the epoch:

- Pick an initial value for the epoch
- Train the the network
- Test the network several and write down the accuracy

Repeat this procedure for different epoch sizes.

- Plot the results

Looking at the plots, there will be a peak in the network´s performance for a certain epoch size. Use this epoch size for further training.

Often, there are some "ready-made" learning rates provided by the vendors, which typically work. In general, change the learning rates until they create

smooth Root-mean- square (RMS) error plots and weight histograms for each layer. IF the RMS error jumps around a lot in the graph, reduce the learning rates for all layers. A rule of thumb is, that the learning rate for the output layer should be half of the rate for the last hidden layer.

The final key decision concerns the amount of hidden layers used in the network. Typically, one should start with the "minimum" number, mostly with one, hidden layers.There is also the problem of deciding how many hidden units are necessary. In general, the less hidden units, the more "generalized" the network results will be. As a guideline, there is a rule that the number of hidden units can be determined by the amount of training examples. The rule is that there should be at least five training examples for each weight. The hidden layer size can be optimized either constructive or destructive. Either one starts with a network with no hidden units and add them unless the performance does not improve on both training and testing set. Or one starts with a network with a large population of hidden units, train the network for a while and disable units and retest the network. It should be left disabled if the network improved its performance.[24]

In case of the churn rate prediction, the resilient backpropagation algorithm will be used with a learning rate of 0.01 and a maximum of steps for training the neural network of 1e6. If the maximum of steps is reached, the training process will be stopped. Regarding hidden layers, we will compare two neural networks and compare the outputs. The first neural network contains four hidden layers starting with 6 hidden neurons:

```
1 library(neuralnet)
2 nn1 <- neuralnet(STORNO~STATISTIKJAHR+GESAMTANZ_HHV+SFOZHHV+
    SPKOMBI+INKASSOART+HHV_ZONE+ABW_BEITRAG+ALTER_CLUSTER+
    LAUFZEIT+BEGINNJAHR,data=data_train_NN, hidden=c(6,4,2,1),
    algorithm="rprop+", learningrate=0.01,threshold= 0.5, rep
    =10, stepmax=1e6, linear.output = FALSE,act.fct = "logistic"
    )
3
```

Various algorithm and activation functions can be used and compared.
In this project, we will not vary these inputs but instead we focus on trying

out different numbers of hidden layers and neurons.
For the second neural network will change the number of hidden layers to three and begin with five hidden neurons, but choose the same learning rate as before:

```
library(neuralnet)
nn2 <- nn2 <- neuralnet(STORNO~STATISTIKJAHR+GESAMTANZ_HHV+
   SFOZHHV+SPKOMBI+INKASSOART+HHV_ZONE+ABW_BEITRAG+ALTER_
   CLUSTER+LAUFZEIT+BEGINNJAHR,data=data_train_NN,hidden=c
   (5,3,1),threshold=0.5, algorithm="rprop+", learningrate
   =0.01, rep=10, stepmax=1e6, linear.output = FALSE,act.fct =
   "logistic")
```

Plotting the neural networks, we can see that the first neural network is more complex than the second one:

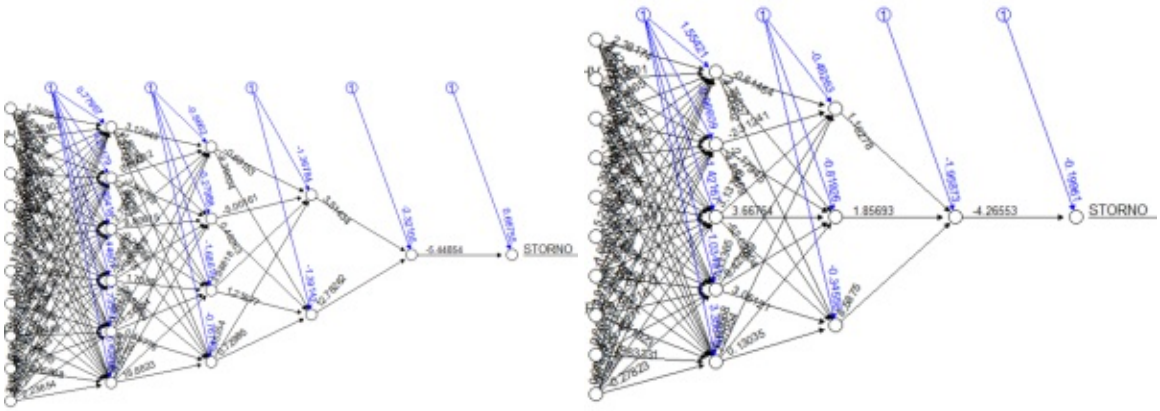


Figure 3.11: neural network 1&2

5. Test and train the network

In a nutshell, testing process consists of plugging in a set of inputs and the desired outputs and comparing them to the outputs the network produces. The occurred error is used to modify the network and adjust the weights. To measure the performance, there are some helpful diagnostic tools like histograms of all the weights, the mean square error of the output layer or the Pearson´s R coefficient. To determine how well the network performs, the test database is used. If there is a huge difference between the generated and the desired output, some corrective actions should be taken.[24]

6. Repeat the steps 1 to 4 as required
7. Position the network in your application [24]

After generating the predictions for our testing set using the computed neural networks, we create a confusion matrix to compare the results and test the accuracy.:

```
result_table1<- table(results1$actual,results1$prediction
  >0.21)
accuracy_Test1 <- sum(diag(result_table1)) / sum(result_table1
  )
```

Because the correct predicted cancellations are of more importance than the correct predictions of loyal customers, we weight the confusion matrix using:

```
weightedstorno <-sum(diag(result_table1)) / sum(result_table1)*
  (result_table1[2,2]/result_table1[1,2])
```

Applying these functions on our data, the accuracy of nn1 is 95.48% with an weightedstorno value of 7.29%. The accuracy of the simpler model is surprisingly higher, i.e. 96.16%, and the weightedstorno value is higher as well, i.e. 7.59%.

To proper evaluate and rank the models, we will do the same procedure as for the generalized linear model: we compute the receiver operating characteristic and the area under the curve.

```
library(pROC)
pROC_obj <- roc(results1$actual,results1$prediction,
smoothed = TRUE,
# arguments for ci
ci=TRUE, ci.alpha=0.9, stratified=FALSE,
# arguments for plot
plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
print.auc=TRUE, show.thres=TRUE)
#Area under the curve - neural network 1
auc(pROC_obj)
```

3.4.5 Results

For the first neural network we get a ROC curve with an area under the curve of 66.30%.

Generating and plotting the second ROC curve, we get a slightly smaller AUC value of 65.99%, indicating that the quality of the first model is a little bit better. Nevertheless is has to be discussed if the additional hidden layer is worth the small increase in accuracy.

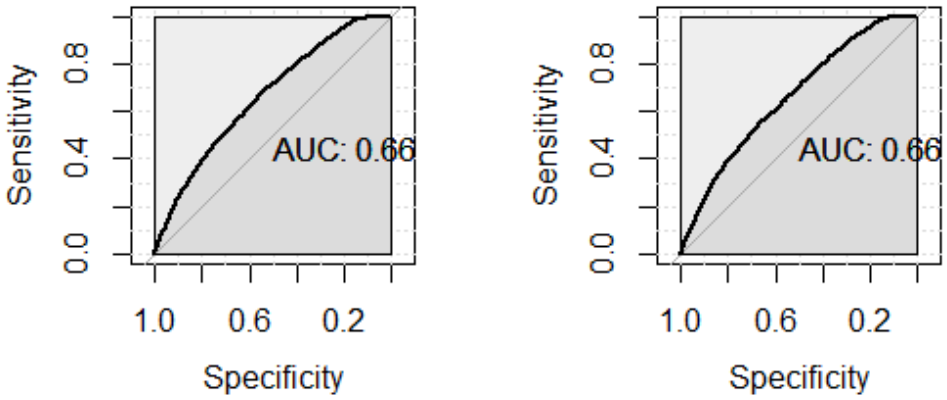


Figure 3.12: ROC NN1 (l) and ROC NN2 (r)

3.5 Comparison

Linear Regression and neural networks are similar in many ways. Both methods try to recognize pattern in the dataset and refit the "parameters" or weights till the optimum is reached. With the resulting model, it is then possible to make predictions of further events.

One main advantage using artificial intelligence is that they are solely driven by the data, on which the neural networks are trained and do not depend on the model. Therefore it is no need for a specific theory on how the pattern should arise. On

the other hand, generalized linear models start with a model and see how good the models fit the data.
Due to the limitation to certain categories of models, GLM is much more transparent than a neural network. It is not always clear how the results were created in artificial networks and therefore limits the confidence and usefulness of the model.[15]
On the other hand, the lack of transparency is a strength of the neural network because the specification of the topology of the network does not have to be too detailed and theoretically the range of functions that can be used to describe the data is unlimited. Additionally, artificial intelligence can be used to describe any real world phenomena, even if they are non-linear.

Looking at the computer time, neural networks are known to take longer to fit than generalized linear models, because they are less refined and the algorithm are more general.
For well-known statistical models like GLM, there are many predefined software packages in R to implement and are therefore widely used in insurance companies. [15]
Nevertheless, in terms of human time, GLMs probably need more time regarding defining, programming and subsequent analysis.

In conclusion, it is hard to say in which sense NNs are superior to GLMs or other non-linear statistical models, but however they do substitute computer power for brain power by eliminating the need to come up with a sophistical models.

3.5.1 Comparison of the results

Our analysed data consists of around $500,000$ policies with information about the customer´s contract and his claim history. Both of the models - GLM and a neural network - compute useful results. Predicting the churn rate with a generalized linear model leads to an area under the curve (AUC) of 0.655, where for the prediction using a neural network with four hidden layers (the network that performed the best in this scenario) the AUC is slightly higher:0.66. Even the neural network with only three hidden layers performs better than GLM.

4 Conclusion

This thesis compares different classification and prediction techniques regarding the retention problem in insurance companies. Based on the findings, the performance of neural networks is better than the performance of a logistic regression. Nevertheless, the complexity of neural networks makes the model less transparent and can likely be over-fitted.

The question therefore is, if the slight increase in accuracy is worth the effort of creating a proper neural network. The results can be used to prevent customer churn and support marketing strategies.

The insurance company could offer special discounts or some additional insurance coverage to potential dissatisfied customers with good claim history.

This thesis could be extended in creating other neural networks or different statistical models in order to improve the accuracy of the prediction.

The computed models could also be tested on other datasets to evaluate the quality of the model.[27]

5 Listing 1

```
library(dplyr)
data_storno <-read.csv2("C:/Users/vicky/OneDrive/Desktop/
    Masterarbeit/GLM_Daten.csv",header=T)
#erase unnecessary data
data_storno$ID <- NULL
data_storno$VUNR <- NULL
data_storno$ZUFALL <- NULL
data_storno$ZUFALL1 <- NULL
data_storno$ANZAHL <- NULL
glimpse(data_storno)
# Check continuous variables
continuous <-select_if(data_storno, is.numeric)
summary(continuous)
# Create factor variables
col_names <- names(data_storno)
data_storno[,col_names[]] <- lapply(data_storno[,col_names[]],factor
    )
# Select categorical column
coln<-c("STATISTIKJAHR","SFHHV","GESAMTANZ_HHV","SFOZHHV","SPKOMBI",
    "VERTRIEB_CL","INKASSOART","HHV_ZONE","ABW_BEITRAG","ALTER_
    CLUSTER","LAUFZEIT","BEGINNJAHR","STORNO")
colnames(data_storno)<-coln
levels(data_storno[,2])<-c("Unbekannt","seit einem Jahr schadenfrei"
    ,"seit zwei Jahren schadenfrei","seit drei Jahren schadenfrei","
    seit vier Jahren schadenfrei","seit fuenf Jahren schadenfrei","
    seit sechs Jahren schadenfrei","seit sieben Jahren schadenfrei","
    Beginn im Statistikjahr","Schaden im Vorjahr")
levels(data_storno[,3])<-c("1","2","3","4","5")
levels(data_storno[,4])<-c("Unbekannt","OZ-Schaden im Vorjahr","OZ-
    Schaden und Schaden mit Zahlung im VJ","Schaden mit Zahlung im VJ
    ","Kein Schaden im Vorjahr")
levels(data_storno[,5])<-c("EHV + HHV","HHV")
levels(data_storno[,6])<-c("MAKLER","SONSTIGE")
```

```
levels(data_storno[,7])<-c("Einzugsermaechtigung","Erlagschein - jaehrlich","Erlagschein - halbjaehrlich","Erlagschein - vierteljaehrlich","Erlagschein - monatlich","Sonstige")
levels(data_storno[,8])<-c("Zone 1","Zone 2","Zone 3","Zone 4","Unbekannt")
levels(data_storno[,9])<-c("Unterdurchschnittlich","Ueberdurchschnittlich")
levels(data_storno[,10])<-c("bis 30 Jahre","bis 45 Jahre","bis 60 Jahre","bis 75 Jahre","ueber 75 Jahre","unbekannt")
levels(data_storno[,11])<-c("Abschluss eines 5-Jahresvertrages","Abschluss eines 10-Jahresvertrages","Sonstige Laufzeit")
levels(data_storno[,12])<-c("bis 2010","nach 2010")
#levels(data_storno[,13])<-c("kein Storno","Storno")
summary(data_storno)
#missing data
library(Amelia)
library(mlbench)
missmap(data_storno, col=c("blue","red"),legend=TRUE)
print(missmap)
#input for plot
factor <- data.frame(select_if(data_storno, is.factor))
ncol(factor)
# Create graph for each column
#install.packages("ggplot2")
library(ggplot2)
graph <- lapply(names(factor),
                function(x)
                  ggplot(factor, aes(get(x))) +
                  geom_bar() +
                  theme(axis.text.x = element_text(angle = 90)))
#plot
graph
help(aes)
#summarize data
ggplot(data_storno, aes(x = GESAMTANZ_HHV, fill = STORNO)) +
  geom_bar(position = "fill") +
  theme_classic()
#summariize for all factors
graph_storno <- lapply(names(factor),
                function(x)
                  ggplot(factor, aes(get(x), fill=STORNO)) +
```

```
                    geom_bar(position="fill") +theme(axis.text.x =
    element_text(angle = 90)))
#output
graph_storno
#relations
graph_relation <-ggplot(factor, aes(STORNO, fill=GESAMTANZ_HHV)) +
                            geom_bar(position="fill") +theme(axis.text.
    x = element_text(angle = 90))
#output
graph_relation
#control random numbers
set.seed(49340249)

#Train and Test set
dummy_sep <- rbinom(nrow(data_storno), 1, 0.8)
data_test <- data_storno[dummy_sep == 0, ]
data_train <-data_storno[dummy_sep == 1, ]
dim(data_train)
#balanced training set
data_train_storno<- subset(data_train,STORNO == 1)
data_train_keinstorno<-subset(data_storno,STORNO == 0)
data_train_keinstorno_klein<-sample_frac(data_train_keinstorno, 0.2,
     replace = TRUE)
data_train<-rbind(data_train_storno,data_train_keinstorno_klein)
dim(data_train)
dim(data_test)
#correlation on training set
library(GGally)
# Convert data to numeric
corr <- data.frame(lapply(data_train, as.integer))
# Plot the graph
ggcorr(corr,
       method = c("pairwise", "spearman"),
       nbreaks = 9,
       digits=2,
       hjust = 0.8,
       label = TRUE,
       label_size = 3,
       geom="circle",
       label_round=3,
       color = "grey50")
help(ggcorr)
```

```
#erase unnecessary data
data_train$SFHHV<-NULL
data_train$VERTRIEB_CL<-NULL
data_test$SFHHV<-NULL
data_test$VERTRIEB_CL<-NULL
################GLM##################
#GLM Model, use binomial to take logistic regression
formula <- STORNO~STATISTIKJAHR+GESAMTANZ_HHV+SFOZHHV+SPKOMBI+
   INKASSOART+HHV_ZONE+ABW_BEITRAG+ALTER_CLUSTER+LAUFZEIT+BEGINNJAHR
model <- glm(formula,data=data_train, family = 'binomial')
summary(model)
#Information criteria
model$aic
################Prediction###############
predict <- predict.glm(model,data_test,type ="response")
# confusion matrix
table_mat1 <- table(data_test$STORNO, predict>0.2)
print(table_mat1)
summary(table_mat1)
#Accuracy test
accuracy_Test <- sum(diag(table_mat1)) / sum(table_mat1)
accuracy_Test
#ROC Plot
library(ROCR)
#normalize data
normalize(predict)
ROCRpred <- prediction(predict, data_test$STORNO)
ROCRperf <- performance(ROCRpred, "tpr", "fpr")
#ROC plot 2nd
library(pROC)
pROC_obj <- roc(data_test$STORNO,predict,
                smoothed = TRUE,
                # arguments for ci
                ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                # arguments for plot
                plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE,
   grid=TRUE,
                print.auc=TRUE, show.thres=TRUE)
#Area under the curve
auc(pROC_obj)
```

Listing 5.1: Source Code GLM

6 Listing 2

```
library(dplyr)
data_storno <-read.csv2("C:/Users/vicky/OneDrive/Desktop/
   Masterarbeit/GLM_Daten.csv",header=T)
#erase unnecessary data
data_storno$ID <- NULL
data_storno$VUNR <- NULL
data_storno$ZUFALL <- NULL
data_storno$ZUFALL1 <- NULL
data_storno$ANZAHL <- NULL
glimpse(data_storno)
#normalize data
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
data_storno <- as.data.frame(lapply(data_storno, normalize))
# Check continuous variables
col_names <- names(data_storno)
data_storno[,col_names[]] <- lapply(data_storno[,col_names[]],as.
   numeric)
continuous <-select_if(data_storno, is.numeric)
summary(continuous)
# col names
coln<-c("STATISTIKJAHR","SFHHV","GESAMTANZ_HHV","SFOZHHV","SPKOMBI",
   "VERTRIEB_CL","INKASSOART","HHV_ZONE","ABW_BEITRAG","ALTER_
   CLUSTER","LAUFZEIT","BEGINNJAHR","STORNO")
colnames(data_storno)<-coln
summary(data_storno)
#contro random numbers
set.seed(49340249)
#Train and Test set
dummy_sep <- rbinom(nrow(data_storno), 1, 0.8)
data_test_NN <- data_storno[dummy_sep == 0, ]
data_train_NN <-data_storno[dummy_sep == 1, ]
dim(data_train_NN)
```

```
#balnaced training set
data_train_storno<- subset(data_train_NN,STORNO == 1)
data_train_keinstorno<-subset(data_storno,STORNO == 0)
data_train_keinstorno_klein<-sample_frac(data_train_keinstorno, 0.2,
    replace = TRUE)
data_train_NN<-rbind(data_train_storno,data_train_keinstorno_klein)
dim(data_train_NN)
dim(data_test_NN)
#numeric data are necessary
col_names <- names(data_train_NN)
data_train_NN[,col_names[]] <- lapply(data_train_NN[,col_names[]],as
    .numeric)
##################Neural Network##################
library(neuralnet)
#first neural network with four hidden layers and starting with 6
    hidden neurons
nn1 <- neuralnet(STORNO~STATISTIKJAHR+GESAMTANZ_HHV+SFOZHHV+SPKOMBI+
    INKASSOART+HHV_ZONE+ABW_BEITRAG+ALTER_CLUSTER+LAUFZEIT+BEGINNJAHR
    ,data=data_train_NN, hidden=c(6,4,2,1), algorithm="rprop+",
    learningrate=0.01,threshold= 0.5, rep=10, stepmax=1e6, linear.
    output = FALSE,act.fct = "logistic")
#results
nn1$result.matrix
#plot neural network
plot(nn1,rep="best",arrow.length=0.15,radius=0.10)

#second neural network with three hidden layers and starting with 5
    hidden neurons
nn2 <- neuralnet(STORNO~STATISTIKJAHR+GESAMTANZ_HHV+SFOZHHV+SPKOMBI+
    INKASSOART+HHV_ZONE+ABW_BEITRAG+ALTER_CLUSTER+LAUFZEIT+BEGINNJAHR
    ,data=data_train_NN,hidden=c(5,3,1),threshold=0.5, algorithm="
    rprop+", learningrate=0.01, rep=10, stepmax=1e6, linear.output =
    FALSE,act.fct = "logistic")
#results
nn2$result.matrix
#plot neural network
plot(nn2, rep="best",arrow.length=0.15,radius=0.10)

#numeric test data necessary
col_names <- names(data_test_NN)
data_test_NN[,col_names[]] <- lapply(data_test_NN[,col_names[]],as.
    numeric)
```

```
59
60 #compariing first nn with actual data
61 nn.results1 <- predict(nn1,data_test_NN)
62 results1 <- data.frame(actual = data_test_NN$STORNO, prediction = nn
       .results1)
63 results1
64 #confusion matrix
65 result_table1<- table(results1$actual,results1$prediction>0.21)
66 result_table1
67 #accuracy test
68 accuracy_Test1 <- sum(diag(result_table1)) / sum(result_table1)
69 accuracy_Test1
70 #weighted accurarcy test
71 weightedstorno <-sum(diag(result_table1)) / sum(result_table1)*(
       result_table1[2,2]/result_table1[1,2])
72 weightedstorno
73 #weight plot
74 help(gwplot)
75 gwplot(nn1, selected.covariate=1)
76 #ROC of neural network nn1
77 library(neuralnet)
78 ROCRpred_nn1 <- ROCR::prediction(results1$prediction, results1$
       actual)
79 library(ROCR)
80 ROCRperf_nn1 <- performance(ROCRpred_nn1, 'tpr', 'fpr')
81 plot(ROCRperf_nn1, colorize = TRUE, text.adj = c(-0.2, 1.7))
82 #ROC and AUC plot - neural network nn11
83 library(pROC)
84 pROC_obj <- roc(results1$actual,results1$prediction,
85                 smoothed = TRUE,
86                 # arguments for ci
87                 ci=TRUE, ci.alpha=0.9, stratified=FALSE,
88                 # arguments for plot
89                 plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE,
       grid=TRUE,
90                 print.auc=TRUE, show.thres=TRUE)
91 #Area under the curve - neural network nn1
92 auc(pROC_obj)
93
94 #comparison second nn with actual data
95 nn.results2 <- predict(nn2, data_test_NN)
```

```
results2 <- data.frame(actual = data_test_NN$STORNO, prediction = nn.results2)
#results
results2
#confusion matrix
result_table2<- table(results2$actual,results2$prediction>0.22)
result_table2
#accuracy test
accuracy_Test2 <- sum(diag(result_table2)) / sum(result_table2)
accuracy_Test2
#weighted accuracy test
weightedstorno2 <-sum(diag(result_table2)) / sum(result_table2)*(result_table2[2,2]/result_table2[1,2])
weightedstorno2
#weight plot
gwplot(nn2, selected.covariate=1)
#ROC of neural network nn2
detach(package:neuralnet,unload = T)
library(ROCR)
ROCRpred_nn2 <- prediction(results2$prediction,results2$actual)
ROCRperf_nn2 <- performance(ROCRpred_nn2, 'tpr', 'fpr')
plot(ROCRperf_nn2, colorize = TRUE, text.adj = c(-0.2, 1.7))
#ROC and AUC plot - neural network nn2
library(pROC)
pROC_obj <- roc(results2$actual,results2$prediction,
                smoothed = TRUE,
                # arguments for ci
                ci=TRUE, ci.alpha=0.9, stratified=FALSE,
                # arguments for plot
                plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
                print.auc=TRUE, show.thres=TRUE)
#Area under the curve - neural network nn2
auc(pROC_obj)
```

Listing 6.1: Source Code GLM

Bibliography

[1] Sanjay Jamwal Afaq Alam Khan and M.M.Sepehri. "Applying data mining to customer prediction in an Internet Service Provider". In: *International Journal of Computer Applications (0975 – 8887) Vol. 9 No.7* (2010).

[2] Robert J. Willis Ai Cheo Yeo Kate A. Smith and Malcolm Brooks. *Modelling the Effect of Premium Changes on Motor Insurance Customer Retention Rates Using Neural Networks*. Technical Report. School of Business Systems, Monash University, 2001.

[3] Terry H.F. Cheuk Andrew Carverhill. *Alternative Neural Network Approach for Option Pricing and Hedging*. Technical Report. University of Hong Kong, 2003.

[4] Dr. Prabin Kumar Panigrahi Anuj Sharma. "A Neural Network based Approach for Predicting Customer Churn in Cellular Network Services". In: *International Journal of Computer Applications Vol. 27 No.11* (2011).

[5] Sekhar Aryasomayajula. *Parametric versus Non-parametric Bond Pricing and Hedging Models*. Ed. by VDM Verlag Dr. Müller. 2002.

[6] Arash Riasi Azarnoush Ansari. "Modelling and evaluating customer loyality using neural networks: Evidence from startup insurance companies". In: *Future Busniess Journal Vol.2 No.1, pp.15-30* (2016).

[7] E. Michael Azoff. *Neural Network Time Series: Forecasting of Financial Markets*. John Wiley Sons, Inc.605 Third Ave. New York, NYUnited States, 1994.

[8] Jason Brownlee. *How to Calculate Correlation Between Variables in Python*. 2020. URL: https://machinelearningmastery.com/how-to-use-correlation-to-understand-the-relationship-between-variables/.

[9] KC Cheung. *10 Use Cases of Ne ural Networks in Business*. 2020. URL: https://algorithmxlab.com/blog/10-use-cases-neural-networks/#Artificial_Neural_Networks_in_Financial_Services.

[10] Michael J Crawley. *The R Book*. John Wiley Sons, 2007.

[11] Ronald J. Williams David E. Rumelhart Geoffrey E. Hinton. *Learning representations by back-propagating errors*. Technical Report. Institute for Cognitive Science, C-015, University of California, 1986.

[12] Tom Fawcett. *An Introduction to ROC analysis*. Technical Report. Institute for the Study of Learning and Expertise, USA, 2005.

[13] Rohith Gandhi. *Introduction to Machine Learning Algorithms: Linear Regression*. 2018. URL: https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a.

[14] Genevieve Hayes. *Beyond Linear Regression: An Introduction to GLMs*. 2020. URL: https://towardsdatascience.com/beyond-linear-regression-an-introduction-to-glms-7ae64a8fad9c.

[15] Louise Pryor Jullian Lowe. *Neural Networks v. GLMs in pricing general insurance*. Technical Report. General Insurance Convention, 1996.

[16] Erika W. Gilbert Krishna Swamy Mary M. Pashley. *Neural Network Applications in Finance*. Conference Paper. Michigan University, Tennessee Technological University, 1997.

[17] David W. Wright Lawrence Kryzanowski Michael Galler. "Using Artifical Neural Networks to Pick Stocks". In: *Financial Analysts Journal Vol.49 No. 4* (1993).

[18] JAmes MCCaffrey. *How to use Resilient Back Propagation to train Neural Networks*. 2020. URL: https://visualstudiomagazine.com/articles/2015/03/01/resilient-back-propagation.aspx.

[19] Michael Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[20] Nils J. Nilsson. *Introdiction to Machine Learning*. Department of Computer Science, Stanford University, 1998.

[21] James M. Hutchinson; Andrew W. Lo; Tomaso Poggio. "A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks". In: *The Journal of Finance Vol. 49, No. 3, pp. 851-889* (1994).

[22] Swati Sharma Ravi Malhotra. *Machine Learning in Insurance*. Technical Report. Accenture, 2018.

[23] Alice E.Smith Ravipim Chaveesuk Chat Srivaree-ratana. *Alternative Neural Network Approaches to Corporate Bond Rating*. Technical Report. University of Pittsburgh, 1997.

[24] Efraim Turban Robert R. Trippi. *Neural Networks in Finance and Investing*. Probus Pub Co, 1992.

[25] Marcus Roth. *AI in Foreign Exchange Trading (Forex) - Current State of the Sector*. Research Paper. Emerj – The AI Research and Advisory Company, 2019.

[26] Maytal Saar-Tsechansky. "Handling Missing Values when Applying Classification Models". In: *Journal of Machine Learning Research 8 1625-1657* (2007).

[27] Sahar F. Sabbeh. *Machine-Learning Techniques for Customer Retention: A Comparative Study*. Technical Report. King AbdulAziz University, 2018.

[28] Murat H. Salzi. "A Brief Review of Feed-Forward Neural Networks". In: *Communications Faculty of Science University of Ankara Vol.50 No.1, pp. 11-17* (2006).

[29] Shashi Shekhar Soumitra Dutta. *Bond rating: A non-conservative application of neural networks*. Conference Paper. IEEE International Conference on Neural Networks, 1988.

[30] Mike Thomas. *How AI Trading Technology is making Stock Market Investors smarter - and richer*. 2020. URL: https://builtin.com/artificial-intelligence/ai-trading-stock-market-tech.