Technische Universität Wien
Finanz- und Versicherungsmathematik

Seminararbeit

Modelling Surrender Risk in Life Insurance

Michael Holzknecht
Matrikelnummer: 11802563

unter Anleitung von
Associate Prof. Dipl.-Ing. Dr.techn. Stefan Gerhold

27. Februar 2022

Inhaltsverzeichnis

1 Einleitung

Vor allem in den letzten Jahren wird im Lebensversicherungsgeschäft immer mehr Wert darauf gelegt, Risiken möglichst frühzeitig zu erkennen und dann entsprechend abzusichern. Um dies umsetzen zu können, muss man diese Risiken nach Möglichkeit realistisch modellieren. Eines dieser Risiken für Lebensversicherungsunternehmen ist das sogenannte *Surrender Risiko*. Ziel dieser Arbeit ist es, einige Alternativen dieses Risikos zu modellieren, zu betrachten und untereinander zu vergleichen. Mittels numerischer Experimente werden dann die verschiedenen Modellierungstypen verglichen und beurteilt. Die gesamte Arbeit orientiert sich stark an [5], unter anderem werden die Ergebnisse aus den Experimenten übernommen.
Begonnen wird in den Kapiteln 2 und 3 mit einigen grundlegenden Definitionen sowie den statistischen Hintergründen, die zur Bewertung benötigt werden. In Kapitel 4 wird dann genauer auf diese Daten eingegangen, welche anschließend in den Experimenten in Kapitel 5 verwendet werden. Die Ergebnisse daraus bieten folglich einen guten Einblick in die sowohl in der Literatur als auch in der Praxis gängigen Methoden Surrender zu modellieren.

2 Grundlegendes

Bevor wir tiefer in die Modellierung des Surrenderrisikos eintauchen, werden in diesem Kapitel einige grundlegende Fragestellungen besprochen. Zuerst wird der Begriff *Surrender* (speziell in unserem Kontext) definiert und darauf eingegangen, warum wir dieses Risiko modellieren wollen bzw. sollten. Im zweiten Teil wird dann noch genauer auf die Zielsetzung dieser Arbeit eingegangen.

2.1 Definition Surrender

In der Literatur wird *Surrender* als Synonym von *Lapse* verwendet. *Lapse Risk* steht in der Versicherung generell, und insbesondere in der Lebensversicherung, für das Stornorisiko. Das Stornorisiko beinhaltet jedoch alle möglichen Optionen des Versicherungsnehmers, welche die zukünftigen Cash-Flows signifikant beeinflussen. In dieser Arbeit wollen wir das jedoch noch weiter einschränken. Wir sprechen von *Surrender*, wenn für das Stornieren des Versicherungsvertrages folgendes gilt: Der Vertrag wird frühzeitg, vollständig und von Versicherungsnehmerseite beendet.[5, 7]
Kommen wir nun zu der Frage, warum wir dieses Surrender Risiko überhaupt modellieren wollen. In folge des Inkrafttretens der Solvency II Regulierungen wurde das Modellieren von Risiken zu einer der wichtigsten Aufgaben von Versicherungsunternehmen. Da das Stornorisiko von der *European Insurance and Occupational Pensions Authority* (EIOPA) als eines der größten Risiken für Lebensversicherungsunternehmen identifiziert wurde, besteht natürlich gerade hier großes Interesse, dieses möglichst genau zu modellieren.[5]

2.2 Zielsetzung

Unsere allgemeine Umgebung ist ein Wahrscheinlichkeitsraum $(\Omega, \mathcal{F}, \mathbb{P})$. Darauf definieren wir einen Versicherungsvertrag zum Zeitpunkt $t \in \mathbb{N}_0$ als $X_t : \Omega \longrightarrow \mathbb{R}^n,\ n \in \mathbb{N}$. Daraus folgt dann, dass $(X_0, X_1, ..., X_T)$ die zeitliche Evolution des Vertrages bis zur Termination T beschreibt. Der jeweilige Zustand eines Vertrages zum Zeitpunkt t wird durch $J_t : \Omega \longrightarrow \mathbb{N}_0$ beschrieben, wobei $J_t = 0, t = 0, ...T - 1$ einen aktiven Vertrag beschreibt. Zur Vereinfachung gehen wir davon aus, dass ein Vertrag nur aus drei Gründen terminiert werden kann, nämlich Surrender, Tod oder Maturität. Klarerweise sind alle diese Zustände $\{J_T = 1\}, \{J_T = 2\}$ und $\{J_T = 3\}$ absorbierend.

Unser Ziel ist es, in einem Ein-Perioden-Setting die Wahrscheinlichkeit zu schätzen, dass in der Periode $[t, t+1)$ Surrender eintritt. Das heißt, wir wollen die bedingte Wahrscheinlichkeit $p(1|x_t) := \mathbb{P}[Y_t = 1|X_t = x_t]$ schätzen, wobei x_t eine Realisation von X_t ist und $Y_t := \mathbf{1}_{\{J_t=1\}}$ das Ziel ist (Surrender tritt ein).
Um diese Wahrscheinlichkeit nun möglichst genau zu bekommen, suchen wir einen Schätzer $\hat{p} : \mathbb{R}^n \longrightarrow [0,1], x \longmapsto \hat{p}(1|x)$, der die *Loss-Funktion* $l : [0,1] \times \{0,1\} \longrightarrow \mathbb{R}$ minimiert. Das bedeutet

$$\hat{p} := arg \min_{p'} l(p'(1|X_t), Y_t) \tag{1}$$

Aus notationstechnischen Gründen werden wir im Rest der Arbeit den Zeitindex t nicht mehr anschreiben. Ebenso werden wir für Realisationen einfach nur Kleinbuchstaben verwenden. Das heißt, dass die tatsächliche Wahrscheinlichkeit als $p(1|x)$ und die Schätzung als $\hat{p}(1|x)$ geschrieben wird. [5]

3 Statistischer Hintergrund

Bei der Modellierung des Surrender Risikos handelt es sich um ein sogenanntes binäres Klassifikationsproblem. Das heißt, die Objekte (hier: Versicherungsverträge) werden auf Grund von zuvor definierten Merkmalen in zwei Klassen eingeteilt (Surrender tritt auf oder nicht). Dieses Einteilen wird von einem sogenannten Klassifikator vorgenommen.
Im folgenden Kapitel werden wir uns zuerst ansehen, wie man solche Klassifikatoren, sowohl frequentistisch als auch probabilistisch, beurteilt. Danach kommen wir auf den Begriff des Rare Events zu sprechen. Die Rare-Event-Natur von Surrender kann zu einigen Problemen führen. Um dem entgegenzuwirken, werden wir uns einige Resampling Methoden ansehen.

3.1 Beurteilung von Klassifikatoren

Wie bereits erwähnt, werden wir uns mit zwei verschiedenen Arten der Beurteilung von Klassifikatoren befassen. Wir beginnen mit klassischen frequentistischen Methoden, bevor wir zu den, für unsere Klassifikation besseren, probabilistischen Beurteilungsmethoden kommen.
Um binäre Klassifikationen zu beurteilen, betrachtet man normalerweise die sogenannte *label prediction*. Die label prediction y kann dann entweder wahr $\hat{y} = y$ oder falsch $\hat{y} \neq y$ sein. Wenn man weiters unterscheidet, ob die Vorhersage positiv oder negativ ist, kommt man auf insgesamt vier verschiedene mögliche Ausgänge: richtig positiv, falsch positiv, richtig negativ und falsch negativ. Diese vier Werte kann man in einer sogenannten *Wahrheitsmatrix* (Abbildung 3) zusammenfassen.

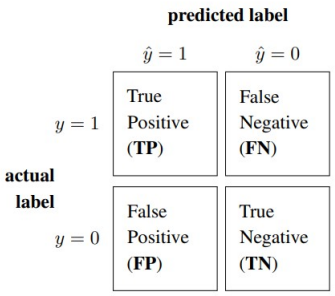


Abbildung 1: Wahrheitsmatrix [5]

Das Hauptproblem an der Wahrheitsmatrix ist für uns das generelle Problem der alleinigen Betrachtung und somit Beurteilung von label predictions für eine probabilistische Zielsetzung. Für Vorhersagen $p(1|x) \in [0,1]$ hängt die gesamte Matrix nur von einem c für die label prediction $\mathbf{1}_{p(1|x)>c}$ ab, was unsere Beurteilung sehr unempfindlich bezüglich c macht.
Weitere frequentistische Beurteilungsmethoden wären die ROC-Kurve beziehungsweise die Precision-Recall-Kurve. Beide bringen uns schon eine ganzheitlichere Betrachtung und somit Evaluation unseres Klassifikators, jedoch bleiben einige Probleme. Mit der ROC-Kurve betrachtet man nur die Verteilung der positiven Vorhersagen, das heißt, wie viele davon richtig und wie viele falsch positiv sind. Die Precision-Recall-Kurve ist im Allgemeinen sehr ähnlich zur ROC-Kurve, jedoch ist sie sehr empfindlich bezüglich unbalancierter Daten. Insgesamt kann man sagen, dass eine frequentistische Evaluation zwar einiges über unseren Klassifikator aussagen kann, aber es geht dennoch einiges an Information verloren. Aus diesem Grund wollen wir probabilistische Beurteilungsmethoden betrachten.
Idealerweise würden wir unsere Vorhersage $\hat{p}(1|x)$ natürlich mit der tatsächlichen Wahrscheinlichkeit $p(1|x)$ vergleichen. Um auf eine bestmögliche Modellierung zu kommen, könnte man beispielsweise den *mean absolute error* ($mae(p,\hat{p})$) minimieren. Dieser ist folgendermaßen definiert

$$mae(p,\hat{p}) := \frac{1}{N}\sum_i |p(1|x_i) - \hat{p}(1|x_i)|. \tag{2}$$

Klarerweise ist das eine rein theoretische Idee, da man in der Praxis die tatsächliche Wahrscheinlichkeit nicht kennt. Die Statistik bietet uns jedoch trotzdem eine gute Möglichkeit, unseren Klassifikator zu optimieren. Wir minimieren die *binäre Krossentrophie*

$$H(\hat{p}) := -\frac{1}{N}\sum_i y_i \log \hat{p}(1|x_i) + (1-y_i)\log\left(1-\hat{p}(1|x_i)\right), \tag{3}$$

um den *Maximum-Likelihood-Schätzer* $\hat{p}$ zu bekommen. Da die Krossentrophie in der Praxis jedoch leider nicht ideal ist, um Surrender zu modellieren, müssen wir noch einen Schritt weitergehen. Wir betrachten den Zentralen Grenzwertsatz von Lindeberg-Feller.

Seien $x_1, ...x_N$ Realisationen des Vertrages X, $Z_1, ..., Z_N$ zufällige, unabhängige Auftritte von Surrender, die $Ber(p(1|x_i))$ verteilt sind, dann gilt, dass wenn ein $\epsilon > 0$ mit $p(1|x) \in [\epsilon, 1-\epsilon]$ für einen Vertrag $x \sim X$ existiert, so folgt

$$\frac{1}{\sigma(S_N)} \sum_i Z_i - p(1|x_i) \longrightarrow \mathcal{N}(0,1) \tag{4}$$

in Verteilung, wobei

$$\sigma(S_N) := \sqrt{Var(\sum_i Z_i)} = \sqrt{\sum_i p(1|x_i)(1-p(1|x_i))} \tag{5}$$

Beweis. Mittels Lindeberg-Feller Bedingung, siehe z.B. [4] □

Die Voraussetzungen in der Proposition führen zu keinen Problemen bezüglich Surrender. Des Weiteren können wir aus dem zuvor gezeigten auch noch ein Konfidenzintervall für die durchschnittliche Surrenderrate bauen:

$$(\frac{1}{N} \sum_i \hat{p}(1|x_i)) \pm z_{1-\alpha/2} \frac{\sqrt{\sum_i \hat{p}(1|x_i)(1-\hat{p}(1|x_i))}}{\sqrt{N(N-1)}}, \tag{6}$$

wobei $\hat{p}(1|x_i)$ der modellierte Schätzer von $p(1|x_i)$ und $z_{1-\alpha/2}$ das jeweilige Perzentil der Standardnormalverteilung sind.
Das Konfidenzintervall werden wir dann vor allem bei den Experimenten verwenden.[5]

3.2 Rare Events

Bei einem binären Klassifikationsproblem sprechen wir von *ausgewogenen Daten* $(x_i, y_i)_{i=1}^N$, wenn die beiden Klassen ungefähr gleich viele Observationen beinhalten, d.h. $\sum_i y_i \approx \sum_i (1-y_i)$. Dementsprechend spricht man von unausgewogenen Daten, wenn eine der Klassen deutlich größer ist. Diese Unausgewogenheit kann mehrere Gründe haben, unter anderem lässt sie sich oft auf die Datensammlung zurückführen. Im Gegensatz dazu zeichnet sich ein *Rare Event* dadurch aus, dass die tatsächliche Wahrscheinlichkeit eines Auftretens sehr gering ist, $\mathbb{P}[Y = 1] < \epsilon$. Klarerweise geht man dabei von einem sehr kleinen Wert ϵ aus. Wenn wir nun Surrender betrachten, trifft man in der Praxis durchaus auf Werte um $\epsilon = 0.05$ oder sogar $\epsilon = 0.01$. In dieser Arbeit gehen wir von "perfekten" Daten aus. Das bedeutet, dass

wir jegliche Unausgewogenheit auf den Rare Event Charakter von Surrender zurückführen können. Trotz alledem führen unsere unausgewogenen Daten zu einigen Problemen. Es gibt jedoch einige Möglichkeiten gegen diese Probleme vorzugehen. Die Literatur verweist vor allem auf zwei Methoden: *Cost-Sensitive-Learning* und *Resampling*. Beide Techniken eignen sich nachweislich gut, um die Aussagekraft von Klassifikatoren auf unbalancierten Datensätzen zu verbessern. In unserem Fall werden wir uns jedoch auf das Resampling konzentrieren. [5]

3.3 Resampling

Wir werden uns nun einige der klassischen Resamplingmethoden ansehen und etwas genauer auf etwaige Vor- und Nachteile für unsere Experimente eingehen. Einfach betrachtet, handelt es sich beim Resampling um einen Versuch, die Unausgewogenheit von Daten auszugleichen. Generell gibt es dafür zwei Möglichkeiten, nämlich *random over-* und *undersampling*. Dabei werden entweder Daten aus der größeren Klasse gestrichen oder man *resampled* die kleinere Klasse (daher der Name). Rein mathematisch definieren wir over- bzw. undersampling folgendermaßen

Definition 1. Seien $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N} i.i.d.$ verteilte Proben aus Daten wie bei Surrender. Seien die Daten unausgeglichen mit Minoritätsklasse $Y = 1$, so dass $\sum_i y_i < \sum_i (1 - y_i)$. Wir nennen S einen *resampling scheme*, wenn es Daten $\mathcal{D}^S = \{(x_i^S, y_i^S)\}_{i=1}^{N^S} \subset \mathcal{D}$ mit $N^S > N$ bzw $N^S < N$ generiert und es gilt

$$\sum_{i=0}^{N}(1 - y_i) = \sum_{i=0}^{N^S}(1 - y_i^S) \; mit \; \{(x_i, y_i) \in \mathcal{D} \; : \; y_i = 0\} \subset \mathcal{D}, \tag{7}$$

bzw.

$$\sum_{i=0}^{N}(y_i) = \sum_{i=0}^{N^S}(y_i^S) \; mit \; \{(x_i, y_i) \in \mathcal{D} \; : \; y_i = 1\} \subset \mathcal{D}, \tag{8}$$

wobei (7) oversampling und (8) undersampling beschreibt.

Um zwischen empirischen und geresampleten Daten zu unterscheiden, schreiben wir die Wahrscheinlichkeiten als $\hat{p}(.|.)$ bzw. $\hat{p}^S(.|.)$. Nun wollen wir noch eine Eigenschaft für einen resampling scheme fordern.

Definition 2. Wir nennen einen resampling scheme *konsistent*, wenn die Wahrscheinlichkeiten durch das Resampling nicht verändert werden, d.h. für beliebiges $x \in X(\Omega)$ gilt für fixes $y \in \{0, 1\}$

$$\hat{p}^S(x|y) = \hat{p}(x|y) \tag{9}$$

Sowohl random over- als auch undersampling erfüllen diese Voraussetzungen und sind somit konsistent. Eine weitere konsistente Resampling Methode ist der SMOTE-Algorithmus[1], welcher dann auch in den hier beschriebenen Experimenten angewendet wird.
Für weitere Anwendungsgebiete und Eigenschaften von Resampling wird auf weiterführende Literatur verwiesen. [5]

4 Daten

In diesem Kapitel werden wir nun etwas genauer auf die Daten eingehen, die den beschriebenen Experimenten zugrunde liegen. Wir werden nicht versuchen, eine ideale Modellierung des Surrender Risikos für jede Art der Lebensversicherung zu finden, sondern uns etwas einschränken. Wir werden dann vier verschiedene Arten von Surrender Modellen aufstellen, die sich in den zugrunde liegenden Faktoren sowie auch in der generellen Komplexität unterscheiden werden. In unserer Modellierung sind diese Faktoren vor allem makro- bzw. mikroökonomischer sowie vertraglicher Natur. Die sogenannten *Surrender Profile* fassen wir dann in einem *Metamodell* zusammen. Bevor wir jedoch zu unseren Profilen kommen, betrachten wir kurz, wie wir unsere Versicherungsverträge simulieren werden.

4.1 Vertragssimulation

Wir betrachten eine klassische kapitalbildende Lebensversicherung. In unseren Simulationen betrachten wir ein Portfolio von $N = 30000$ Verträgen und gehen von einem Neugeschäft von sechs Prozent per anno aus.
Jeder Vertrag $x_i = (x_i^{(1)}, ..., x_i^{(n)})$ setzt sich aus folgenden Einträgen zusammen, welche auch in Abbildung 2 zu sehen sind:

- Alter der Versicherungsnehmer
- Nominalbetrag der Polizze
- Vertragsdauer
- bereits abgelaufene Zeit
- Restlaufzeit
- jährliche Prämie
- Prämienfrequenz
- Kalenderjahr

Dieses Portfolio sowie die realistische Verteilung von X beziehen wir aus [6], wo detaillierte Statistiken zu Lebensversicherungsverträgen aus einem US-amerikanischen Portfolio dargestellt werden.

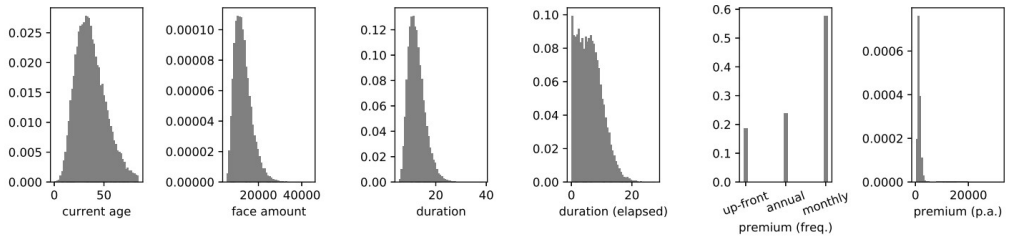


Abbildung 2: Randverteilung des Portfolios im Jahr 0 [5]

Auch wenn es sich dabei nicht um die exakt richtige Art von Verträgen handelt, gibt es uns dennoch eine gute Basis für unsere Experimente. Zur Berechnung der Prämien verwenden wir das Äquivalenzprinzip mit folgenden Annahmen:

- Zinsrate i ist konstant mit $i = 0.02$
- Kosten für das Versicherungsunternehmen werden dargestellt durch $(\alpha, \beta, \gamma) = (0.025, 0.03, 0.001)$
- Prämien werden bis zum Alter 67 gezahlt (deutsches Pensionsantrittsalter)
- für die Mortalität wenden wir das Sterbegesetz von Makeham an

$$ {}_tp_x := e^{-At - \frac{B}{\log c} c^x (c^t - 1)}, \tag{10} $$

  wobei wir die klassischen in der Literatur vertretenen Werte $A = 0.00022$, $B = 2.7 \cdot 10^-7$ und $c = 1.124$ verwenden.

Für etwaige andere Faktoren, die Einfluss auf die Vertragssimulationen haben könnten, gehen wir von konstanten Werten aus. Generell wird die Zeit in 15 Einjahresperioden unterteilt.[5]

4.2 Metamodell

Für gegebenes $x = (x^{(1)}, ..., x^{(n)})$ verwenden wir Logistische Regression, um auf die tatsächliche Surrender Wahrscheinlichkeit $p(1|x)$ zu kommen:

$$ p(1|x) := (1 + e^{-\beta_0 - \beta_x' x})^{-1}. \tag{11} $$

Die Regressionskoeffitienten $\beta_0 \in \mathbb{R}$ und $\beta'_x : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ hängen vom jeweiligen Surrender Profil ab. Im Unterschied zum Großteil der Literatur definieren wir β' als eine Funktion in x. Dies lässt uns den Effekt $\beta^i_x x^i$ der i-ten Komponente besser anpassen, was hilfreich bei der Aufstellung der Surrender Profile ist. Die spezifischen Werte von $\beta^i_x x^i$ werden unter anderem aus [3, 6, 8] übernommen. Der sogenannte *intercept* β_0 wird so angepasst, dass die Surrender Rate pro Kalenderjahr in eine realistische Region zwischen 1 und 5 Prozent fällt.[5]
Nun werden noch kurz die vier Surrender Profile betrachtet und etwas genauer beschrieben, worin sie sich unterscheiden.

Surrender Profile 1 und 2 Wie in [11] verwenden wir für die ersten beiden Surrender Profile sogenannte *odd-ratios*. Um diese zu berechnen, betrachten wir zwei Verträge x und x' die sich nur in einer Komponente (hier die i-te) unterscheiden. Dann sieht die odd-ratio folgendermaßen aus:

$$\frac{p(1|x)/(1-p(1|x))}{p(1|x')/(1-p(1|x'))} = e^{\beta^{(i)}_x x^{(i)} - \beta^{(i)}_{x'} x'^{(i)}} \tag{12}$$

Nun setzen wir $\beta^{(i)}_{x'} x'^{(i)} := 0$ als Baseline des Risikos in der i-ten Komponente und können daraus dann für alle j die Werte $\beta^{(j)}_x x^{(j)}$ berechnen. Es werden dann für Profil 1 empirische und für Profil 2 modellierte odd-ratios verwendet. Generell betrachten wir in diesen Profilen nur das Alter der Versicherungsnehmer, die bereits abgelaufene Zeit, die Prämienfrequenz und die jährliche Prämie.

Surrender Profil 3 und 4 Im Vergleich mit den Profilen 1 und 2 kommen noch die Vertragsdauer sowie die Restlaufzeit dazu. Außerdem wird die Prämienfrequenz sowie die Restlaufzeit etwas genauer betrachtet, nämlich auf einer "pro-Vertrag-Basis". Für das vierte Profil wird dann noch zusätzlich das Kalenderjahr als Risikotreiber hinzugefügt. Es werden dabei die Jahre 1991-2007 betrachtet, wobei man unter dem, aus dem Kalenderjahr resultierenden Risiko jenes versteht, das sich aus dem ökonomischen Umfeld ergibt.

4.3 Simulation der Surrender Aktivität

Generell betrachten wir jeden Vertrag x bis zu seiner Beendigung, also bei uns Maturität, Tod oder Surrender. Klarerweise ist Maturität am einfachsten zu

betrachten. Wenn über die gesamte Vertragsdauer weder Tod noch Surrender eintritt, muss Maturität auftreten. Für den Tod invertieren wir einfach ${}_tp_x$ als eine Funktion in t und kommen so zur Terminationszeit $T = ({}_tp_x)^{-}1(u), u \sim \mathcal{U}(0,1)$. Ähnlich gehen wir bei Surrender vor. Surrender tritt im folgenden Jahr auf, genau dann wenn $p(1|x) >= v$, $v \sim \mathcal{U}(0,1)$ für einen Vertrag x gilt. Sollten Surrender und entweder Tod oder Maturität im selben Jahr auftreten, gehen wir wieder von Gleichverteilung aus.[5]

4.4 Präparation der Daten

Bevor mit den Experimenten begonnen werden kann, müssen die vorhandenen Daten noch vorbereitet werden. Wir setzen die errechneten Prämien ein und skalieren unsere Verträge auf $[0,1]^n$. Nun spalten wir unsere Daten $\{(x_i, y_i)\}_{i=1}^N$ in zwei Sätze auf. Wir betrachten jenes Kalenderjahr, in welchem mindestens 70 Prozent der Daten betrachtet wurden. Je nachdem welches Surrender Profil man betrachtet, kommt man auf eine Teilung zum Zeitpunkt $t \in \{7,8\}$. Der erste Datensatz $\mathcal{D}_{train}$ bildet unsere Trainingsdaten und der zweite Datensatz $\mathcal{D}_{test}$ dann unsere Testdaten. Zusätzlich werden zufällige 30 Prozent der Trainingsdaten $\mathcal{D}_{train}$ für Validierungszwecke gesondert betrachtet.[5]

5 Experimente

Im folgenden Kapitel wird nun auf die Experimente sowie deren Ergebnisse von [5] eingegangen. Zu Beginn wird kurz das zugrunde liegende Basismodell beschrieben, bevor genauer auf die verschiedenen Modellierungstypen in den Experimenten eingegangen wird. Zum Abschluss werden dann noch die daraus resultierenden Ergebnisse genauer betrachtet und interpretiert.

5.1 Baseline Modell

Bevor wir verschiedene Modellierungstypen betrachten und auch bewerten können, wollen wir eine gewisse Basis haben, womit dann Vergleiche angestellt werden können. Dafür werden wir einen Klassifikator mit konstanter Surrender Rate verwenden. Aus der Einfachheit dieses *Baseline Modells* lässt sich dann sehr gut ableiten, ob die immer komplizierter werdenden Klassifikatoren notwendig sind bzw. überhaupt mehr Aussagekraft haben. Zu jedem Surrender Profil und den zugehörigen Trainingsdaten $\mathcal{D}_{train}$ definieren wir unser $\hat{p}$ folgendermaßen:

$$\hat{p}(1|x') := \frac{1}{|\mathcal{D}_{train}|} \sum_{(x,y)\in\mathcal{D}_{train}} y, \tag{13}$$

wobei x' einen beliebigen Vertrag darstellt. [5]

5.2 Modellierungstypen

Bei den numerischen Experimenten werden drei verschiedene Arten von Modellierungstypen verwendet. Dabei handelt es sich um *Logistische Regression*, *Tree Based Klassifikatoren* und *Neuronale Netze*. Diese werden jeweils in ihrer *bagged* und *boosted* Version angewendet.
Der Unterschied von *bagged* und *boosted* im Maschinellen Lernen ist einfach gesagt der folgende: Beim *bagging* werden Vorhersagen gleicher Art zusammengeführt, beim *boosting* Vorhersagen unterschiedlicher Art. Man kann im Allgemeinen davon ausgehen, dass *bagging* die Varianz verringert wobei *boosting* den Bias verringert.[10]

Logistische Regression Ähnlich wie schon beim Bilden des Metamodells, sieht die Struktur der Logistischen Regression folgendermaßen aus:

$$\hat{p}(1|x) := (1 + e^{-\beta_0 - \beta' x})^{-1}. \tag{14}$$

Es ist jedoch anzumerken, dass im Vergleich zum Meta Modell die Regressionskoeffizienten β_i keine Funktionen in x mehr sind, sondern Konstanten. Weiters sollte man darauf hinweisen, dass die Surrender Profile untereinander nicht interagieren. Da eine einzelne Logistische Regression eine viel zu hohe Volatilität für Rare Events hat, kombinieren wir insgesamt $N_{bag} = 10$ Schätzer, um unser Modell zu bilden. [5]

Tree Based Klassifikatoren Einen Tree Based Klassifikator kann man sich, wie schon der Namen sagt, als eine Art Baumdiagramm vorstellen. Man zerlegt den Ursprungsraum rekursiv immer in zwei disjunkte Unterregionen. Dies führt dann zu einer Menge disjunkter Regionen $\mathcal{R}_1, ..\mathcal{R}_m$. Die Surrender Wahrscheinlichkeit $\hat{p}(1|x)$ errechnet sich dann aus den relativen Frequenzen pro Region $\mathcal{R}_k$. Nun betrachten wir für einen beliebigen Punkt $x' \in \mathcal{R}_k$ die Teilmenge $\mathcal{D}_{\mathcal{R}_k} = \{(x, y) \in \mathcal{D}_{train} \mid x \in \mathcal{R}_k\}$ und setzen

$$\hat{p}(1|x') := \frac{1}{|\mathcal{D}_{\mathcal{R}_{k|}}} \sum_{(x,y)\in\mathcal{D}_{\mathcal{R}_k}} y. \tag{15}$$

Wie bereits angemerkt, betrachten wir sowohl bagged als auch boosted Varianten.Wir kombinieren $N_{bag} = 100$ Bäume zu einem *Random Forest*[9] (bagged Version) und $N_{boost} = 100$ Bäume zu einem *XGBoost*[2] (boosted Version). [5]

Neuronale Netze Analog zum vorigen Kapitel gibt es auch bei den Neuronalen Netzen sowohl eine bagged als auch eine boosed Version. Für den bagged Schätzer kombinieren wir $N_{bag} = 5$ Modelle $\hat{p}(1|.)$, die alle ident aufgebaut sind. Es werden sogenannte *feed-forward* Netze mit drei *Hidden Layers* verwendet. Der Schätzer setzt sich dann wie folgt zusammen:

$$\hat{p}(1|x) := \frac{1}{N_{bag}} \sum_k \hat{p}^{(k)}(1|x). \tag{16}$$

Neben dieser verwenden wir auch noch eine boosted Version. Es werden einschichtige Neuronale Netze $\hat{p}^{(k)}$, $k = 1, ..., N_{boost}$ mit $N_{boost} = 20$ verwendet. Der boosting Prozess beginnt mit der Basisrate von $\hat{p}^{(1)}(1|x) := \sigma^{-1}(\frac{1}{|\mathcal{D}_{train}|} \sum_i y_i)$ für Trainingsdaten $(x_i, y_i) \in \mathcal{D}_{train}$. Dazu werden dann interativ die Netze $\hat{p}^{(k)}$ hinzugefügt, bis wir zu unserem finalen geboosteten

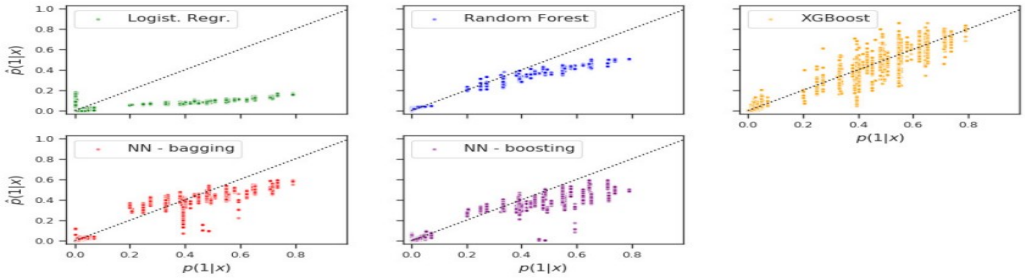
Neuronalen Netz kommen

$$\hat{p}(1|x) := \sigma(\sum_{k=1}^{N_{boost}} \hat{p}^{(k)}(1|x)). \tag{17}$$

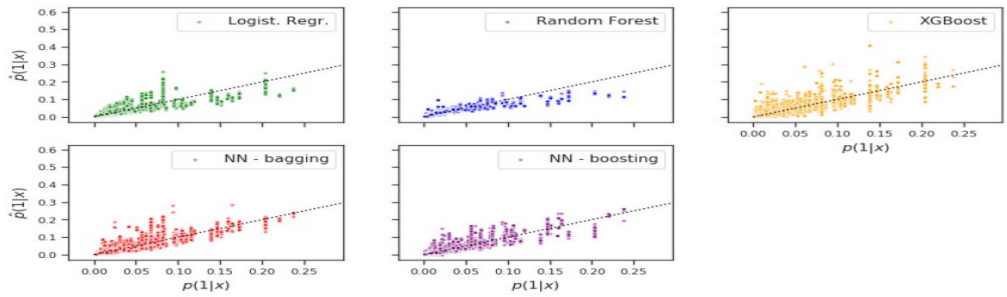
Bei σ handelt sich sich um eine sogenannte *sigmoid function*, die uns hier als Aktivierungsfunktion dient. In unserem Fall handelt es sich dabei um den Tangens hyperbolicus. [5]

5.3 Numerische Ergebnisse

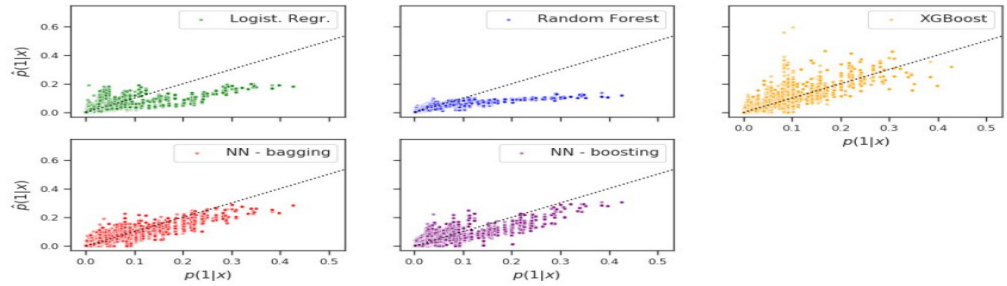
Nun kommen wir zu den Ergebnissen aus unseren numerischen Experimenten. Es werden also die Klassifikatoren auf alle Surrender Profile angewendet und beurteilt. Wir beginnen mit einem rein hypothetischen Ansatz, indem wir davon ausgehen, dass wir Zugriff auf die tatsächliche Surrender Wahrscheinlichkeit $p(1|x)$ haben. Auch wenn dies in der Praxis natürlich nicht möglich ist, gibt es uns einen guten Einblick in die erbrachte Leistung unserer Klassifikatoren. Wir werden die Modelle zuerst im Hinblick auf Varianz und Bias untersuchen, bevor wir uns dann die durchschnittliche Surrender Rate in einem praktischen Setting genauer ansehen. In Abbildung 3 sehen wir nun die Auswirkungen der verschiedenen Klassifikatoren auf die Surrender Profile, wobei bereits Testdaten $(x, y) \in \mathcal{D}_{test}$ verwendet wurden. Es werden dafür die modellierten Werte $\hat{p}(1|x)$ gegen die tatsächliche Wahrscheinlichkeit $p(1|x)$ geplottet. Es ist wichtig anzumerken, dass hier ausbalancierte Daten verwendet wurden, nachdem SMOTE Resampling angewendet wurde.
Man sieht sofort, dass für das Surrender Profil 1 alle Klassifikatoren, außer dem XGBoost, der annähernd keinen Bias hat, die tatsächliche Wahrscheinlichkeit recht deutlich unterschätzen. Dafür hat der XGBoost eine ziemlich hohe Varianz. Diese bleibt auch beim Surrender Profil 2 bestehen. Die Neuronalen Netze haben sowohl in der bagged als auch in der boosted Version eine deutlich geringere Varianz, aber im Vergleich zum ersten Profil auch keinen deutlichen Bias in irgendeine Richtung. Wenn wir uns das Surrender Profil 3 ansehen, fällt auf, dass wir sehr ähnliche Ergebnisse zum ersten Profil haben. Die Logistische Regression bietet wieder keinen guten Schätzer (weder bzgl. Varianz, noch bzgl. Bias). Der Random Forest hat zwar eine sehr kleine Varianz, unterschätzt die Wahrscheinlichkeit jedoch weiter. Im Surrender Profil 4 ist unter anderem interessant zu sehen, dass die Neuronalen Netze verschiedene Probleme haben.



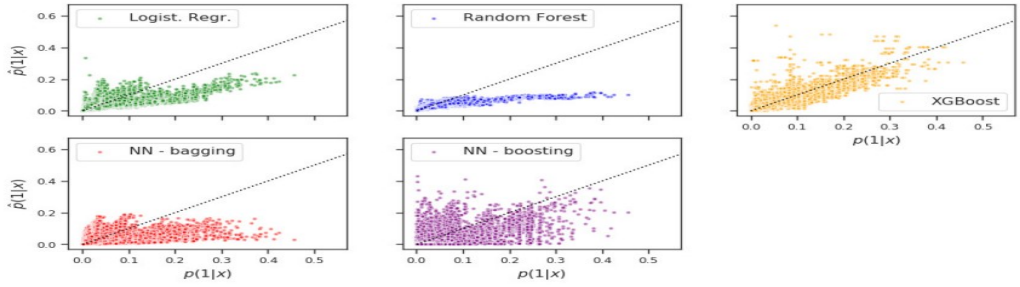
(a) Profil 1



(b) Profil 2



(c) Profil 3

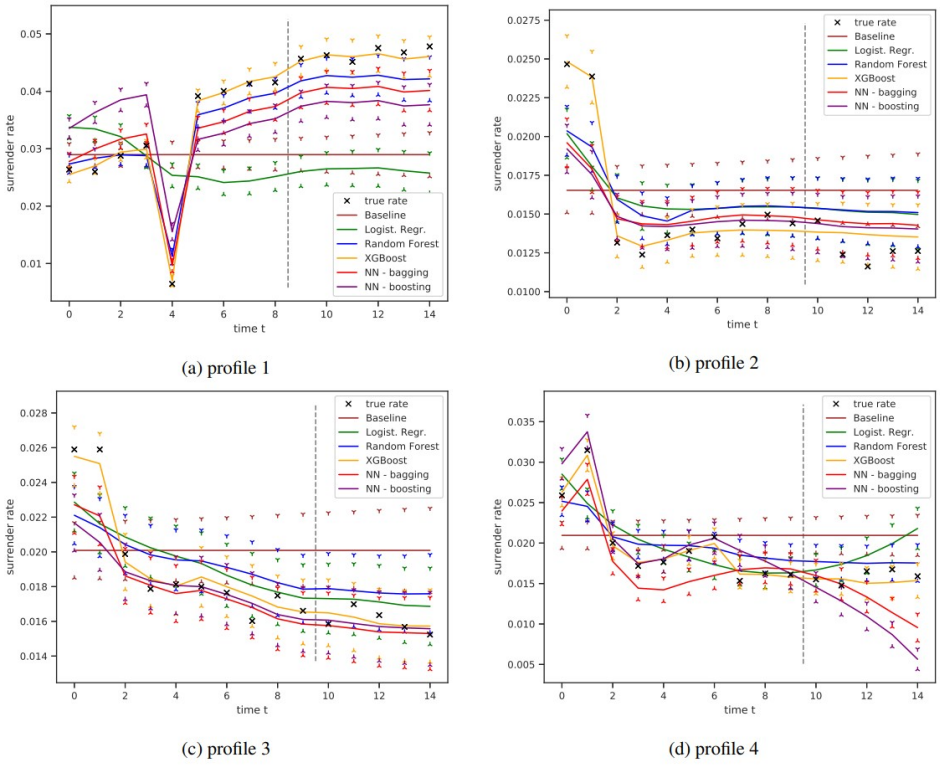


(d) Profil 4

Abbildung 3: Ergebnisse nach Anwendung der Klassifikatoren auf die Surrenderprofile[5]

Die bagged Version hat einen recht deutlichen Bias dazu, die tatsächliche Wahrscheinlichkeit zu unterschätzen, wohingegen die boosted Version eine hohe Varianz aufweist. Der XGBoost schätzt das $p(1|x)$ wieder relativ gut mit geringem Bias und trotzdem ohne hohe Varianz.
Wenn man sich die Ergebnisse dann noch etwas genauer ansieht, kommt man schnell zu dem Schluss, dass der XGBoost wohl die beste Möglichkeit der Modellierung ist.

Zum Schluss wollen wir nun noch die durchschnittlichen Surrender Raten in einem realistischen Setting betrachten. Gemeinsam mit einem 95% Konfidenzband plotten wir die Raten über die Kalenderjahre in Abbildung 4.



(a) profile 1 (b) profile 2

(c) profile 3 (d) profile 4

Abbildung 4: Durchschnittliche Surrender Raten[5] - vertikale Linie zeigt den Split in Trainigs- und Testdaten

Man kann sofort erkennen, dass die durchschnittlichen Surrender Raten nicht stationär sind. Das führt klarerweise augenblicklich zu der Erkenntnis, dass

die naive Baseline keinen guten Klassifikator liefert. Wenn man die ersten drei Profile betrachtet, fällt auf, dass der XGBoost sehr nahe an der tatsächlichen Rate dran ist. Für das Surrender Profil 4 fällt die Exaktheit dann leicht ab, wobei der XGBoost noch immer der einzige Klassifikator ist, für den die wahre Rate immer innerhalb des Konfidenzbandes liegt. Auch die Neuronalen Netze schneiden in den Profilen 1-3 sehr gut ab, unterschätzen die Surrender Rate im vierten Profil gegen Ende hin jedoch sehr deutlich. Aus dem letzen Surrender Profil lässt sich generell sehr viel ablesen. Man kann unter anderem auch sehen, dass Tree Based Klassifikatoren sehr stabil sind, vor allem nach dem Split von Trainings- zu Testdaten.

6 Conclusio

Nachdem wir die Ergebnisse der Experimente anhand der vier verschiedenen Surrender Profile betrachtet haben, lassen sich einige interessante Schlüsse ziehen, die vor allem für weitere und genauere Forschung in dem Bereich zum Tragen kommen. Da wir mit Logistischer Regression, Tree Based Klassifikatoren und Neuronalen Netzen jeweils in der bagged und boosted Variante die klassischen Möglichkeiten zur Modellierung von Surrender abgedeckt haben, haben wir nun einen ganzheitlichen Überblick bzgl. Vor- und Nachteile dieser Methoden. Es lässt sich wie schon in Kapitel 5 besprochen sagen, dass der XGBoost, auf allen Surrender Profilen betrachtet das überlegene Modell ist, wobei wir uns bei der Bewertung vor allem auf Bias und Varianz der Ergebnisse beziehen. Man kann außerdem sehen, dass Resampling tatsächlich bei der Beurteilung von Klassifikatoren hilft, jedoch in der Praxis aufgrund eines stark erhöhten Bias nicht sinnvoll ist. Bei der besonders für die Praxis interessanten Betrachtung der durchschnittlichen Surrender Raten ist es wichtig anzumerken, dass aus den Experimenten ganz klar hervorgeht, dass es nicht reicht, eine naive Baseline mit einem Risikopuffer zu versehen.

Literatur

[1] N. V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (2002), S. 321–357.

[2] T. Chen und C. Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), S. 785–794.

[3] M. Eling und D. Kiesenbauer. "What Policy Features Determine Life Insurance Lapse? An Analysis of the German Market". In: *The Journal of Risk and Insurance* 81.2 (2014), S. 241–269.

[4] Karl Grill. *Maß- und Wahrscheinlichkeitstheorie*. Institut fur Stochastik und Wirtschaftsmathematik TU Wien. Jan. 2022.

[5] Mark Kiermayer. *MODELLING SURRENDER RISK IN LIFE INSURANCE: THEORETICAL AND EXPERIMENTAL INSIGHT*. Jan. 2021.

[6] X. Milhaud und C. Dutang. "Lapse tables for lapse risk management in insurance: a competing risk approach". In: *European Actuarial Journal* 8.1 (2018), S. 97–126.

[7] *QIS5 Technical Specifications*. European Comission. 2010.

[8] M. Edwards R. R. Cerchiara und A. Gambini, Hrsg. *Generalized Linear Models in Life Insurance: Decrements and Risk Factor Analysis Under Solvency II*. 18th AFIR Colloquium. Rome, 2008.

[9] R. Tibshirani T. Hastie und J. H. Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. Second. New York: Springer, 2017.

[10] Pavan Vadapalli. *Bagging vs Boosting in Machine Learning: Difference Between Bagging and Boosting*. `https://www.upgrad.com/blog/bagging-vs-boosting/`. Aufgerufen am 09/02/2022. Nov. 2020.

[11] S. Loisel X. Milhaud und V. Maume-Deschamps. "Surrender Triggers in Life Insurance: Classification and Risk Predictions". In: *Bulletin Français d'Actuariat* 22.11 (2011), S. 5–48.

Abbildungsverzeichnis