# REAL ESTATE INSURANCE CLAIMS PREDICTION WITH MACHINE LEARNING ALGORITHMS

Lappeenranta–Lahti University of Technology LUT

Master's Programme in Strategic Finance and Analytics

2022

Alexander Karzov

Examiner(s): Post-doctoral Researcher Christoph Lohrmann, Ph.D

Associate Professor Jan Stoklasa

## ABSTRACT

Lappeenranta–Lahti University of Technology LUT

LUT School of Business and Management

Business Administration

Alexander Karzov

**Real Estate Insurance Claims Prediction with Machine Learning Algorithms**

Master's thesis

2022

78 pages, 6 figures, 25 tables and 5 appendices

Examiner(s): Post-doctoral Researcher Christoph Lohrmann, Ph.D. (Computational Science), Associate Professor Jan Stoklasa

Keywords: real estate insurance, clams prediction, classification algorithms, machine learning

Nowadays insurance companies are increasingly implementing machine learning algorithms in their business routine. An ability to determine beforehand an emergence of claims could offer a tool to increase a profitability of the insurance business. This thesis focuses on real estate insurance and explores opportunities of machine learning algorithms implementation for claims emergence prediction. The goal of the study is to build a set of binary classification models, evaluate their performance, compare those models with each other and suggest the best algorithm.

The selection of algorithm is based the most frequently used models in previous studies. According to the literature review, the selection of the algorithms is the following: Logistic Regression (LR), Naïve Bayes (NB), Neural Networks (NN), Random Forest (RF) and eXtreme Gradient Boosted Tree (XGBoost). The results are measured using Accuracy, Sensitivity, Specificity, Area Under the Curve (AUC) and Cohen's Kappa. Based on the summarized outcome of those five performance metrics, NB show the best performance. However, during the performance evaluation all algorithms show certain disadvantages. Actual implementation of any of the tested models for claims emergence prediction as a business routine demands additional researches.

**TIIVISTELMÄ**

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUT-kauppakorkeakoulu

Kauppatieteet

Alexander Karzov

**Kiinteistövakuutusten Vahinkojen Ennustaminen Koneoppimismenetelmillä**

Kauppatieteiden pro gradu -tutkielma

2022

78 sivua, 6 kuvaa, 25 taulukkoa ja 5 liitettä

Tarkastajat: Tutkijatohtori Christoph Lohrmann, Apulaisprofessori Jan Stoklasa

Avainsanat: kiinteistövakuutus, vahinkojen ennustaminen, luokittelualgoritmit, koneoppiminen.

Nykyään vakuutusyhtiöt ottavat jatkuvasti käyttöön koneoppimisalgoritmeja. Mahdollisuus ennustaa vahinkojen ilmaantumisen etukäteen voisi tarjota työkalun vakuutustoiminnan kannattavuuden parantamiseen. Tämä Pro gradu -tutkielma keskittyy kiinteistövakuutuksiin ja tutkii koneoppimisalgoritmien mahdollisuuksia vahinkojen ilmaantumisen ennustamisessa. Tutkimuksen tavoitteena on rakentaa joukko binäärisiä luokittelualgoritmeja, arvioida niiden suorituskyvyn, verrata niitä keskenään ja valita niistä parhaan.

Algoritmien valinta perustuu aikaisemmissa tutkimuksissa eniten käytettyihin malleihin. Kirjallisuuskatsauksen perusteella algoritmien valinta on seuraava: Logistinen Regressio (LR), Naiivi Bayesin luokitin (NB), Neuroverkot (NN), Satunnaismetsä (RF) ja eXtreme Gradient Boosted Tree (XGBoost). Tulokset mitataan käyttämällä Tarkkuutta, Herkkyyttä, Spesifisyyttä, Käyrän Alla Olevaa Pinta-alaa (AUC) ja Cohenin Kappaa. Näiden viiden suorituskykymittarin yhteenvedon perusteella NB näyttää parhaan suorituskyvyn. Suorituskyvyn arvioinnin aikana kaikissa algoritmeissa on kuitenkin löydetty haittoja, joten niiden todellinen käyttö liiketoimintarutiinina vaatii lisätutkimuksia.

ABBREVIATIONS

| | |
|---|---|
| AB | AdaBoost (Adaptive Boosting Algorithm) |
| AUC | Area Under the Curve |
| BT | Boosted Tree |
| DT | Decision Tree |
| GB | Gradient Boost |
| KNN | k-Nearest Neighbours |
| LR | Logistic Regression |
| NB | Naïve Bayes |
| NN | Neural Network |
| RF | Random Forest |
| ROC | Receiver Operating Characteristic |
| SVM | Support Vector Machine |
| XGBoost | Extreme Gradient Boosted Tree |

**Table of contents**

Abstract

Abbreviations

1. Introduction ..... 9
   1.1. Focus of the study ..... 10
   1.2. Objectives, research questions and delimitations ..... 10
   1.3. Structure ..... 11
2. Literature review ..... 12
   2.1. Methodology of the Literature Review ..... 12
   2.2. Applications of machine learning algorithms in the insurance industry ..... 13
   2.3. Studies of claims emergence prediction in property insurance ..... 15
   2.4. Summary of the literature review ..... 17
3. Theoretical Background ..... 20
   3.1. Overview of Methods ..... 20
      3.1.1. Logistic regression ..... 21
      3.1.2. Naïve Bayes ..... 22
      3.1.3. Neural Network ..... 24
      3.1.4. Random forest ..... 27
      3.1.5. Boosted Trees ..... 29
   3.2. Overview of Performance measures ..... 31
      3.2.1. Accuracy ..... 32
      3.2.2. Sensitivity ..... 33
      3.2.3. Specificity ..... 34
      3.2.4. ROC and AUC ..... 34
      3.2.5. Cohen's Kappa ..... 35
4. Data ..... 37
   4.1. Description of initial data ..... 37
   4.2. Data modification ..... 38
      4.2.1. Datasets merging and explained variable creation ..... 38

4.2.2. Handling missing values ..... 39
4.2.3. Categorical variable recoding ..... 39
4.2.4. Normalization ..... 40
4.3. Data split ..... 40
5. Model specification ..... 42
5.1. Model selection and specification process description ..... 42
5.2. Logistic regression tuning ..... 44
5.2.1. LR: Default model ..... 44
5.2.2. LR: Tuning process ..... 45
5.2.3. LR: Selected model ..... 46
5.3. Naïve Bayes tuning ..... 47
5.3.1. NB: Default model ..... 47
5.3.2. NB: Tuning process ..... 48
5.4. Neural Network tuning ..... 49
5.4.1. NN: Default model ..... 49
5.4.2. NN: Tuning process ..... 50
5.4.3. NN: Selected model ..... 51
5.5. Random Forest tuning ..... 52
5.5.1. RF: Default model ..... 52
5.5.2. RF: Tuning process ..... 52
5.5.3. RF: Selected model ..... 53
5.6. eXtreme Gradient Boosted tree tuning ..... 54
5.6.1. XGBoost: Default model ..... 54
5.6.2. XGBoost: Tuning process ..... 55
5.6.3. XGBoost: Selected model ..... 57
6. Results ..... 59
6.1. Confusion Matrices ..... 60
6.2. Models evaluation using performance metrics ..... 61
6.3. Selection of the best performed model ..... 62
7. Conclusions ..... 64
7.1. Answers to the research questions ..... 64
7.2. Analysis of the results ..... 66
7.3. Limitations and ideas for future researches ..... 66

References.....68

Appendices

Appendix 1. LR ROC Curve

Appendix 2. NB ROC Curve

Appendix 3. NN ROC Curve

Appendix 4. RF ROC Curve

Appendix 5. XGBoost ROC Curve

Figures

Figure 1. Perceptron

Figure 2. Neural Networks with two hidden layers

Figure 3. Bagging: Parallel connection

Figure 4. Boosting: Sequential connection

Figure 5. ROC Curve

Figure 6. Confusion matrices of testing dataset

Tables

Table 1. Summary of related studies

Table 2. Frequency of methods used in related studies

Table 3. Frequency of performance measures used in related studies

Table 4. Confusion matrix

Table 5. Confusion matrix of LR with default values of hyperparameters

Table 6. LR hyperparameter tuning

Table 7. Changes in LR confusion matrix

Table 8. LR: performance of selected parameters compared to default parameters

Table 9. Confusion matrix of NB with default values of hyperparameters

Table 10. NB hyperparameter tuning

Table 11. Confusion matrix of NN with default values of hyperparameters

Table 12. NN hyperparameter tuning

Table 13. Changes in NN confusion matrix

Table 14. NN: performance of selected parameters compared to default parameters

Table 15. Confusion matrix of RF with default values of hyperparameters

Table 16. RF hyperparameter tuning

Table 17. Changes in RF confusion matrix

Table 18. RF: performance of selected parameters compared to default parameters

Table 19. Confusion matrix of XGBoost with default values of hyperparameters

Table 20. XGBoost hyperparameter tuning

Table 21. Changes in XGBoost confusion matrix

Table 22. XGBoost: performance of selected parameters compared to default parameters

Table 23. Summary of selected hyperparameters

Table 24. Results of selected models based on performance metrics

Table 25. Scored results of selected models

# 1. Introduction

**Insurance** is a financial product provided by insurance companies. It guaranties for clients financial protection against losses from agreed upon beforehand risks. A price the clients pay to the insurance company for receiving those guaranties is called a **premium**. The agreement between the client and the insurance company is called a **policy**. In case if the insured risk is realised, the client makes a request of that to the insurance company for receiving a financial compensation. Such request is called a **claim**. (AM Best, 2019)

Insurance companies work based on probability and the law of large numbers. They collect small premiums from individual clients and pool them together to cover the losses. An idea behind the business is that most of the clients do not face the loss during a lifetime of the policy. Based on that one of the key challenges in the insurance industry is such an accurate prediction of claims emergence as possible. Successful solving of this issue could let the company set optimal premium prices or make a more general decision of carrying a certain risk or not. However, it is not a trivial task, since every client is unique, so the risk differs from one client to another. (Kafkova, 2015)

As Baecke and Bocca mentioned (2017) artificial intelligence development, increase of data availability and growing computational power provide excellent opportunities for performance improvements in various fields including insurance. Most of insurance companies use nowadays machine learning algorithms in their decision-making process. Steiner and Ruiz (2019) reported that more than 2/3 of companies found a positive impact of applying models based on machine learning algorithms in their business routine including such cases as claims emergence, cost estimation and fraud prediction. According to Louzada et al. (2016) application of machine learning algorithms in insurance industry is also rising popularity among academic literature.

## 1.1. Focus of the study

Insurance companies are working in different fields. One of the main fields is a property insurance. It can be divided into personal and commercial product lines. Two main types of products within the personal line are an auto insurance and a real estate insurance. In many countries those types of insurance are obligatory by the law. It generates a constant demand among clients and interest to this field among insurance companies. Another one typical characteristic of those types of insurance is a presence of a concrete insured object with measurable features. (AM Best, 2019)

This thesis focuses on the real estate insurance. A database for the research was provided by a European insurance company. Currently the company uses an algorithm for setting an optimal premium price based on the features of the insured object. However, the current algorithm only includes the predicted risk in the premium price. An idea of this research is to investigate a possibility to predict beforehand emergence of claims for considering an opportunity of declining high-risk policies at all. In terms of machine learning it represents a binary classification problem.

## 1.2. Objectives, research questions and delimitations

The main goal of this thesis is to build a set of claims emergence predictive algorithms, evaluate their performance and compare them against each other for finding the best algorithm. For making this comparison as reliable as possible, the selection of the algorithms is made based on the existing scientific knowledge of the subject. The same principal is applied for performance measures. Based on the goal of the thesis the research questions are formulated in the following way:

1. *Which machine learning algorithms have been frequently used in the previous studies?*
2. *Which performance measures have been frequently used in the previous studies?*

3. *How would different classification methods perform compared to each other in the real estate insurance claims emergence prediction on the provided dataset?*
4. *Which of the selected classification methods performed the best in the real estate insurance claims emergence prediction on the provided dataset?*

This thesis does not try to estimate an exact type of claims or an exact sum of loss occurred by each claim. It is focused on a binary classification between policies where some claim is expected and policies without claims expectation. The target is to estimate only the claim emergence, since only the fact of that could affect the decisions of the policy accepting and the price setting.

Number of applied methods is also limited. Because algorithms are selected based on the literature review, in this thesis there are applied only those methods, which have been frequently used in previous researches in property (not necessarily real estate) claims emergence predictions.

## 1.3. Structure

Chapter 2 presents a literature review, starting from the types of implementations of machine learning algorithms in the insurance industry and continuing to the studies with classification problems in the property insurance field. Chapter 3 provides a description of machine learning algorithms and performance measures that are applied in the current thesis. Chapter 4 describes data preparation and modification. In Chapter 5 selected models are initially applied on the training dataset with the default values of hyperparameters. Then the hyperparameters are tuned trying to improve the performance of the models with default values. Chapter 6 provides the result of the tuned models on the testing dataset and compares them using selected performance measures. Finally, Chapter 7 contains the answers to the research questions, an analysis of the results, a discussion about the limitations and ideas for future researches.

# 2. Literature review

This Chapter contains a review of existed academic knowledge from the field of the thesis. It describes a searching process of the related studies, the results of it and goes through different branches of scientific thoughts about implementations of machine learning algorithms in the insurance industry. Then it goes deeply into the applications of machine learning algorithms in claims emergence prediction in property insurance field and in the end summarizes key findings and provides conclusions of the literature review.

## 2.1. Methodology of the Literature Review

Keywords for the search of the related studies were selected based on the topic of this Master's thesis, which is "Real Estate Insurance Claims Prediction with Machine Learning Algorithms". As the main keywords for the search in databases were used "real estate insurance" (or "property insurance"), "claim" (or "loss"), "prediction" and "machine learning".

For a wider coverage simultaneously were used such portals as Elsevier ScienceDirect, U-M Library Articles Search, Research Gate, Google Scholar and even a regular Google search. During the process, it was noticed that there were many relatively related topics, but it was hard to find such articles, which would exactly follow the idea of this thesis. For getting a broader search results the one of the keywords was changed from "real estate insurance" to a more general "insurance".

For each keyword and their combination were scanned up to the first 10 pages in each databased. Every time when there was found some work that could contribute the current thesis its abstract was read. If the hypothesis of suitability of the chosen research was confirmed, that research was reviewed in more details including its results, conclusions,

chosen models and performance measures. In addition, if such information was available, a data, choice of variables and their importance to the results were explored too.

Following the methodology proposed by Webster and Watson (2002) the sources used in each of the reviewed articles were also scanned for finding more suitable works that could be missed using the database search. All works, which were found using this approach, were studied in the same way including check of the used sources. The process was repeated until it was noticed that all mentioned in the sources works were already explored.

## 2.2. Applications of machine learning algorithms in the insurance industry

According to Denney and Tewksbury (2013) a comprehensive literature review should not be concentrated only on the totally matching works, but should also cover related studies from similar fields. Based on this recommendation this section briefly describes application of machine learning algorithms in three branches of the researches in the insurance industry. Every described branch has some similarities with the topic of this thesis, but also has some differences. In general, this section covers where else in insurance industry applications of machine learning algorithms were studied.

The first one of mentioned above branches is concentrated on the claims prediction the same as the current thesis. However, the target of the researches is not prediction of claims emergence. Instead, they are focused on estimating a size of the expected loss payments. It is not a classification but a regression problem, so usually it was solved using regression methods. The most common tool there was a linear regression in its various modifications. Also, as Tino Tuomi-Nikula (2018) noticed in the majority of those researches the loss is estimated for the whole risk portfolio in general, while there also are some works, where the loss is estimated individually for each claim.

The second branch is focused on fraud detection. In opposite to the first branch it is a classification problem, but it is not related to the claims. As Palacio (2019) found, among

property insurances there were many studies which are related to vehicle insurance. As an example he mentioned the following works: Belhadji et al., 2000; Stefano and Gisella, 2001; Brockett et al., 2002; Phua et al., 2004; Viaene et al., 2007; Wilson, 2009; Nian et al., 2016. In the same time, according to his findings, real estate fraud detection is much rarer topic.

One of the classic examples of fraud detection in real estate insurance is Bentley's (2000) article. He applied an algorithm based on fuzzy logic on the real estate claim dataset and got a number of correct predictions of the positive class of 60%, which was considered as an impressive result, because among two classes the negative result was observed 10 000 times, while the positive one only 49 times.

More recent research was made by Severino and Peng (2021). In the dataset they used there was a small number of building related variables, the main focus was on customers demographic data and behaviour as a client of the insurance company. They compared Logistic Regression, Penalized Logistic Regression, Naïve Bayes, K-neighbours classifier, Polynomial Kernel SVM, Gaussian Kernel SVM, Deep Neural Network, Random Forest and Gradient Boosting Machine with such performance measures as Accuracy, Precision, Recall, F1 Score, Kappa and MCC. Severino and Peng summarized that among those six measures Random Forest showed the best performance, Deep Neural Network and Gradient Boosting Machine were quite close, K-neighbours classifier and Naïve Bayes had the poorest results and Logistic Regression was somewhere in the middle.

The third branch is focused on retention of the customers. It is less popular compared to the previous two, but still is relevant in terms of used methods and performance measures. The same as the previous one it is a classification problem, but the explained variable there is a customer churn.

Günther et al. (2014) studied Logistic Regression and Generalized Additive Models. The first one showed a good success based on ROC, but the second one proved its usefulness in the cases with nonlinear relationships. Mau, Pletikosa and Wagner (2018) tested the

effectiveness of the Random Forest on the vehicle and real estate insurance data. Based on Accuracy, F-score, ROC (Receiver Operating Characteristic) curve and AUC (Area Under the Curve) their model performed well enough. Stucki (2019) compared Logistic Regression, Random Forest, AdaBoost with Decision Trees, K-neighbours classifier, Artificial Neural Network and Support Vector Machine using Accuracy, F-score, ROC and AUC. He concluded that the best models were Random Forest and AdaBoost (a bit poorer), while other models were quite close.

## 2.3. Studies of claims emergence prediction in property insurance

The closest work to the current thesis is Wanyanga's (2021) article. It represents a classification problem in real estate insurance claims prediction. He used a dataset provided by a building insurance company and using feature importance analysis found that the most important variable among 14 presented in the dataset was a size of the insured building with importance of 67%. Other five important variables were geographical code, duration of insurance policy, type of the building, date of the first occupation and years policy have been observed (presented in order of importance decreasing). In the same time, the following five variables were also included in the analysis but had no actual importance: is the building residential or not, is the building painted or not, is the building fenced or not, is the building gardened or not and rural/urban area.

Wanyanga used Decision Trees (DT), Gradient Boost (GB), Random Forest (RB) and Support Vector Machine (SVM). Applying k-fold cross-validation he conducted that Gradient Boost showed the highest accuracy of 79%, however SVM and GB were also very close to it with an accuracy of 77% both. DT showed much poorer accuracy of 69%.

In addition to Wanyanga's article there were found another 6 closely related researches. The first of them was completed by Smith, Willis and Brooks (2000). In their research they were concentrated on two separate questions. The first one was about customer's retention and represented the third branch of the studies about the insurance industry, but the second one

was an identification of customers with a higher risk of claims emergence. They tested Decision Trees (DT) and Neural Networks (NN). Neural Networks showed higher performance based on Accuracy and ROC.

The topic of the next five researches is a vehicle insurance claims prediction. Although it is not a real estate insurance, it is still property insurance and is closer to the topic of the thesis than for example health insurance, because there is also insured some physical object with concrete and measurable features.

The first of mentioned vehicle insurance researches was implemented in 2017 by Jing et al. They used a one-year insurance policy dataset, which included general demographic characteristics of the customers like gender, years of driving experience or residential area and vehicle specific variables. Researchers compared Naïve Bayes (NB) and Bayesian Network in terms of accuracy and both methods gave the same accuracy. Jing et al. (2017) mentioned that Bayesian model is not so universal due to its limitations, however it could be even more effective, because it reached the same Accuracy as Naïve Bayes using limited number of variables.

Abdelhadi, Elbahnasy and Abdelsalam (2020) completed a similar test. They used publicly available Kaggle dataset where the variables were the same as in the article of Jing et al. (2017), but in the addition, there was such variable like miles driven by the client, which represented individual driving pattern. In addition, the dataset contained customers' credit and claims history. The selection of the methods in this research was wider – there were included Decision Trees (J48), Naïve Bayes, Neural Networks and XGBoost. XGBoost performed the best in terms of Accuracy (92,53%) and ROC (98,6%). DT showed just a little bit poorer result (Accuracy of 92,22% and ROC equal to 98,1%). NB had the worst Accuracy (83,84%) with the worst ROC (68,0%).

If Abdelhadi et al. (2020) included only one driving pattern variable, Pesantez-Narvaez et al. (2019) developed this idea and used a whole customers' driving patterns dataset collected

by telematic system with wider selection of driving pattern variables. However, the choice of the models was limited – researchers focused only on comparison of Logistic Regression (LR) versus XGBoost. In terms of performance measures in addition to Accuracy and ROC used also Sensitivity, Specificity and RSME. Pesantez-Narvaez et al. (2019) concluded that XGBoost had definitely higher performance on training dataset, but it was explained by overfitting. They mentioned that XGBoost was a very effective method with an Accuracy of 84%, but it could be too sensitive to the data. On the other hand, LR performed poorer achieving only 66% accuracy, but it did not face so sever Sensitivity problem as XGBoost.

The good performance of XGBoost was confirmed by Fauzan and Murfi (2018) on a car insurance big data which contained 2 116 753 missing values in total which were filled using medians and means. Here XGBoost again showed the highest Accuracy among AdaBoost (AB), Stochastic Gradient Boost, Random Forest and Neural Networks, however Stochastic Gradient Boost was quite close. Random Forest had the lowest result.

Among other researchers Hanafi and Ming (2021) compared the largest number of different models. In their article they described the test of such models as Logistic Regression, XGBoost, Random Forest, Decision Trees (CART, J48 and C50), Naïve Bayes, and K-neighbours classifier. They used large insurance dataset from a Brazilian company, which was not described due to company's privacy policies. They also applied a large number of different performance measures among previously described researches: Accuracy, Error Rate, Kappa, AUC, Sensitivity, Specificity, Precision, Recall and F1. Based on the analysis of all those measures they concluded that for their dataset Random Forest was the best predictive model. Hanafi & Ming also suggested to use various performance measures in further researches for receiving more comprehensive results.

## 2.4. Summary of the literature review

Literature review shows that predictive algorithms are widely used for solving insurance problems. Besides the topic of the current thesis, there are three another branches of machine

learning algorithms implementations in the insurance industry: prediction of an exact loss occurred by a claim (regression problem), prediction of an emergence of a fraud (classification problem) and prediction of a customer churn (classification problem). The real estate insurance claims prediction was found to be quite a rare topic in the scientific world, so it could be considered as a gap in the existed scientific knowledge. There was only one article, which totally matched with the idea of the current thesis. Other product line of the property insurance, the auto insurance, was much more commonly studied.

The summary of the seven closest studies of property insurance claims emergence prediction is combined in the Table 1. It shows which algorithms were frequently used in previous studies, which ones were concluded to be the best ones and which performance measures were applied.

Table 1. Summary of related studies

| Year | Authors | Methods | Performance measure | Best method |
|---|---|---|---|---|
| 2000 | Smith, Willis & Brooks | DT, NN | Accuracy, ROC | NN |
| 2017 | Jing, Zhao, Sharma & Feng | NB, Bayesian Network | Accuracy | Both |
| 2018 | Fauzan & Murfi | BT(AB), BT(Stochastic GB), NN, RF, BT(XGBoost) | Accuracy | BT(XGBoost) |
| 2019 | Pesantez-Narvaez, Guillen & Alcañiz | LR, BT(XGBoost) | Sensitivity, Specificity, Accuracy, RMSE, ROC | BT(XGBoost) |
| 2020 | Abdelhadi, Elbahnasy & Abdelsalam | DT, NB, NN, BT(XGBoost) | Accuracy, ROC | BT(XGBoost) |
| 2020 | Hanafi & Ming | LR, DT, KNN, NB, RF, BT(XGBoost) | Accuracy, Error Rate, Kappa, AUC (ROC), Sensitivity, Specificity, Precision, Recall, F1 | RF |
| 2021 | Wanyanga | DT, BT(GB), RF, SVM | Accuracy | BT(GB) |

The most common best performed algorithms were different variations of Boosted Trees, but also many other algorithms showed good performance. Among Boosted Trees XGBoost

was the best algorithm in thre researches and Gradient Boosting only in one. However, they were compared against each other only ones and then XGBoost performed better. As many of authors noted, success of the algorithms is highly data specific. The most common performance metrics were Accuracy and ROC.

# 3. Theoretical Background

In the previous chapter there were mentioned various models and their evaluation techniques which were used by earlier researchers. This chapter goes into details of them. Section 3.1 describes the methods and section 3.2 performance measures applied in the current thesis.

## 3.1. Overview of Methods

Table 2 was created based on the seven researches in property insurance claims emergence prediction described in section 2.3. It shows the frequency of each used classification method. Table is organised in order of frequency and in cases of equal frequency in alphabetic order. As one can see, the most frequent ones were Boosted Trees: among them XGBoost was the most popular algorithm and was used four times, while other two were used only once each. Basic Decision Trees have the same frequency of being applied in mentioned above studies as XGBoost. In opposite such methods as K-neighbours classifier, Support Vector Machine and Bayesian Network were not so popular and were used only once each. All methods which were used at least in two different researches are described in this section.

Table 2. Frequency of methods used in related studies

| Method | Frequency |
|---|---|
| Boosted Tree (XGBoost) | 4 |
| Decision Tree | 4 |
| Naïve Bayes | 3 |
| Neural Networks | 3 |
| Random Forest | 3 |
| Boosted Tree (Gradient Boost) | 2 |
| Logistic Regression | 2 |
| Bayesian Network | 1 |
| Boosted Tree (AdaBoost) | 1 |
| K-neighbours classifier | 1 |
| Support Vector Machine | 1 |

### 3.1.1. Logistic regression

The logistic function initially was invented in 19th century for population growth measurement (Cramer, 2002). The currently used logistic regression is based on the works of Berkson (1944) and Cox (1958).

Logistic regression could be considered as a modification of a widely used linear regression. Despites it also has a "regression" word in its name, it is applicable for classification problems instead of regression. The difference is that the explained variable in the logistic regression represent the classes and should be categorical, while explanatory variables could be either numeric or categorical. (Bishop, 2006, 205-206)

Logistic function is a probability function. The output of it is always range between 0 and 1. The simplest implementation of the logistic regression is a binary one with only two classes. In that case regression divide the observations into two groups by the certain point in range from 0 to 1 and they are grouped between the classes based on this division. If there are three or more classes, the discussion is about a multi-class logistic regression, but the process is the same just with more than one cut point. (Cramer, 2002)

The whole logistic regression equation can be written as:

$$ln\left(\frac{P}{1-P}\right) = a + \beta_1 x_1 + \cdots + \beta_m x_m \quad (1)$$

Where $P$ is a probability that a certain observation would be classified to a certain class, $a$ and $\beta$ represent regression coefficients and $x$ represent explanatory variables. (Park, 2013)

Machine learning methods have arguments, which have to be set before the start of the learning process. Those arguments are called hyperparameters. Selection of optimal values

for those parameters can improve a performance of the model. (Feurer and Hutter, 2019) According to Brownlee (2019) Logistic Regression does not have any critical hyperparameters, however, regularisation (a penalty for increase of model's complexity which prevents overfitting) can sometimes increase the performance of the model. A popular regularisation hyperparameter is a loss function. Two of its widely used applications are L1 which uses an absolute value of magnitude of coefficient as penalty term and L2 which uses a squared value of magnitude of coefficient as penalty term. (Yang, 2021) The second common regularisation hyperparameter is a cost. The higher value of this parameter set the higher penalty for a model's complexity. (Brownlee, 2019) Pothabattula (2019) mentioned a tolerance of termination as a possible hyperparamater. However, he also concluded that high value of this parameter could cause too early stop of iteration process, which could lead to a poorer performance of the model. Based on that Pothabattula (2019) suggested to use the default tolerance value.

Logistic regression is a simple algorithm with a short training time, which is easily interpretable and does not require much computational power. In the low dimensional datasets (where number of observations is higher than number of features) it does not have a high tendency for overfitting and is very effective in prediction when the classes are separated linearly. However, it cannot solve non-linear problems, is not efficient in an identification of complex relationships between variables, has a tendency for overfitting in high dimensional datasets (where number of features is higher than number of observations), is sensitive to multicollinearity and outliers and is more suitable for large datasets. (Tu, 1996)

### 3.1.2. Naïve Bayes

Naïve Bayes is based on the Bayes's theorem, which was presented by Bayes in 1763. The idea of the theorem is that it is possible to predict a class $c$ based on prior probabilistic knowledge. The formula of this theorem is:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{2}$$

$P(c|x)$ here is a posterior probability, which should be calculated. It is a probability of $c$ being true when $x$ is true (for example, how often red object is a tomato). $P(x|c)$ is a likelihood. It means how often $x$ is true if $c$ is true (percentage of cases when object defined as a tomato is red). $P(c)$ is a prior probability. It is an initial probability of $c$ is being true before testing evidence (which percentage of objects could be defined tomatoes). $P(x)$ is a marginal probability. It is just a probability of an evidence (percentage of $x$ is being true in a sample – which percentage of objects could be defined as red). (Bayes, 1763)

Naïve Bayes has an assumption of an independency among explanatory variables. In other words, influence of some feature to a class prediction is unrelated to influence of other features. In reality this assumption does not always hold. That is the reason why this algorithm is called "naïve". However, if this assumption is hold, Naïve Bayes demands a shorter training data and outperforms other popular algorithms. It is simple and easily interpretable. In addition, it does not require much computational power, so could be useful for working with large datasets. (Domingos & Pazzani, 1997)

Another on weakness occurs if in the testing dataset categorical variable has a category, which was not presented in the training dataset. In such cases Naïve Bayes simply suggests a zero probability. It can be solved with smoothing techniques. One of them is a Laplace correction, which adds an alpha parameter, which forces the posterior probability being higher than zero. Laplace correction is a tunable hyperparameter for Naïve Bayes. (Domingos & Pazzani, 1997) Majka (2020) mentioned that a type of distribution could also be considered as a tuneable hyperparameter for likelihood modelling. For continuous features by default there is applied a Gaussian distribution, however, it is possible to use instead a kernel distribution to estimate a probability density function in a non-parametric way. In addition, he proposed another one hyperparameter - bandwidth adjustment, which increases a flexibility of kernel density estimation. The higher value this hyperparameter get the smoother the distribution is. (Majka, 2020).

### 3.1.3. Neural Network

The idea of artificial neurons, which are the basis of Neural Networks, was proposed in 1943 (McCulloch & Pitts). It was developed until 1969 when it became to be considered inapplicable in practice based on ideas of Minsky and Papert, who found that Neural Networks are able solve only simple problems, while solving of more sophisticated ones could be possible in theory but would require unrealistically long time or unrealistically high computational power (Minsky and Papert, 1969). However, Neural Networks again became a field of interest in 1980s and gained popularity in 2010s due to a significant increase of computational power availability. (Kurenkov, 2020)

Neural Networks got their name because the principles behind the method are based on a human brain's behaviour. The human brain has a vast number of neurons (nerve cells). Each neuron is connected to many others and gains inputs from them. Inspired by the principals behind the neurons Rosenblatt (1957) proposed an idea of Perceptron (a single layer neural network). Inside the Perceptron a set of inputs should be multiplied by certain weights as presented in the following formula:

$$s = \sum_{i=0}^{n} x_i * w_i \tag{3}$$

The inputs *(x)* there are represented by a sample of explanatory variables. The weights *(w)* are parameters that are specified during the training process. A high weight reflects to an importance of the inputs. After the sum *(s)* is calculated, it is evaluated using an activation function, which decides how it should be transformed into the outputs. In binary classification problems if *s* is high enough based on the activation function, neuron is activated (get the output of 1), otherwise it get the output of 0. An example of the Perceptron is presented on Figure 1. (Kurenkov, 2020)

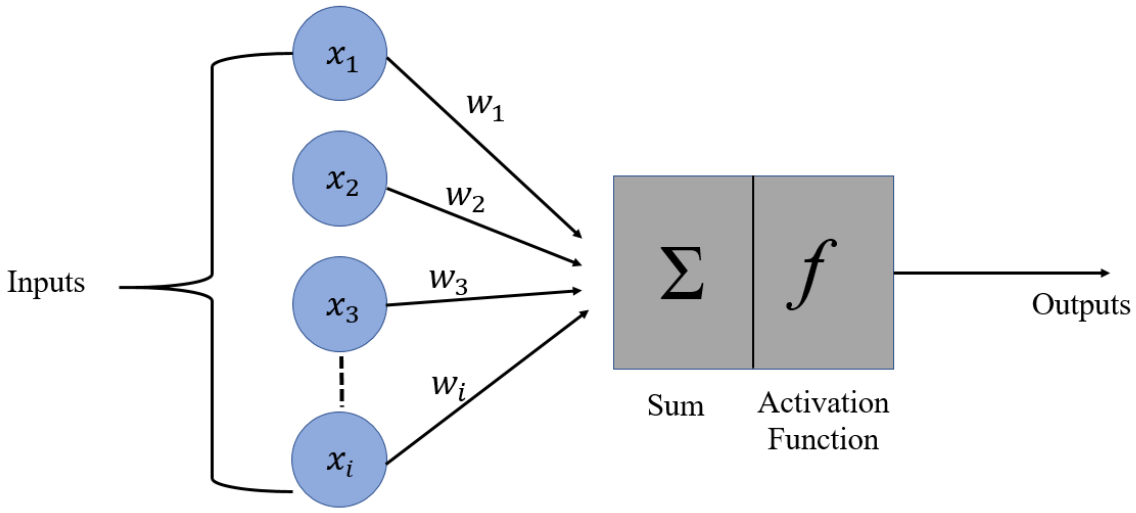



Figure 1. Perceptron

Nowadays Neural Networks uses combinations of multiple perceptron (nodes). Those nodes are grouped into ordered sets of layers. Nodes from the one layer are connected to the nodes in the following and in the previous layers, and the outputs of every middle layer are used as the inputs for the next one. (Bishop, 2006, 225-241)

The first layer, which receives an external data, is called an input layer. There are performed calculation in the nodes and the outputs are generated for being used as inputs on the following layer. Traditional number of nodes for the input layer is a number of features in the training dataset plus one. The last layer is called an output layer. There inputs are taken from the previous layer, calculations performed, and the final result produced. As the result is final, it has only one node. The layers between them are called hidden layers because the true values inside their nodes are unknown for the user, who can investigate only input and output layers. Usually, all hidden layers have the same number of nodes. With increasing number of hidden layers, Neural Networks can gain a possibility to solve problems that are more complex but also become more demandable to the computational power. Number of layers is a tuneable hyperparameter. Neural Networks with two or more hidden layers are called Deep Neural Networks. The structure of the Neural Networks with multiple hidden layers is presented on the Figure 2. (Bishop, 2006, 225-241)

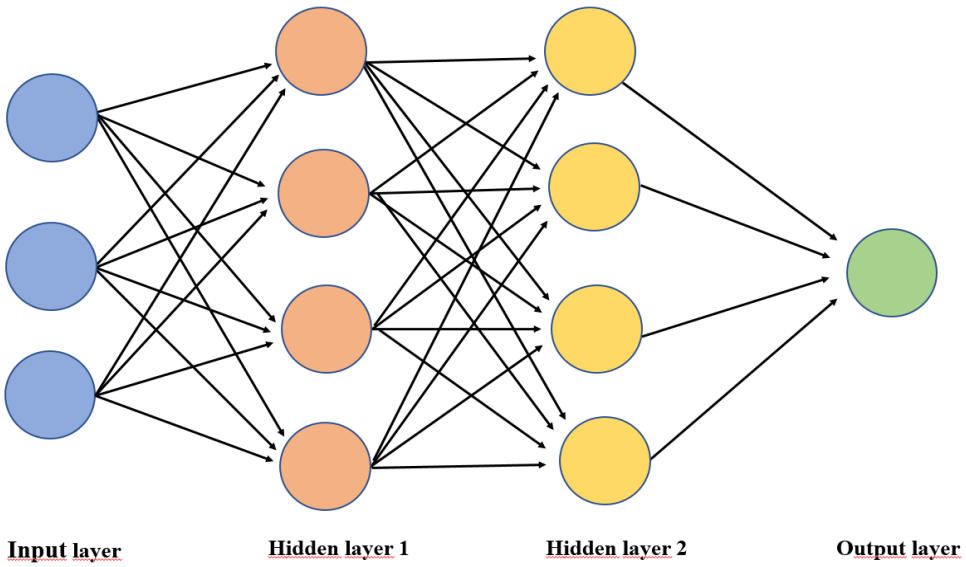



Figure 2. Neural Networks with two hidden layers

In the Neural Networks each neuron gets its unique weights. Initially weights are selected randomly but during the training process they are tuned to the most suitable ones for the selected dataset using a feedback loop that is called backpropagation. On the testing stage weights are fixed. (Chakraborty et al., 2019)

In addition to number of layers, another one tuneable hyperparameters for Neural Networks is called weight decay. The same as loss function in case of Logistic Regression it is a regularisation parameter, because it adds a penalty for model's complexity. The higher the value of weigh decay is, the higher is this penalty. (Vasani, 2019)

Neural Networks have such advantages like high tolerance to noise, missing values and errors in the data. They have distributed memory. Regarding to disadvantages the biggest problem is its "black box" principle – the user receives a solution but does not get an information how it has been achieved, because studying of the Neural Networks function does not provide any insights of this structure, only the result. With increasing number of layers Neural Networks become hardware dependent and computationally expensive. In addition, the method works only with numerical data. (Mijwel, 2018)

### 3.1.4. Random forest

Before a description of Random Forest by itself there should be explained a concept of Decision Tree. It was proposed for solving classification problems by Breiman et al. (1984) and its structure reminds a tree. Decision Tree starts from the root node, where the first splitting happens based on a certain splitting point of some variable. Therefore, the root node divides into several sub-nodes that are called decision nodes. In the decision nodes the same process continues. The final nodes that does not split anymore and contains the answers, are called leaf nodes or terminal nodes. Nodes that are divided into sub-nodes, are named parent nodes and their sub-nodes are named child nodes. (Breiman et al., 1984)

Subsection, which starts from some decision node and contains multiple leaf nodes is called a branch. After the tree is build there can exist some irrelevant nodes (for example if the variable used for the split contains outliers). Those branches could have a negative impact to the tree performance, so they are removed. This process is called pruning. (Breiman et al., 1984)

The variable for the root node are chosen based on some criteria, which is calculated for all variables in the dataset. One of the popular criterias, which is also used in current thesis, is Gini Index, which can be expressed by the following formula:

$$Gini\ Index = 1 - \sum_{i=1}^{n} (P_i) \tag{4}$$

Where $(P_i)$ is a probability that a certain observation would be classified to a certain class. (Daniya et al., 2020)

One of the main advantages of the Decision Trees is its easy interpretability. In addition, Decision Trees directly show which variables are used for split and therefore are the most

important for the class determination. Other strengths of decision trees are an ability to be applied to both continuous and categorical data and an ability to be applied with lower data cleaning compared to other methods. However, the predictions of continuous variables quite often are not accurate enough. Also, Decision Trees have a tendency for overfitting. They are sensitive to the noise in the data. In the end, in case of large datasets Decision Trees could become too complex and computationally expensive. (Hamad & Zeki, 2018)

The idea of Random Forest was generated by Ho (1995). He suggested that the result provided by the group of simultaneously growing Decision Trees could provide more accurate results than a single tree by decreasing variance. The method by itself was formulated by Breiman (2001).

Random Forest uses bagging as an ensemble technique. Random subsamples are selected from the dataset. For each subsample is built its own model and those models are trained independently of each other. An overall result is selected based on the combined results of all models according to a majority principle. (Breiman, 2001). The graphical representation of the idea behind bagging is provided on Figure 3.

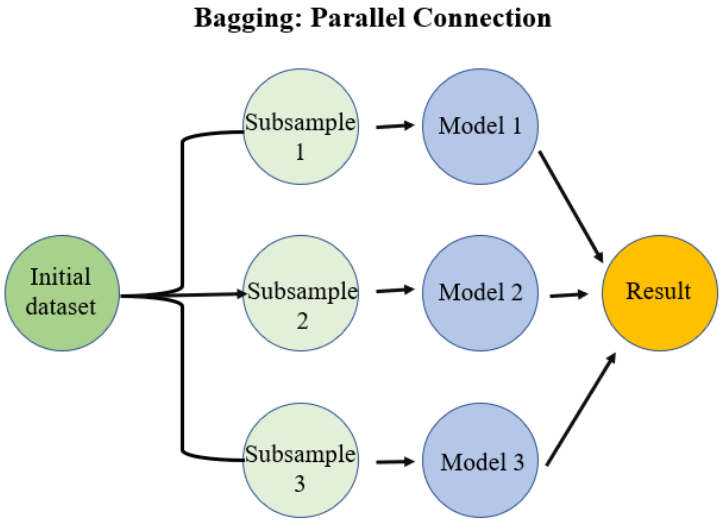



Figure 3. Bagging: Parallel connection

According to Probst et al. (2019), among all hyperparameters in Random Forest, the most influential is a number of variables, which are randomly selected as candidates at each split. The default value for it is a square root from the number of explanatory variables.

Random Forest has a lower tendency to overfitting compared to a single decision tree. It gives a suitable way of dealing with missing data. It is generally more accurate and highly stable as it presents an average result of various independent models. However, it is more complicated and less interpretable, demands longer training time and computational power to be implemented. Also, as Linden and Yarnold (2019) noticed, it could find relationships in a random data.

### 3.1.5. Boosted Trees

Boosted Trees are based on another ensemble technique, which is called boosting. The graphical representation of the idea behind bagging is provided on Figure 4. While bagging uses independent subsamples with their specific models parallelly, boosting uses them sequentially. While bagging tends to decrease variance, boosting decreases bias. False predictions of subsample are identified and on the following step are modified until the algorithm starts to classify outputs correctly. So, boosted trees combined weak learners (decision trees) in series for creating strong learners, which are weighted averages of weak learners. Comparing to Random Forest, new models are not used in parallel manner, but each one learns on the errors of the previous ones and tries to improve them. (Schapire, 1999)

The first boosting model which have gained a popularity is AdaBoost, an adaptive boosting algorithm, which was invented by Freund and Schapire (1996). In AdaBoost wrongly classified observations get higher weights on the next step. If they are wrongly classified on the next step too, they again get even higher weights. This process continues until they are correctly classified.

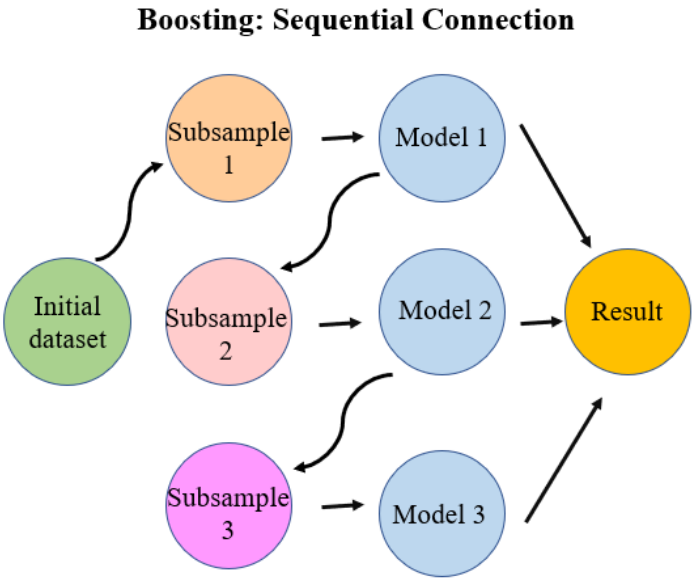



Figure 4. Boosting: Sequential connection

Gradient Boosting is another one sequential connection method, which was proposed by Friedman (2001). The difference from AdaBoost is that in case of Gradient Boosting the weights are not increased. Instead, the loss function is updated in the new model.

eXtreme Gradient Boosting or XGBoost is a modern and improved variant of Gradient Boosting created by Chen (2016). It increases efficiency and computational speed by implementing parallel calculations, distributed computing, out-of-core computing and cache optimization. As Alam (2021) mentioned XGBoost has about 35 hyperparameters in its documentation, but only part of them are frequently used. The most frequent ones are number of boosting iterations, which is a number of decision trees applied in the model and maximum tree depth, which sets how deep each tree can grow. Three other frequently tuned hyperparameters are subsample, shrinkage and minimum loss reduction. Subsample means a percentage of training set used for trees growing. Its default value is 1 which means the whole training dataset and the range of possible values is from 0 to 1. Shrinkage is a regularisation parameter, which shrinks the weight of the feature on each boosting step for overfitting prevention. Its default value is 0.3 and the range of possible values is from 0 to 1. Minimum loss reduction is another one regularisation parameter for pruning. It specifies

the minimum loss reduction for a tree. Its default value is 0 and the range of possible values is from 0 to infinity. In addition, XGBoost has some more specific hyperparameters that are less popular. One example is subsample ratio of columns (a percentage of columns used for a tree growing) with a range of possible values from 0 to 1. It reminds an idea of number of variables selected to the split from RF, but uses all columns as a default value. Another one example is minimum sum of instance weight – a minimum sum of instance weight a leaf node should contain. (Alam, 2021)

Jafarzadeh et al. (2021) mentioned that Boosting Trees have similar advantages and disadvantages as Random Forest. They often outperform other methods in case of predictability and can be implemented on datasets with missing data. However, they are even more computationally expensive, hardly interpretable, sensitive to outliers, have a tendency for overfitting and finding trends in random samples.

## 3.2. Overview of Performance measures

In the same way as previous Section described methods used in previous studies, this Section covers performance measures. In Table 3 performance measures used in property insurance claims emergence prediction studies are presented in order of frequency and in cases of equal frequency in alphabetic order.

Table 3. Frequency of performance measures used in related studies

| Performance measure | Frequency |
|---|---|
| Accuracy | 7 |
| ROC (AUC) | 3 |
| Sensitivity | 2 |
| Specificity | 2 |
| Error rate | 1 |
| F1 | 1 |
| Kappa | 1 |
| Precision | 1 |
| Recall | 1 |
| RMSE | 1 |

As Table 3 shows, Accuracy is the most common measure, which was used in every research. The second most popular one, ROC, was applied in four researches. As it was mentioned in literature review and could be observed from the Table 1, there are only two researches, where there were applied simultaneously more than two measures. Common measures it those two researches were Sensitivity and Specificity. The rest of the measures were applied only once among selected studies. Accuracy, ROC, Sensitivity and Specificity are described below as frequently used performance measures. In addition there was described Cohen's Kappa, which was faced only once among previous studies, but was considered relevant for the current research.

Most of the performance measures are based on the concept of confusion matrix or error matrix. In the simplest case with binary results it is a 2x2 table as presented in Table 4:

Table 4. Confusion matrix

| | **Predicted: Yes** | **Predicted: No** |
|---|---|---|
| **Actual: Yes** | TP | FN |
| **Actual: No** | FP | TN |

True Positive (TP) here means frequency of cases where the tested algorithm proposed a positive result and in reality it is also positive; True Negative (TN) means frequency of cased where of cases where the tested algorithm proposed a negative result and in reality it is also negative; False Positive (FP) means that the tested algorithm proposed a positive result although in reality it should be negative; False Negative (FN) means that the tested algorithm proposed a negative result although in reality it should be positive.

### 3.2.1. Accuracy

According to Ben-David (2007) for a long-time Accuracy was the most commonly used performance measure. It is the most intuitive metric the idea of which follows directly from

the confusion matrix. It is simply the ratio of correct predictions to total predictions. It is calculated by the following formula:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (5)$$

As one of the performance metrics Accuracy could be used in most of the cases, however, it is the most suitable when positive and negative results are similarly distributed in the dataset. In case of imbalanced datasets Accuracy could be biased. For example, if the major class occurs in 90% of the cases and minor class only in 10% of the cases, 90% Accuracy could be achieved by mapping all observations to the major class. It this case a real prediction power of the model will be poor, while Accuracy won't show it. It this case some other performance metrics could be considered. (Provost et al., 1998).

### 3.2.2. Sensitivity

Accuracy is a summary, which tries to combine all information into one single number. That approach cannot cover every aspect of the problem. (Habib et al., 2015) However, the same formula could be divided into two parts: sensitivity (positive accuracy) and specificity (negative accuracy).

Sensitivity is a ratio of true positive values to total positive outcomes. It is written as the following:

$$Sensitivity = \frac{TP}{TP + FN} \quad (6)$$

Sensitivity is useful if determination of correctness of true positive outcomes if essential or a presence of false negatives is critical. It is an important metric when researcher would

rather prefer to ignore false positive outcomes than have any false negative ones. (Habib et al., 2015)

### 3.2.3. Specificity

In opposite to Sensitivity, Specificity is a ratio of true negative values to total negative outcomes. It could be calculated by the following formula:

$$Specificity = \frac{TN}{TN + FP} \tag{7}$$

Specificity is applicable when a presence of false positive is critical. It is an important metric when researcher would rather prefer to ignore false negative values than have any false positive ones. (Habib et al., 2015)

### 3.2.4. ROC and AUC

Receiver Operating Characteristic curve or ROC curve is one of the metrics, which combines information contained in Sensitivity and Specificity. It is a plot of the True Positive Rate (TPR) on y-axis against False Positive Rate (FPR) on x-axis as presented on the Figure 5. TPR is equal to Sensitivity. FPR could be determined as 1 – Specificity. It is calculated by the following formula (Fawcett, 2006):

$$FPR = 1 - Specificity = \frac{FP}{TN + FP} \tag{8}$$

The low values of x-axis represent the situation where TN are high and FP are low. The high values of y-axis represent the situation where TP are high and FN are low. So, the best model would have a high y and a low x. If there is such a clear point on the curve, the best model will be located near it. If the curve is smoother, then the choice of the best threshold is up to

researcher based on the task. If the curve looks like a straight diagonal line, the predictive power of the model is equal to a random guess. (Fawcett, 2006)

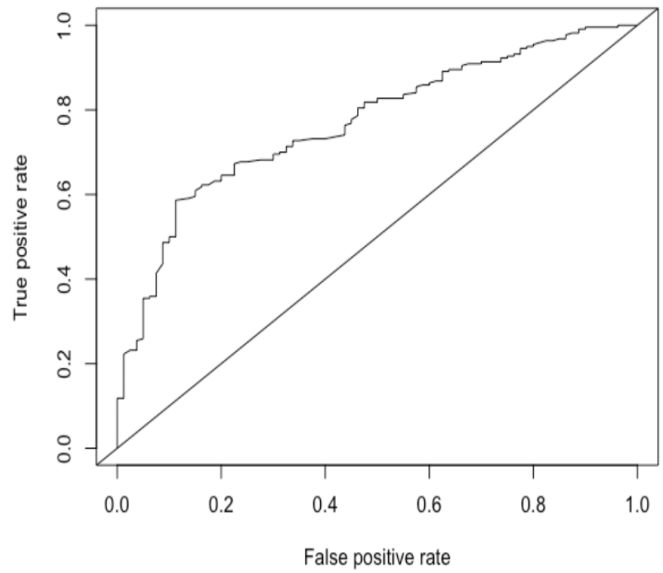



Figure 5. ROC Curve

The Area Under the Curve (AUC) is a numerical representation of ROC, but could be also considered as a separate performance metrics. It measures a separability, which answers the question how well the model predicts TP and TN. The higher is AUC the better is the model. The value of 50% represents a random guess and is used as a benchmark for models evaluation. (Fawcett, 2006)

### 3.2.5. Cohen's Kappa

Cohen's Kappa (or Kappa) is a performance measure, which is widely used when target classes are imbalanced. It tries to fix the bias of accuracy by taking into account a priori distribution. It is calculated by the following formula:

$$k = \frac{p_0 - p_e}{1 - p_e} \tag{9}$$

Where $p_0$ is an overall accuracy of the model, $p_e$ is an overall accuracy which could be achieved with a random guess and 1 indicates to a perfect model with Accuracy equal to 100%. The overall accuracy for a random guess is calculated with a strong assumption that predictions are not affected by the a priori distribution of the target class. (McHugh, 2012)

Cohen's kappa has a range from -1 to 1, where 1 correspond to a perfect model. Kappa is not intuitive nor easily interpretable. However, it is effective in comparing the same model variations with different parameters. (McHugh, 2012)

# 4. Data

This Chapter describes the data used in this research. It starts from the description of data, which was initially available and then goes through its modification, cleaning and preprocessing. The last Section explains data split into training and testing samples and those samples are tested to adequate similarity of distribution among classes.

## 4.1. Description of initial data

The data used in this research was provided by a European insurance company. It is protected by corporate privacy policies, so a detailed description of features could not be presented.

Initially there were two separate datasets in Excel format. The first one contained a list of insured objects for six recent years from 2016 to 2021. All variables in this dataset could be divided into two groups. The larger group contained constructional features of insured objects. The smaller group represented variables, which were insurance policy specific. Most of the variables were numerical, while some of them were categorical. Features from this dataset were used as independent variables. The first dataset initially had 23 variables and 10702 observations in total.

The second dataset contained information about claims for the same time period. This dataset contained 2476 claims. One of the presented variables was policy number. This variable was common with the first dataset and could be used for merging those two datasets. The rest of the information in claims dataset was irrelevant in purpose of the current research or protected by companies policy. However, only the fact that a certain policy number was included in claims dataset was enough for further analysis.

## 4.2. Data modification

This Section describes the process of fata preparation step by step. Initial data modification was implemented in the Excel. Datasets were merged and the combined dataset was used in a further analysis. An explained variable was created in that merged dataset based on the information from the second dataset. The merged dataset was cleaned from missing values. After that categorical variables were one-hot encoded and numerical variables were normalized using R.

### 4.2.1. Datasets merging and explained variable creation

As it was noted, the second dataset included an information about claims emergence. It was noticed that in claims dataset some policies could be presented several times, because there were cases with more than one claim related to the same policy. Before the merging of datasets was started, those observations were merged in such a way that any policy number was presented only once. All policy numbers were copied to a separate Excel sheet. If some policy number was observed more than once, all cases where it was observed were merged. For every policy number was created a new variable where was counted how many times claims were occurred to it.

The first dataset did not require any specific preparation, just some columns with irrelevant information (mostly variables where all observation have the same result like name of the company, type of the insurance, currency etc.) were removed. After that, two described datasets were merged using Excel's Power Query tool. In the result the variable, which contained a number of claims, occurred to each observation was added to the first dataset with a name "Claims total". A total number of claims in the combined dataset was 2465. For eleven claims policy number was not found in the first dataset, so they were considered as having incorrect policy number and dropped from the further analysis.

The “Claims total” variable contained an information how many claims were occurred to a certain policy and had a range of values from 0 to 4. However, the purpose of this thesis was only a binary classification. So, this variable was recoded to a new one for analysing a probability of claims emergence estimation. It was called “Claims emergence” and got values “1” if there was at least one claim occurred to a certain observation or “0” it there was zero claims. This variable had only 1935 observations; it meant that in 530 cases there were more than one claim occurred for a certain policy.

### 4.2.2. Handling missing values

Each variable was tested for missing values. At least some missing data was observed in four variables. In total there were found 63 observations where at least 1 of the independent variables had a missing value. In addition, in most cases missing values were observed in more than one features for the same observation, which could mean that those observations were damaged. As those observations represented less than 1% from the population they were decided to be removed. After that the dataset contained 10 639 policies with claims emerged in 1930 of them or in 18,14% of all cases.

### 4.2.3. Categorical variable recoding

As was mentioned, the dataset contained few categorical variables. All of them were nominal. Most of them were binary with “Yes” and “No” categories. However, in all those variables one of the categories was dominated in about 99% of the cases. Also, some of those variables reminded the ones which Wanyanga (2021) found being unimportant. Therefore, all those variables were considered irrelevant to be included in the analysis.

In addition, the dataset contained one variable with three categories. For dealing with it was applied one-hot encoding. For each category was created a separate column with value “1” if this category is presented for a certain policy and with “0” if not.

#### 4.2.4. Normalization

After categorical variable was recoded, there were 15 explanatory variables in total. 3 variables represented a recoded categorical variable. The rest 12 were numerical variables with different scales. Those variables were from different scales, which according to Loukas (2020) could affect their unequal impact to the results due to increase of bias. For avoiding that situation was applied normalization. Because different classification methods were planned to be compared, for unification purposes the whole dataset was normalized using a very common and basic min-max normalization, which could be described by the following formula:

$$x_{new} = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{10}$$

After it was applied all values in explanatory variables were set inside the range from 0 to 1.

### 4.3. Data split

Data was divided into training and testing datasets with commonly used hold-out split. Although it provides only one training and only one testing dataset, while k-fold cross-validation provides multiple train-test splits, hold-out split makes the testing dataset fully independent. (Kelley, 2017) As Awwalu and Nonyelum (2019) mentioned hold-out split was found to being performed better in the classifier selection tasks, while k-fold cross-validation performed better in parameter optimization tasks. Data was divided in traditional 70/30 proportion into training and testing samples. Train dataset contained 7 447 observations and test dataset 3 192 observations.

Datasets were tested for similarity of distribution among classes using explained variable *Claims emergence*. Total dataset *Claims emergence* had 8 709 observations with 0 value and 1 930 observations with 1, so the distribution was 81,8% / 18,1%. Training sample had 6 096

observations with 0 value and 1 351 observations with 1, so the distribution was the same 81,8% / 18,1%. Testing sample had 2 613 observations with 0 value and 579 observations with 1, so distribution was again the same 81,8% / 18,1%. Therefore, the explained variable was similarly distributed in both training and testing datasets.

# 5. Model specification

This Chapter starts from the selection of the models. Selected models are run with their default parameters on the training dataset and evaluated. Then the Chapter goes through hyperparameters tuning process of each model. After that, the specifications of each model are presented and their performance compared to performance of models with default parameters. All manipulations described in this Chapter are implemented on the training dataset. The final results and performance evaluation on the testing dataset are described in the Chapter 6.

## 5.1. Model selection and specification process description

Selection of the models applied in this research was made based on the previous related studies. All models, which were presented in more than one study were included in the analysis, except of Decision Tree, which was dropped because there were two its more advanced modifications. Among Boosted Trees was selected XGBoost, which was the most common best performed boosted tree application among seven related researches described in literature review. So, the choice of the methods is the following:

1. Logistic Regression
2. Naïve Bayes
3. Neural Networks
4. Random Forest
5. eXtreme Boosted Gradient Tree

The selection of performance measures was done using the same principals: all measures, which were applied in more than one previous study, were selected. As was mentioned before, those measures were the following: Accuracy, Sensitivity, Specificity and AUC (as a numerical representation of ROC Curve). In addition to them was selected Kappa, which was applied only in one previous study, but according to McHugh (2012) was useful in cases with imbalanced data. This feature was considered important based on the distribution of

explained variable among classes. The total number of selected performance measures was five. According the suggestions of Hanagi and Ming (2021) it was higher than number of performance measures applied in most previous researches. Thus, the following performance measures were selected:

1. Accuracy
2. Sensitivity
3. Specificity
4. AUC
5. Kappa

For a better comparability all models were optimized following the same process. Hyperparameter tuning was applied using *train* function in R and methods available in it. Initially all models were run with their default parameters on the training dataset. Then those hyperparameters, which were available for tuning in R *train* function for a chosen method were tuned. Following the findings of Awwalu and Nonyelum (2019) for avoiding overfitting hyperparameter tuning was applied using k-fold cross-validation, with widely common $k = 10$. This mean that training dataset was randomly split into 10 folds where each fold was used as a test dataset and remaining 9 folds as a training dataset. (Brownlee, 2018)

Due to a computational power limitation hyperparameters were tuned one by one. Logical and categorical parameters were tuned with all possible values. For numeric parameter was applied a grid search with 6 or more values from a range, which was relevant for a certain hyperparameter. Hyperparameters were tuned using Kappa, which was chosen due its ability to measure imbalanced data. After a value with the highest Kappa was found, it was used as a fixed value for tuning of the next hyperparameter. When all hyperparameters were tuned, the model with selected tuned hyperparameters was again applied to the whole training dataset described and compared with the model with default parameters.

## 5.2. Logistic regression tuning

For Logistic Regression was applied *"regLogistic"* method from *LiblineaR* package. The method had 3 hyperparameters available for tuning:

- Loss Function (*loss*, character);
- Cost (*cost*, numeric);
- Tolerance (*epsilon*, numeric).

### 5.2.1. LR: Default model

The default parameters for Logistic Regression were the following:

- *loss = L1;*
- *cost = 1;*
- *epsilon = 0.01.*

Initially Logistic Regression was applied to the training dataset with default values of hyperparameters. Resulted confusion matrix is presented in Table 5.

Table 5. Confusion matrix of LR with default values of hyperparameters

| **LR** | **Predicted: Yes** | **Predicted: No** |
|---|---|---|
| **Actual: Yes** | 14 | 1 337 |
| **Actual: No** | 20 | 6 076 |

As it is possible to see, Logistic Regression defined significant number of observations to the majority class. Thus, Accuracy of 81,78% was achieved. However estimation cannot be considered as good and Kappa which was just 1,14% illustrated this. Sensitivity was 1,04%, Specificity 99,67% which means what the model fails to predict positive values. AUC is 50,35% which is close to a random outcome.

### 5.2.2. LR: Tuning process

10-fold cross-validated Kappa of the model with default parameters was 0,92%. Hyperparameters were tuned trying to improve validated kappa in the following order: loss function, cost, tolerance. Loss function was tested with all available values *L1, L2 dual* and *L2 primal*. Kappa was less than 1 percent for all cases, but for *L2 dual* was the highest 0,94%. So, *L2 dual* was selected.

According to its natural limitations, cost should be a positive value; however, its maximum is not limited. Therefore for the tuning were selected few values from the units that are higher and lower than the default value. The selection of values was the following: 0.01, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20 and 50. Kappa got the highest value of 2,04% with *cost* = 50, which was the highest among the selected values. This could mean that a higher penalty for model's complexity improves the model.

Epsilon have a range of possible values from 0 to 1. For its tuning were selected few values from the units that are higher and lower than the default value, including also maximum and minimum possible values. The selection of the values was the following: 0, 0.001, 0.002, 0.005, 0.01, 0.05, 0.1, 0.2, 0.5 and 1. The highest kappa 2,25% was achieved with *epsilon* = 0.1, so it was selected. The tuning results are summarized in Table 6. Hyperparameters are presented in the order they were tuned.

Table 6. LR hyperparameter tuning

| **Hyperparameter** | **Default value** | **Tested values** | **Selected value** |
|---|---|---|---|
| Loss function (loss) | L1 | L1, L2 dual, L2 primal | L2 dual |
| Cost (cost) | 1 | 0.01, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50 | 50 |
| Tolerance (epsilon) | 0.01 | 0, 0.001, 0.002, 0.005, 0.01, 0.05, 0.1, 0.2, 0.5, 1 | 0.1 |

### 5.2.3. LR: Selected model

Table 7 shows changes in Confusion matrix after Logistic Regression with selected values of hyperparameters was applied on the training dataset. There can be observed an improvement of TP and FN predictions. However, predictions of FP and TN were poorer. In general, updated algorithm provided more positive predictions.

Table 7. Changes in LR confusion matrix

| **LR** | **Predicted: Yes** | **Predicted: No** |
|---|---|---|
| **Actual: Yes** | 14 → 24 | 1 337 → 1 327 |
| **Actual: No** | 20 → 32 | 6 076 → 6 064 |

Table 8 shows how performance measures were changed after the model with tuned values was applied on the whole training dataset. Accuracy was almost the same 81,75%. Specificity slightly decreased from 99,67% to 99,47%. The rest of the parameters improved: Kappa from 1,14% to 2,00%, sensitivity from 1,04% to 1,78% and AUC from 50,35% to 50,63%. Such results let make a conclusion that hyperparameter tuning was useful and the model was improved compared to the default one. However, it still did not look like a good model for imbalanced class prediction because even improved sensitivity indicates problems in prediction of positive outcomes and even improved AUC is close to a random one.

Table 8. LR: performance of selected parameters compared to default parameters

| **LR** | **Performance with default values** | **Performance with selected values** |
|---|---|---|
| **Accuracy** | 81,78% | 81,75% |
| **Sensitivity** | 1,04% | 1,78% |
| **Specificity** | 99,67% | 99,47% |
| **AUC** | 50,35% | 50,63% |
| **Kappa** | 1,14% | 2,00% |

## 5.3. Naïve Bayes tuning

For Naïve Bayes was applied "*naive_bayes*" method from *naivebayes* package. It has 3 hyperparameters available for tuning:

- Distribution Type (*usekernel*, logical);
- Bandwidth Adjustment (*adjust*, numeric);
- Laplace Correction (*laplace*, numeric).

### 5.3.1. NB: Default model

The default parameters for Naive Bayes were the following:

- *usekernel* = FALSE;
- *adjust* = 1;
- *laplace* = 0.

Initially Naive Bayes was applied to the training dataset with default values of hyperparameters. Resulted confusion matrix is presented in Table 9.

Table 9. Confusion matrix of NB with default values of hyperparameters

| **NB** | **Predicted: Yes** | **Predicted: No** |
|---|---|---|
| **Actual: Yes** | 382 | 969 |
| **Actual: No** | 988 | 5 108 |

Compared to Logistic Regression, default Naïve Bayes succeed to define considerably more TP. However, number of FN and FP also increased. This could be also observed in the Accuracy of default algorithm of 73,72% which is lower than Accuracy of the majority class. Kappa was 12%, Sensitivity was 28,27%, Specificity was 83,79% and AUC was 56,03%.

### 5.3.2. NB: Tuning process

10-fold cross-validated Kappa of the model with default parameters was 10,67%. Hyperparameters were tuned trying to improve validated Kappa in the following order: distribution type, bandwidth adjustment, Laplace correction.

At the first step was tuned distribution type. It had only two possible outcomes: TRUE or FALSE. TRUE value of this parameter decreased Kappa, so this parameter remained unchanged, which meant that Gaussian distribution was applied. Based on that the tuning of the second hyperparameter became unnecessary, because as Majka (2020) mentioned in was designed specifically for kernel distribution. Despites of that for confirming an absence of its impact it was tuned with set of values from the units that are higher and lower than the default value: 0.1, 0.5, 1, 2, 5, 10, 50, and 100. As was expected Kappa was the same for all those values, so the default value *adjust* = 1 remained.

The third tuneable hyperparameter was Laplace correction, which had a range of possible values from 0 to infinity. In addition to default value of 0 were selected few values from different units. The selection of the values was the following: 0, 0.1, 0.5, 1, 2, 5, 10, 50 and 100. Kappa was the same for all those values, so the default value *laplace* = 0 remained.

All three hyperparameters remained unchanged, which can be observed from the results of tuning presented in Table 10. Hyperparameters are presented in the order they were tuned. Therefore, in case of Naïve Bayes hyperparameter tuning did not provide any additional value. So, the selected model remained the same as the default one with the same results provided by performance measures.

Table 10. NB hyperparameter tuning

| Hyperparameter | Default value | Tested values | Selected value |
|---|---|---|---|
| Distribution Type (usekernel) | FALSE | TRUE, FALSE | FALSE |
| Bandwidth Adjustment (adjust) | 1 | 0.1, 0.5, 1, 2, 5, 10, 50, 100 | 1 |
| Laplace Correction (laplace) | 0 | 0, 0.1, 0.5, 1, 2, 5, 10, 50, 100 | 0 |

## 5.4. Neural Network tuning

For Neural Network was applied "*nnet*" method from *nnet* package. It has 2 hyperparameters available for tuning:

- Number of Hidden Units (*size*, numeric);
- Weight Decay (*decay*, numeric).

### 5.4.1. NN: Default model

The default parameters for Neural Networks were the following:

- *size* = 1;
- *decay* = 1.

Initially Neural Networks were applied to the training dataset with default values of hyperparameters. Resulted confusion matrix is presented in Table 11.

Table 11. Confusion matrix of NN with default values of hyperparameters

| NN | Predicted: Yes | Predicted: No |
|---|---|---|
| **Actual: Yes** | 5 | 1 346 |
| **Actual: No** | 5 | 6 091 |

Default Neural Networks had the same problem as Logistic Regression, but in case of NN it was even more extreme. It assigned almost all observations to a major class. Thus, was achieved accuracy of 81,86% with kappa equal to 0,47%. Sensitivity was 0,37%, Specificity 99,92% and AUC 50,14%.

### 5.4.2. NN: Tuning process

10-fold cross-validated Kappa of the model with default parameters was 0,35%. Hyperparameters were tuned trying to improve validated kappa in the following order: number of hidden units first and weight decay second. Number of hidden layers should be a natural number. Following the ideas of Luwes (2010) the grid search for this hyperparameter was applied with Fibonacci numbers 1, 2, 3, 5, 8, 13, 21 and 34. Highest Kappa of 0,81% was achieved with *size* = 13, so this value was selected.

Weight decay has a range of possible values from 0 to infinity. For tuning were selected few values from the units that are higher and lower than the default value, including also the minimum possible value. The selection of the values was the following: 0, 0.001, 0.002, 0.005, 0.01, 0.05, 0.1, 0.2, 0.5, 1, 2, 5 and 10. Highest Kappa was achieved with *decay* = 0.005, so some penalty for model's complexity was useful. The tuning results are summarized in the Table 12. Hyperparameters are presented in the order they were tuned.

Table 12. NN hyperparameter tuning

| **Hyperparameter** | **Default value** | **Tested values** | **Selected value** |
|---|---|---|---|
| Number of Hidden Units (size) | 1 | 1, 2, 3, 5, 8, 13, 21, 34 | 13 |
| Weight Decay (decay) | 1 | 0, 0.001, 0.002, 0.005, 0.01, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10 | 0.005 |

### 5.4.3. NN: Selected model

Table 13 shows changes in Confusion matrix after Neural Networks with selected values of hyperparameters was applied on the training dataset. The same as was observed in case of LR, predictions of TP and FN were improved. However, predictions of FP and TN were poorer. In general, updated algorithm provided more positive predictions.

Table 13. Changes in NN confusion matrix

| **NN** | **Predicted: Yes** | **Predicted: No** |
|---|---|---|
| **Actual: Yes** | 5 → 23 | 1 346 → 1 328 |
| **Actual: No** | 5 → 23 | 6 091 → 6 073 |

Table 14 shows how performance measures were changed after the model with tuned values was applied on the whole training dataset. The results are also similar with the results of LR tuning. Accuracy was absolutely the same. Specificity slightly decreased from 99,92% to 99,62%. The rest of the parameters improved: Kappa from 0,47% to 2,12%, Sensitivity from 0,37% to 1,70% and AUC from 50,14% to 50,66%. Such results let make a conclusion that hyperparameter tuning was useful and the model was improved compared to default one. However, it still did not look like a good model for imbalanced class prediction because even improved sensitivity indicates problems in prediction of positive outcomes and even improved AUC is close to a random one.

Table 14. NN: performance of selected parameters compared to default parameters

| **NN** | **Performance with default values** | **Performance with selected values** |
|---|---|---|
| **Accuracy** | 81,86% | 81,86% |
| **Sensitivity** | 0,37% | 1,70% |
| **Specificity** | 99,92% | 99,62% |
| **AUC** | 50,14% | 50,66% |
| **Kappa** | 0,47% | 2,12% |

## 5.5. Random Forest tuning

For Random Forest was applied "*rf*" method from *randomForest* package. It has only 1 hyperparameter available for tuning:

- Number of Randomly Selected Predictors (*mtry*, numeric).

### 5.5.1. RF: Default model

The only default parameter *mtry* for Random Forest was the following:

- *mtry* = 4 (rounded square root from number of variables).

Initially Random Forest was applied to the training dataset with default values of hyperparameters. Resulted confusion matrix is presented in Table 15.

Table 15. Confusion matrix of RF with default values of hyperparameters

| **RF** | **Predicted: Yes** | **Predicted: No** |
|---|---|---|
| **Actual: Yes** | 1 307 | 44 |
| **Actual: No** | 0 | 6 096 |

Default Random Forest looked like it had a perfect performance. There were only 44 FN observations. The rest observations were predicted correctly. Accuracy was 99,41%, kappa 97,99%, sensitivity 96.74%, specificity 100% and AUC 98,37%. Such comparably high measures could mean that model was overfitted.

### 5.5.2. RF: Tuning process

10-fold cross-validated Kappa of the model with default parameters was 9,94%. The reduction of validated Kappa compared to Kappa from the whole training dataset proved that the model was overfitted. The only available for tuning hyperparameter (number of

randomly selected predictors) was tuned with all possible values: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 and 15. Highest kappa 10,79% was achieved with *mtry* = 7. The results of tuning process are summirezed in Table 16.

Table 16. RF hyperparameter tuning

| Hyperparameter | Default value | Tested values | Selected value |
|---|---|---|---|
| Number of Randomly Selected Predictors (mtry) | 4 | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 | 7 |

### 5.5.3. RF: Selected model

Table 17 shows changes in Confusion matrix after Random Forest with selected values of hyperparameters was applied on the training dataset. Number of TP increased from 1 307 to 1 348. In the same time number of FN decreased from 44 to 3. According to Confusion matrix the fitting of the model on the training dataset was improved.

Table 17. Changes in RF confusion matrix

| RF | Predicted: Yes | Predicted: No |
|---|---|---|
| **Actual: Yes** | 1 307 → 1348 | 44 → 3 |
| **Actual: No** | 0 → 0 | 6 096 → 6 096 |

Table 18 shows how performance measures were changed after the model with tuned values was applied on the whole training dataset. All of the performance measures showed an improvement, which let make a conclusion that hyperparameter tuning improved model's fitting on training dataset. However, 10-fold cross-validation also proved that model is overfitted, therefore a similarly high performance on the testing dataset is not guaranteed.

Table 18. RF: performance of selected parameters compared to default parameters

| RF | Performance with default values | Performance with selected values |
|---|---|---|
| **Accuracy** | 99,41% | 99,96% |
| **Sensitivity** | 96,74% | 99,78% |
| **Specificity** | 100,00% | 100,00% |
| **AUC** | 98,37% | 99,89% |
| **Kappa** | 97,99% | 99,86% |

## 5.6. eXtreme Gradient Boosted tree tuning

For eXtreme Gradient Boosted Tree was applied "*xgbTree*" method which demanded *xgboost* and *plyr* packages. It had 7 hyperparameters available for tuning:

- Number of Boosting Iterations (*nrounds*, numeric);
- Max Tree Depth (*max_depth*, numeric);
- Shrinkage (*eta*, numeric);
- Minimum Loss Reduction (*gamma*, numeric);
- Subsample Percentage (*subsample*, numeric);
- Subsample Ratio of Columns (*colsample_bytree*, numeric);
- Minimum Sum of Instance Weight (*min_child_weight*, numeric).

### 5.6.1. XGBoost: Default model

The default parameters for eXtreme Gradient Boosted Tree were the following:

- *nrounds = 100;*
- *max_depth = 6;*
- *eta = 0.3;*
- *gamma = 0;*
- *subsample = 1;*
- *colsample_bytree = 1;*
- *min_child_weight = 1.*

Initially XGBoost was applied to the training dataset with default values of hyperparameters. Resulted confusion matrix is presented in Table 19.

Table 19. Confusion matrix of XGBoost with default values of hyperparameters

| **XGBoost** | **Predicted: Yes** | **Predicted: No** |
|---|---|---|
| **Actual: Yes** | 587 | 764 |
| **Actual: No** | 11 | 6 085 |

XGBoost with default parameters looked like it performed relatively well. It did not assign all observations to the same class, instead succeeded to correctly predict 587 TP. Confusion matrix showed had just few FP, but there was a significant number of FN. Accuracy was 89,59% which was higher than major class. Kappa was 55,25%, Sensitivity was 43,45%, Specificity was 99,82% and AUC 71,63%.

### 5.6.2. XGBoost: Tuning process

10-fold cross-validated Kappa of the model with default parameters was 7,46%. The same as in case of Random forest, reduction of validated Kappa compared to Kappa from the whole training dataset proved that the model was overfitted. Hyperparameters were tuned trying to improve validated Kappa in the following order: number of boosting iterations, max tree depth, shrinkage, minimum loss reduction, subsample percentage, subsample ratio of columns and minimum sum of instance weight.

Number of boosting iterations was tuned for set of values with a step of 50 following the principals implemented in previous researches (Hanafi and Ming, 2021). The set of the tuned values was the following: 50, 100, 150, 200, 250, 300, 350, 400, 450 and 500. Highest kappa was achieved with *nrounds* = 350, so a number higher than the default one provided better performance.

Max tree depth should be a natural number. For tuning were selected few numbers that are higher and lower than the default value. The selection of the values was the following: 1, 2, 4, 6, 8, 10, 15, 20, 30 and 40. Highest kappa was achieved with *max_depth* = 15, so a number higher than the default one provided better performance.

Shrinkage had a range of possible values from 0 to 1. It was tuned for the set of values with a step of 0.1. The selection of the values was the following: 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1. Highest Kappa was achieved with *eta* = 0.4, which was a slightly higher than the default value.

Minimum loss reduction had a range of possible values from 0 to infinity. For tuning were selected few values from the units that are higher and lower than the default value, including also the minimum possible value. The selection of the values was the following: 0, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5 and 10. Highest Kappa was achieved with *gamma* = 0.005, so a small positive value provided better performance than default zero. Improvements of Kappa reasoned by increase of shrinkage and minimum loss reduction indicated that the model benefited from a slight increase of penalty for its complexity.

Subsample had a range of possible values from 0 to 1. It was tuned for the set of values with a step of 0.1. The selection of the values was the following: 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1. Highest Kappa was achieved with default *subsample* = 1, so this parameter remained unchanged.

Subsample ratio of columns had a range of possible values from 0 to 1. It was tuned for the set of values with a step of 0.1. The selection of the values was the following: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1. Highest Kappa was achieved with *colsample_bytree* = 0.4. Thus, using of 40% of the columns was more beneficial compared to the use of all of them.

Minimum sum of instance weight had a range of possible values from 0 to infinity. For tuning were selected few values from the units that are higher and lower than the default value, including also the minimum possible value. The selection of the values was the following: 0, 0.001, 0.01, 0.1, 1, 2, 5 and 10. Highest Kappa was achieved with *min_child_weight* = 0.01, which was significantly lower than the default value. The tuning results are summarized in the Table 20. Hyperparameters are presented in the order they were tuned.

Table 20. XGBoost hyperparameter tuning

| **Hyperparameter** | **Default value** | **Tested values** | **Selected value** |
|---|---|---|---|
| Number of Boosting Iterations (nrounds) | 100 | 50, 100, 150, 200, 250, 300, 350, 400, 450 and 500 | 350 |
| Max Tree Depth (max_depth) | 6 | 1, 2, 4, 6, 8, 10, 15, 20, 30, 40 | 15 |
| Shrinkage (eta) | 0.3 | 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1 | 0.4 |
| Minimum Loss Reduction (gamma) | 0 | 0, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10 | 0.005 |
| Subsample Percentage (subsample) | 1 | 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6,0.7, 0.8, 0.9, 1 | 1 |
| Subsample Ratio of Columns (colsample_bytree) | 1 | 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1 | 0.4 |
| Minimum Sum of Instance Weight (min_child_weight) | 1 | 0, 0.001, 0.01, 0.1, 1, 2, 5, 10 | 0.01 |

### 5.6.3. XGBoost: Selected model

Table 21 shows changes in Confusion matrix after XGBoost with selected values of hyperparameters was applied on the training dataset. Number of TP increased from 527 to 1 350. TN were increased from 6 085 to 6 094 and FP decreased from 11 to 2. However, the most important change was happened with FN predictions. Default model contained a significant amount of FN, while the model with selected parameters had only two of them.

Table 21. Changes in XGBoost confusion matrix

| XGBoost | Predicted: Yes | Predicted: No |
|---|---|---|
| **Actual: Yes** | 587 → 1 350 | 764 → 1 |
| **Actual: No** | 11 → 2 | 6 085 → 6 094 |

Table 22 shows how performance measures were changed after the model with tuned values was applied on the whole training dataset. All of the performance measures showed an improvement, which let make a conclusion that hyperparameter tuning improved model's fitting on training dataset. However, 10-fold cross validation also proved that model is overfitted, therefore a similarly high performance on the testing dataset is not guaranteed. This observation was in line with findings of Pesantez-Narvaez et al. (2019) who also reported a relatively high performance of XGBoost on the training dataset, which was explained by overfitting. In general, situation with XGBoost looks similar with situation with Random Forest. The difference is that Random Forest had a performance of all hyperparameters close to 100% even before hyperparameter tuning, while XGBoost achieved such performance just during the tuning process.

Table 22. XGBoost: performance of selected parameters compared to default parameters

| XGBoost | Performance with default values | Performance with selected values |
|---|---|---|
| **Accuracy** | 89,59% | 99,96% |
| **Sensitivity** | 43,45% | 99,93% |
| **Specificity** | 99,82% | 99,97% |
| **AUC** | 71,63% | 99,95% |
| **Kappa** | 55,25% | 99,86% |

# 6. Results

This Chapter starts from the summary of selected values for hyperparameters for all five methods applied in this thesis, which is presented in Table 23. After the performance of each model is presented in confusion matrices. Then the results are compared using such performance measures as Accuracy, Sensitivity, Specificity, AUC (linked to the ROC curves presented in Appendices) and Kappa. In the end, results provided by each model are scored based their performance according to each performance measure and the best model was selected based on the sum of the scores.

Table 23. Summary of selected hyperparameters

| **Model** | **Hyperparameter** | **Selected value** |
|---|---|---|
| LR | Loss function (loss) | L2 dual |
| | Cost (cost) | 50 |
| | Tolerance (epsilon) | 0.1 |
| NB | Distribution Type (usekernel) | FALSE |
| | Bandwidth Adjustment (adjust) | 1 |
| | Laplace Correction (laplace) | 0 |
| NN | Number of Hidden Units (size) | 13 |
| | Weight Decay (decay) | 0.005 |
| RF | Number of Randomly Selected Predictors (mtry) | 7 |
| XGBoost | Number of Boosting Iterations (nrounds) | 350 |
| | Max Tree Depth (max_depth) | 15 |
| | Shrinkage (eta) | 0.4 |
| | Minimum Loss Reduction (gamma) | 0.005 |
| | Subsample Percentage (subsample) | 1 |
| | Subsample Ratio of Columns (colsample_bytree) | 0.4 |
| | Minimum Sum of Instance Weight (min_child_weight) | 0.01 |

## 6.1. Confusion Matrices

Confusion matrices for all five methods are presented in Figure 6. They show that LR and NN failed to predict imbalanced classes and just assigned all values to the major class. NB, RF and XGBoost predicted both positive and negative outcomes, but all three models had a significant number of FP and FN predictions. NB showed the highest number of false predictions (FP+FN). Also, it has the lowest number of TN. In the same time, it has the highest number of TP.

| **LR** | **Predicted: Yes** | **Predicted: No** |
|---|---|---|
| **Actual: Yes** | 9 | 570 |
| **Actual: No** | 12 | 2 601 |

| **NB** | **Predicted: Yes** | **Predicted: No** |
|---|---|---|
| **Actual: Yes** | 262 | 317 |
| **Actual: No** | 727 | 1 886 |

| **NN** | **Predicted: Yes** | **Predicted: No** |
|---|---|---|
| **Actual: Yes** | 11 | 568 |
| **Actual: No** | 16 | 2 597 |

| **RF** | **Predicted: Yes** | **Predicted: No** |
|---|---|---|
| **Actual: Yes** | 70 | 509 |
| **Actual: No** | 153 | 2 460 |

| **XGBoost** | **Predicted: Yes** | **Predicted: No** |
|---|---|---|
| **Actual: Yes** | 103 | 476 |
| **Actual: No** | 212 | 2 401 |

Figure 6. Confusion matrices of testing dataset

## 6.2. Models evaluation using performance metrics

This section presents the results of all models based on selected performance metrics. The results are summarized in Table 24.

Table 24. Results of selected models based on performance metrics

| | Accuracy | Sensitivity | Specificity | AUC | Kappa |
|---|---|---|---|---|---|
| **LR** | 81,77% | 1,55% | 99,54% | 50,55% | 1,75% |
| **NB** | 67,29% | 45,25% | 72,18% | 58,71% | 13,66% |
| **NN** | 81,70% | 1,90% | 99,39% | 50,64% | 2,05% |
| **RF** | 79,23% | 12,09% | 94,11% | 53,10% | 8,14% |
| **XGBoost** | 78,45% | 17,79% | 91,89% | 54,84% | 11,76% |

Logistic Regression has the highest Accuracy (81,77%), but algorithm did not found a way to make correct predictions and just assigned almost all observation to the majority class, so this Accuracy is biased. Specificity is too high (99,54%) but at the same time Sensitivity is too low (1,55%). AUC is close to random results (50,55%) and the ROC Curve is close to a straight line (Appendix 1). Kappa is also very low (1,75%). The results of LR application on the testing dataset are in line with the results on the training dataset.

Naïve Bayes showed poorer Accuracy (67,29%) than any other model. However, it provided the most balanced combination of Sensitivity (45,25%) and Specificity (72,18%), while in case of all other models Specificity was many times higher than Sensitivity. AUC (58,71%) and Kappa (13,66%) were the highest among other models, but still not so high in absolute terms. ROC Curve (Appendix 2) is clearly above the diagonal line and looks clearly better than ROC Curves of all other models. The results of NB application on the testing dataset are in line with the results on the training dataset.

In case of Neural Networks situation is the same as with LR. Accuracy is closed to a one of majority class (81,70%), Sensitivity is too low (1,90%) while Specificity is too high (99,39%), Kappa is also too low (2,05%), AUC is close to a random one (50,64%) and ROC

Curve (Appendix 3) is close to a straight line. The results of NN application on the testing dataset are in line with the results on the training dataset.

Random Forest provided an average performance. Accuracy is a little bit lower (79,23%), than in case of LR or NN. On the other hand, Sensitivity is about 6 times higher (12,09%), while Specificity is also not so close to 1 (94,11%) and Kappa is also about 4 times higher (8,14%). AUC is slightly higher (53,10%) than the random guess value. The same can be observed from the ROC Curve (Appendix 4) which is only slightly above the diagonal line. The results of RF application on the testing dataset are significantly poorer compared to the results on the training dataset.

In case of XGBoost Accuracy (78,45%) and Specificity (91,89%) are slightly lower than in RF. In the same time Sensitivity (17,79%) and Kappa (11,76%) are about 1,5 times higher. AUC is slightly higher (54,84%). The same could be observed from the ROC Curve of XGBoost (Appendix 5), which reminds the ROC Curve of RF, but is slightly above it. The results of XGBoost application on the testing dataset are significantly poorer compared to the results on the training dataset.

## 6.3. Selection of the best performed model

For the selection of the best performed model the results of all five models were scored from 1 to 5 based on each performance metric. After that, scores for each models were summarized to the total score, which was used for the final comparison of the models. All scores are presented in Table 25.

Table 25. Scored results of selected models

| | Accuracy | Sensitivity | Specificity | AUC | Kappa | Total score |
|---|---|---|---|---|---|---|
| **LR** | 5 | 1 | 5 | 1 | 1 | 13 |
| **NB** | 1 | 5 | 1 | 5 | 5 | 17 |
| **NN** | 4 | 2 | 4 | 2 | 2 | 14 |
| **RF** | 3 | 3 | 3 | 3 | 3 | 15 |
| **XGBoost** | 2 | 4 | 2 | 4 | 4 | 16 |

LR and NN got high scores according to Accuracy and Specificity, however it was explained by assigning all results to the majority class. According to other three performance measures they got the worst scores. Total scores of those two models were also the lowest. RF provided an average performance according to all performance measures and also is the third best model according to the sum of scores. XGBoost showed the second worst Accuracy and Specificity, but the second best Sensitivity, AUC and Kappa. Based on the sum of scores it is the second best model. In the same way Naïve Bayes performed the worst in case of Accuracy and Specificity, but showed the best Sensitivity, AUC and Kappa. The highest sum of scores let conclude that NB can be considered as the best performed model among the selected ones.

# 7. Conclusions

This Chapter contains three Sections. It starts from the answers to the research question. In the second Section the results presented in the Chapter 6 are analysed and discussed. In the last Section there are presented some ideas for further researches.

## 7.1. Answers to the research questions

Two first research questions were related to the previous studies, so the thesis starts from the literature review. It was found that application of machine learning algorithms in the insurance industry is widely discussed in the scientific literature. Focuses of the majority of studies are loss payment estimation, fraud prediction and customer retention. Among different implementations of machine learning algorithms there were found seven studies, where the topic was a prediction of claims emergence in the property insurance field.

The first research question was formulated as the following:

1. *Which machine learning algorithms have been frequently used in the previous studies?*

Boosted Trees were the most frequent method in general. Different types of Boosted Trees were applied seven times. Among them the most frequent one was eXtreme Gradient Boosted Tree, which was applied four times. Decision Tree has the same frequency of implementation. Naïve Bayes, Neural Networks and Random Forest were applied three times each. Logistic Regression was used in two studies.

The second research question was:

2. *Which performance measures have been frequently used in the previous studies?*

In the same way the answer was presented in Table 3 in the Section 3.2. The most frequently used performance measure was Accuracy, which was applied in all seven studies. The secondly frequent was ROC, which was applied in four studies. Sensitivity and Specificity were applied only twice.

The rest two questions were related to performance evaluation of the selected models. The third question was the following:

3. *How would different classification methods perform compared to each other in the real estate insurance claims emergence prediction on the provided dataset?*

Summarizing the results Logistic Regression and Neural Networks failed to predict imbalanced classes and assigned all observations to majority "no claim" class. Thus, they achieved an Accuracy of majority class, which was higher than Accuracy provided by other models, and Specificity close to 100%. It terms of Sensitivity, AUC and Kappa those two algorithms were the poorest ones. The rest three algorithms predicted both classes, but also made many false predictions. Naïve Bayes provided the highest Sensitivity, AUC and Kappa, but showed the lowest Accuracy and Specificity. Also, it got the lowest TN and the highest number of false predictions. XGBoost provided the second best result according to Sensitivity, AUC and Kappa and the second worst according to Accuracy and Specificity. Performance of Random Forest was in between the performance of other algorithms according to all metrics. More details related to this question were provided in Sections 6.2 and 6.3.

The last question was formed as:

4. *Which of the selected classification methods performed the best in the real estate insurance claims emergence prediction on the provided dataset?*

It was also asked in the Section 6.3. When the results of each performance measures were ranged from 1 to 5 among all selected models, Naïve Bayes got the highest score. Based on that it could be chosen as the best performed algorithm.

## 7.2. Analysis of the results

Although some algorithms showed the result which was better compared to a random guess, the results are not satisfied in case of practical usability. Some model performed better than others did in terms on specific performance measures, but in absolute terms none of them could be called successful. Even Naïve Bayes, which performed the best according to summarized scores, have an Accuracy of only 67,27%, which is lower than Accuracy of assigning all observations to majority class without any analysis. All of the algorithms showed certain disadvantages, which made their implementation in business routine highly questionable at least at current stage. The current research was limited by several factors, which are discussed in more details in the next Section. Changes in those factors could potentially improve the performance of solving the problem of claims emergence prediction.

An alternative way of implementation of the results of this research was proposed by findings provided by Logistic Regression and Neural Networks. By assigning all predictions to the majority "no claim" class, those algorithms supported an idea of randomness of claims emergence. So, those algorithms suggested to make an assumption that all policies have no claims expectations and none of them should be excluded from the risk portfolio. Despites the poor performance of those two algorithms according to the majority of the selected performance measures, the solution suggested by LR and NN could be considered as a reasonable. It also confirms the business routine, which is currently existing in the company.

## 7.3. Limitations and ideas for future researches

Limitations of this study start from the data. There was used the data provided by a single insurance company for a limited time period. In addition, the number of features was limited

by those, which were included in the dataset, while in reality insured objects had more measurable features. Including of additional features into analysis could potentially improve the results.

The second limitation is related to the methods applied in this study. There were applied only those methods, which were frequently used in the previous related researches. In further researches an application of some new method, which was not tested before for claims emergence prediction could be considered. As another option, the same algorithms could be used with their modern complications.

Hyperparameters in this research were tuned one by one and only for a limited number of possible values. A higher computational power could make a possibility to implement the grid search for a higher number of different possible values and to test all hyperparameters simultaneously.

# References

Abdelhadi, S., Elbahnasy, K. and Abdelsadam, M. 2020. A proposed model to predict auto insurance claims using machine learning techniques. Journal of Theoretical and Applied Information Technology. Vol.98. No 22.

Alam, M. 2021. A guide to XGBoost hyperparameters. Towards Data Science. Accessed 29 May 2022. Available: https://towardsdatascience.com/a-guide-to-xgboost-hyperparameters-87980c7f44a9

AM Best. 2019. Best's Guide to Understanding the Insurance Industry. Supplement to Best's Review November 2019. Accessed 29 May 2022. Available: https://www.ambest.com/review/UnderstandingInsuranceGuide.pdf

Awwalu, J. and Nonyelum, O. F. 2019. On Holdout and Cross Validation A Comparison between Neural Network and Support Vector Machine. nternational Journal of Trend in Research and Development, Volume 6(2), ISSN: 2394-9333. Kaduna. Nigeria.

Baecke, P., & Bocca, L. 2017. The value of vehicle telematics data in insurance risk selection processes. Decision Support Systems, 98, 69.

Bayes, T. 1763. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to John Canton, A. M. F. R. SPhil. Trans. R. Soc.53370–418.

Belhadji, E.B., Dionne, G and Tarkhani, F. 2000. A model for the detection of insurance fraud. The Geneva Papers on Risk and Insurance-Issues and Practice, 25(4): 517–538.

Ben-David, A. 2007. A lot of randomness is hiding in accuracy. Engineering Applications of Artificial Intelligence, 20(7), 875–885.

Bentley, P. J. 2000. Evolutionary, my dear Watson investigating committee-based evolution of fuzzy rules for the detection of suspicious insurance claims. In Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation. Morgan Kaufmann Publishers Inc., pp. 702–709.

Berkson, J. 1944. Application of the Logistic Function to Bio-Assay. Journal of the American Statistical Association, 9, 357-365.

Bishop, C. M. 2006. Pattern recognition and machine learning. New York, NY: Springer Science Business Media, LLC.

Breiman, L. 2001. Random Forests. Machine Learning 45, 5–32.

Breaiman, L., Friedmen, J., Stone, C. J. and Olshen, R. A. 1984. Classification and Regression Trees. Chapman and Hall/CRC. 1st edition.

Brockett, PL, Derrig, RA, Golden, LL, Levine, A and Alpert, M. 2002. 'Fraud classification using principal component analysis of ridits'. Journal of Risk and insurance, 69(3): 341–371.

Brownlee, J. 2018. A Gentle Introduction to k-fold Cross-Validation. Machine Learning Mastery. Statistics. Accessed 29 May 2022. Available: https://machinelearningmastery.com/k-fold-cross-validation/

Brownlee, J. 2019. Tune Hyperparameters for Classification Machine Learning Algorithms. Machine Learning Mastery. Python Machine Learning. Accessed 29 May 2022. Available: https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/

Chacraborty, K., Bhattacharyya, S., Bad, R. and Hassanien, A. A. 2019. 7 - Sentiment Analysis on a Set of Movie Reviews Using Deep Learning Techniques. Social Network Analytics. Computational Research Methods and Techniques. Pages 127-147.

Chen, T. and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. KDD'16, 785-794.

Cox, D. 1958. The regression analysis of binary sequences. Journal of the Royal Statistical Society. Series B (Methodological), 20(2), 215-242.

Cramer, J. S. 2002. The Origins of Logistic Regression. Vol. 119. Tinbergen Institute. pp. 167–178.

Daniya, T., Geetha, M. and Suresh Kumar, K. 2020. Classification and regression trees with gini index. Advances in Mathematics Scientific Journal 9(10):1857-8438.

Denney, A. S & Tewksbury, R. 2013. How to Write a Literature Review. Journal of criminal justice education.

Domingos, P. and Pazzani, M. 1997. On the optimality of the simple Bayesian classifier under zero-one loss. Machine learning, 29(2), 103-130.

Fauzan, M. A. and Murfi, H. 2018. The Accuracy of XGBoost for Insurance Claim Prediction. International Journal of Advances in Soft Computing and its Applications 10(2):159-171.

Fawcett, T. 2006. Introduction to ROC analysis. Pattern Recognition Letters 27(8):861-874.

Feurer, M and Hutter, F. 2019. Hyperparameter Optimization. Methods, Systems, Challenges, pages 3–38.

Freund, Y and Schapire, R. 1996. Experiments with a New Boosting Algorithm. ICML. Machine Learning: Proceedings of the Thirteenth International Conference.

Friedman, J. H. 2001. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics, Vol. 29, No. 5, pp. 1189-1232.

Günther, C.C., Tvete, I. F., Aas, K., Sandnes, G. I. & Borgan, Ø. 2014. Modelling and predicting customer churn from an insurance company, Scandinavian Actuarial Journal, 2014:1, 58-71.

Hamad, M. A. and Zeki, A. M. 2018. Accuracy vs. Cost in Decision Trees: A Survey. International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT).

Hanafi, M. and Ming, R. 2021. Machine Learning Approaches for Auto Insurance Big Data. Risks 9: 42.

Habib, A. H., Alalyani, M. and Hussain, I. 2015. Brief review on Sensitivity, Specificity and Predictivities. IOSR Journal of Dental and Medical Sciences. Volume 14, Issue 4 Ver. V, PP 64-68.

Ho, T. K. 1995. Random Decision Forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.

Jafarzadeh, H., Mahdianpari, M., Gill, E., Mohammadimanesh, F. and Homayouni, S. 2021. Bagging and Boosting Ensemble Classifiers for Classification of Multispectral, Hyperspectral and PolSAR Data: A Comparative Evaluation. Remote Sens, 13(21), 4405.

Jing, L., Zhao, W., Sharma, K. and Fend, R. 2017. Research on Probability-based Learning Application on Car Insurance Data. Advances in Engineering Research, vol. 150. 4th International Conference on Machinery, Materials and Computer.

Kafkova, S. (2015). Bonus-malus systems in vehicle insurance. Procedia Economics and Finance, 23, 216-222.

Kelley, F. 2017. Making Predictive Models Robust: Holdout vs Cross-Validation. Accessed 29 May 2022. Available: https://www.kdnuggets.com/2017/08/dataiku-predictive-model-holdout-cross-validation.html

Kurenkov, A. 2020. A Brief History of Neural Nets and Deep Learning. Skynet Today. Accessed 29 May 2022. Available: https://www.skynettoday.com/overviews/neural-net-history

Linden, A. and Yarnold, P. R. 2019. Some Machine Learning Algorithms Find Relationships Between Variables When None Exist -- CTA Doesn't. Optimal Data Analysis, LLC. Vol. 8, 64-67.

Loukas, S. 2020. Everything you need to know about Min-Max normalization: A Python tutorial. Towards Data Science. Accessed 29 May 2022. Available: https://towardsdatascience.com/everything-you-need-to-know-about-min-max-normalization-in-python-b79592732b79

Louzada, F., Ara, A., & Fernandes, G. B. (2016. Classification methods applied to credit scoring: Systematic review and overall comparison. Surveys in Operations Research and Management Science, 21(2), 117-134.

Luwes, N. J. 2010. Fibonacci numbers and the golden rule applied in neural networks. Interim: Interdisciplinary Journal. Vol. 9, No. 1.

Majka, M. 2020. Introduction to naivebayes package. Repository CRAN. Accessed 29 May 2022. Available: https://cran.r-project.org/web/packages/naivebayes/vignettes/intro_naivebayes.pdf

Mau, S., Pletikosa, I. and Wagner, J. 2018. Forecasting the next likely purchase events of insurance customers: A case study on the value of data-rich multichannel environments, International Journal of Bank Marketing, 36(6), pp. 1125–1144.

Mijwel, M. S. 2018. Artificial Neural Networks Advantages and Disadvantages. Computer science, college of science, University of Baghdad. Accessed 29 May 2022. Available: https://www.researchgate.net/profile/Maad-Mijwil/publication/323665827_Artificial_Neural_Networks_Advantages_and_Disadvantages/links/5aa2c01faca272d448b5a23d/Artificial-Neural-Networks-Advantages-and-Disadvantages.pdf

Minsky, M and Paper, S. A. 1969. Perceptrons: an introduction to computational geometry. MIT Press.

McHugh, M. L. 2012. Interrater reliability: The kappa statistic. Biochemia Medica. 22 (3): 276–282.

McCulloch, W. and Pitts, W. 1943. A Logical Calculus of Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics. 5 (4): 115–133.

Nian, K, Zhang, H, Tayal, A, Coleman, T and Li, Y. 2016. 'Auto insurance fraud detection using unsupervised spectral ranking for anomaly'. The Journal of Finance and Data Science, 2(1): 58–75.

Palacio, S. M. 2019. Abnormal Pattern Prediction: Detecting Fraudulent Insurance Property Claims with Semi-Supervised Machine-Learning. Data Science Journal, 18(1), p.35.

Park, N.-A. 2013. An Introduction to Logistic Regression: From Basic Concepts to Interpretation with Particular Attention to Nursing Domain. J Korean Acad Nurs Vol.43 No.2, 154-164.

Pesantez-Narvaez, J., Guillen, M. and Alcañiz, M. 2019. Predicting Motor Insurance Claims Using Telematics Data—XGBoost versus Logistic Regression. Risks, 7(2), 70.

Phua, C., Alahakoon, D. and Lee, V. 2004. 'Minority report in fraud detection: classification of skewed data'. Acm sigkdd explorations newsletter, 6(1): 50–59.

Pothabattula, S. K. 2019. The Math and Intuition behind Logistic Regression. Analytics Vidhya. Accessed 29 May 2022. Available: https://medium.com/analytics-vidhya/logistic-regression-mathematics-and-intuition-behind-logistic-regression-8ca864286bff

Probst, P., Wright, M. and Boulesteix, A.-L. 2019. Hyperparameters and Tuning Strategies for Random Forest. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 9(3).

Provost, F. J., Fawcett, T., and Kohavi, R. 1998. The case against accuracy estimation for comparing induction algorithms. In ICML Vol. 98, pp. 445-453.

Rosenblatt, F. 1957. The Perceptron, a Perceiving and Recognizing Automaton Project Para. Cornell Aeronautical Laboratory.

Schapire, R. E. 1999. A Brief Introduction to Boosting. Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence. IJCAI'99: Proceedings of the 16th international joint conference on Artificial intelligence - Volume 2. Pages 1401–1406.

Severino, M. K. & Peng, Y. 2021. Machine learning algorithms for fraud prediction in property insurance: Empirical evidence using real-world microdata. Accessed 29 May 2022. Available: https://www.researchgate.net/publication/352658980_Machine_learning_algorithms_for_fraud_prediction_in_property_insurance_Empirical_evidence_using_real-world_microdata

Smith, Kate A., Robert J. Willis, and Malcolm Brooks. 2000. An analysis of customer retention and insurance claim patterns using data mining: A case study. Journal of the Operational Research Society 51: 532–41.

Stefano, B and Gisella, F. 2001. Insurance fraud evaluation: a fuzzy expert system. In Fuzzy Systems. The 10th IEEE International Conference, vol. 3. IEEE, pp. 1491–1494.

Steiner, K. and Ruiz, A. 2019. Predictive analytics speeds innovation for life insurers. Life Predictive Analytics Survey Report (Americas). Willis Towers Watson. Accessed 29 May 2022. Available: https://www.willistowerswatson.com/-/media/WTW/Insights/2019/06/predictive-analytics-speeds-innovation-for-life-insurers.pdf?modified=20190621024449

Stoltzfus, J. C. 2011. Logistic regression: a brief primer. Acad Emerg Med. 18(10): 1099-104.

Stucki, O. 2019. Predicting the Customer Churn with Machine Learning Methods: Case: Private Insurance Customer Data. Master's dissertation, LUT University, Lappeenranta, Finland. Accessed 29 May 2022. Available: https://lutpub.lut.fi/bitstream/handle/10024/160081/Thesis_Oskar_stucki.pdf?sequence=1&isAllowed=y

Tu, J. V. 1996. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. Journal of Clinical Epidemiology. Volume 49, Issue 11, 1225-1231.

Tuomi-Nikula, T. 2018. Insurance Claim Risk Scoring with Machine Learning Algorithms: A Case Study on Developing a Predictive System for Assessing Corporate Customers' Claim Risk. Accessed 29 May 2022. Available: https://lutpub.lut.fi/bitstream/handle/10024/158310/MSF_Thesis_Tino-Tuomi-Nikula.pdf?sequence=1&isAllowed=y

Vasani, D. 2019. This thing called Weight Decay. Towards Data Science. Accessed 29 May 2022. Available: https://towardsdatascience.com/this-thing-called-weight-decay-a7cd4bcfccab

Viaene, S, Ayuso, M, Guillen, M, Van Gheel, D & Dedene, G. 2007. Strategies for detecting fraudulent claims in the automobile insurance industry. European Journal of Operational Research, 176(1): 565–583.

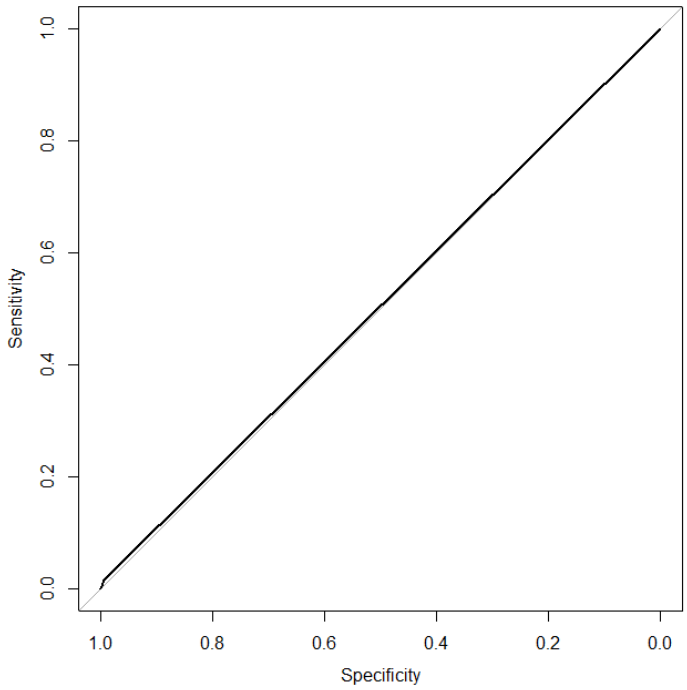
Wanyanga, P. 2021. Insurance Claim Prediction Using Machine Learning Ensemble Classifier. Accessed 29 May 2022. Available: https://medium.com/analytics-vidhya/insurance-claim-prediction-using-machine-learning-ensemble-classifier-14652907a65e

Webster, J., & Watson, R. T. 2002. Analyzing the past to prepare for the future: Writing a literature review. MIS Quarterly, 26(2), 13-23.

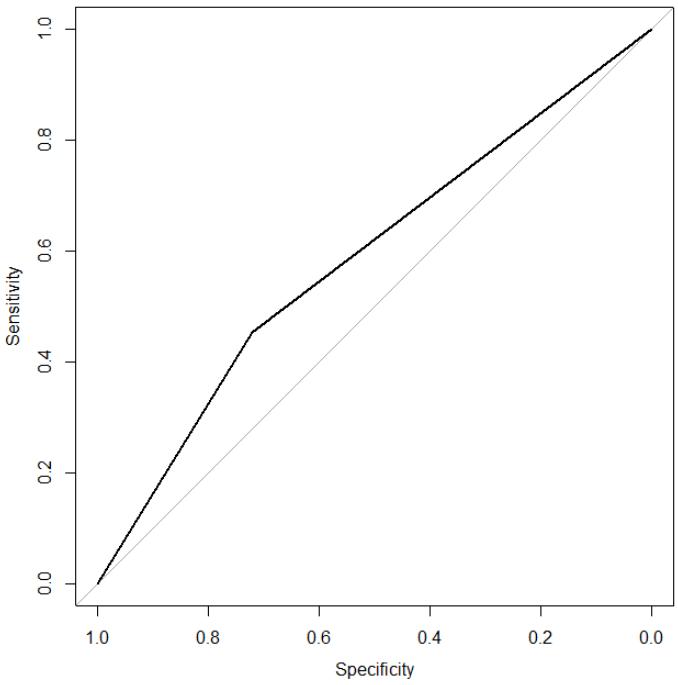
Wilson, JH. 2009. An analytical approach to detecting insurance fraud using logistic regression. Journal of Finance and Accountancy, 1: 1.

Yang, Y. 2021. Sparse Logistic Regression with the Hybrid L1/2+1 Regularization. 6th International Conference on Mathematics and Artificial Intelligence. Pages 8–13.

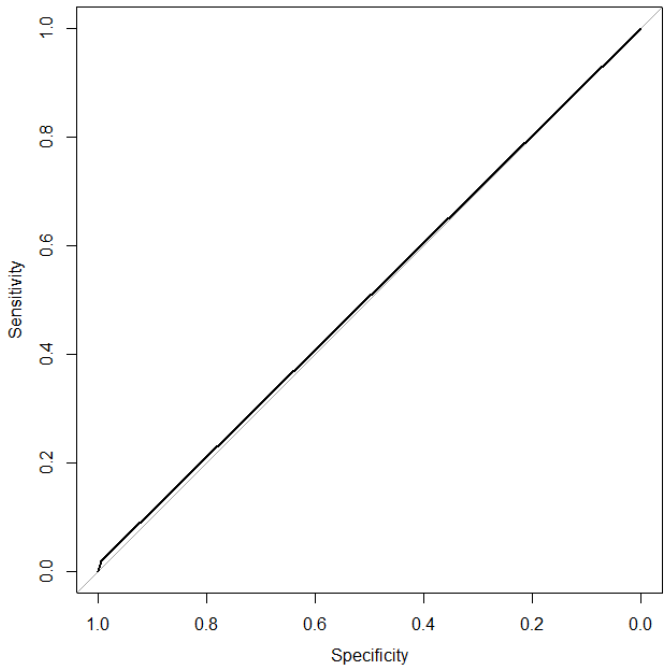
Appendix 1. LR ROC Curve



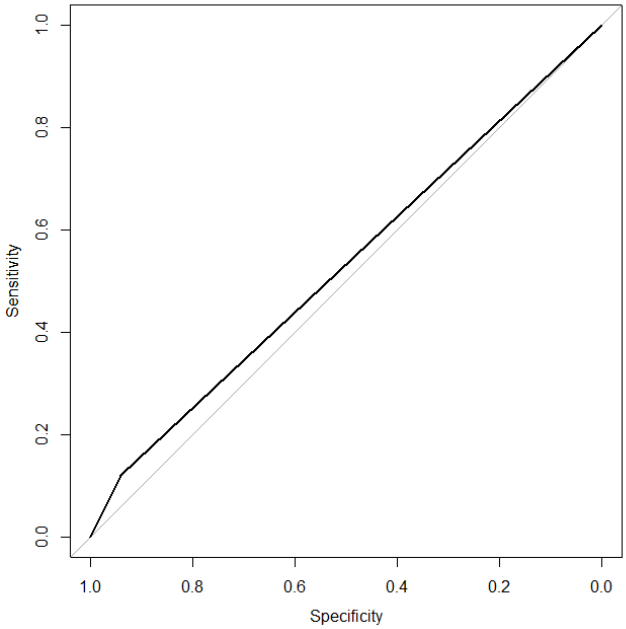
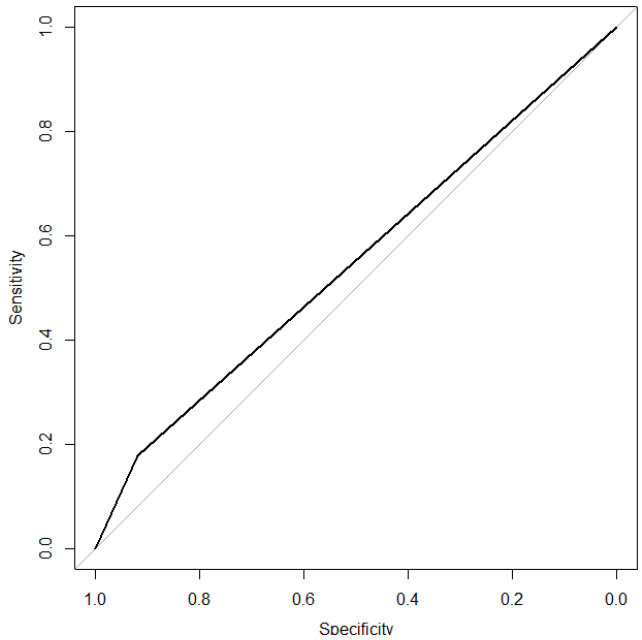

Appendix 2. NB ROC Curve



Appendix 3. NN ROC Curve




Appendix 4. RF ROC Curve



Appendix 5. XGBoost ROC Curve