TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Über die Optimierung von Modellpunkten für Cashflowmodelle in der Lebensversicherung

Zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Finanz- und Versicherungsmathematik

eingereicht von

Lisa Maria Wanka, BSc
Matrikelnummer: 01325141

ausgeführt am Institut für Stochastik und Wirtschaftsmathematik der Fakultät für Mathematik und Geoinformation der Technischen Universität Wien

Betreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Friedrich Hubalek
Mitbetreuer: Dipl.-Ing. Werner Matula

(Unterschrift Verfasserin) (Unterschrift Betreuer)

Wien, am 24 Oktober 2019

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 24 Oktober 2019

Lisa Maria Wanka

Zusammenfassung

Ziel dieser Arbeit ist es, mithilfe von Clusterverfahren, insbesondere dem k-Means Algorithmus, einen Teil von Versicherungsverträgen aus einem Versicherungsbestand auszuwählen, der den gesamten Bestand gut repräsentiert und die gleichen Eigenschaften wie dieser aufweist. Anstatt für das gesamte Versicherungsportfolio die Cashflowberechnungen durchzuführen, soll dies nur mit den erhaltenen Repräsentanten geschehen. Es wird zuerst eine Einführung in die Thematik der Clusteranalyse gegeben und der k-Means Algorithmus wird genauer betrachtet. Zusätzlich werden einige Verfahren vorgestellt, die eine bessere Leistung diese Clusterverfahrens gewährleisten sollen. Im Zuge dessen soll auch eine einheitliche Notation eingeführt werden. In weiterer Folge wird der erstellte Algorithmus im Anwendungsbereich Lebensversicherung getestet. Hierbei werden zuerst sehr vereinfachte Beispiele herangezogen, um die Ergebnisse einfacher veranschaulichen und überprüfen zu können. Danach werden auch kompliziertere Datenbestände zum Testen der erstellten Algorithmen verwendet. Unter anderem wird dazu ein reales Versicherungsportfolio herangezogen.

Abstract

The aim of this master thesis is to select some insurance contracts from an insurance portfolio in order to represent the whole portfolio. To find a set of contracts, which has the same properties as the whole portfolio, clustering, especially the k-means algorithm, is used. Instead of calculating the cashflow values of the whole insurance portfolio, this should be only done with the representative set of insurance contracts. First an introduction to cluster analysis and the k-means algorithm is given. Furthermore some techniques to receive a better performance of the cluster method are discussed and a consistent notation is introduced. Moreover the created algorithm will be tested in the application field life insurance. At first only simplified examples will be given to be able to verify the results easily. Afterwards also more complex test portfolios are used to test the introduced algorithms. Also a real insurance portfolio will be used.

Danksagung

An dieser Stelle möchte ich mich bei all jenen bedanken, die mich während der Anfertigung dieser Diplomarbeit unterstützt haben. Insbesondere gilt mein Dank Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Friedrich Hubalek und Dipl.-Ing. Werner Matula, die meine Diplomarbeit betreut und begutachtet haben. Ich bedanke mich für die konstruktive Kritik und Verbesserungsvorschläge.

Ebenfalls möchte ich mich bei der VIG Vienna Insurance Group AG Wiener Versicherungs Gruppe bedanken, welche mir die in dieser Arbeit verwendeten Daten zur Verfügung gestellt hat. Des Weiteren danke ich Michal Kujovic, M.A. und Alexandru Tindeche, M.Sc. für ihre Unterstützung und Hilfsbereitschaft.

Außerdem bedanke ich mich bei meinen Eltern und meiner Familie, die mir durch ihre Unterstützung dieses Studium ermöglicht und mich immer motiviert haben.

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit beschäftigt sich mit der Verdichtung von Modellpunkten eines Versicherungsbestandes, um die Berechnung der Cashflows in der Lebensversicherung zu optimieren. Das Ziel ist es, einen Bestand, der aus vielen Versicherungsverträgen besteht, nur durch einen kleinen Teil dieser Policen zu beschreiben, um daraufhin nur mit diesem versicherungsmathematische Kalkulationen durchzuführen, deren Ergebnisse das Verhalten des gesamten Bestandes vertreten sollen. Dafür müssen diejenigen Versicherungsverträge gefunden werden, die den gesamten Datenbestand am Besten repräsentieren können. In dieser Arbeit wird mithilfe von Clusterverfahren versucht, solche Repräsentanten ausfindig zu machen.

Zuerst sollen jedoch die verwendeten Verfahren selbst, ohne den speziellen Fall der Anwendung in der Versicherung, im Allgemeinen erklärt und eine einheitliche Notation eingeführt werden.

1.1 Clusteranalyse

Für dieses Kapitel wird auf das Buch [XW09] verwiesen, welches einen sehr guten Überblick über Clustering gibt. In der Clusteranalyse wird versucht, einen Datenbestand in Gruppen, auch Cluster genannt, einzuteilen. Dies wird in vielen verschiedenen Bereichen, die mit großen Datensätzen zu tun haben, wie beispielsweise in der Marktforschung oder in der Biologie verwendet. Da es keine strikte Definition von einem Cluster gibt, haben sich viele verschiedene Verfahren entwickelt, um Gruppierungen von Datenpunkten zu finden, die aufgrund ihrer unterschiedlichen Annahmen meistens nicht die gleichen Ergebnisse liefern. Die einzelnen Algorithmen können vorwiegend in hierarchische und partitionierende Verfahren eingeteilt werden. Hierarchische Verfahren starten entweder mit dem gesamten Datenbestand als einer großen Gruppe und teilen sie immer weiter in kleinere auf (das divisive oder Top-Down-Verfahren) oder starten mit jedem Modellpunkt als einelementiges Cluster und fassen sie in jedem Schritt zu einem größeren Cluster zusammen (das agglomerative oder Bottom-Up-Verfahren). Nach welchem Kriterium zusammengefasst wird ist unterschiedlich. Ein Beispiel dazu ist das Single-Linkage-Verfahren, bei dem diejenigen zwei Gruppen zusammengelegt werden, die die Objekte mit dem geringsten Abstand beziehungsweise mit der größten Ähnlichkeit enthalten. Dazu sei beispielsweise auf Dendrogramme, als eine graphische Darstellung solcher Algorithmen verwiesen.

Partitionierende Verfahren brauchen als zusätzliche Information die gewünschte Anzahl der Gruppen oder eine vorgegebene Start-Gruppierung. Sie weisen Objekte anderen Clustern, als jenen, in denen sie sich aktuell befinden, zu, bis sich ein Optimum ergibt. Wann ein Optimum erreicht wird, ist je nach Methode unterschiedlich und oft wird der Algorithmus vorzeitig abgebrochen, wenn er eine Bedingung erfüllt, die sicherstellen soll, dass die Clusterzuordnung eine akzeptable Güte erreicht hat. Dazu wird ein Schwellenwert definiert, der über- oder unterschritten werden muss, damit das Ergebnis als gut genug betrachtet wird. Oft wird hierbei ein Varianzkriterium verwendet und ein Maximalwert angegeben, der unterschritten werden soll. Grund dafür ist, dass es sehr lange dauern kann, bis tatsächlich die optimale Zuordnung der Modellpunkte zu den Gruppen gefunden oder in manchen Fällen das Optimum überhaupt nicht erreicht wird, wodurch der Algorithmus unendlich lange weiter laufen würde.

[Mir96] liefert eine gute Einführung in das Thema der Clusteranalyse und geht etwas genauer auf die Unterschiede zwischen hierarchischen und partitionierenden Verfahren ein. Weiters erhält man in [XW09] einen guten Überblick über verschiedene Clusteralgorithmen und ihre Funktionsweise. Es kann aus diversen Gründen erwünscht sein, Anhäufungen von Modellpunkten zu erkennen. Einerseits kann dadurch die Struktur eines Datenbestandes analysiert werden, es können die einzelnen Punkte in Klassen eingeteilt werden und man kann solche Verfahren auch zur Verdichtung eines Datenbestandes verwenden, indem beispielsweise die Zentren der Cluster als Repräsentanten gewählt werden. Letzteres kommt später in dieser Arbeit zur Anwendung.

Zusätzlich muss bei der Bildung der Cluster entschieden werden, ob Ausreißer auch zu den Gruppen zugeordnet werden sollen oder nicht. Des Weiteren kann erlaubt oder auch verboten werden, dass ein Modellpunkt zu mehr als nur einem Cluster gehört, also dass sich die Gruppen überlappen dürfen. Im Fuzzy-Clustering andererseits wird ein Datenpunkt jedem Cluster zu einem bestimmten Anteil zugeteilt.

Eine hohe Diversität zwischen zwei unterschiedlichen und eine hohe Homogenität innerhalb eines Clusters sind wünschenswert. Also sollen Objekte, die sich ähnlich sind, demselben Cluster zugeordnet werden und Modellpunkte, die sich stärker voneinander unterscheiden, sollten in verschiedenen Gruppen zu finden sein. Hierbei besteht nun die Schwierigkeit zu definieren, wann zwei Objekte sich ähnlich sind beziehungsweise ein Maß zu finden, das die Stärke der Ähnlichkeit von zwei Datenpunkten misst. Hierfür gibt es viele verschiedene Ansätze, die auch oft unterschiedliche Ergebnisse liefern, obwohl derselbe Datenbestand analysiert wird.

1.2 Distanz- und Ähnlichkeitsmaße

Ein Möglichkeit, um zwei Objekte miteinander zu vergleichen, ist die Zuhilfenahme von Metriken, auch Distanzmaße genannt. Demzufolge werden zwei Modellpunkte als ähnlich betrachtet, wenn sie eine geringe Distanz zueinander aufweisen, und je größer der Abstand zwischen zwei Punkten ist, desto größer ist auch der Unterschied zwischen den beiden.
Um die mathematische Definition von solchen Funktionen zu betrachten, soll jedoch zuerst der Aufbau eines Datenbestandes X erklärt werden. Die betrachteten Objekte beziehungsweise Personen, die analysiert werden sollen, werden als Vektoren x der Länge m dargestellt. Die Einträge x_i mit $i = 1 \dots m$ dieser Datenvektoren sind die Eigenschaften des jeweiligen Modellpunktes. Beispielsweise sind die hier gespeicherten Merkmale:

- Alter
- Geschlecht
- Körpergewicht
- Körpergröße

Es wird jedoch nicht nur ein Objekt oder eine Person betrachtet, sondern es werden die Daten von einer Vielzahl gesammelt und in den Datenbestand X eingetragen, sodass jede Zeile eine Beobachtungseinheit darstellt. Demzufolge ist X eine $n \times m$ - Matrix, wobei n die Anzahl der Modellpunkte bezeichnet. Das bedeutet, ein Datenbestand besteht aus n Objekten von denen jeweils m Merkmale erhoben wurden. In Tabelle 1 ist ein Beispiel

für einen Datenbestand mit $n = 5$ und $m = 4$ gegeben. Am linken Rand der Tabelle sind die beobachteten Personen und über jeder Spalte die Namen der erfassten Merkmale angegeben. Zu den Kalkulationen werden diese Informationen jedoch nicht verwendet.

	Alter	**Geschlecht**	**Körpergewicht in kg**	**Körpergröße in cm**
Person 1	34	m	86	182
Person 2	18	m	77	184
Person 3	57	w	68	170
Person 4	46	m	90	179
Person 5	24	w	54	162

Tabelle 1: Beispiel für einen Datenbestand.

Da nun der theoretische Aufbau eines Datenbestandes besprochen wurde, kehren wir zurück zu den Distanzmaßen. Siehe für Definitionen und für einige Beispiele von Distanzen [Heu84] und [Doo94]. Eine Metrik ist eine Abbildung

$$d: \ X \times X \to \mathbb{R}_+,$$

die folgende Eigenschaften $\forall\ x, y, z \in X$ erfüllt:

$$\begin{aligned} &d(x,y) \geq 0, \\ &d(x,y) = d(y,x), \\ &d(x,y) = 0 \ \Leftrightarrow \ x = y, \\ &d(x,y) \leq d(x,z) + d(z,x). \end{aligned} \tag{1}$$

Da in einem Datenbestand X nicht immer die gesamte Information über ein Objekt x gespeichert werden kann, ist die dritte Bedingung nicht immer erfüllt. Es können die erhobenen Daten zwar identisch sein, jedoch kann es sich trotzdem um beispielsweise zwei verschiedene Patienten handeln. Etwa wenn nur Alter, Geschlecht, Körpergewicht und Körpergröße notiert werden und nicht der Name der Person und eben genau diese Merkmale bei beiden Patienten dieselben sind. Also kann es passieren, dass mehrere Modellpunkte die gleichen relevanten Eigenschaften x_i besitzen und deshalb scheinbar identische Punkte in den verwendeten Daten zu finden sind, obwohl sie eigentlich verschiedene Beobachtungsobjekte repräsentieren. Es gilt jedoch trotzdem die Implikation

$$x = y \ \Rightarrow \ d(x,y) = 0. \tag{2}$$

Solche Funktionen werden als Pseudometriken bezeichnet. Siehe zu diesem Thema [Doo94]. Auch die Dreiecksungleichung (die letzte angeführte Eigenschaft in (1)) muss nicht immer für alle Datensätze erfüllt sein. Somit gelten in der Datenanalyse für eine Distanzfunktion nur die ersten zwei Eigenschaften mit Sicherheit und statt der dritten ist die Implikation (2) erfüllt.

Nun betrachten wir Beispiele für solche Funktionen, die unter anderem in der Clusteranalyse verwendet werden. Hierbei ist die Menge $X = \mathbb{R}^m$. Die wahrscheinlich bekanntesten Metriken sind die Euklidische Distanz:

$$d(x,y) = \|x - y\|_2 = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2};$$

und die Manhattan Distanz:

$$d(x,y) \quad = \quad \sum_{i=1}^{m} |x_i - y_i|.$$

Die Verallgemeinerung, sowohl der Euklidischen ($p = 2$), als auch der Manhattan Distanz ($p = 1$), ist die Minkowski Distanz:

$$d(x,y) \quad = \quad \left(\sum_{i=1}^{m} |x_i - y_i|^p \right)^{1/p}, \ \forall\ p \in [1, +\infty).$$

Für zusätzliche Beispiele von Distanzmaßen siehe [PG11].

Des Weiteren kann auch ein Ähnlichkeitsmaß verwendet werden, um die Beziehung zwischen zwei Modellpunkten darzustellen. Auch hierfür liefern [XW09] und [Mir96] einen guten Überblick. Im Gegensatz zu den Metriken bedeutet hier ein kleiner Wert als Output, dass sich die Objekte nur wenig ähneln und je größer die erhaltenen Werte sind, desto kleiner ist der Unterschied zwischen den betrachteten Punkten. Mathematisch beschrieben ist eine Ähnlichkeitsfunktion eine Abbildung

$$s : X \times X \to \bar{\mathbb{R}}, \text{mit } \bar{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$$

für die der ausgegebene Wert die Stärke der Ähnlichkeit darstellt. Also wie bereits erklärt, erhält man von dieser Funktion große Ergebnisse für Objekte, die sich ähnlich sind und kleine, falls die Eingabeobjekte sich stärker unterscheiden. Für ein Ähnlichkeitsmaß s gelten folgende Eigenschaften $\forall\ x, y \in X$:

$$\begin{aligned} s(x,x) &\geq s(x,y), \\ s(x,y) &= s(y,x). \end{aligned}$$

Oft werden sie auch normiert, so dass $s(x,y) \in [0;1]$ und $s(x,x) = 1$ gilt. Ein Beispiel für ein Ähnlichkeitsmaß ist die Kosinus-Ähnlichkeit die folgendermaßen definiert ist:

$$s(x,y) = \frac{x \cdot y}{\|x\|_2 \cdot \|y\|_2} = \frac{\sum_{i=1}^{m} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{m} x_i^2} \cdot \sqrt{\sum_{i=1}^{m} y_i^2}}.$$

Es kann auch mithilfe einer gegebenen Metrik d ein normiertes Ähnlichkeitsmaß definiert werden:

$$s(x,y) := \frac{1}{1 + d(x,y)}.$$

2 Clusterverfahren

Wie zuvor schon besprochen wurde, gibt es verschiedene Methoden, um in einem gegebenen Datenbestand Cluster zu finden. Für diese Arbeit wird ein partitionierendes Clusterverfahren verwendet, das mitunter eines der bekanntesten ist: der k-Means Algorithmus. Dazu siehe unter anderem [Jai10].

2.1 Der k-Means Algorithmus

Der k-Means Algorithmus ist ein relativ einfacher, oft verwendeter Algorithmus, der von mehreren Forschern unabhängig voneinander entdeckt wurde. Der Begriff "k-Means" wurde 1967 von MacQueen [Mac67] eingeführt, aber das Prinzip wurde erstmalig 1956 von Steinhaus [Ste56] beschrieben. Am weitgehendsten wird unter k-Means jedoch die Methode von Lloyd [Llo82] beziehungsweise Forgy [For65], die unabhängig voneinander einen sich überwiegend deckenden Algorithmus entwickelt haben, verstanden.

Wie bereits erwähnt, wird bei partitionierenden Clusterverfahren versucht, ein Optimierungsproblem zu lösen. In dem Fall von k-Means soll der quadratische Fehler zwischen den Mittelwerten der Cluster und den jeweils zugehörigen Modellpunkten minimiert werden. Es ist notwendig, dass die Anzahl der Gruppen $k \in \mathbb{N}$ im Vorhinein gegeben ist. Die Annahme, dass vor der Clusteranalyse bereits bekannt ist, in wie viele Cluster der betrachtete Datenbestand eingeteilt werden soll, ist ein erheblicher Nachteil dieser Methode, da die Struktur der Daten vorher oft noch nicht ersichtlich ist. Dieses Thema wird in dieser Arbeit jedoch erst zu einem späten Zeitpunkt in Kapitel (4) genauer behandelt.

Sei nun $1 < k < n$ die Anzahl der Gruppen, in die der Datenbestand eingeteilt werden soll fix gewählt. Des Weiteren bezeichne man mit C_j, $j = 1 \dots k$ die Cluster und mit c_j, $j = 1 \dots k$ die dazu gehörenden Zentren. Für einen Datenbestand X, der aus n Modellpunkten $x \in X$ besteht, ist der quadratische Fehler zwischen dem Clusterzentrum c_j und den zum Cluster C_j zugeordneten Punkten wie folgt definiert:

$$f(C_j) \quad := \quad \sum_{x \in C_j} \|x - c_j\|_2^2.$$

Nun soll die folgende Funktion $F(C)$, der die Gruppeneinteilung $C = \{C_1, C_2, \dots, C_k\}$ übergeben wird, um die Summe der quadratischen Fehler der einzelnen Cluster zu berechnen, minimiert werden:

$$F(C) := \sum_{j=1}^{k} f(C_j) = \sum_{j=1}^{k} \left(\sum_{x \in C_j} \|x - c_j\|_2^2 \right). \tag{3}$$

Um eine Lösung für dieses gegebene Optimierungsproblem zu finden, wird nun der k-Means Algorithmus folgendermaßen schrittweise durchgeführt:

1. Wähle k zufällige, voneinander verschiedene Modellpunkte aus X als erste Clusterzentren $c_1, \dots, c_k$.

2. Weise jeden Datenpunkt x aus X dem Cluster C_j zu, zu dessen Zentrum c_j der euklidische Abstand am geringsten ist. Also sodass

$$j = \text{argmin}_{l=1\dots k} \left(\|x - c_l\|_2 \right)$$

gilt.

3. Berechne die neuen Clusterzentren $c_1, \ldots, c_k$, sodass sie die Mittelwerte der Modellpunkte, die dem jeweiligen Cluster zugewiesen wurden, darstellen.

$$c_j := \frac{1}{|C_j|} \sum_{x \in C_j} x, \ j = 1 \ldots k.^1 \tag{4}$$

4. Wiederhole die Schritte 2. und 3. so oft, bis sich an der Aufteilung der Cluster nichts mehr ändert.

Da es sehr lange dauern kann, bis es tatsächlich keine Änderungen mehr in der Zuweisung zu den Gruppen gibt und es sogar passieren kann, dass sich Punkte im Datenbestand befinden, die immer zwischen zwei Clustern hin und her springen und daher nie eine stationäre Lösung gefunden wird, sollte eine maximale Anzahl an Iterationen angegeben werden, bei der der Algorithmus abgebrochen wird, falls nicht zuvor schon ein Optimum erreicht und der Vorgang beendet wurde. Für den Fall, dass die maximale Anzahl an Iterationen erreicht wurde, werden die zuletzt kalkulierten Cluster und ihre Zentren als Lösung verwendet.

In der zumeist in der Statistik verwendeten Programmiersprache R, die auch für die Berechnungen in dieser Arbeit verwendet wurde, gibt es im R-Package *stats* eine Funktion namens *kmeans*, die den k-Means Algorithmus für ein gegebenes Dataframe und eine gegebene Anzahl an Clustern k durchführt. Hier beträgt der Defaultwert der maximalen Anzahl an Iterationen zehn, dies kann aber bei Bedarf geändert werden. Anstatt der Anzahl der Cluster k kann auch eine Matrix, deren Zeilen als Vektoren, die die Positionen von Startzentren angeben, aufgefasst werden, der Funktion übergeben werden. In diesem Fall werden diese Clusterzentren verwendet, um die erste Clusterzuordnung durchzuführen, und nicht eine zufällige Menge an Modellpunkten herangezogen.

2.2 Startzentren

Im k-Means Algorithmus werden zu Beginn die Zentren zufällig aus allen Modellpunkten in dem betrachteten Datenbestand ausgewählt. Diese per Zufall initialisierten Startzentren können jedoch unterschiedlich gut sein und in verschiedenen Ergebnissen resultieren. Selbst wenn der Algorithmus nicht schon frühzeitig unterbrochen wird, sondern wirklich bis zum Ende gewartet wird und eine sich nicht mehr ändernde Zuweisung in Gruppen gefunden wurde, ist dies in den meisten Fällen nur ein lokales Optimum, welches aufgrund der zufällig gewählten Startzentren nicht mehr reproduzierbar ist. Es bietet sich an, den Algorithmus mehrmals durchzuführen und dann das beste Ergebnis zu verwenden, also jenes, welches den kleinsten Wert für die Funktion (3) und somit den kleinsten quadratischen Fehler annimmt. Wenn jedoch für sehr große Datenmengen eine Aufteilung in Cluster gesucht werden soll, kann dieser Prozess zu sehr langen Rechenzeiten führen.

[1] Hier wird natürlich davon ausgegangen, dass keine leeren Cluster vorhanden sind und deshalb niemals durch 0 dividiert werden kann. Da als Startzentren Modellpunkte verwendet werden, die sich im Datenbestand befinden, ist auf jeden Fall das Zentrum selbst, also zumindest ein Element in jedem Cluster vorhanden.

Um reproduzierbare Startzentren und somit auch reproduzierbare Ergebnisse zu erhalten, schlägt [Del17] folgende Vorgehensweise zur Initialisierung der ersten Clusterzentren vor:

Man nehme an, es liegt eine nummerierte Liste von den zu analysierenden Modellpunkten vor. Es sollen nun k dieser Punkte gewählt werden, um als Startzentren c_i, $i = 1 \dots k$ zu fungieren. Gesucht ist eine Formel, die eine Auswahl an Indizes j_i festlegt, um daraufhin die Zentren wie folgt zu definieren:

$$c_i := x[j_i],\ i = 1 \dots k.$$

Hierfür ist unter $x[l]$, $l = 1 \dots n$ der Datenpunkt aus dem Datenbestand X zu verstehen, welchem die Nummer l zugewiesen wurde. Sei nun

$$t := \left\lfloor \frac{\text{Anzahl Modellpunkte}}{\text{Anzahl Cluster}} \right\rfloor = \left\lfloor \frac{n}{k} \right\rfloor.$$

Die Indizes j_i sollen folgendermaßen berechnet werden:

$$j_i := \left\lfloor \frac{(2 \cdot i - 1) \cdot t}{2} \right\rfloor,\ i = 1 \dots k.$$

Daher ergibt sich die folgende Auswahl an Indizes der als Startzentren zu wählenden Modellpunkte:

$$j_1 = \left\lfloor \frac{t}{2} \right\rfloor,\ j_2 = \left\lfloor \frac{3t}{2} \right\rfloor, \dots,\ j_k = \left\lfloor \frac{(2 \cdot k - 1) \cdot t}{2} \right\rfloor. \tag{5}$$

Als Nummerierung kann einfach die Stelle der Zeile, in der sich der Modellpunkt befindet, gewählt werden. Somit sind die Ergebnisse reproduzierbar, jedoch ist damit nicht sichergestellt, dass ein globales Minimum gefunden wird. Die Lösung, die durch eine solche Initialisierung der Startzentren erhalten wird, ist weder besser, noch schlechter, als wenn per Zufall die ersten Clusterzentren gewählt werden. Auch hier wird in der Regel nur ein lokales Optimum gefunden. In diesem Fall kann nicht mehr der Trick verwendet werden, dass der Algorithmus mehrmals durchgeführt und das beste Ergebnis herangezogen wird, da die Startzentren nun nicht mehr zufällig sind.

In [Ste99] wird das Thema Startzentren genauer betrachtet und versucht, eine Methode zu finden, um die ersten Clusterzentren gut zu wählen, sodass ein gutes Ergebnis erhalten wird. Dabei wird versucht zu verhindern, dass der Algorithmus in dem nächstgelegenen lokalen Minimum gefangen ist und dadurch eine schlechte Clusteraufteilung als Ergebnis liefert. [Ste99] schlägt einige Varianten vor, wie die Startzentren gewählt werden können, beziehungsweise entwickelte Algorithmen, die eine zufällige Wahl an Clusterzentren so bearbeiten, dass sie die Daten gut abdecken und eine besserte Ausgangsposition ergeben.

Dazu wird versucht, schlecht positionierte Zentren zu verschieben und an bessere Stellen zu setzen. Für diese Arbeit wird der Algorithmus *Decreasing and Jumping Prototypes* aus [Ste99], welcher eine Kombination aus zwei solcher Methoden darstellt, herangezogen. Hierbei wird der Prozess mit mehr Startzentren, als am Ende erwünscht, begonnen. Zuerst wird das *Jumping*-Verfahren angewandt und daraufhin das *Decreasing*-Verfahren, bei dem die Anzahl der Cluster so lange reduziert wird, bis die gewünscht Anzahl an Gruppen k erreicht ist.

2.2.1 Jumping-Verfahren[2]

Bei dem *Jumping*-Verfahren werden Clusterzentren, die schlecht liegen, entfernt und stattdessen ein neues Zentrum an einer besseren Lage positioniert. Nun muss überlegt werden, wann ein Clusterzentrum schlecht beziehungsweise gut platziert ist. Wenn sich in einem Cluster nur sehr wenige Modellpunkte befinden, bedeutet das dass ein zweites Zentrum zu nahe ist und dadurch nur wenige Punkte in dem Cluster liegen können. Dadurch wird das Cluster, welches am wenigsten Elemente besitzt, als schlecht platziert angesehen und aufgelöst.

Als nächsten Schritt soll nun ein neues Clusterzentrum positioniert werden. Dafür bietet sich ein Modellpunkt aus einem nicht so gut repräsentierten Bereich an, also ein Datenpunkt, der sich im Cluster mit der größten Anzahl an Elementen befindet. Dieser Punkt wird dann zufällig aus den zu dieser Gruppe zugeordneten Modellpunkten gewählt. Sobald das geschehen ist, wird wie im k-Means Algorithmus die Clusterzugehörigkeit der Modellpunkte aktualisiert und die neuen Clusterzentren laut Gleichung (4) berechnet. Dieser Prozess wird für eine zuvor definierte Anzahl an Iterationen durchgeführt.

Sei nun k die Anzahl an Cluster in die der Datenbestand X eingeteilt werden soll. Wähle des Weiteren die Anzahl der Iterationen it. Der Pseudocode für diesen Algorithmus lautet wie folgt:

$$
\begin{aligned}
centers = \ & \textbf{JUMP}(X,\ k,\ it)\{ \\
& \quad c = randomsample(X, k) \\
& \quad \textbf{for}(Anzahl\ der\ Iterationen\ \leq\ it)\{ \\
& \quad\quad \textbf{for}(x \in X)\{ \\
& \quad\quad\quad \textbf{for}(j = 1 \ldots k)\{ \\
& \quad\quad\quad\quad d_j = dist_{Eukl}(x, c_j) \\
& \quad\quad\quad \} \\
& \quad\quad\quad l = argmin_j(d_j) \\
& \quad\quad\quad x\ zu\ C_l\ zuordnen \\
& \quad\quad \} \\
& \quad\quad \textbf{for}(j = 1 \ldots k)\{ \\
& \quad\quad\quad c_j = mean(x \in C_j) \\
& \quad\quad \} \\
& \quad\quad j_{min} = argmin_j(|C_j|) \\
& \quad\quad j_{max} = argmax_j(|C_j|) \\
& \quad\quad c_{j_{min}} = randomsample(C_{j_{max}}, 1) \\
& \quad \} \\
& \quad \textbf{return}\ c \\
& \}
\end{aligned}
$$

Zu Beginn werden k zufällige Modellpunkte aus dem Datenbestand X gewählt und als erste Clusterzentren $c_j,\ j = 1 \ldots k$ verwendet. Daraufhin beginnt eine for-Schleife, die

[2] [Ste99, S.53 f]

it-mal läuft. In der nächsten for-Schleife wird für jeden Modellpunkt x aus X der Abstand zu den aktuellen Clusterzentren c_j gemessen und daraufhin wird der Punkt x dem Cluster zugeordnet, zu dessen Zentrum er den geringsten Abstand hat. Danach werden die neuen Clusterzentren c_j laut Gleichung (4) berechnet, indem der Mittelwert aus den Modellpunkten, die sich im entsprechenden Cluster C_j befinden, kalkuliert wird. Bis zu dieser Stelle ist die Vorgehensweise dieselbe, wie beim k-Means Algorithmus, bei dem nun wieder eine neue Zuordnung zu den Gruppen stattfinden würde, um danach wieder die Zentren zu aktualisieren. Bei dem *Jumping*-Verfahren gibt es aber noch einen Zwischenschritt, bevor die neue Clustereinteilung und Kalkulation der zugehörigen Zentren erfolgt. Es wird das Cluster mit den wenigsten und das mit den meisten Elementen gesucht. Das Clusterzentrum, das die geringste Anzahl an Modellpunkten besitzt, wird angepasst, indem es durch ein zufällig gewähltes Element aus dem Cluster, welches die meisten Punkte enthält, ersetzt wird. Sobald alle *it* Iterationen durchgeführt worden sind, endet der Algorithmus und die aktuellen Zentren werden zurückgegeben.

Da das *Jumping*-Verfahren nach einer fixen Anzahl an Iterationen stoppt, ist es möglich, dass die letzte Verschiebung eines Zentrums zu einer schlechteren Aufteilung der Clusterzentren als der vorherigen führt. Das wäre zum Beispiel der Fall, wenn das Cluster, das aufgelöst wurde, mehr als die Hälfte der Elemente enthalten hat als die größte Gruppe, aus der das neue Zentrum gewählt wurde.

2.2.2 Decreasing-Verfahren[3]

Im *Decreasing*-Verfahren startet man mit einer höheren Anzahl an Startzentren h als eigentlich gewünscht ist und entfernt Schritt für Schritt schlechte Cluster, bis nur mehr $k < h$ Gruppen übrig sind, wobei k die Anzahl an Cluster ist, in die der Datenbestand X geteilt werden soll. Wie auch beim *Jumping*-Verfahren wird das Clusterzentrum mit den wenigsten zugehörigen Modellpunkten als schlecht bewertet und somit entfernt. Daraufhin wird die Aufteilung der Cluster aktualisiert und die Berechnung der Zentren laut Gleichung (4) neu durchgeführt.

[3] [Ste99, S.52 f]

Der Programmiercode dazu sieht wie folgt aus:

```
centers = DECR(X, k, h){
            c = randomsample(X, h)
            while(h > k){
              for(x ∈ X){
                for(j = 1 ... h){
                  d_j = dist_Eukl(x, c_j)
                }
                l = argmin_j(d_j)
                x zu C_l zuordnen
              }
              for(j = 1 ... h){
                c_j = mean(x ∈ C_j)
              }
              j_min = argmin_j(|C_j|)
              remove(c_{j_min})
              h = h − 1
            }
            return c
          }
```

Als erstes werden h zufällige Modellpunkte aus X gewählt und als Clusterzentren verwendet. Es beginnt eine for-Schleife, die so lange läuft bis die aktuelle Anzahl an Gruppen h dem Wert k entspricht. Die Zuordnung der Datenpunkte zu den Clustern und die Berechnung der neuen Clusterzentren erfolgen wie im k-Means beziehungsweise wie im *Jumping*-Verfahren. Auch im *Decreasing*-Verfahren gibt es einen Zwischenschritt, bevor die Clusterzuordnung aktualisiert wird. Hier wird ebenfalls das Cluster mit der geringsten Menge an zugehörigen Elementen gesucht und das entsprechende Clusterzentrum wird in diesem Fall gelöscht, ohne ein neues Zentrum an eine andere Stelle zu setzen. Damit wird die Anzahl der Cluster um eins reduziert und $h = h - 1$. Sobald nur mehr k Gruppen übrig sind wird der Algorithmus beendet und die aktuellen Zentren werden als Ergebnis des Programmes erhalten.

2.2.3 Zusammensetzung der Verfahren[4]

Um nun das *Jumping*- und das *Decreasing*-Verfahren zu kombinieren, wähle man $h > k$. Als erstes wird der *Jumping*-Algorithmus für eine vorher festgelegte Anzahl an Iterationen *it* durchgeführt. Dazu werden h Modellpunkte zufällig aus dem Datenbestand als Startzentren ausgewählt. Daraufhin wird dann die Anzahl der Cluster mithilfe des *Decreasing*-Verfahrens immer weiter minimiert, bis nur mehr die gewünschte Menge k an Gruppen übrig bleibt.

[4] [Ste99, S.54 f]

Zur Veranschaulichung ist auch hierfür ein Pseudocode erstellt worden:

```
centers = JUMPDECR(X, k, h, it){
            c = randomsample(X, h)
            while(h > k){
              for(x ∈ X){
                for(j = 1 ... k){
                  d_j = dist_Eukl(x, c_j)
                }
                l = argmin_j(d_j)
                x zu C_l zuordnen
              }
              for(j = 1 ... k){
                c_j = mean(x ∈ C_j)
              }
              if(Anzahl der Iterationen ≤ it){
                j_min = argmin_j(|C_j|)
                j_max = argmax_j(|C_j|)
                c_{j_min} = randomsample(C_{j_max}, 1)
              } else {
                j_min = argmin_j(|C_j|)
                remove(c_{j_min})
                h = h − 1
              }
            }
            return c
          }
```

Wie bereits erwähnt wird zu Beginn, wie auch bei den anderen beiden Algorithmen, eine zufällige Auswahl an Modellpunkten aus dem Datenbestand X herangezogen, um als Startzentren zu fungieren. In diesem Fall werden wie auch beim *Decreasing*-Verfahren mehr Zentren verwendet, als am Ende erwünscht sind. Wieder wird zuerst die Clusterzuordnung durchgeführt, um danach die Zentren neu zu berechnen. Als nächstes kommt nun eine if-Abfrage, die für die ersten it Iterationen den Zwischenschritt aus dem *Jumping*-Algorithmus durchführt und schlecht platzierte Clusterzentren in bessere Bereiche verschiebt. Danach wird im else-Abschnitt das *Decreasing*-Verfahren mit h Gruppen startend durchlaufen, solange bis nur mehr k Cluster vorhanden sind. Sobald das geschehen ist, wird der Algorithmus beendet und die aktuellen Clusterzentren werden als Rückgabe ausgegeben. [Ste99] empfiehlt für $it = h - k$ zu wählen, sodass beide Verfahren gleich oft durchlaufen werden.

Eigentlich sollten nach der letzten Entfernung eines Zentrums die Modellpunkte ein weiteres Mal in Cluster eingeteilt werden. Da dieses Verfahren jedoch nur dazu dient, gute Startzentren zu finden, die daraufhin dem k-Means Algorithmus übergeben werden,

muss dieser Schritt nicht durchgeführt werden. Im k-Means Algorithmus muss ohnedies erneut eine Zuteilung stattfinden.

Das Problem, dass die letzte Verschiebung eines Zentrums ein schlechteres Ergebnis liefern könnte, welches bei dem *Jumping*-Verfahren aufgetreten ist, ist nun durch die nachträgliche Entfernung von schlechten Clusterzentren durch den *Decreasing*-Algorithmus behoben. Falls nämlich durch das Versetzen des Zentrums ein kleineres Cluster entsteht als zuvor, wird dieses als erstes entfernt.

Die Clusterzentren, die als Ergebnis von dem *Jumping and Decreasing*-Algorithmus erhalten werden, liefern gut verteilte Startzentren für den gegebenen Datenbestand, die nun als Input für den k-Means-Algorithmus verwendet werden können und auf deren Basis die Optimierung durchgeführt werden kann. Jedoch sind die Ergebnisse trotzdem nicht reproduzierbar, da aufgrund der unsystematischen Auswahl der ersten Zentren, immer noch eine zufällige Komponente Einfluss auf die Resultate hat. Um die Ergebnisse jedoch reproduzierbar zu machen, wähle die Datenpunkte laut der Nummerierung (5). Dafür muss die erste Zeile des Programmes JUMPDECR durch folgende Programmzeilen ersetzt werden:

$$
\begin{aligned}
&t = \left\lfloor \frac{nrow(X)}{h} \right\rfloor \cdot \tfrac{1}{2} \\
&t^* = \lfloor (1, 3, \ldots, (h \cdot 2 - 1)) \cdot t \rfloor \\
&c = \{x[i],\ i \in t^*\}
\end{aligned}
$$

Die Funktion $nrow(X)$ gibt die Anzahl der Zeilen einer Matrix, die in unserem Fall der Datenbestand X ist, der in den Zeilen die Modellpunktvektoren gespeichert hat, zurück. t^* ist eine Folge, die die Nummerierungen der zu wählenden Datenpunkte gespeichert hat, und die Menge c enthält eben diese Modellpunkte, die nun als Startzentren verwendet werden sollen.

Somit wurde ein Verfahren erstellt, das eine Vorarbeit zu dem k-Means Algorithmus leistet, um gute und reproduzierbare Ergebnisse zu erhalten. Dadurch ist das Risiko, dass die Optimierung ein lokales Minimum mit einem vergleichsweise hohen quadratischen Fehler annimmt, verringert.

3 Datenaufbereitung

In Kapitel 1.2 wurde schon einmal kurz der Aufbau eines Datenbestandes X erläutert. Was jedoch noch nicht besprochen wurde, ist die Existenz von nominalen Skalen, also Merkmalen, die keine metrischen Werte, wie beispielsweise ein Gewicht oder Alter, angeben, sondern eine Kategorie darstellen. Als Beispiele seien hierfür das Geschlecht und die Blutgruppe einer Person genannt. Des Weiteren sind auch verschiedene Größenordnungen in einem Datenbestand zu finden und man muss sich damit beschäftigen, ob die Daten skaliert werden sollen und, wenn ja, wie.

3.1 Nominale Skalen

In dem k-Means Algorithmus wird die Euklidische Distanz verwendet, um die Ähnlichkeit von zwei Datenpunkten zu messen. In vielen Datensätzen sind jedoch nicht nur metrische, sondern auch nominale Variablen zu finden. Wenn zum Beispiel eine erfasste Variable die Modellpunkte den Kategorien A, B und C zuordnet, ist die Euklidische Distanz nicht dafür geeignet, den Unterschied zwischen solchen Variablen zu messen. Auch wenn die Kategorien in Nummern 1, 2 und 3 unterteilt werden, kann zwar eine Distanz gemessen werden, aber der Wert, der als Ergebnis erhalten wird hat keine Aussagekraft. Für den Abstand zwischen den Gruppen 1 und 3 würde der Wert 2 gemessen werden und für die Distanz zwischen 1 und 2 beziehungsweise 2 und 3, erhält man den Wert 1. Das würde bedeuten, dass der Unterschied zwischen den Kategorien 1 und 3 größer ist, als der der anderen. Das macht jedoch keinen Sinn, denn nominale Variablen sind entweder gleich oder unterschiedlich und haben keine Abstufungen von Ähnlichkeit. Deshalb soll als Distanzmaß für solche nominalen Merkmale die Diskrete Metrik verwendet werden:

$$d(x,y) = \begin{cases} 0, & \text{falls } x = y \\ 1, & \text{falls } x \neq y \end{cases} \tag{6}$$

Damit nicht der Programmcode für den k-Means Algorithmus geändert werden muss, soll der Datenbestand angepasst werden, bevor das Clusterverfahren auf ihn angewandt wird. Wenn ein nominales Merkmal nur zwei Kategorien annehmen kann, beziehungsweise in den verwendeten Daten nur zwei auftreten, dann sollen diese mit 0 und 1 bezeichnet werden. Ein typischen Beispiel für solch eine Variable in einem Versicherungsbestand ist das Geschlecht, weiblich $\hat{=}$ 0 und männlich $\hat{=}$ 1 oder auch umgekehrt. Für den Fall, dass mehr als nur zwei Ausprägungen eines nominalen Merkmals beobachtet worden sind, werden sogenannte Dummy-Variablen eingeführt, die anstelle der nominalen Variable im Datenbestand gespeichert werden sollen. Dadurch vergrößert sich auch die Dimension der Modellpunkte m. Hat eine Variable r verschiedene Kategorien, so werden r Dummy-Variablen $dummy_j$, $j = 1 \dots r$ statt dieser angelegt, wobei jede Dummy-Variable $dummy_j$ für eine Kategorie j steht. Somit wächst der Datenbestand X von einer $n \times m$ zu einer $n \times (m+r-1)$ Matrix. Wenn ein Modellpunkt zur Klasse j gehört, so nimmt die Dummy-Variable $dummy_j$ den Wert 1 und die restlichen Variablen $dummy_i$, $i = 1 \dots r$ wobei $i \neq j$ gilt, den Wert 0 an. Wichtig ist, dass bei einem nominalen Merkmal genau eine der zugehörigen Dummy-Variablen den Wert 1 annimmt.

Als Veranschaulichung wird in Tabelle 2 ein Beispiel gegeben, in dem die Augenfarbe als nominales Merkmal betrachtet wird.

	Augenfarbe	$\mathbf{dummy_{bl}}$	$\mathbf{dummy_{gr}}$	$\mathbf{dummy_{br}}$
Person 1	braun	0	0	1
Person 2	grün	0	1	0
Person 3	blau	1	0	0
Person 4	grün	0	1	0

Tabelle 2: Beispiel für Dummy-Variablen.

In der ersten Spalte ist die originale Variable gegeben, die die verschiedenen Kategorien angibt. Die anderen drei Spalten stellen die entsprechenden Dummy-Variablen dar, die das ursprüngliche Merkmal im Datenbestand ersetzen sollen.

Durch das Verwenden von solchen Hilfsvariablen erfüllt nun die Euklidische Distanz Gleichung (6) für nominale Merkmale und somit können auch Daten die sowohl metrische als auch nominale Variablen enthalten mit dem k-Means Algorithmus in Cluster eingeteilt werden. Es ist jedoch zu beachten, dass durch das k-Means Verfahren der Mittelwert der Modellpunkte eines Clusters berechnet wird und das Zentrum somit Werte für Dummy-Variablen annehmen kann, die weder 0 noch 1 sind, sondern im Intervall $(0, 1)$ liegen. Aufgrund dessen sollen, nachdem das Clusterverfahren angewandt wurde, nicht die Clusterzentren selbst, sondern der Datenpunkt, der den geringsten Abstand zu dem jeweiligen Zentrum hat, als Repräsentant gewählt werden. Diese Wahl macht auch für die spätere Anwendung im Bereich der Lebensversicherung Sinn, da anstatt von Mittelwerten eine Teilmenge von echten Verträgen, die tatsächlich im Bestand existieren, als Vertreter für diesen herangezogen werden.

Damit nicht für jedes Zentrum die Distanz zu allen Modellpunkten berechnet werden muss, wird jeweils nur der Abstand zu den Punkten gemessen, die sich im entsprechenden Cluster befinden, da das ohnedies jene Datenpunkte sind, die dem jeweiligen Zentrum am nächsten liegen. Dadurch wird auch der Rechenaufwand verringert.

Des Weiteren beschäftigen [BCK08] sich mit Äquivalenzmaßen für nominale Daten und schlägt viele verschiedene Möglichkeiten zur Messung ihrer Ähnlichkeit vor. Unter anderem entspricht das Ähnlichkeitsmaß *Overlap Measure*[5], Gleichung (7), dem in dieser Arbeit verwendetem Distanzmaß für nominale Daten (6).

$$s(x, y) = \begin{cases} 1, & \text{falls } x = y \\ 0, & \text{falls } x \neq y \end{cases} \tag{7}$$

3.2 Skalierung von Daten

Bei einem großen Datenbestand können viele verschiedene metrische Skalen enthalten sein, die unterschiedlich große Spannweiten besitzen. Beispielsweise können das Alter und das Gehalt der beobachteten Personen im Datenbestand gespeichert werden. Ein Altersunterschied von 10 Jahren ist relativ groß, wobei eine Gehaltsabweichung von 10€ vergleichsweise gering ist, wenn man betrachtet, dass eine Person selten älter wird als 100 Jahre und ein Gehalt von mehr als 2000€ keine Seltenheit ist. Durch die Euklidische

[5]Bemerke, dass hier der Wert 1 für die gleiche und 0 für unterschiedliche Kategorien angenommen wird. Vgl. hierfür mit Kapitel (1.2) den Unterschied zwischen Distanz- und Ähnlichkeitsmaßen.

Distanz würden nun Datenpunkte, die einen Altersunterschied von 10 Jahren und für alle anderen Merkmale dieselben Werte haben, als ähnlicher bewertet werden, als zwei Modellpunkte, die sich nur durch das Gehalt um 15€ unterscheiden. Demzufolge hat das Merkmal Gehalt einen stärkeren Einfluss auf den Abstand der Datenpunkte und beeinflusst somit auch die Aufteilung des Datenbestandes in Cluster. Damit die starke Wirkung von nicht sehr bedeutsamen Unterschieden in Variablen, wie in diesem Beispiel dem Gehalt, aufgehoben werden kann, muss der Datenbestand skaliert werden.

Falls es es Merkmale gibt, die bedeutsamer sind und deshalb einen stärkeren Einfluss auf die Clusteraufteilung haben sollten, können auch Gewichte für die Merkmale eingeführt werden.

Nun werden zwei verschiedene Varianten der Skalierung von Daten betrachtet. Einerseits können die Modellpunkte $x \in X$ so skaliert werden, dass die Ausprägungen jedes Merkmals im Bereich $[0; 1]$ liegen. Damit wird die Spannweite der Variablen vereinheitlicht. Betrachte dazu einen Datenpunkt x mit den Merkmalen x_i, wobei $i = 1 \dots m$. Die Einträge des entsprechenden, skalierten Modellpunktes $\hat{x}$ berechnen sich wie folgt:

$$\hat{x}_i = \frac{x_i - \min(X_i)}{\max(X_i) - \min(X_i)}, \quad i = 1 \dots m.$$

Dabei ist X_i der Vektor, der die Werte des Merkmals i von allen Modellpunkten in X enthält.

Andererseits kann anstatt der Spannweite, die Varianz der Variable vereinheitlicht werden. Dazu müssen die Werte des Merkmales durch die jeweils entsprechende Standardabweichung σ_i dividiert werden. Um die Daten auch zu zentrieren wird zuvor noch der Mittelwert μ_i des jeweiligen Merkmales von dem Eintrag x_i abgezogen. Dementsprechend werden die Werte des skalierten Datenpunktes folgendermaßen kalkuliert:

$$\hat{x}_i = \frac{x_i - \mu_i}{\sigma_i}, \quad i = 1 \dots m. \tag{8}$$

Hierzu ist anzumerken, dass in diesem Fall auch die nominalen Merkmale, beziehungsweise die bereits erstellten zugehörigen Dummy-Variablen, skaliert werden. Die Werte, die sich dadurch ergeben, sind welche, die diese Variablen nicht annehmen können. Die Cluster sollen trotzdem mit diesem überarbeiteten Datenbestand aufgestellt werden und die Modellpunkte können danach zurück transformiert werden, um wieder die originalen Datenwerte zu erhalten. In dieser Arbeit wird die Skalierung nach Formel (8) verwendet.

4 Optimale Anzahl an Clustern

Wie zuvor schon erwähnt ist es notwendig, dass die Anzahl k der Cluster, in die ein Datenbestand mithilfe des k-Means Algorithmus unterteilt werden soll, im Vorhinein gegeben ist. Diese Tatsache ist einer der größten Nachteile der Verwendung des k-Means Verfahrens. Wenn die Struktur des Datenbestandes noch nicht bekannt ist und es nicht möglich ist, aufgrund von Expertenwissen eine Anzahl der Gruppen festzulegen, muss ein anderer Weg gefunden werden, um die Clusteranzahl zu bestimmen.

Es gibt mittlerweile viele verschiedene Methoden, um heraus zu finden, welche Anzahl k an Clustern für einen gegebenen Datenbestand eine gute Wahl ist. Dafür muss der k-Means Algorithmus für mehrere Werte für die Clusteranzahl durchgeführt und die erhaltenen Ergebnisse daraufhin miteinander verglichen werden. Daher muss eine minimale und eine maximale Anzahl an Gruppen, in die der Datenbestand unterteilt werden darf, bestimmt werden. Wie die Güte gemessen wird, hängt von dem Verfahren ab. Dadurch kann es auch passieren, dass unterschiedliche Zahlen für die optimale Anzahl an Clustern erhalten werden. In dieser Arbeit werden 13 verschiedene Methoden zum Messen der Qualität der Clusteraufteilung verwendet. Jedes dieser Verfahren liefert einen Vorschlag für die optimale Anzahl an Gruppierungen und der Wert, der am häufigsten auftritt, wird dann als Anzahl für die zu suchenden Cluster gewählt.

Dazu gibt es in der Programmiersprache R eine bereits vorgefertigte Funktion namens *NbClust*, die im gleichnamigen Package zu finden ist. Diese Funktion kann für verschiedene Clusterverfahren, unter anderem dem k-Means Algorithmus, die optimale Anzahl der Cluster mithilfe diverser Verfahren zur Qualitätsmessung bestimmen. Es wird ihr der Datenbestand X, ein minimaler und ein maximaler Wert für die Clusteranzahl k übergeben. Diese Funktion verwendet den k-Means Algorithmus ohne das zuvor erstellte Verfahren zur Erstellung von guten Startzentren. Es soll an dieser Stelle nur eine Anzahl von Gruppen gefunden werden, die für den gegebenen Datenbestand passt, und daraufhin wird das Verfahren wie in Kapitel 2 beschrieben, unter Verwendung von reproduzierbaren, gut gelegenen Startzentren für das erhaltene k durchgeführt.

Es werden nun drei der verwendeten Methoden zur Feststellung der Qualität einer Clustereinteilung vorgestellt. Diese stehen alle in der Funktion *NbClust* zur Auswahl zur Verfügung. Ein Einblick in die verschiedenen Verfahren kann unter anderem auch in [CLAS10] und [LJB06] erhalten werden.

Dunn Index

Sei die Anzahl der Cluster k fix gewählt und C_j, $j = 1 \dots k$ die Clustereinteilung von X, deren Qualität überprüft werden soll. Weiter sei $d(C_i, C_j)$, $i \neq j$ die Distanz zwischen zwei Clustern, auch Intercluster-Distanz genannt, und $\hat{d}(C_j)$ die Intracluster-Distanz, also die Distanz innerhalb eines Clusters. Es gibt verschiedene Distanzmaße die für d beziehungsweise $\hat{d}$ gewählt werden können. Für die Intracluster-Distanz ist das beispielsweise der maximale Abstand zwischen zwei Modellpunkten, die im Cluster liegen, oder die Summe der Abstände aller Datenpunkte der Gruppe zu ihrem Mittelwert. Als Beispiel für die Intercluster-Distanz kann der Abstand der beiden Mittelwerte der Cluster herangezogen werden oder die kleinste Distanz zwischen zwei Modellpunkten, wobei aus jedem Cluster genau ein Punkt gewählt werden muss. Weitere Beispiele für Intercluster-Distanzen sind

in [XW09] zu finden. Damit ist der Dunn Index folgendermaßen definiert:

$$D = \frac{\min_{1 \leq i < j \leq k} d(C_i, C_j)}{\max_{1 \leq i \leq k} \hat{d}(C_i)}$$

Da bei einem Clusterverfahren versucht wird, eine Einteilung der Daten zu finden, bei der die Unterschiede innerhalb eines Clusters minimiert und die Differenz zwischen den Gruppen maximiert werden, bedeuten große Werte für den Dunn Index, dass die Qualität einer Clusterzuordnung gut ist. Demzufolge wird beim Vergleich von Clustereinteilungen beziehungsweise beim Betrachten verschiedener Anzahlen k von Clustern jene Gruppierung beziehungsweise Anzahl von Gruppen bevorzugt, die den höchsten Dunn Index D aufweist.

Silhouettenkoeffizient

Sei $x \in X$ ein Modellpunkt und C_j, $j = 1 \ldots k$ eine Clusteraufteilung des Datenbestandes. Dann berechnet sich die Silhouette von $x \in C_i$ wie folgt:

$$S(x) = \frac{b(x) - a(x)}{\max(a(x), b(x))}$$

Hierbei ist $a(x)$ der durchschnittliche Abstand von dem Datenpunkt x zu den anderen Punkten innerhalb des selben Clusters C_i und $b(x)$ ist als das Minimum der durchschnittlichen Abstände von x zu allen Modellpunkten eines Clusters C_j, $j \neq i$, in welchem sich der Punkt x selbst nicht befindet, definiert. $b(x)$ ist also die mittlere Distanz von x zu den Datenpunkten, die sich in der nächst gelegenen Gruppe befinden, die x nicht enthält.

$S(x)$ nimmt Werte in dem Intervall $[-1; 1]$ an. Wenn für einen Punkt seine Silhouette nahe bei 1 liegt, bedeutet das, dass die Zuordnung zu seinem Cluster sehr gut ist. Werte um 0 sagen aus, dass der Datenpunkt genau so gut zu einer anderen Gruppe gehören könnte und etwa in der Mitte zwischen den beiden liegt. Wenn die Silhouette einen Wert in der Nähe von -1 annimmt, wurde x einem falschen Cluster zugeordnet.

Der Silhouettenkoeffizient ist der Mittelwert aus allen Silhouetten $S(x)$, $x \in X$ und, entsprechend dem zuvor Erwähnten, sind große Werte ein Zeichen für eine gute Qualität der Clusterzuordnung. Wenn nun entschieden werden soll, welche Anzahl an Clustern für einen Datenbestand ideal ist, soll jene gewählt werden, die den Silhouettenkoeffizient maximiert. Siehe zu dieser Methode der Qualitätskontrolle für Clustereinteilungen auch [Rou87].

SD Index

Um den SD Index für eine Clustereinteilung zu bestimmen, müssen zuerst die Varianz des gesamten Datenbestandes X und die Varianzen innerhalb der einzelnen Cluster C_j, $j = 1 \ldots k$ ermittelt werden. Dazu sei μ_X der Mittelwert des ganzen Datenbestandes und c_j der des Clusters C_j, also das jeweilige Zentrum. Aus den folgenden komponentenweise

zu verstehenden Berechnungen ergeben sie die Varianzvektoren.

$$\begin{aligned} \sigma_X &= \frac{1}{n}\sum_{x\in X}(x-\mu_X)^2; \\ \sigma_{C_j} &= \frac{1}{|C_j|}\sum_{x\in C_j}(x-c_j)^2, \quad j=1\ldots k. \end{aligned}$$

In weiterer Folge wird nun die durchschnittliche Streuung der Cluster und die gesamte Streuung zwischen den Clustern kalkuliert:

$$\begin{aligned} \mathrm{str}_d &= \frac{1}{k}\sum_{j=1}^{k}\frac{\|\sigma_{C_j}\|}{\|\sigma_X\|}; \\ \mathrm{str}_g &= \frac{\max_{i,j=1\ldots k}(\|c_i-c_j\|)}{\min_{i,j=1\ldots k,\ i\neq j}(\|c_i-c_j\|)}\sum_{l=1}^{k}\left(\sum_{h=1}^{k}\|c_l-c_h\|\right)^{-1} \end{aligned}$$

Letztendlich wird mit dem Gewichtungsfaktor α, der dem Wert str_g für die maximale Anzahl an Clustern entspricht, der SD Index berechnet:

$$\mathrm{SD} = \alpha\cdot\mathrm{str}_d + \mathrm{str}_g\,.$$

Im Gegensatz zu den zuvor besprochenen Qualitätsmaßen bedeuten hohe Werte für den SD Index, dass die betrachtete Clusteraufteilung schlecht ist, und niedrige Werte sind daher wünschenswert. Siehe zu diesem Index auch [HVB00].

5 Anwendungsbereich Lebensversicherung

Bis hierher wurden die Verfahren und Algorithmen an sich betrachtet und erklärt. Es ist Ziel dieser Arbeit, eine repräsentative Gruppe aus einem Versicherungsbestand auszuwählen, um versicherungsmathematische Berechnungen zukünftiger Cashflows zu optimieren. Die Kalkulationen sollen nicht für den gesamten Bestand an Policen durchgeführt werden müssen, da dies einen enormen Zeitaufwand bedeutet. Wenn ähnliche Verträge in Gruppen zusammengefasst werden können und für jede dieser Klassen ein repräsentativer Vertrag gefunden wird, für welchen die Cashflowberechnungen anstatt für jede einzelne Police aus seinem Cluster durchgeführt werden, erhält man eine Annäherung an die Cashflows des Bestandes. Diese Approximation soll das Verhalten des gesamten Versicherungsbestandes widerspiegeln.

Der erwartete Vorteil von Clusterverfahren im Gegensatz zur Optimierung mittels linearer Regression unter Verwendung des Non-Negative Least Squares Algorithmus ist, dass nur die Informationen aus dem Vertrag verwendet und keine Cashflows der einzelnen Verträge gebraucht werden. Dadurch ist die Wahl der Repräsentanten unabhängig von Annahmen zweiter Ordnung, die zur Berechnung der Cashflows benötigt werden, und damit auch im Falle von Schockszenarien gültig. Bei der Optimierung mittels linearer Regression ist nicht sichergestellt, dass die repräsentative Gruppe auch noch unter stark veränderten Bedingungen, wie beispielsweise einer gravierenden Änderung der Sterblichkeitsraten, immer noch eine passende Wahl zur Widerspiegelung der Eigenschaften des gesamten Versicherungsbestandes darstellt. Da die Informationen, die zum Erstellen von Clustern mithilfe des k-Means Algorithmus benötigt werden keine solchen Annahmen verwenden, sind die erhaltenen Ergebnisse für alle möglichen Szenarien, die Einfluss auf die Kalkulation der Cashflows haben, gültig.

Nun soll das bisher Erarbeitete auf einen Versicherungsbestand angewandt werden. Dazu wird zuerst betrachtet, wie solch ein Datenbestand aufgebaut ist und welche Informationen in ihm enthalten sind. Des Weiteren werden auch das Versicherungsprodukt, welches in den Beispielen verwendet wird, und die bisher angewandte Methode kurz beschrieben.

5.1 Versicherungsportfolio

In einem Versicherungsbestand sind die Modellpunkte die einzelnen Versicherungsverträge und die Merkmale die Informationen, die ein Versicherungsunternehmen über die versicherte Person und die Eigenschaften des Vertrages hat. Da ein Versicherungsunternehmen im Regelfall nicht nur ein einziges Produkt, sondern eine Vielzahl von verschiedenen Versicherungen anbietet, für die unterschiedliche Merkmale des Versicherungsvertrages eine wichtige Rolle spielen, wird für jedes Versicherungsprodukt ein separater Datenbestand angefertigt.

Ein wichtiges Merkmal einer Versicherungspolice ist der Versicherungstarif, der die Berechnungsgrundlagen, wie beispielsweise den zu verwendenden Zins, die Abschlusskosten und die zugrunde liegende Sterbetafel festlegt. Da der Tarif die Grundlage zur Kalkulation zukünftiger Cashflows darstellt, soll für jede Tarifgruppe extra eine Auswahl an Repräsentanten gesucht werden. Also soll ein Versicherungsbestand, der mehr als eine Tarifgruppe beinhaltet, in so viele Datensätze geteilt werden, wie er Tarife enthält, um das

Clusterverfahren für jeden Versicherungstarif separat anzuwenden. Auch das Merkmal Währung soll sich in einem Datenbestand, für den eine Clusterzerlegung gesucht wird, nicht unterscheiden. Demzufolge wird für jede Kombination aus Tarif und Währung ein eigener Datenbestand angelegt, um diesen mithilfe der zuvor erläuterten Algorithmen in Gruppen zu teilen.

Die restlichen zur Berechnung der Cashflows relevanten Eigenschaften sollen nun dazu dienen, Cluster der Modellpunkte zu finden. Typische Merkmale sind:

- das Alter der versicherten Person bei Abschluss des Vertrages,
- ihr Geschlecht,
- das Datum des Vertragsabschlusses,
- die versicherte Summe,
- die Höhe der Prämie,
- wie oft die Prämie gezahlt wird,
- die gültige Laufzeit des Vertrages,
- die Information, ob ein zweites versichertes Leben vorhanden ist,
- falls ja, sein Alter und Geschlecht,
- usw.

Falls für eine der Variablen bei allen Datenpunkten der selbe Wert auftritt, kann diese Spalte aus dem Datenbestand entfernt werden, da sie keine Informationen zum Finden von Clustern enthält.

Das Versicherungsprodukt, das in den folgenden Beispielen verwendet wird, ist eine Gemischte Versicherung mit einer Einmalprämie. Die Währung der Versicherungsverträge ist der Rumänische Leu (RON), als zugrundeliegende Sterbetafel wird *RO 2013 unisex* verwendet und der technische Rechnungszins beträgt 2%.

Die Höhe der Prämie Π ergibt sich aus folgender Gleichung, die durch Anwendung des Äquivalenzprinzips erhalten wird:

$$\Pi \cdot S = A_{x,n\rceil} \cdot S + \alpha_1 \cdot \Pi \cdot S + \alpha_2 \cdot \Pi \cdot S + \gamma \cdot \ddot{a}_{x,n\rceil} \cdot S. \tag{9}$$

Hierbei ist S die Versicherungssumme. Die Abschlusskosten α werden in α_1 und α_2 aufgeteilt und die Verwaltungskosten sind durch γ gegeben. $A_{x,n\rceil}$ ist der Barwert einer gemischten Versicherung und $\ddot{a}_{x,n\rceil}$ der Barwert einer jährlichen, vorschüssigen Leibrente auf ein Leben x mit einer Laufzeit von n Jahren. Die Kostenparameter sind folgendermaßen definiert:

$$\begin{aligned} \alpha_1 &= \min\left(0.8\%, \frac{135}{S}\right), \\ \alpha_2 &= 2.95\%, \\ \gamma &= \min\left(0.4\%, \frac{40}{S}\right). \end{aligned}$$

Durch Umformen der Gleichung (9) erhält man:

$$\Pi = \frac{A_{x,n\rceil} + \gamma \cdot \ddot{a}_{x,n\rceil}}{1 - \alpha_1 - \alpha_2}.$$

Um ein Gefühl zu bekommen, wie viele Informationen über eine Police tatsächlich betrachtet werden müssen, wird nun ein Beispiel eines Modellpunktes gegeben.

Merkmale eines Modellpunktes

Es werden 66 Merkmale, die Informationen über die versicherte Person und die Konditionen der Police darstellen, pro Versicherungsvertrag gespeichert. Da für den Tarif, der in den Beispielen dieser Arbeit verwendet wird, nicht alle dieser erhobenen Variablen erforderlich sind, sollen nur die 44 relevanten Merkmale aufgelistet und erklärt werden.

In Tabelle 3 sind in der ersten Spalte die Namen der Variablen zu finden, in der zweiten sind die Werte eines Modellpunktes als Beispiel und in der dritten Spalte eine kurze Beschreibung zu dem jeweiligen Merkmal gegeben.

AGE_ AT_ ENTRY	39	Alter der versicherten Person bei Vertragsbeginn.
CURRENCY	"RON"	Zugrundeliegende Währung des Versicherungsvertrages.
DURATIONIF_ M	13	Bereits abgelaufene Vertragsdauer zum Start der Projektionsperiode in Monaten angegeben.
ENTRY_ DAY	1	Tag des Versicherungsbeginns der Vertragsscheibe.
ENTRY_ MONTH	12	Monat des Versicherungsbeginns.
ENTRY_ YEAR	2017	Jahr des Versicherungsbeginns.
EXTRA_ PREM	0	Extra Prämie, falls zum Beispiel eine erhöhte Mortalität oder Ähnliches vorhanden ist.
INIT_ DECB_ IF	51.36	Aktueller Bonus der gültigen Police. Dies ist ein Betrag, der über die Jahre durch Gewinnbeteiligung angesammelt wird und am Ende des Vertrages oder beim Tod der versicherten Person ausbezahlt wird.
INIT_ POLS_ IF	1	Anzahl der Policen, die genau die Merkmale dieses Modellpunktes besitzen. Dadurch können mehrere Datenpunkte, die identisch sind, zu einem zusammengefasst werden und müssen nicht alle extra im Datenbestand angeführt werden.
ORIG_ ENTRY_ MONTH	12	Ursprünglicher Monat des Versicherungsbeginns.
ORIG_ ENTRY_ YEAR	2017	Ursprüngliches Jahr des Versicherungsbeginns.

POL_ TERM_ Y	5	Vertragsdauer in Jahren angegeben.
POLICYNUMBER	12345	Die Policennummer.
PREM_ DISC_ PC	0	Prämien Nachlass als Prozentsatz der Bruttoprämie angegeben.
PREM_ FREQ	Annually	Modalität der Prämienzahlungen.
PREM_ PAYBL_ Y	1	Anzahl an Jahren in denen eine Prämie fällig ist.
PROFIT_ SHARING	Yes	Zeigt an, ob es sich um einen gewinnberechtigten Vertrag handelt.
RECO_ PREM_ DB	4979.38	Höhe der Bruttoprämie.
RIDER_ PREM_ 1	0	Höhe der Prämie für die erste Zusatzversicherung.
RIDER_ PREM_ 2	0	Höhe der Prämie für die zweite Zusatzversicherung.
RIDER_ PREM_ 3	0	Höhe der Prämie für die dritte Zusatzversicherung.
RIDER_ PREM_ 4	0	Höhe der Prämie für die vierte Zusatzversicherung.
RIDER_ PREM_ 5	0	Höhe der Prämie für die fünfte Zusatzversicherung.
SEGMENT	0	Nummer der Vertragsscheibe (Segment), das heißt ein Vertrag kann aus mehreren Modellpunkten bestehen. Hierbei muss darauf geachtet werden, dass bei einer Police mit beispielsweise zwei Segmenten die Variable NO_ POLS_ IF auf 0.5 gesetzt werden muss.
SEX	Female	Das Geschlecht der versicherten Person.
SUM_ ASSURED	5179.37	Die Versicherungssumme.
TARIFF	Tariff_ A	Der Versicherungstarif.
TARIFF_ GROUP	Group_ A	Die Tarifgruppe.
USE_ ALPHA_ 0	1	Verwendung von Alpha0 Kosten.
USE_ ALPHA_ 1	1	Verwendung von Alpha1 Kosten.
USE_ ENTRY_ DATE	0	Zeigt an, ob der Versicherungsbeginn oder die bereits abgelaufene Versicherungsdauer verwendet wird, um den jeweils anderen Parameter zu ermitteln.
WT_ POLS	1	Gewichtung des Merkmals *Anzahl an Policen.*
MTHS_ TO_ SALE	0	Monate bis zum Versicherungsbeginn, falls zukünftiges Neugeschäft im Datenbestand enthalten ist.

BUSINESS	"EB"	Art des Geschäftes. *EB* steht für *existing business*, also für bereits bestehendes Geschäft.
SA_ MAT_ FAC	1	Der Faktor der mit der Versicherungssumme multipliziert wird, um die Höhe der Leistung im Erlebensfall zu erhalten.
RIDER_ 1	0	Ob erste Zusatzversicherung vorhanden ist.
RIDER_ 2	0	Ob zweite Zusatzversicherung vorhanden ist.
RIDER_ 3	0	Ob dritte Zusatzversicherung vorhanden ist.
RIDER_ 4	0	Ob vierte Zusatzversicherung vorhanden ist.
RIDER_ 5	1	Ob fünfte Zusatzversicherung vorhanden ist.
LOB	1	Geschäftssparte.
EXTRA_ PREM_ MULT_ PC	0	Falls eine Extra Prämie vorhanden ist, wird dieser Wert als Multiplikator auf die Bruttoprämie angewandt.
PROP_ RESQ_ PC	100	Diese Variable wird zur Anpassung der Mortalitätsraten, die zur Kalkulation der mathematischen Reserve verwendet werden, benutzt. Sie ist ein Prozentsatz der mit den Werten der zu verwendeten Sterbetafel multipliziert wird.
PROP_ VALQ_ PC	100	Diese Variable wird zur Anpassung der Mortalitätsraten, die zur Kalkulation der Prämie verwendet werden, benutzt. Sie ist ein Prozentsatz der mit den Werten der zu verwendeten Sterbetafel multipliziert wird.

Tabelle 3: Merkmale eines Versicherungsvertrages.

Es gibt einige Merkmale die *ja* oder *nein* als Einträge haben, wobei 0 *nein* und 1 *ja* bedeutet.

Für solch einen hochdimensionalen Datensatz ist es schwer eine gute optische Darstellung zu finden. Eine Möglichkeit ist, dass jede Kombination aus zwei Merkmalen in einer Ebene geplottet wird. Dies resultiert jedoch in einer enormen Menge an Grafiken, die nur selten hilft, sich die Aufteilung des Datenbestandes räumlich vorstellen zu können.

Als eine weitere Variante zur grafischen Darstellung der Gesamtheit der betrachteten Modellpunkte kann die Hauptkomponentenanalyse (Principal Component Analysis), siehe [Pea01], herangezogen werden. Unter Verwendung dieses statistischen Verfahrens wird

versucht, in dem zu untersuchenden hochdimensionalen Raum jene Ebene zu finden, die die meiste Information enthält. Die Vorgehensweise dazu soll nun kurz erklärt werden.

1. Suche jene Gerade beziehungsweise Linearkombination der Merkmale, die die größte Varianz aufweist.
2. Fixiere diese Gerade.
3. Suche jene Linearkombination mit der maximalen Varianz, die normal auf die fixierte Gerade steht.

Damit wird jene Ebene erhalten, die den größten Anteil der Varianz des Datenbestandes und damit die meiste Information beschreibt. Das Bild dieser zweidimensionalen Ebene gewährt einen Einblick in die Struktur des Portfolios.

Dieses Verfahren wird im Folgenden unter anderem zur grafischen Aufbereitung der Ergebnisse verwendet.

5.2 Betrachtete Cashflows

Bevor der erstellte Algorithmus auf das erste Beispiel angewandt wird, sollen die betrachteten Cashflows kurz erwähnt werden:

- Todesfallleistung: Die Leistung, die erhalten wird, falls die versicherte Person vor Ende der Vertragslaufzeit stirbt.
- Stornoleistung: Die Leistung, die von der versicherten Person erhalten wird, falls sie den Versicherungsvertrag storniert.
- Leistung zur Fälligkeit des Vertrages: Die Leistung, die die versicherte Person erhält, falls sie am Ende der Vertragslaufzeit noch am Leben ist.
- Leistungen aus Zusatzversicherungen: Die Leistungen, die aus den Zusatzversicherungen anfallen.
- Kosten: Kosten, die das Versicherungsunternehmen aufgrund dieser Police hat.
- Anstieg der Rücklagen: Differenz der Rücklagen verglichen zur vorherigen Periode.
- Kapitalerträge
- gesetzliche Rücklagen: Die vom Gesetz vorgeschriebenen Rücklagen, die für die Police zu halten sind.
- Risikosumme: Der Teil der Versicherungssumme, der noch nicht durch die Reservierung von eingezahlten Prämien abgedeckt ist.

Nun soll der Cashflow *Stornoleistungen* als Beispiel genauer untersucht werden. Falls ein Versicherungsvertrag storniert wird, erhält der/die Versicherungsnehmer/in das Deckungskapital V_t zu diesem Zeitpunkt t abzüglich eines Abschlags $f \in [0, 1]$,

$$\mathrm{SV}_t = \mathrm{V}_t \cdot (1 - f).$$

Zur Erzeugung der wahrscheinlichkeitsgewichteten Cashflows wird ein Dekrementmodell zu Hilfe genommen. Dazu wird der Monat t betrachtet. Am Anfang des Monats gibt es NO_ POLS_ IFSM(t) Policen. Diese berechnen sich aus den Anzahl der Policen NO_ POLS_ IF($t-1$) am Ende des vorherigen Monats $t-1$ abzüglich der Anzahl NO_ MATS(t) jener, deren Fälligkeit zum Zeitpunkt t eintritt.

$$\text{NO\_ POLS\_ IFSM}(t) = \text{NO\_ POLS\_ IF}(t-1) - \text{NO\_ MATS}(t).$$

Am Ende des Monats t sind NO_ POLS_ IF(t) Policen in Kraft, dieser Wert unterscheidet sich von der Menge der Verträge zu Beginn des Monats um die Anzahl der Todesfalle NO_ DEATHS(t) und der Storni NO_ SURRS(t), die in diesem Monat auftreten.

$$\text{NO\_ POLS\_ IF}(t) = \text{NO\_ POLS\_ IFSM}(t) - \text{NO\_ DEATHS}(t) - \text{NO\_ SURRS}(t).$$

Zur Kalkulation der Anzahl der Todesfälle und der Storni werden eine Sterbetafel mit Sterbewahrscheinlichkeiten q'_x und Stornowahrscheinlichkeiten s'_x benötigt, die Annahmen zweiter Ordnung[6] sind. Damit ergibt sich

$$\begin{aligned} \text{NO\_ DEATHS}(t) &= q'_x \cdot \text{NO\_ POLS\_ IFSM}(t), \\ \text{NO\_ SURRS}(t) &= s'_x \cdot (1 - q'_x) \cdot \text{NO\_ POLS\_ IFSM}(t). \end{aligned}$$

Damit wird der tatsächliche Cashflow *Stornoleistungen* zum Zeitpunkt t folgendermaßen erhalten:

$$\text{SURR\_ OUTGO}(t) = \text{SV}_t \cdot \text{NO\_ SURRS}(t)$$

5.3 Non-Negative Least Squares

Da die vom k-Means Algorithmus erhaltenen Ergebnisse mit jenen der bisher verwendeten Methode zur Optimierung verglichen werden sollen, wird in diesem Kapitel die lineare Regression unter Anwendung des Non-Negative Least Squares Algorithmus behandelt.

Dabei handelt es sich um eine lineare Methode der kleinsten Quadrate (Linear Least Squares)[7], die eine zusätzliche Bedingung an die Lösung hat, welche ist, dass diese nicht negativ sein darf.

Dabei gibt es zu jedem beobachteten Objekt $j = 1 \dots m$, n unabhängige Variablen $x_{j,i}$, $i = 1 \dots n$ und eine abhängige Variable y_j. Bei der Methode der kleinsten Quadrate wird jene lineare Funktion f von $x_j = (x_{j,1}, \dots, x_{j,n})^\intercal$ gesucht, die den quadratischen Fehler QF (10) zwischen den tatsächlichen Werten y_j und den von der Funktion approximierten Werten $\hat{y_j} = f(x_j) = a_0 + a_1 \cdot x_{j,1} + \dots + a_n \cdot x_{j,n}$ der m gegebenen Objekte minimiert.

$$\text{QF} = \sum_{j=1}^{m} (\hat{y_j} - y_j)^2 \tag{10}$$

[6] Diese Wahrscheinlichkeiten unterscheiden sich von den Annahmen erster Ordnung, die beispielsweise zur Berechnung der Prämien verwendetet werden. Die Annahmen erster Ordnung sind meist vorsichtiger gewählt, als die zweiter Ordnung, welche die Wirklichkeit realistischer darstellen sollen.

[7] [Bjö96]

Das bedeutet insbesondere, dass jener Vektor $a = (a_0, \ldots, a_n)^\intercal$ gesucht wird, der den quadratischen Fehler QF (10) minimiert, wobei noch die zusätzliche Bedingung $a \geq 0$ komponentenweise erfüllt werden soll. Mit der $m \times (n+1)$ - Matrix $X = (\mathbf{1}, (x_1, \cdots, x_m)^\intercal)$, wobei $\mathbf{1}$ ein m-dimensionaler Vektor mit lauter 1 als Einträgen ist und mit $y = (y_1, \ldots, y_m)^\intercal$ ergibt sich dadurch folgendes Optimierungsproblem:

$$\min_a \|Xa - y\|_2, \quad a \geq 0.$$

Um dieses Optimierungsproblem zu lösen wird ein komplizierter Algorithmus angewandt, der in [HH81] vorgestellt wird. Dabei ist zu Beginn a ein Nullvektor, der in jeder Iteration einen neuen Wert ungleich 0 für einen seiner Einträge erhält. Nicht nur Stellen, an denen der Wert 0 gespeichert ist, können ersetzt werden, sondern auch welche, die bereits einen echt positiven Eintrag besitzen, falls dafür ein besserer Wert von dem Algorithmus gefunden wurde. In dem bisher verwendeten Programm werden dabei höchstens $3n$ Iterationen durchgeführt, bevor der Algorithmus abgebrochen wird. Siehe dazu auch [Del16].

In dem hier betrachteten Fall sind die Objekte j die verschiedenen Cashflows, die eine Police erzeugen kann. Dementsprechend ist m die Anzahl der betrachteten Cashflows. Die unabhängigen Variablen $x_{j,i}$, $i = 1 \ldots n$ sind die Werte der Cashflows der einzelnen Versicherungsverträge und somit ist n die Anzahl der Policen in dem betrachteten Portfolio. Die abhängige Variable y_j ist der kumulierte Wert des Cashflows j für den gesamten Versicherungsbestand. Der gesuchte Vektor a hat als Einträge die Gewichte für die einzelnen Modellpunkte des Versicherungsbestandes, die das gesamte Portfolio repräsentieren sollen, und die Summe $\sum_{i=1}^{n} a_i$ aller Gewichte muss der Anzahl der Policen im Bestand n entsprechen.

In dieser Anwendung wird $a_0 = 0$ gesetzt, da das Ziel ist, die Versicherungspolicen zu gewichten, und damit reduzieren sich die Dimensionen der Matrix X zu $m \times n$.

Zur besseren Veranschaulichung wird nun ein einfaches Beispiel gegeben. Angenommen es wird ein Versicherungsportfolio, welches aus vier Versicherungsverträgen besteht, betrachtet. Mithilfe von der Modellierungssoftware Prophet wird der Cashflow *Prämieneinnahmen* PREM_ INC für die nächsten fünf Jahre, für jeden Vertrag einzeln berechnet. Das bedeutet, dass in diesem Beispiel $m = 5$ und $n = 4$ gilt.

	Police 1	**Police 2**	**Police 3**	**Police 4**
PREM_ INC Jahr 1	100	102	101	114
PREM_ INC Jahr 2	90	87	80	105
PREM_ INC Jahr 3	80	70	75	97
PREM_ INC Jahr 4	70	66	68	88
PREM_ INC Jahr 5	60	59	61	76

Tabelle 4: Fiktives Beispiel für den Cashflow *Prämieneinnahmen* der nächsten fünf Jahre eines Versicherungsbestandes mit vier Policen.

Die Einträge der Tabelle 4 entsprechen den Einträgen der 5×4 - Matrix X. Dadurch

ergibt sich für dieses Beispiel nachfolgendes Optimierungsproblem.

$$\min_{a} \left\| \begin{pmatrix} 100 & 102 & 101 & 114 \\ 90 & 87 & 80 & 105 \\ 80 & 70 & 75 & 97 \\ 70 & 66 & 68 & 88 \\ 60 & 59 & 61 & 76 \end{pmatrix} \times \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} - \begin{pmatrix} 417 \\ 362 \\ 322 \\ 292 \\ 256 \end{pmatrix} \right\|_2, \quad a \geq 0.$$

Die triviale Lösung für dieses Optimierungsproblem ist $a = (1, 1, 1, 1)^\intercal$, da die Werte der abhängigen Variable y der Summen der jeweiligen Zeile in Matrix X entsprechen. Ziel ist es jedoch, eine echte Teilmenge des Bestandes auszuwählen, um das Portfolio zu repräsentieren. Daher wird als zusätzliche Einschränkung verlangt, dass höchstens $k < n$ Einträge des Vektors a einen Wert ungleich 0 annehmen dürfen. In diesem Beispiel könnte $k = 2$ gewählt werden. Dadurch werden höchstens zwei der Modellpunkte als Repräsentanten des Versicherungsportfolios gewählt.

Wie zuvor bereits erwähnt wurde, wird in jeder Iteration ein Gewicht ungleich 0 in den Vektor a eingetragen. Sobald k Einträge des Vektors einen echt positiven Wert zugeordnet bekommen haben oder bereits $3n$ Iterationen durchgeführt wurden, wird der Algorithmus abgebrochen und dadurch wird gewährleistet, dass maximal k Policen als Repräsentanten für das Versicherungsportfolio ausgewählt werden.

Die Gewichte a_i zeigen dann an, wie oft der Vertrag i in dem repräsentativen Bestand enthalten ist, und können nicht nur ganzzahlige, sondern auch reelle Werte annehmen. Da eine negative Gewichtung der Modellpunkte nicht erwünscht ist, wird der Non-Negative Least Squares Algorithmus verwendet, dem Aufgrund seiner Zusatzbedingung eine negative Gewichtung gar nicht erst möglich ist.

Bei der tatsächlichen Anwendung in der Lebensversicherung werden viele verschiedene Cashflows jeweils für mehrere Jahre berechnet und in den Zeilen der Matrix X eingetragen. Dies geschieht für alle Verträge des zu repräsentierenden Bestandes, welche die Spalten darstellen. Auf diese Art kann unter Zuhilfenahme von Cashflows, zu deren Kalkulation Annahmen zweiter Ordnung notwendig sind, ein repräsentativer Teil eines Versicherungsportfolios ausgewählt werden.

5.4 Erste vereinfachte Beispiele

Als erstes Beispiel wird ein fiktiver Bestand erstellt, für welchen bekannt ist, wie die Clustereinteilung auszusehen hat. Im ersten Schritt wird eine Police aus einem realen Versicherungsbestand, bei welchem es sich um ein rumänisches Portfolio handelt, herangezogen und vervielfacht, um daraufhin nur die Werte eines einzelnen Merkmals zu variieren. Hiermit wird es auch einfacher, den Datenbestand optisch darzustellen, da nicht alle Dimensionen betrachtet werden müssen.

Für den ersten Testbestand wird das Merkmal *Alter bei Abschluss des Vertrages* variiert. Dazu werden zuerst drei Clusterzentren gewählt und daraufhin die Modellpunkte um diese herum erstellt, sodass das fixierte Zentrum dem Mittelwert der umliegenden Datenpunkte entspricht. Damit wird ein Datenbestand erzeugt, für den sowohl die Clusterzentren, als auch die Zuordnung aller Datenpunkte zu den Clustern und die Größe der Cluster bekannt sind.

Mithilfe der aktuariellen Modellierungssoftware Prophet[8] werden für jeden Vertrag dieses fiktiven Portfolios die Prämie und die anzulegende Reserve pro Prämie kalkuliert. Diese neuen Variablen ergeben sich aus den in den Modellpunkten gespeicherten Merkmalen und liefern zusätzliche Informationen, die zur Clusteranalyse verwendet werden können. Da diese neuen Variablen unter anderem von dem Alter der versicherten Person anhängig sind, ergeben sich unterschiedliche Werte und es gibt nun drei Dimensionen in dem betrachteten Versicherungsbestand, die nicht nur den selben Wert annehmen und damit Einfluss auf das Finden der Cluster haben. Diese zwei neuen Merkmale sind aber nicht von Annahmen abhängig, die in einem Schockszenario andere Ergebnisse liefern könnten. Somit ist eine Auswahl an Repräsentanten, die unabhängig von äußeren Einflüssen ist, immer noch gewährleistet.

In Tabelle 5 ist der erste Testbestand dargestellt. Um Platz zu sparen werden hier nur die Merkmale angeführt, die unterschiedliche Werte annehmen. Unter der Variable *Alter* ist das Alter der versicherten Person zum Zeitpunkt des Vertragsabschlusses zu verstehen. Die versicherte Summe der betrachteten Verträge, die einen großen Einfluss auf die Werte der neuen Variablen *Prämie* und *Reserve pro Prämie* hat, beträgt 5000 RON. Des Weiteren beträgt die Vertragslaufzeit für alle Datenpunkte zehn Jahre.

Dieser Testbestand besteht aus 30 fiktiven Versicherungsverträgen. Die drei Modellpunkte, die mit dicker Schrift hervorgehoben wurden, sind die gewählten Clusterzentren. Darunter sind die Datenpunkte aufgelistet, die für das jeweilige Zentrum erstellt wurden, also die zugehörigen Elemente des Clusters. Das bedeutet, es gibt in diesem Datenbestand nun drei Gruppen, wobei eine von ihnen acht Versicherungspolicen und die anderen zwei elf Verträge beinhalten.

Der zuvor erarbeitete Algorithmus wird nun auf den soeben erstellten Versicherungsbestand angewandt, um seine Leistung zu testen. Als minimale beziehungsweise maximale Anzahl an Clustern wurden dem Programm die Werte 2 und 5 übergeben. Es liefert, wie gewünscht als Ergebnis drei Clusterzentren, die genau den fett gedruckten Verträgen in Tabelle 5 entsprechen. Auch die Zuordnung der einzelnen Datenpunkte stimmt mit dem Erwünschten überein und die Clustergrößen betragen acht, elf und elf.

Es ist wichtig, dass die Größe des Clusters in der Variable INIT_ POLS_ IF (Anzahl der gültigen Policen) des Clusterzentrums gespeichert wird, da diese Versicherungspolice nun nicht mehr nur sich selbst, sondern alle Verträge in ihrem Cluster repräsentiert. Dies ist wichtig für die Cashflowberechnung, die nun so durchgeführt wird, als ob acht mal der Vertrag, der dem ersten Clusterzentrum entspricht, und elf mal die Police des zweiten und dritten Zentrums im Versicherungsportfolio enthalten wären und nicht mehr die vielen verschiedenen Datenpunkte.

In Abbildung 1 sind die Datenpunkte des fiktiven Versicherungsbestandes mit den drei Merkmalen abgebildet, die nicht für alle Verträge denselben Wert annehmen. Die Modellpunkte sind jeweils in einer von drei Farben eingefärbt, um ihre Zugehörigkeit zu einem der drei Cluster darzustellen.

Wie vorher bereits erwähnt, kann auch mithilfe der Hauptkomponentenanalyse eine grafische Darstellung des Datenbestandes erstellt werden. Dies ist in Abbildung 2 zu sehen. Auch hier werden die Cluster farblich hervorgehoben. Die jeweiligen Zentren

[8] prophet-web.com

Alter	Prämie	Reserve pro Prämie
20	**4665.97**	**4496.76**
21	4665.94	4496.73
19	4665.95	4496.74
18	4665.92	4496.71
22	4665.94	4496.73
18	4665.92	4496.71
18	4665.92	4496.71
23	4665.91	4496.70
40	**4667.86**	**4498.53**
45	4670.43	4500.95
35	4666.75	4497.49
41	4668.23	4498.88
39	4667.57	4498.25
42	4668.67	4499.29
38	4667.27	4497.97
43	4669.17	4499.76
37	4667.03	4497.75
36	4666.89	4497.62
44	4669.74	4500.30
60	**4686.50**	**4516.03**
61	4687.86	4517.31
59	4685.08	4514.70
65	4695.01	4524.04
59	4685.08	4514.7 0
56	4681.26	4511.13
62	4689.56	4518.90
58	4683.82	4513.52
63	4691.12	4520.37
57	4682.59	4512.36
60	4686.50	4516.03

Tabelle 5: Testbestand mit drei Clustern, unter Variation des Merkmals *Alter der versicherten Person bei Abschluss des Vertrages*.

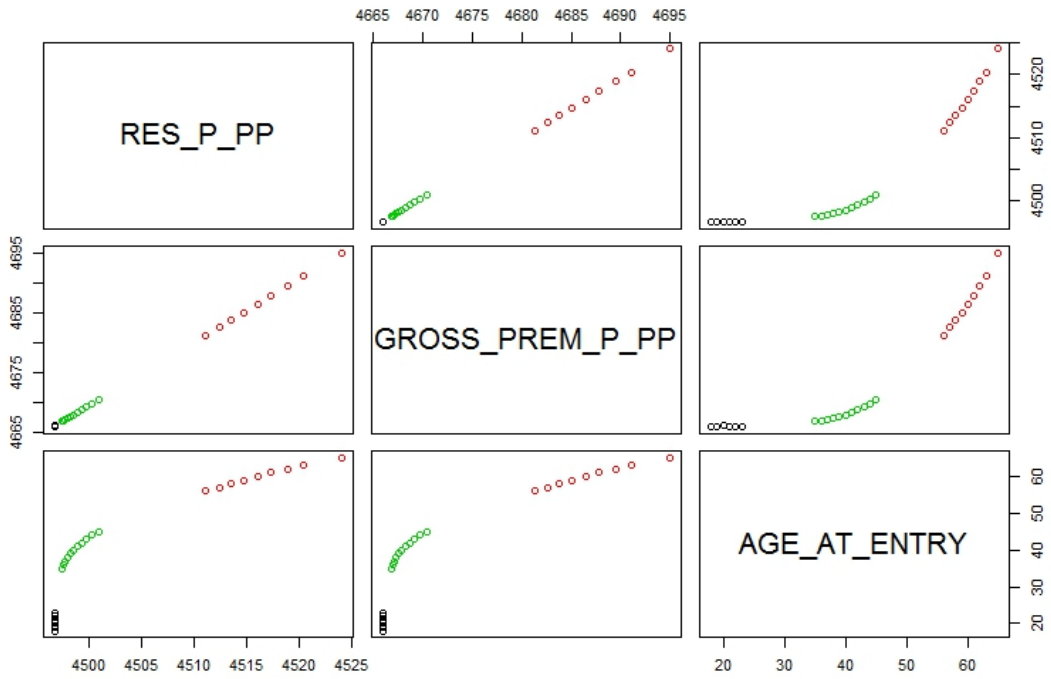


Abbildung 1: Grafiken der Ebenen aus jeweils zwei der Merkmale *Reserve pro Prämie*, *Prämie* und *Alter zu Versicherungsbeginn* für den Datenbestand mit drei Clustern. Mithilfe des Programmes R erstellt.

der Gruppen werden mit einem größeren Symbol als die restlichen Modellpunkte dargestellt. Die hier eingetragenen Zentren sind die tatsächlichen Clusterzentren, die von dem k-Means Algorithmus als Ergebnis erhalten werden, und nicht unbedingt existierende Modellpunkte aus dem betrachteten Datenbestand.

An den Achsen der Abbildung kann abgelesen werden, wie viel Prozent der Varianz des Datenbestandes anhand dieser Dimension, die aus einer Linearkombination der Merkmale besteht, erklärt werden kann.

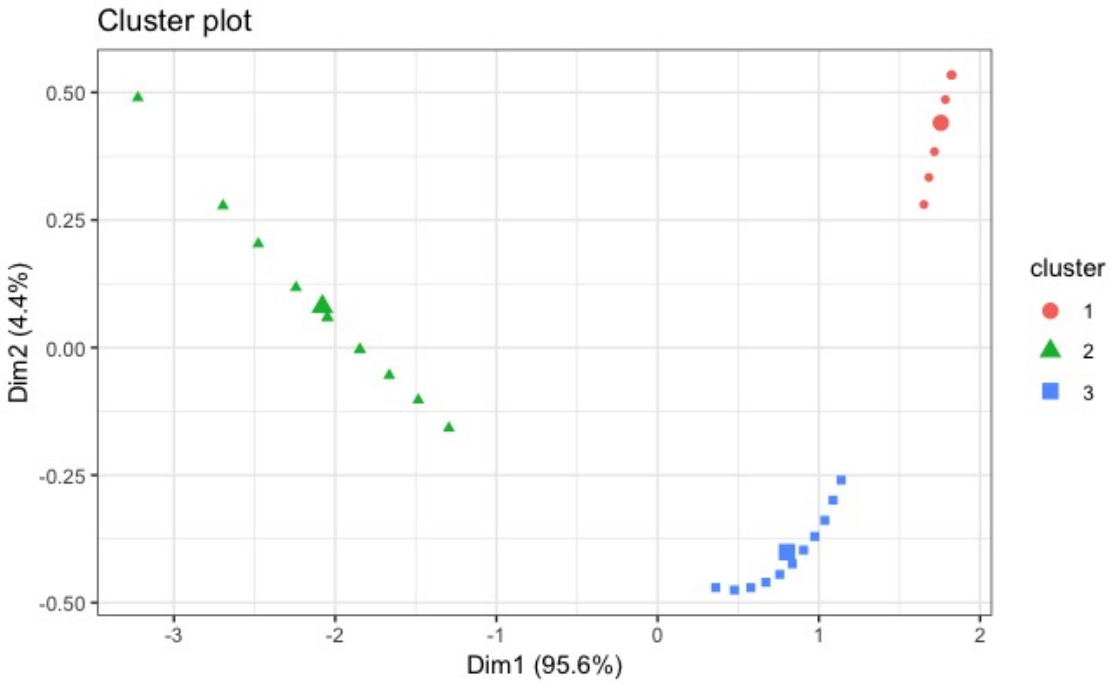
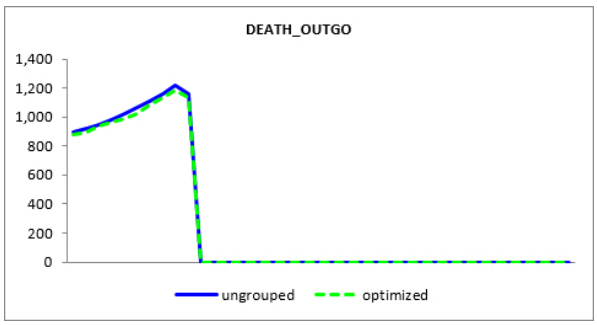


Abbildung 2: Grafische Darstellung der drei Cluster unter Verwendung der Hauptkomponentenanalyse. Mithilfe des Programmes R erstellt.

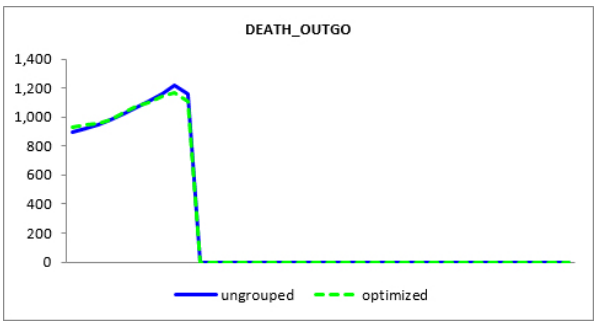
Da die Auswahl von Repräsentanten des Versicherungsportfolios jenen Sinn hat, die Cashflows des gesamten Bestandes zu approximieren, sollen nun die Ergebnisse der Cashflowberechnungen für den gesamten Versicherungsbestand mit jenen der Repräsentanten verglichen und des Weiteren überprüft werden, wie gut diese miteinander übereinstimmen.

Dazu werden die Werte der Cashflows mithilfe von Prophet berechnet und daraufhin unter Verwendung von Excel miteinander verglichen. Da nur das Merkmal *Alter* variiert wurde, stimmen alle Cashflows mit Ausnahme der Todesfallleistungen miteinander überein. Diese Beobachtung ist nicht überraschend, da das Alter der versicherten Person einen großen Einfluss auf die Todesfallleistung hat und in diesem Testbestand nur an dieser Stelle approximiert wird.

Zum Vergleich wird auch das bisher verwendete Verfahren, das mithilfe des Non-Negative Least Squares Algorithmus arbeitet, auf das Testportfolio angewandt. Dies liefert bessere Ergebnisse, als der in dieser Arbeit eingeführte Algorithmus, welcher das k-Means Verfahren verwendet. In Abbildung 3 können die Resultate für die Todesfallleistungen der nächsten 40 Projektionsjahre betrachtet werden.



(a) Todesfallleistungen für die Approximation durch k-Means.



(b) Todesfallleistungen für die Approximation durch lineare Regression.

Abbildung 3: Vergleich der Ergebnisse der beiden Optimierungsmethoden für den Cashflow *Todesfallleistungen*. Die blaue Linie *ungrouped* sind die Werte der Cashflows für den gesamten Bestand und die grüne Linie *optimized* die der Repräsentanten.

Um die Güte der Ergebnisse zu bewerten wird ein Schwellenwert eingeführt. Solange die Abweichung des Wertes eines Cashflows der Repräsentanten von dem des gesamten Versicherungsbestandes weniger als 1% beträgt, ist das Resultat zufriedenstellend. In Tabelle 6 sind die prozentuellen Abweichungen für beide Verfahren gegeben. Es werden nur die ersten zehn Projektionsjahre angegeben, da die Verträge alle eine Laufzeit von zehn Jahren haben und alle Werte der Cashflows nach Vertragsende und somit auch die Differenzen 0 betragen.

Jahr	**2019**	**2020**	**2021**	**2022**	**2023**	**2024**	**2025**	**2026**	**2027**	**2028**
***k*-Means**	1.5%	3.0%	0.8%	2.2%	3.1%	3.8%	2.3%	2.7%	2.6%	2.3%
lin Reg	3.9%	2.2%	0.5%	0.1%	0.7%	0.9%	0.9%	1.6%	4.1%	4.5%

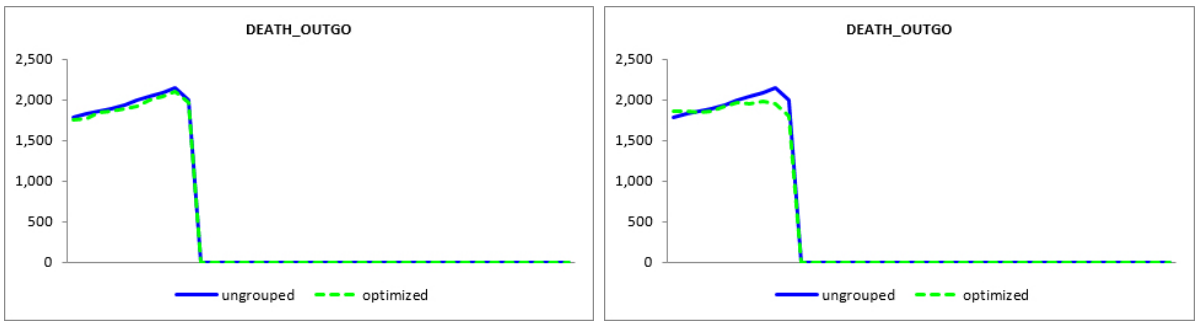
Tabelle 6: Prozentuelle Abweichungen der Todesfallleistungen für die ersten zehn Projektionsjahre. Vergleich zwischen dem erstellten k-Means Verfahren und der bisher verwendeten Methode mittels linearer Regression.

Unter Verwendung des k-Mean Algorithmus wird nur im Jahr 2021 das Gütekriterium erfüllt. In allen anderen Jahren beträgt die Abweichung mehr als 1%. Für das Verfahren der linearen Regression erfüllen nur die ersten zwei und die letzten drei Jahre das Kriterium nicht und somit liefert es eine bessere Approximation. Es ist jedoch zu bemerken, dass die größten Abweichungen unter Verwendung des Non-Negative Least Squares Algorithmus auftreten.

Da bei Zuhilfenahme der linearen Regression mehr Information, insbesondere Information über die Cashflows verwendet wird, ist es jedoch nicht verwunderlich, dass dieses Verfahren bessere Ergebnisse liefert, als der k-Means Algorithmus, der keine Annahmen zweiter Ordnung zur Auswahl der Repräsentanten benutzt. Der Vorteil, der durch das Weglassen solcher Informationen erhofft wird, ist, dass auch bei starken Veränderungen der Annahmen zweiter Ordnung die Wahl der Repräsentanten immer noch richtig ist.

Um zu untersuchen, ob dieser Vorteil tatsächlich besteht, wird ein Mortalitätsschockszenario erstellt. Unter der Annahme, dass sich die Mortalität verdoppelt, werden nun ein weiteres Mal die Cashflows für die zuvor erhaltenen Repräsentanten berechnet. In

Abbildung 4 sind die Todesfallleistungen der nächsten 40 Projektionsjahre unter dem Schockszenario zu sehen.



(a) Todesfallleistungen für die Approximation durch k-Means unter einem Mortalitätsschockszenario.

(b) Todesfallleistungen für die Approximation durch lineare Regression unter einem Mortalitätsschockszenario.

Abbildung 4: Vergleich der Ergebnisse der beiden Optimierungsmethoden für den Cashflow *Todesfallleistungen* unter einem Mortalitätsschockszenario. Die blaue Linie *ungrouped* sind die Werte der Cashflows für den gesamten Bestand und die grüne Linie *optimized* die der Repräsentanten.

Es ist gut erkennbar, dass der k-Means Algorithmus für solch ein Szenario bessere Ergebnisse liefert. In Tabelle 7 sind zur genaueren Analyse die prozentuellen Werte der Abweichungen gegeben.

Jahr	**2019**	**2020**	**2021**	**2022**	**2023**	**2024**	**2025**	**2026**	**2027**	**2028**
k-Means	1.5%	3.0%	0.7%	2.0%	2.9%	3.5%	1.8%	2.1%	1.9%	1.5%
lin Reg	3.9%	1.8%	0.3%	1.4%	1.1%	1.5%	4.0%	5.6%	8.8%	10.4%

Tabelle 7: Prozentuelle Abweichungen der Todesfallleistungen für die ersten zehn Projektionsjahre unter einem Mortalitätsschockszenario. Vergleich zwischen dem erstellten k-Means Verfahren und der bisher verwendeten Methode mittels linearer Regression.

Das k-Means Verfahren erfüllt das Gütekriterium zwar wieder nur im Jahr 2021, aber das gilt auch für die Methode der linearen Regression, bei der gegen Ende hin die Abweichung immer größer wird und einmal sogar mehr als 10% beträgt. Diese Resultate bestätigen den erhofften Vorteil der Verwendung des Clusterverfahrens gegenüber der Anwendung des Non-Negative Least Squares Algorithmus.

Für den nächsten Testbestand wird wieder das Merkmal *Alter der versicherten Person bei Abschluss des Vertrages* herangezogen. Dieses Mal werden die Modellpunkte so erstellt, dass es fünf Cluster gibt. In Tabelle 8 werden die Zentren der Cluster aufgelistet. Der erzeugte Versicherungsbestand besteht in diesem Beispiel aus 50 Policen und auf die Auflistung der einzelnen Datenpunkte wird an dieser Stelle verzichtet. In der vierten Spalte ist die Anzahl der Verträge, die sich in der erstellten Gruppen befinden, abzulesen.

Bei einer Eingabe von 3 als minimale und 10 als maximale Anzahl an Gruppen werden die erwarteten Zentren von dem Programm erhalten. In Abbildung 5 und Abbildung

Alter	Prämie	Reserve pro Prämie	Anzahl Verträge
20	4665.97	4496.76	8
30	4666.09	4496.87	14
40	4667.86	4498.53	11
50	4674.73	4504.99	6
60	4686.50	4516.03	11

Tabelle 8: Zentren des Testbestands mit fünf Clustern, unter Variation des Merkmals *Alter der versicherten Person bei Abschluss des Vertrages*.

6 sind die grafischen Darstellungen des Datenbestandes zu sehen. Die Cluster sind wieder farblich gekennzeichnet und in Abbildung 6 werden die Clusterzentren durch größere Symbole hervorgehoben. An dieser Stelle ist ein weiteres Mal anzumerken, dass die hier dargestellten Zentren, die tatsächlichen Mittelwerte, berechnet aus den zur Gruppe zugeordneten Modellpunkten und nicht die in Tabelle 8 angegebenen Verträge sind, die jene Datenpunkte darstellen, die den geringsten Abstand zu den Zentren annehmen.

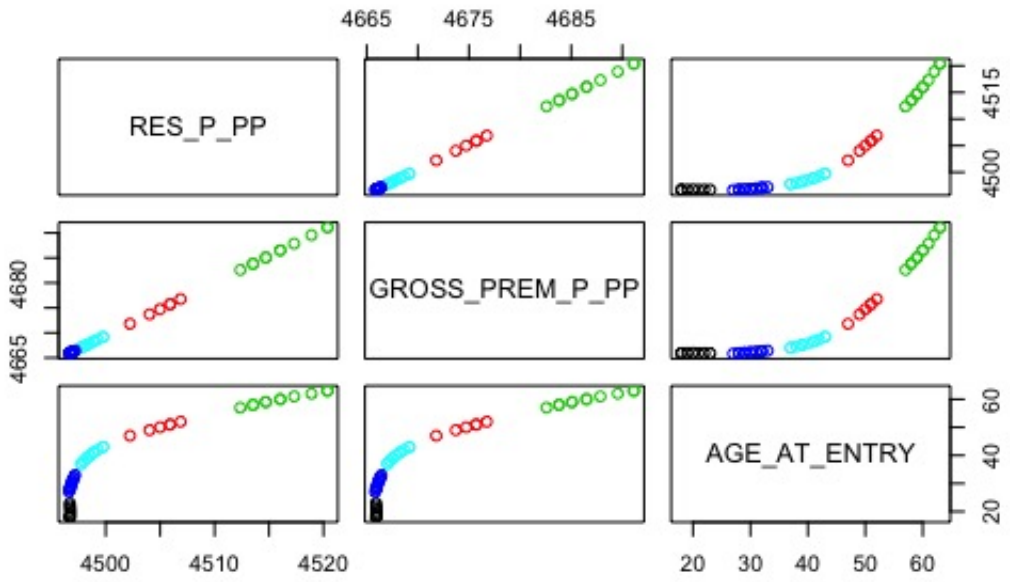


Abbildung 5: Grafiken der Ebenen aus jeweils zwei der Merkmale *Reserve pro Prämie*, *Prämie* und *Alter zu Versicherungsbeginn* für den Datenbestand mit fünf Clustern. Mithilfe des Programmes R erstellt.

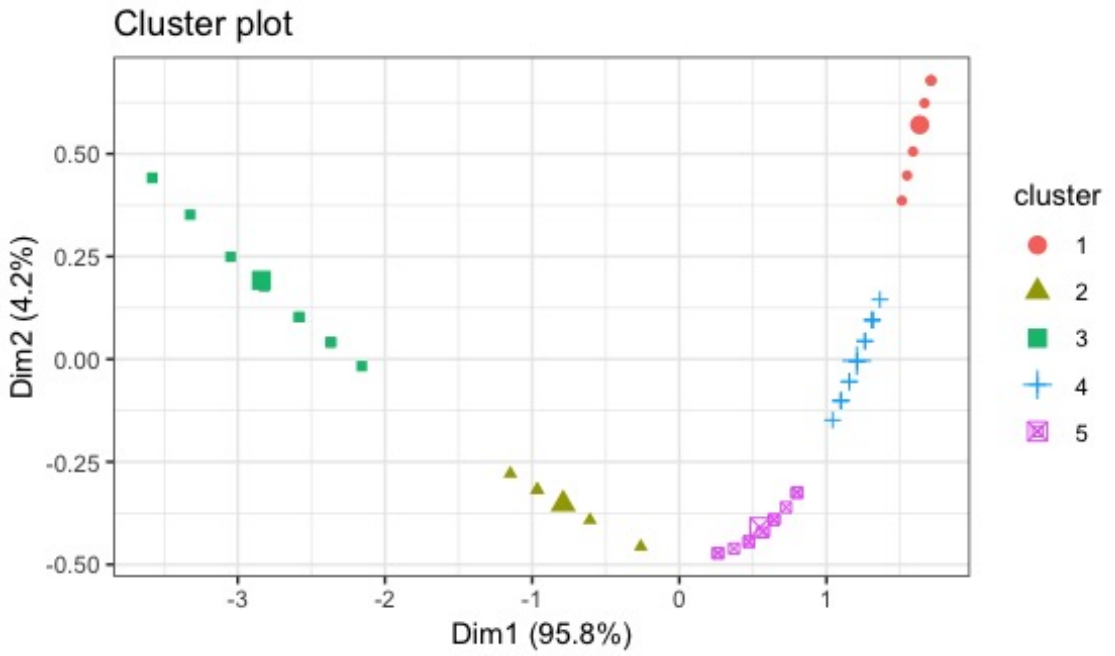
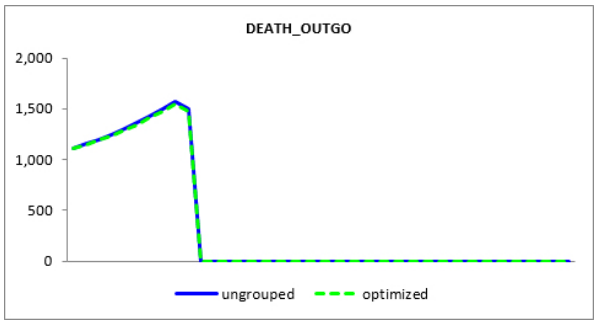
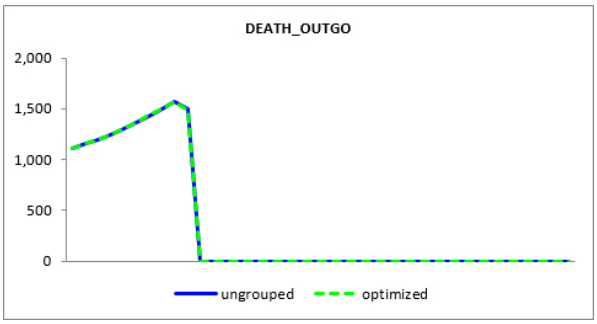


Abbildung 6: Grafische Darstellung der fünf Cluster unter Verwendung der Hauptkomponentenanalyse. Mithilfe des Programmes R erstellt.

Auch für diesen Testbestand wurden die Cashflows mit Prophet berechnet und mithilfe von Excel die Abweichungen zu dem gesamten Versicherungsbestand kalkuliert. In Abbildung 7 sind die dadurch erhaltenen Graphiken zu sehen. Dabei stellt wieder die blaue Linie die Werte des gesamten Portfolios und die grüne die der Repräsentanten dar.



(a) Todesfallleistungen für die Approximation durch k-Means.



(b) Todesfallleistungen für die Approximation durch lineare Regression.

Abbildung 7: Vergleich der Ergebnisse der beiden Optimierungsmethoden für den Cashflow *Todesfallleistungen*. Die blaue Linie *ungrouped* sind die Werte der Cashflows für den gesamten Bestand und die grüne Linie *optimized* die der Repräsentanten.

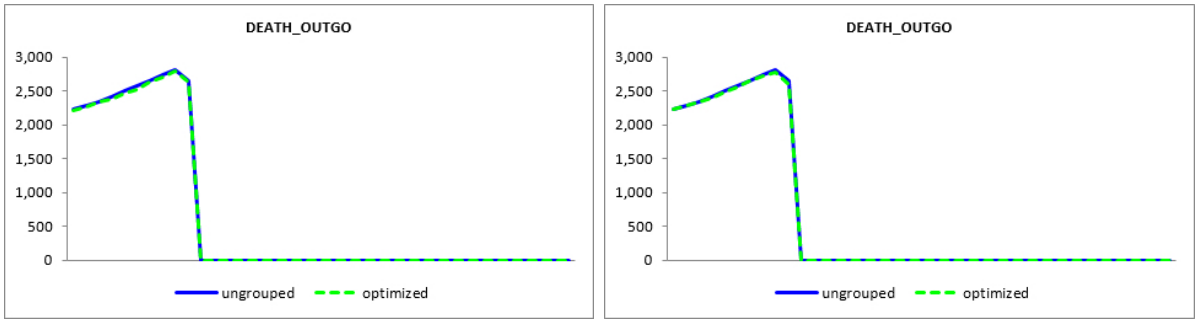
In Tabelle 9 sind die genauen Werte der prozentuellen Abweichungen der ersten zehn Projektionsjahre angegeben. Wie bei dem ersten Beispiel ist auch hier das Ergebnis der bisher verwendeten Optimierung besser, als das des neu erstellten k-Means Algorithmus. Das Gütekriterium wird nur in zwei Jahren erfüllt, im Gegensatz zum Verfahren, welches die lineare Regression verwendet. Jedoch beträgt die Abweichung nie mehr als 1.8%. In

Abbildung 7(a) ist gut zu sehen, dass die blaue und die grüne Linie nicht weit voneinander abweichen.

Jahr	**2019**	**2020**	**2021**	**2022**	**2023**	**2024**	**2025**	**2026**	**2027**	**2028**
***k*-Means**	0.6%	1.2%	0.1%	1.3%	1.5%	1.8%	1.2%	1.3%	1.2%	1.2%
lin Reg	0.0%	0.3%	0.0%	0.5%	0.3%	0.1%	0.4%	0.4%	0.1%	0.3%

Tabelle 9: Prozentuelle Abweichungen der Todesfallleistungen für die ersten zehn Projektionsjahre. Vergleich zwischen dem erstellten k-Means Verfahren und der bisher verwendeten Methode mittels linearer Regression.

Wie für den Testbestand zuvor, soll auch hier ein Mortalitätsschockszenario betrachtet werden, bei dem sich die Sterblichkeitsraten verdoppeln. In Abbildung 8 und Tabelle 10 sind die erhaltenen Resultate angeführt.



(a) Todesfallleistungen für die Approximation durch k-Means unter einem Mortalitätsschockszenario.

(b) Todesfallleistungen für die Approximation durch lineare Regression unter einem Mortalitätsschockszenario.

Abbildung 8: Vergleich der Ergebnisse der beiden Optimierungsmethoden für den Cashflow *Todesfallleistungen* unter einem Mortalitätsschockszenario. Die blaue Linie *ungrouped* sind die Werte der Cashflows für den gesamten Bestand und die grüne Linie *optimized* die der Repräsentanten.

Jahr	**2019**	**2020**	**2021**	**2022**	**2023**	**2024**	**2025**	**2026**	**2027**	**2028**
***k*-Means**	0.6%	1.2%	0.0%	1.2%	1.4%	1.6%	1.0%	1.0%	0.9%	0.8%
lin Reg	0.0%	0.2%	0.2%	0.8%	0.8%	0.5%	0.3%	0.6%	1.2%	1.8%

Tabelle 10: Prozentuelle Abweichungen der Todesfallleistungen für die ersten zehn Projektionsjahre unter einem Mortalitätsschockszenario. Vergleich zwischen dem erstellten k-Means Verfahren und der bisher verwendeten Methode mittels linearer Regression.

In diesem Fall liefert auch unter dem hier gewählten Schockszenario der Non-Negative Least Squares Algorithmus die besseren Lösungen und nicht wie gehofft der k-Means Algorithmus.

5.5 Beispiel: Variation zweier Merkmale

Da Versicherungsbestände in der Realität eine Vielzahl an Verträgen beinhalten und bisher die Testbestände nur 30 beziehungsweise 50 Policen enthalten haben, sollen nun größere Versicherungsportfolios erstellt und getestet werden. Da für das in diesen Beispielen verwendete Versicherungsprodukt das Alter der versicherten Person nur zwischen 18 und 65 Jahren liegen darf und dieses Merkmal deshalb nur 48 verschiedene Werte annehmen kann, macht es nicht viel Sinn, in einem größeren Bestand nur diese Variable zu variieren. Deshalb sollen von nun an auch für das Merkmal *Vertragslaufzeit* verschiedene Werte betrachtet werden. Die restlichen Vertragsmerkmale werden wieder für jede Police im Testbestand gleich sein.

Dieser Versicherungsbestand besteht aus 300 fiktiven Verträgen. Die Ausprägungen des Merkmals *Alter bei Abschluss des Vertrages* wurden mit normal verteilten Zufallsvariablen in dem Programm R generiert. Dazu wurden mit den in Tabelle 11 angegebenen Mittelwerten μ und Standardabweichungen σ jeweils 100 Werte erzeugt.

μ	23	40	59
σ	4	4	4

Tabelle 11: Mittelwerte und Standardabweichungen zur Erzeugung verschiedener Werte für das Merkmal *Alter bei Abschluss des Vertrages* mithilfe normal verteilter Zufallsvariablen.

Dabei beträgt das minimal mögliche Alter 18 Jahre und das maximal mögliche 65 Jahre. Die Werte für die Variable *Vertragslaufzeit* werden mithilfe einer uniform verteilten Zufallsvariable ausgewählt. Hierfür sind 2 Jahre die minimale und 25 Jahre die maximale Dauer der Laufzeit, die ein Vertrag besitzen kann. Das Merkmal *Versicherungssumme* nimmt, wie auch im Testbestand zuvor, für alle Policen denselben Wert an und beträgt nun 5314.61 RON.

Mit der Modellierungssoftware Prophet werden auch hier die Prämie und die Reserve pro Prämie für jeden der 300 Verträge kalkuliert und als zusätzliche Information für die Clustereinteilung mittels dem k-Means Algorithmus benutzt. Dieser liefert als Ergebnis drei Cluster, deren Zentren in Tabelle 12 abzulesen sind. Auch die Anzahl der Elemente im jeweils zugehörigen Cluster ist in Tabelle 12 eingetragen.

Alter	**Vertragslaufzeit**	**Prämie**	**Reserve**	**Anzahl Verträge**
39	20	4300.24	4203.1	127
27	9	5035.73	4928.3	94
54	7	5195.16	5082.6	79

Tabelle 12: Zentren des erstellten Testbestands unter Variation der Merkmale *Alter der versicherten Person bei Abschluss des Vertrages* und *Laufzeit des Versicherungsvertrages*.

In Abbildung 9 wird der Versicherungsbestand mithilfe der Hauptkomponentenanalyse grafisch dargestellt und die drei erhaltenen Cluster werden wieder mit drei verschiedenen Farben und unterschiedlichen Symbolen gekennzeichnet. Die größeren Symbole stellen

auch hier die Clusterzentren dar, welche die kalkulierten Zentren der jeweiligen Gruppe sind, und nicht die in Tabelle 12 angegebenen Policen, die die Datenpunkte mit der kleinsten Distanz zu den eigentlichen Clustermittelpunkten darstellen. Im Gegensatz zu den Beispielportfolios, die zuvor betrachtet wurden, ist bei diesem Testbestand keine eindeutige Einteilung in Cluster mit freiem Auge erkennbar.

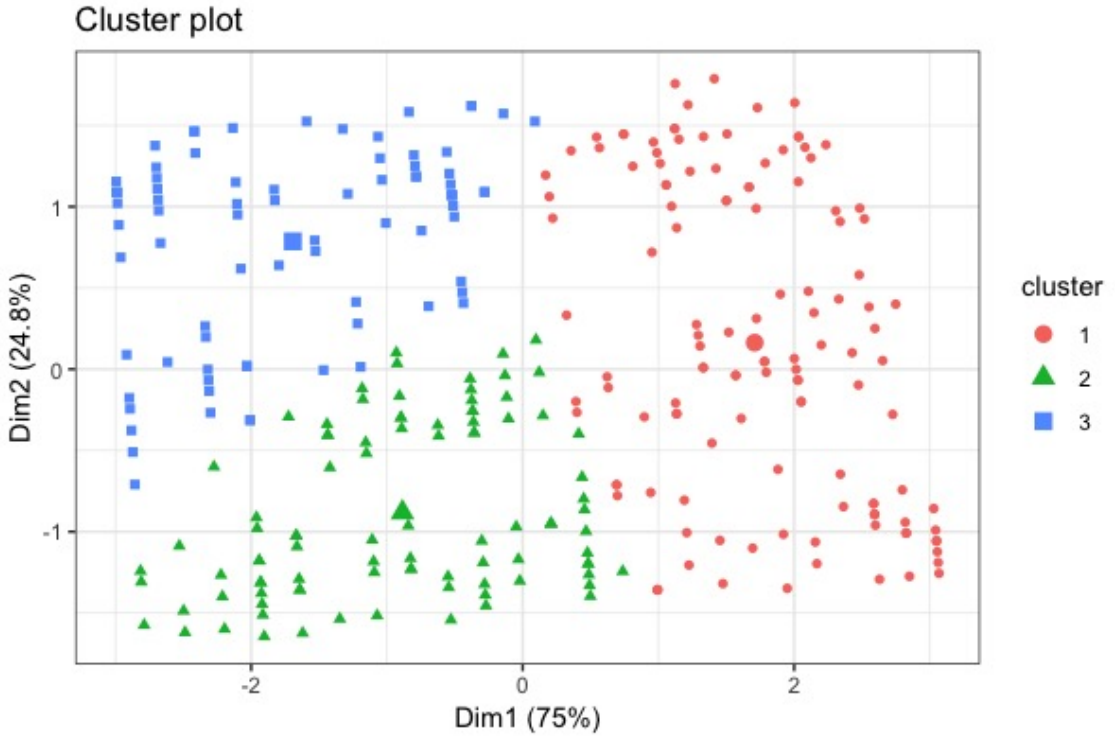
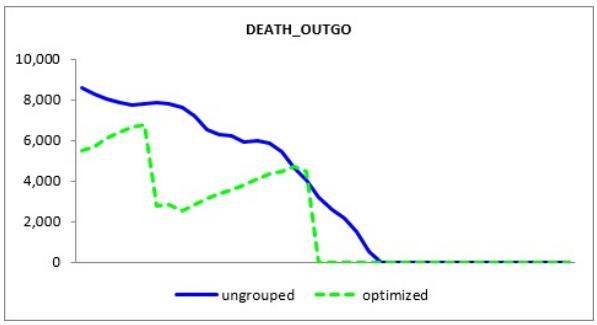


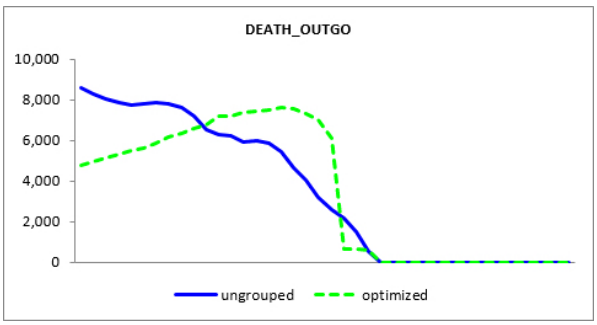
Abbildung 9: Grafische Darstellung der drei Cluster unter Verwendung der Hauptkomponentenanalyse. Mithilfe des Programmes R erstellt.

Als nächsten Schritt werden, wie auch zuvor, die Cashflows in Prophet für den gesamten Bestand und für die Repräsentanten berechnet und in Excel miteinander verglichen. Da nun nicht mehr nur das Merkmal *Alter bei Abschluss des Vertrages* verschiedene Werte annimmt, ist im Gegensatz zu den vorherigen Bespielen nicht nur der Cashflow der Todesfallleistung zu untersuchen. Zusätzlich zu den Todesfallleistungen werden ab jetzt auch die Stornoleistungen und die Leistungen zur Fälligkeit des Vertrages betrachtet.

Es werden auch mithilfe der bisher verwendeten Methode der Non-Negativ Least Squares drei Repräsentanten aus dem Bestand herausgesucht, um daraufhin die Cashflowberechnungen zum Vergleich durchzuführen. In den Abbildungen 10, 11 und 12 sind die Ergebnisse der beiden Methoden für die Cashflows Todesfallleistung, Stornoleistung und Leistungen zur Maturität des Vertrages nebeneinander gestellt.

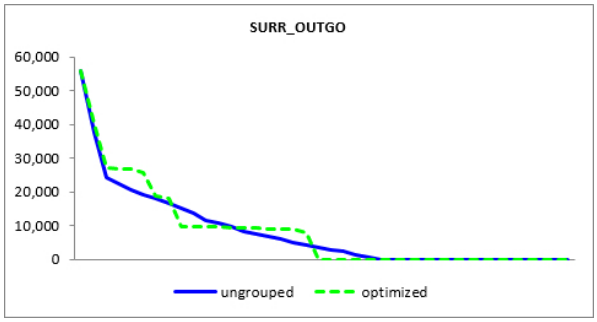


(a) Todesfallleistungen für die Approximation durch k-Means.

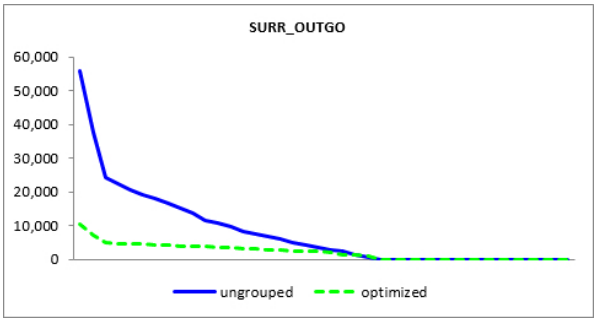


(b) Todesfallleistungen für die Approximation durch lineare Regression.

Abbildung 10: Vergleich der Ergebnisse der beiden Optimierungsmethoden für den Cashflow *Todesfallleistungen*. Die blaue Linie *ungrouped* sind die Werte der Cashflows für den gesamten Bestand und die grüne Linie *optimized* die der Repräsentanten.

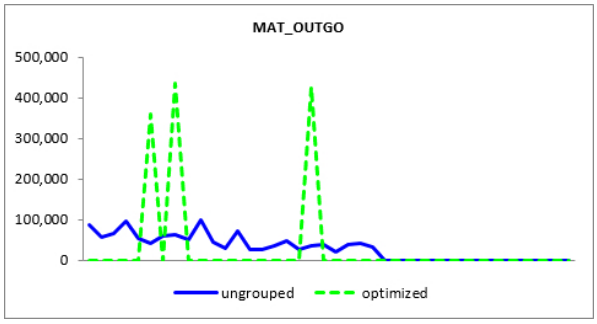


(a) Stornoleistungen für die Approximation durch k-Means.

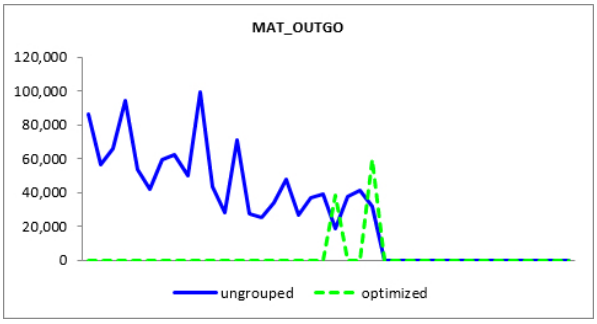


(b) Stornoleistungen für die Approximation durch lineare Regression.

Abbildung 11: Vergleich der Ergebnisse der beiden Optimierungsmethoden für den Cashflow *Stornoleistungen*. Die blaue Linie *ungrouped* sind die Werte der Cashflows für den gesamten Bestand und die grüne Linie *optimized* die der Repräsentanten.



(a) Leistungen zur Maturität für die Approximation durch k-Means.



(b) Leistungen zur Maturität für die Approximation durch lineare Regression.

Abbildung 12: Vergleich der Ergebnisse der beiden Optimierungsmethoden für den Cashflow *Leistungen zur Maturität*. Die blaue Linie *ungrouped* sind die Werte der Cashflows für den gesamten Bestand und die grüne Linie *optimized* die der Repräsentanten.

Es ist sofort erkennbar, dass beide Methoden keine gute Approximation für den gesamten Bestand liefern. Das kann daran liegen, dass das gesamte Versicherungsportfolio von 300 Verträgen durch nur 3 Policen repräsentiert wird. Das erklärt vor allem auch, dass der Cashflow *Leistungen zur Maturität*, Abbildung 12, für beide verwendeten Algorithmen nur in den Jahren, in denen die Fälligkeit der repräsentativen Verträge eintritt, einen Wert ungleich 0 annimmt. Unter Verwendung des k-Means Algorithmus wird in jenen Jahren, in denen eine Leistung für den Erlebensfall ausbezahlt wird, der tatsächliche Wert um ein Vielfaches überschritten, da angenommen wird, dass in diesen Jahren die Maturität für 79, 94 beziehungsweise 127 Verträge gleichzeitig eintritt. Bei Anwendung der bisher verwendeten Methode, sind jene Werte, die nicht 0 entsprechen, relativ gute Annäherungen an den Wert des gesamten Portfolios.

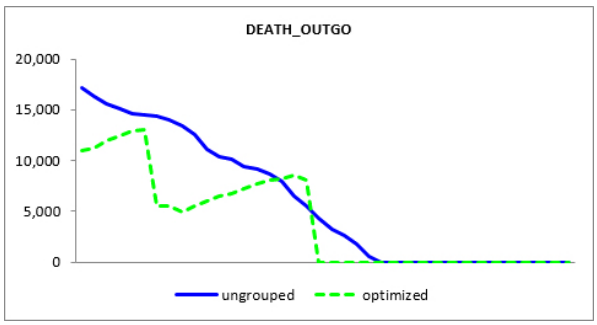
Wird jedoch die Summe aller zukünftigen Werte dieses Cashflows, diskontiert auf den Zeitpunkt des Projektionsbeginns (31.12.2018) betrachtet, schneidet das k-Means Verfahren viel besser ab. Die Abweichung des optimierten Cashflows vom tatsächlichen beträgt nur 0.3% und erfüllt somit das Gütekriterium, da der Schwellenwert von 1% nicht überschritten wurde. Das liegt daran, dass die zu hohen Werte in den drei Jahren, in denen die Erlebensleistungen der repräsentativen Policen fällig sind, die zu niedrigen Werte gleich 0 in den restlichen Jahren kompensieren.

Bei Anwendung der Methode mithilfe linearer Regression sind zwar jene zwei Werte, die nicht 0 betragen, bessere Annäherungen an den tatsächlichen Cashflow im jeweiligen Jahr als im Falle der Anwendung vom k-Means Algorithmus. Jedoch liegen sie mit einer Abweichung von 105.7% im Jahr 2039 und 86.5% im Jahr 2042 weit über dem Schwellenwert von 1% und auch die Summe der auf den Zeitpunkt der Projektion diskontierten Werte weicht mit $95,7\%$ sehr stark von dem Ergebnis des gesamten Versicherungsbestandes ab.

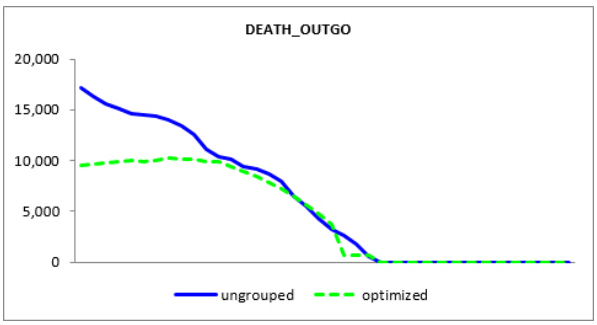
Im Falle der *Stornoleistungen*, Abbildung 11, schneidet der k-Means Algorithmus in diesem Beispiel besser ab als der Non-Negative Least Squares Algorithmus, der abgesehen von den letzten Jahren immer weit unter den Werten des Versicherungsbestandes liegt. Da die längste Vertragsdauer der Repräsentanten, welche mithilfe der k-Means Methode gefunden wurden, 20 Jahre beträgt und die maximale Dauer jedoch bei 25 Jahren liegt, nimmt der Cashflow schon ab dem Jahr 2038 den Wert 0 an, obwohl dies erst ab 2043 geschehen sollte.

Wenn nun auch die *Todesfallleistungen*, Abbildung 10, betrachtet werden, ist es schwer zu sagen, welche der beiden Methoden bessere Ergebnisse liefert. Beide Algorithmen liefern keine gute Approximation an das gesamte Versicherungsportfolio. Auch hier ist in der Approximation durch den k-Means Algorithmus der Wert des Cashflows ab dem Jahr 2038 gleich 0, da die Vertragsdauer aller Repräsentanten spätestens zu diesem Zeitpunkt abgelaufen ist.

In weiterer Folge soll nun auch noch untersucht werden, wie sich ein Mortalitätsschockszenario, bei dem angenommen wird, dass sich die Mortalität der versicherten Personen verdoppelt, auf die Werte der Cashflows auswirkt. Dazu werden in den Abbildungen 13, 14 und 15 wieder die Cashflows von beiden Methoden gegenübergestellt, um sie gut miteinander vergleichen zu können.

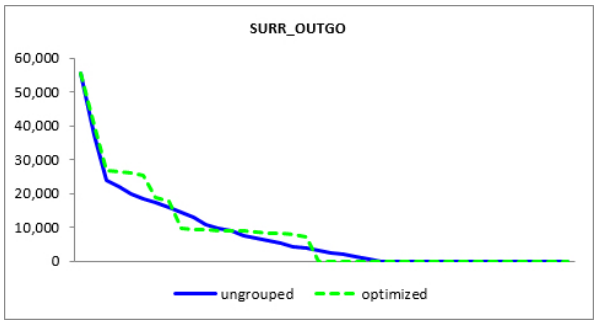


(a) Todesfallleistungen für die Approximation durch k-Means unter einem Mortalitätsschockszenario.

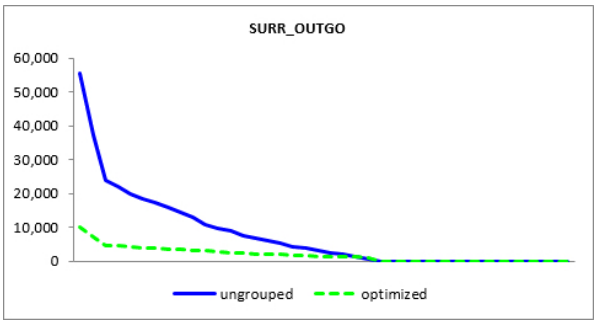


(b) Todesfallleistungen für die Approximation durch lineare Regression unter einem Mortalitätsschockszenario.

Abbildung 13: Vergleich der Ergebnisse der beiden Optimierungsmethoden für den Cashflow *Todesfallleistungen* unter einem Mortalitätsschockszenario. Die blaue Linie *ungrouped* sind die Werte der Cashflows für den gesamten Bestand und die grüne Linie *optimized* die der Repräsentanten.

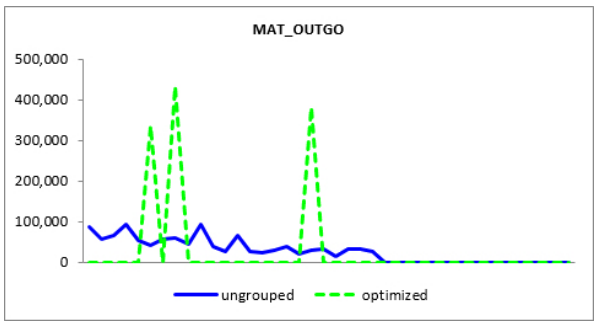


(a) Stornoleistungen für die Approximation durch k-Means unter einem Mortalitätsschockszenario.

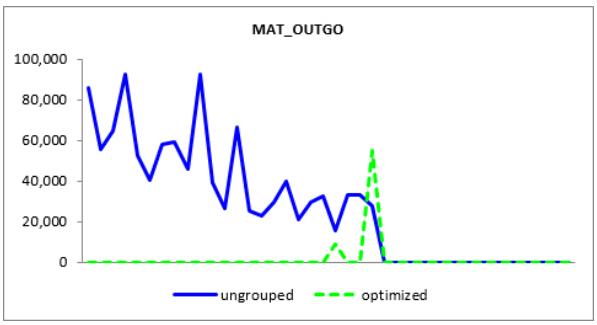


(b) Stornoleistungen für die Approximation durch lineare Regression unter einem Mortalitätsschockszenario.

Abbildung 14: Vergleich der Ergebnisse der beiden Optimierungsmethoden für den Cashflow *Stornoleistungen* unter einem Mortalitätsschockszenario. Die blaue Linie *ungrouped* sind die Werte der Cashflows für den gesamten Bestand und die grüne Linie *optimized* die der Repräsentanten.



(a) Leistungen zur Maturität für die Approximation durch k-Means unter einem Mortalitätsschockszenario.



(b) Leistungen zur Maturität für die Approximation durch lineare Regression unter einem Mortalitätsschockszenario.

Abbildung 15: Vergleich der Ergebnisse der beiden Optimierungsmethoden für den Cashflow *Leistungen zur Maturität* unter einem Mortalitätsschockszenario. Die blaue Linie *ungrouped* sind die Werte der Cashflows für den gesamten Bestand und die grüne Linie *optimized* die der Repräsentanten.

Auf den ersten Blick ist, abgesehen von den *Todesfallleistungen*, Abbildung 13, nicht wirklich ein Unterschied zu dem Szenario ohne Mortalitätsschock erkennbar. Durch die erhöhte Sterblichkeit der versicherten Personen erhöht sich der Cashflow *Todesfallleistungen* für den gesamten Bestand in allen Projektionsjahren auf etwa den doppelten Wert, wie auch unter Anwendung des k-Means Algorithmus. Unter Verwendung der Non-Negative Least Squares Methode flacht die Cashflowkurve (grün) im Gegensatz zu dem zuvor betrachteten Szenario ab und nähert sich in den letzten Jahren den Werten des gesamten Versicherungsportfolios (blau) an.

Die anderen beiden Cashflows *Stornoleistungen*, Abbildung 14, und *Leistungen zur Maturität*, Abbildung 15, nehmen niedrigere Werte im Mortalitätsschockszenario an. Diese Beobachtung ist sinnvoll, da bei erhöhter Sterblichkeit die Wahrscheinlichkeit, dass es zu einer Auszahlung im Erlebensfall kommt, geringer wird. Ansonsten sind die selben Eigenschaften wie im zuvor betrachteten Szenario für die *Leistungen zur Maturität* zu beobachten. Die Abweichung der Summe der auf den Projektionsbeginn diskontierten Cashflows, erfüllt unter Anwendung des k-Means Algorithmus auch in dem Schockszenario mit 0.5% das Gütekriterium. Im Fall der bisher verwendeten Methode ergibt sich eine Abweichung von 97.1%, womit der Schwellenwert von 1% wieder weit überschritten wird.

Auch für die *Stornoleistungen* sind geringere Werte unter erhöhter Sterblichkeit nicht verwunderlich, da die Stornowahrscheinlichkeit von der Sterbewahrscheinlichkeit abhängig ist. Einer bereits verstorbenen Person ist es nicht mehr möglich, den Versicherungsvertrag zu stornieren. Hier sind ansonsten ebenfalls die selben Eigenschaften wie im Szenario ohne Mortalitätsschock erkennbar.

5.6 Testen des Algorithmus an einem tatsächlichen Portfolio

Der letzte zum Testen des Algorithmus verwendete Bestand ist ein reales Versicherungsportfolio, aus welchem auch für die vorigen Beispiele jeweils eine Police ausgewählt wurde, um diese zu vervielfältigen und nur einzelne Merkmale zu verändern. Dabei handelt es sich um ein rumänisches Portfolio an gemischten Versicherungen, welches 4782 Policen

enthält. Die Währung ist der rumänische Leu (RON), das Alter zu Beginn der Versicherung kann zwischen 18 und 65 Jahren betragen und die Laufzeit des Versicherungsvertrags bewegt sich zwischen 5 und 10 Jahren. Die Beträge der Versicherungssummen nehmen einen minimalen und maximalen Wert von 5160.2 RON beziehungsweise 1113693.64 RON an.

Da die Policen nicht alle zum gleichen Zeitpunkt in Kraft treten, was in den zuvor verwendeten Testbeständen immer der Fall war, sollen nun nicht mehr die Merkmale *Alter bei Abschluss des Vertrages* und *Vertragslaufzeit* verwendet werden, sondern stattdessen das aktuelle Alter der versicherten Person zum Zeitpunkt der Projektion der Cashflows (31.12.2018) und die restliche Vertragslaufzeit, die zu diesem Zeitpunkt noch vorhanden ist. Diese beiden Werte können mit der Modellierungssoftware Prophet berechnet werden.

Zur visuellen Darstellung des Datenbestandes wird ein weiteres Mal die Hauptkomponentenanalyse verwendet. Für dieses Versicherungsportfolio wird die Grafik ohne einer Clusterzuordnung erstellt und kann in Abbildung 16 betrachtet werden.

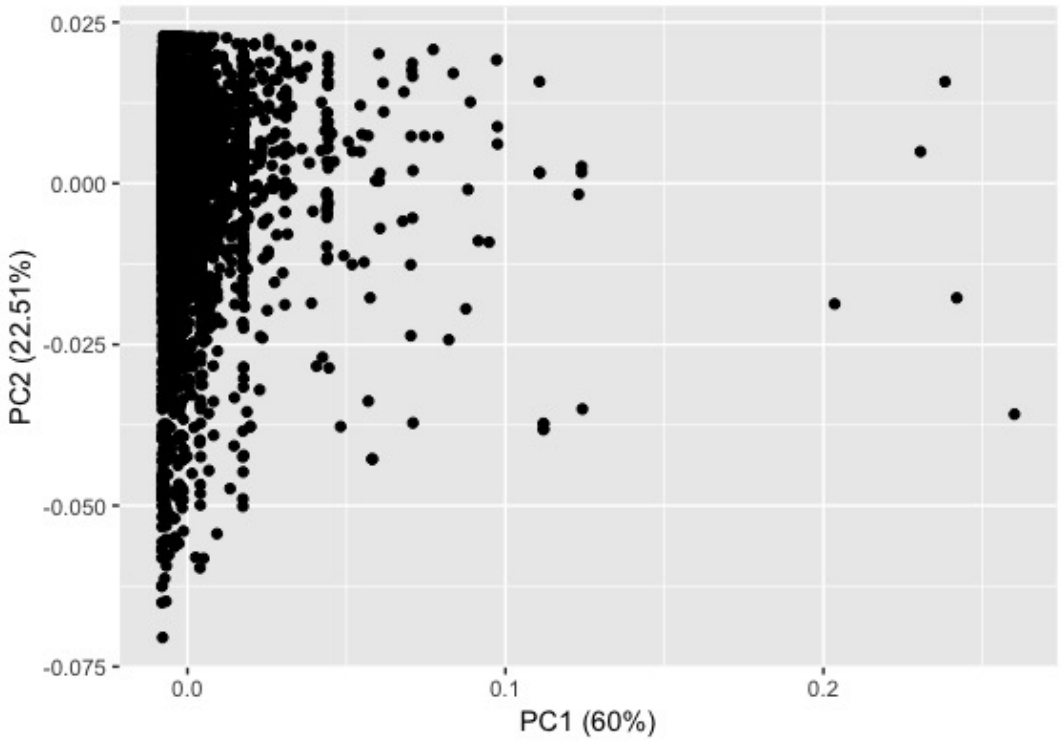


Abbildung 16: Grafische Darstellung des realen Versicherungsportfolios unter Verwendung der Hauptkomponentenanalyse. Mithilfe des Programmes R erstellt.

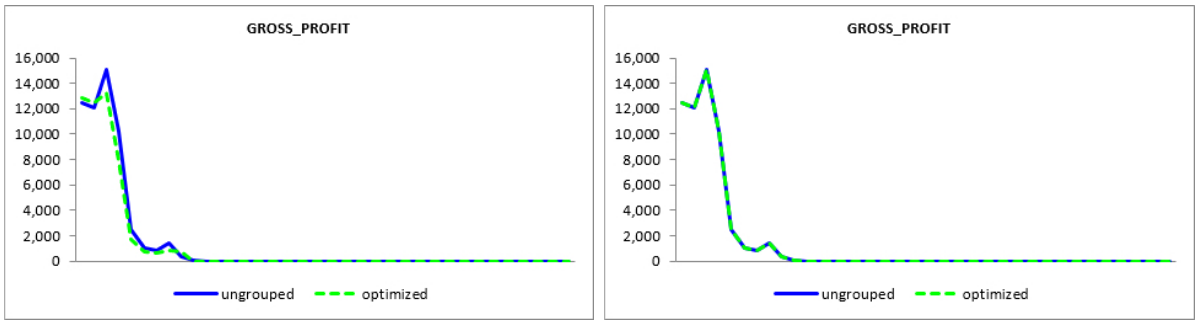
Es ist deutlich zu sehen, dass dieser Datenbestand komplizierter ist als jene, die in den vorherigen Beispielen betrachtet wurden. Hier sind nicht nur ein beziehungsweise zwei Merkmale, so wie zuvor, variiert worden und deshalb können die zwei Dimensionen der Grafik, die durch die Hauptkomponentenanalyse erstellt wurde, nur 82.51% der Varianz des Bestandes erklären. Für die Portfolios, die bisher betrachtet wurden, konnte mithilfe der Hauptkomponentenanalyse 99.8% beziehungsweise 100% der Varianz erklärt werden.

Das reale Versicherungsportfolio ist in 100 Cluster unterteilt worden, welchen zwischen 5 und 92 Modellpunkte zugeordnet wurden. Diese Anzahl wiederum, entspricht jeweils

dem Gewicht, welches dem repräsentativen Vertrag, der für das jeweilige Cluster gewählt wurde, zugeschrieben wird.

In weiterer Folge sind wieder die Cashflows für den repräsentativen Teilbestand und für das gesamte Versicherungsportfolio kalkuliert worden, um die erhaltenen Ergebnisse miteinander zu vergleichen. Insbesondere soll nun der Cashflow *Bruttogewinn* genauer betrachtet werden. Ein weiteres Mal wird auch die Methode, die mithilfe von linearer Regression arbeitet, auf den betrachteten Versicherungsbestand angewandt, um daraufhin die beiden Algorithmen miteinander vergleichen zu können.

In Abbildung 17 ist der Verlauf des Cashflows *Bruttogewinn* grafisch dargestellt.



(a) Bruttogewinn für die Approximation durch k-Means.

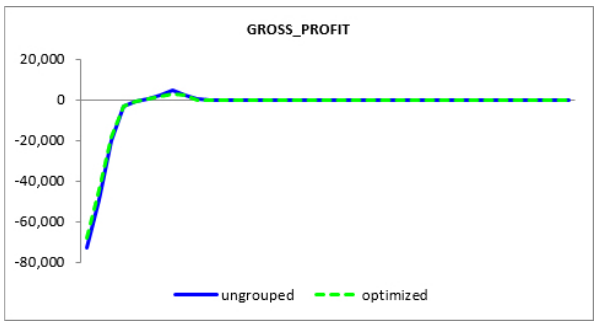
(b) Bruttogewinn für die Approximation durch lineare Regression.

Abbildung 17: Vergleich der Ergebnisse der beiden Optimierungsmethoden für den Cashflow *Bruttogewinn*. Die blaue Linie *ungrouped* sind die Werte der Cashflows für den gesamten Bestand und die grüne Linie *optimized* die der Repräsentanten.

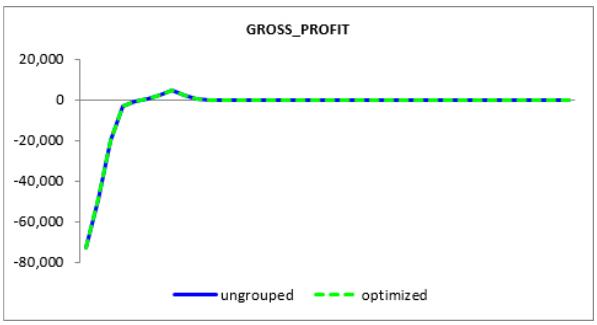
Der Cashflow, der mit den Repräsentanten, die mithilfe der bisher verwendeten Methode gewählt wurden, berechnet wird, stimmt perfekt mit dem von dem gesamten Versicherungsbestand erzeugten überein. Obwohl der von dem k-Means Algorithmus erhaltene repräsentative Bestand auch eine recht gute Annäherung darstellt, liefert er ein schlechteres Ergebnis als unter Verwendung der linearen Regression. Abgesehen von den ersten zwei Jahren und dem Jahr 2027, sind die Werte des optimierten Cashflows geringer als ihre tatsächliche Höhe.

Da unter Anwendung der Non-Negative Least Squares Methode, wie in Kapitel (5.3) erläutert, die Cashflows des Versicherungsportfolios verwendet werden, um passende Policen für den repräsentativen Bestand auszuwählen, und daher mehr Informationen benutzt werden als bei Verwendung des k-Means Algorithmus, ist es nicht verwunderlich, dass diese Methode eine bessere Lösung liefert.

Des Weiteren soll auch für das reale Versicherungsportfolio ein Mortalitätsschockszenario betrachtet werden. Es wird ein weiteres Mal angenommen, dass sich die Sterblichkeit der versicherten Personen verdoppelt. Die erhaltenen Werte des Cashflows *Bruttogewinn* werden in Abbildung 18 dargestellt, um auch diese miteinander vergleichen zu können.



(a) Bruttogewinn für die Approximation durch k-Means unter einem Mortalitätsschockszenario.



(b) Bruttogewinn für die Approximation durch lineare Regression unter einem Mortalitätsschockszenario.

Abbildung 18: Vergleich der Ergebnisse der beiden Optimierungsmethoden für den Cashflow *Bruttogewinn* unter einem Mortalitätsschockszenario. Die blaue Linie *ungrouped* sind die Werte der Cashflows für den gesamten Bestand und die grüne Linie *optimized* die der Repräsentanten.

Obwohl nicht in allen Jahren das Gütekriterium, dass die Abweichung nicht mehr als 1% des tatsächlichen Wertes betragen darf, erfüllt wird, liefern beide Methoden eine sehr gute Annäherung an die tatsächlichen Werte des Cashflows. Insbesondere ist bemerkenswert, dass der k-Means Algorithmus in dem Schockszenario eine bessere Performance liefert als zuvor.

Trotz der Tatsache, dass die Anwendung von linearer Regression zur Auswahl von Repräsentanten Annahmen zweiter Ordnung verwendet, im Speziellen die Mortalität der versicherten Personen, die zur Kalkulation dieser Cashflows stark verändert wurde, passen die Werte des dadurch erhaltenen Cashflows, entgegen dem Erwarteten, sehr gut.

6 Fazit

Das Ziel dieser Arbeit ist zu untersuchen, ob mithilfe des Clusterverfahrens, dem k-Means Algorithmus, eine repräsentative Teilmenge eines Versicherungsbestandes gefunden werden kann, um die Berechnung der Cashflows zu optimieren. Speziell ist es wünschenswert, die Auswahl solcher Repräsentanten unabhängig von Annahmen zweiter Ordnung zu gestalten, was durch dieses Verfahren gewährleistet wird.

Da die bisher verwendete Methode unter Anwendung des Non-Negative Least Squares Algorithmus eben solche Annahmen zur Auswahl von repräsentativen Verträgen benutzt, ist eben diese speziell auf das zugrundeliegende Szenario, welches durch die Annahmen zweiter Ordnung bestimmt wird, zugeschnitten und muss deshalb nicht unbedingt auch unter anderen Voraussetzungen eine passende Auswahl darstellen.

Der Vorteil des k-Means Algorithmus ist, dass nur die Merkmale der Police benutzt werden, um repräsentative Verträge aus dem Versicherungsportfolio auszuwählen. Die Methode, die mithilfe linearer Regression arbeitet, verwendet diese Informationen nicht wirklich, da nur die Cashflows betrachtet werden, in denen nur indirekt durch ihre Berechnung diese Merkmale enthalten sind.

Deshalb wurden die Auswirkungen eines Mortalitätsschocks auf die Cashflows untersucht und die Ergebnisse des neu erstellten Verfahrens mit jenen der bisher angewandten Methode verglichen. Dabei wurden die Repräsentanten nicht erneut unter Berücksichtigung der erhöhten Sterblichkeit berechnet, sondern die bereits zuvor erhaltenen verwendet.

Das bedeutet, dass der Non-Negative Least Squares Algorithmus unter Anwendung Annahmen zweiter Ordnung, die sich als falsch herausgestellt haben, repräsentative Verträge ausgewählt hat. Der k-Means Algorithmus hingegen verwendet nur Informationen, die immer gleich bleiben und sich nicht in verschiedenen Szenarien ändern können, nämlich die Merkmale des Versicherungsvertrages. Dies ist der große Vorteil der Anwendung dieses Clusterverfahrens zur Auswahl von repräsentativen Policen.

Beispielsweise bei der Variation des Alters konnte gut erkannt werden, dass das Ergebnis der bisherigen Methode nicht ideal ist und in einem Schockszenario den Bestand nicht mehr ausreichend gut repräsentieren kann. Da das Merkmal *Alter der versicherten Person* stark mit der Sterblichkeit in Verbindung steht, treten in dem Beispiel, in dem genau diese Variable variiert wurde, stärkere Abweichungen in einem Mortalitätsschockszenario für die bisher verwendete Methode auf, die unter anderen Sterblichkeitsannahmen eine Auswahl an repräsentativen Verträgen getätigt hat.

Leider liefert der neu entwickelte Algorithmus im Grundszenario noch keine sehr guten Ergebnisse und er müsste weiter bearbeitet werden, um eine bessere Anpassung an den gesamten Versicherungsbestand liefern zu können. Trotz dessen sieht die Anwendung von Clusterverfahren zur Optimierung von Cashflowberechnungen sehr vielversprechend aus. Obwohl bei dem Beispiel mit dem tatsächlichen Versicherungsportfolio die Auswahl der Repräsentanten, die von dem k-Means Algorithmus erhalten wurden, erst nicht so gute Ergebnisse liefert, sieht die Annäherung des Cashflows im Mortalitätsschockszenario sehr gut aus.

Zusammenfassend scheint der in dieser Arbeit erstellte Algorithmus noch nicht ganz ausgereift zu sein, jedoch ist die Anwendung des k-Means Verfahrens zur Auswahl repräsentativer Versicherungsverträge, um die Berechnung der Cashflows zu optimieren, sehr vielversprechend.

7 Anhang

Nachfolgend sind die Programme, die in der Statistiksoftware R erstellt wurden, aufgelistet.

```
#used packages:
library("NbClust", lib.loc="/Library/Frameworks/R.framework/
Versions/3.5/Resources/library")
library("factoextra", lib.loc="/Library/Frameworks/R.framework/
Versions/3.5/Resources/library")
library("cluster", lib.loc="/Library/Frameworks/R.framework/
Versions/3.5/Resources/library")
library("psych", lib.loc="/Library/Frameworks/R.framework/
Versions/3.5/Resources/library")
library("RANN", lib.loc="/Library/Frameworks/R.framework/
Versions/3.5/Resources/library")
```

```
Repr <- function(X, kmin, kmax, categ){

        #transformed and unscaled data:
        TransData_us <- PrepData(subset(X, select =
                - POLICYNUMBER), categ)

        #scale data:
        sigma = SD(TransData_us)
        sigma[sigma==0] = 1
        muh = colMeans(TransData_us)

        TransData <- as.data.frame(scale(TransData_us,
                muh, sigma))

        if(kmin < kmax){
                optnr <- OptNum(TransData, kmin, kmax)
                } else{
                optnr = kmin
                }

        InitCent <- JumpDecr(TransData, optnr,
                max(floor(optnr*1.5), optnr+5))
        #InitCent <- JumpDecr(TransData, optnr, optnr+15)
        #for big datasets like the real portfolio

        ClustData <- kmeans(TransData, InitCent$centers, 50)
        CentMP <- X[nn2(TransData, ClustData$centers,1)$nn.idx,]

        if(any(colnames(X) == "INIT_POLS_IF")){
```

```
            CentMP <- InitPols(X, CentMP, ClustData$cluster)
            #sum of all values of INIT_POLS_IF of the
                modelpoints within this cluster
            }

    return(list(centers = CentMP, size = ClustData$size,
            cluster = ClustData$cluster))
    }
```

```
PrepData <- function(X, categ){

    m = as.integer(length(categ))
    nr = as.integer(nrow(X))

    for(i in 1:m){
            val.i = sort(unique(as.matrix(X)[,categ[i]]))
            lv = as.integer(length(val.i))

            #make for each category a dummy variable with
                yes=1 and no=0:
            if(lv > 2 | (lv == 2 & (val.i[1] != 0 |
                val.i[2] != 1))){
                dummat = matrix(0, ncol=lv, nrow=nr)

                for(j in 1:lv){
                    dummat[X[,categ[i]] == val.i[j],
                        j] = 1
                    }

                #original name of the modelpoint plus
                    the different category values:
                categ_names = paste(names(X)[categ[i]],
                    val.i[1:lv], sep = "..")
                dumfr = data.frame(dummat)
                names(dumfr) = categ_names

                #data matrix including dummy variables,
                    without original categorical
                    variable:
                X = cbind(X[-categ[i]], dumfr)

                #new places of columns with categorical
                    data categ[j], after removing
                    the column:
                categ[i+1:m] = categ[i+1:m]-1
```

```
                }
        }

    #remove columns with only one value as entry:
    uniquelength <- sapply(X, function(x) length(unique(x)))
    X <- subset(X, select=uniquelength > 1)

    for(i in 1:ncol(X)){
        X[,i] = as.numeric(unlist(X[,i]))
        }

    return(X)
    }
```

```
OptNum <- function(X, kmin, kmax){

    selected <- c("kl", "ch", "hartigan", "cindex", "db",
        "silhouette", "ball", "ptbiserial", "frey",
        "mcclain", "dunn", "sdindex", "sdbw")
    optnr <- numeric(length(selected))

    for (i in 1:length(selected)) {
        optnr[i] <- NbClust(X, min.nc = kmin,
            max.nc = kmax, method = "kmeans",
            index = selected[i])$Best.nc[[1]]
        }

    optnr <- as.numeric(names(which.max(table(optnr))))

    return(optnr)
    }
```

```
JumpDecr <- function(X,k,m){

    nr = as.integer(nrow(X))
    k = as.integer(k)
    m = as.integer(m)

    it = 0              #number of iterations
    s = numeric(nr)

    t = floor(nr/m)/2
    t = floor(seq(1, m*2-1, by = 2)*t)
    center = X[t,]          #reproducible initial centers
```

```
while(m > k){
        #assign modelpoints to clusters:
        for(i in 1:nr){
                dist.i = numeric(m)

                for(j in 1:m){
                        dist.i[j] = norm(as.matrix(
                                center[j,]-X[i,]),
                                type = "F")
                        }

                #number of clustercenter with least
                        distance to x_i:
                s[i] = which.min(dist.i)
                }

        #calculate new cluster centers:
        for(j in 1:m){
                center[j,] = colMeans(as.matrix(
                        X[s == j,]))
                }

        h = numeric(m)

        #number of modelpoints in cluster j:
        for(j in 1:m){
                h[j] = length(s[s == j])
                }

        if(it < m-k){

                #jumping process:
                jmin = which.min(h)
                jmax = which.max(h)
                Y = X[s == jmax,]
                center[jmin,] = Y[sample(h[jmax],1),]
                it = it+1
                } else {

                #decreasing process:
                jmin = which.min(h)
                center = center[-jmin,]
                m = m-1
                }
        }
```

```
        return(list(centers = center))
        }
```

```
InitPols <- function(X, centers, cluster){

        for(i in 1:nrow(centers)){
                centers[i, "INIT_POLS_IF"] = sum(X[cluster == i,
                        "INIT_POLS_IF"])
                }

        return(centers)
        }
```

Literatur

[BCK08] S. Boriah, V. Chandola, and V. Kumar. *Similarity Measures for Categorical Data: A Comparative Evaluation*, 2008. Conference Paper: Proceedings of the SIAM International Conference on Data Mining.

[Bjö96] Å. Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics (SIAM), 1996.

[CLAS10] M. Charrad, Y. Lechevallier, M. B. Ahmed, and G. Saporta. On the number of clusters in block clustering algorithms. In *FLAIRS Conference*, 2010.

[Del16] Deloitte GmbH Wirtschaftsprüfungsgesellschaft. *Non Negative Least Square Algorithmus*, 2016. Power Point Präsentation.

[Del17] Deloitte GmbH Wirtschaftsprüfungsgesellschaft. *Clustering mit dem Deloitte Grouping Optimizer*, 2017. Power Point Präsentation.

[Doo94] J. L. Doob. *Measure Theory*. Springer-Verlag New York, 1994.

[For65] E. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–780, 1965.

[Heu84] H. Heuser. *Lehrbuch der Analysis. Teil 1.* B. G. Teubner Verlag Stuttgart, 1984.

[HH81] K. H. Haskell and R. J. Hanson. An algorithm for linear least squares problems with equality and nonnegativity constraints. *Mathematical Programming*, 21:98 – 118, 1981.

[HVB00] M. Halkidi, M. Vazirgiannis, and Y. Batistakis. Quality scheme assessment in the clustering process. In D.A. Zighed, J. Komorowski, and J. Żytkow, editors, *Principles of Data Mining and Knowledge Discovery. PKDD 2000*, volume 1910 of *Lecture Notes in Computer Science*, pages 265–276. Springer, Berlin, Heidelberg, 2000.

[Jai10] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31:651 – 666, June 2010.

[LJB06] C. Legány, S. Juhász, and A. Babos. Cluster validity measurement techniques. In *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, AIKED'06, pages 388–393, Stevens Point, Wisconsin, USA, 2006. World Scientific and Engineering Academy and Society (WSEAS).

[Llo82] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[Mir96] B. Mirkin. *Mathematical Classification and Clustering*. Kluwer Academic Publishers, 1996.

[Pea01] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.

[PG11] S. Pandit and S. Gupta. A comparative study on distance measuring approaches for clustering. *International Journal of Research in Computer Science*, 2:29 – 31, December 2011.

[Rou87] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987.

[Ste56] H. Steinhaus. Sur la division des corp materiels en parties. *Bulletin de l'Académie Polonaise des Sciences*, 4:801–804, 1956.

[Ste99] G. S. Steiner. *Statistical Data Compression by Optimal Segmentation - Theory, Algorithms and Experimental Results*. PhD thesis, Wirtschaftsuniversität Wien, September 1999.

[XW09] R. Xu and D. C. Wunsch II. *Clustering*. Wiley - IEEE Press, 2009.