UNIVERSITÄT SIEGEN

BACHELORARBEIT
IM FACH INFORMATIK

Kodierungsstrategien von boole'schen Konfigurationsräumen für Ansätze zum Feature-Modell-Lernen

vorgelegt von

El Mehdi RAHALI

Betreuer:

Prof. Dr. M. LOCHAU,
Professur für Modellbasierte Entwicklung,
Universität Siegen

M. Sc. M. WEISS,
Professur für Modellbasierte Entwicklung,
Universität Siegen

Beginn der Arbeit: 16.10.2023
Ende der Arbeit: 26.02.2024

Eidesstattliche Erklärung

Ich versichere nach § 39 (1) der Einheitlichen Regelungen, meine Arbeit (bei einer Gruppenarbeit meinen entsprechend gekennzeichneten Anteil der Arbeit) selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht zu haben.

Ort, Datum

Unterschrift

UNIVERSITÄT SIEGEN

Zusammenfassung

Kodierungsstrategien von boole'schen Konfigurationsräumen für Ansätze zum Feature-Modell-Lernen

von El Mehdi RAHALI

In der SPL-Entwicklung werden Feature-Modelle eingesetzt, um eine Vielzahl von ähnlichen, aber unterschiedlichen Produktvarianten in einem konfigurierbaren Softwaresystem zu modellieren. Feature-Modelle werden in der Praxis jedoch meistens nicht manuell von Grund auf neu erstellt, sondern durch Ansätze wie das Feature-Modell-Lernen (FML) rekonstruiert. FML verwendet maschinelle Lernalgorithmen, um aus einer bereits implementierten Menge von Produktkonfigurationen automatisch ein Feature-Modell zu lernen.

Ein kritisches Problem, das jedoch bei maschinellen Lernalgorithmen auftritt, ist der sogenannte Curse-of-Dimensionality. Mit jedem hinzugefügten Feature (Dimension) wächst die Größe des Konfigurationsraums exponentiell, was zur Folge hat, dass die vorhandenen Datenpunkte im Vergleich zur Gesamtgröße des Raums zunehmend spärlicher verteilt sind. Diese spärlichere Verteilung der Daten führt dazu, dass es für maschinelle Lernalgorithmen schwieriger wird, signifikante Muster aus dem Datensatz zu erkennen und effektive Klassifikationsmodelle zu erstellen. Um diese Herausforderung zu adressieren, untersuchen wir in dieser Arbeit den Einsatz von Dimensionality-Reduction (DR). Im Rahmen der DR-Technik wenden wir automatisierte Kodierungsstrategien an, die den ursprünglich hochdimensionalen Datensatz in eine niedriger dimensionierte Repräsentation transformieren. Insbesondere konzentrieren wir uns auf zwei spezifische Methoden: Linear-Principal-Component-Analysis (Linear PCA) und Logistic-Principal-Component-Analysis (Logistic PCA). Bei der Linear PCA erreichen wir die Reduktion, indem wir die Daten auf die Hauptachsen (Principal-Components) projizieren, die den größten Teil der Informationen des Datensatzes erfassen. Die Logistic PCA passt das Prinzip der Linear PCA an, um auch mit der booleschen Natur der Konfigurationsräume umzugehen.

Durch experimentelle Evaluierung auf drei verschiedenen Fallstudien vergleichen wir die Effektivität (Präzision und Recall) und Effizienz (Rechenaufwand) von FML sowohl mit als auch ohne den Einsatz von Linear und Logistic PCA. Wir zeigen, dass die Präzision für FML ohne Dimensionality-Reduction nur leicht effektiver ist. Der Recall weist allerdings deutlich höhere Werte im Fall von FML mit Einsatz von DR auf. Der Rechenaufwand von FML verbessert sich durch die Anwendung von PCA in seinen verschiedenen Varianten im Durchschnitt um bis zu 9 % im Vergleich zu FML ohne den Einsatz von PCA. Die Ergebnisse verdeutlichen den Trade-off zwischen Effektivität und Effizienz beim Einsatz von Dimensionality-Reduction im Kontext des FMLs.

UNIVERSITY OF SIEGEN

Abstract

Encoding Strategies of Boolean Configuration Spaces for Approaches for Feature Model Learning

by El Mehdi RAHALI

In SPL development, feature models are used to model a large number of similar but different product variants in a configurable software system. In practice, however, feature models are usually not created manually from scratch, but are reconstructed using approaches such as feature model learning (FML). FML uses machine learning algorithms to automatically learn a feature model from an already implemented set of product configurations.

However, a critical problem that occurs with machine learning algorithms is the so-called curse of dimensionality. With each added feature (dimension), the size of the configuration space grows exponentially, which means that the existing data points are increasingly sparsely distributed compared to the total size of the space. This sparser distribution of data means that it becomes more difficult for machine learning algorithms to recognize significant patterns from the dataset and create effective classification models. To address this challenge, we investigate the use of dimensionality reduction (DR) in this work. As part of the DR technique, we apply automated coding strategies that transform the original high-dimensional dataset into a lower dimensional representation. In particular, we focus on two specific methods: Linear Principal Component Analysis (Linear PCA) and Logistic Principal Component Analysis (Logistic PCA). In Linear PCA, we achieve the reduction by projecting the data onto the principal components, which capture most of the information in the data set. Logistic PCA adapts the principle of Linear PCA to also deal with the Boolean nature of the configuration spaces.

Through experimental evaluation on three different case studies, we compare the effectiveness (precision and recall) and efficiency (computational effort) of FML both with and without the use of Linear and Logistic PCA. We show that the precision for FML without dimensionality reduction is only slightly more effective. However, the recall shows significantly higher values in the case of FML with the use of DR. The computational effort of FML improves on average by up to 9 % by applying PCA in its different variants compared to FML without the use of PCA. The results illustrate the trade-off between effectiveness and efficiency when using dimensionality reduction in the context of FML.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

SPL	**S**oftware-**P**roduktlinie
FM	**F**eature-**M**odel
FML	**F**eature-**M**odel-**L**ernen
ML	**M**aschinelles-**L**ernen
(Auto)ML	(**A**utomatic)-**M**achine-**L**earning
DR	**D**imensionality-**R**eduction
PCA	**P**rincipal-**C**omponent-**A**nalysis
PC	**P**rincipal-**C**omponent
BCS	**B**ody-**C**omfort-**S**ystem
SVM	**S**upport-**V**ector-**M**achine

Kapitel 1

Einführung

1.1 Hintergrund und Motivation

Bei der heutigen, sich ständig weiterentwickelnden Welt der Softwareentwicklung stellt die Erfüllung der unterschiedlichen Bedürfnisse und Vorlieben der Benutzer eine zentrale Herausforderung dar. Traditionelle Ansätze zur Softwareentwicklung, bei denen manuell separate Versionen der Software für verschiedene Benutzeranforderungen erstellt werden, sind oft nicht mehr praktikabel [6]. Hier kommt das Konzept der *Software-Produktlinien* (SPL) [14] als Lösung ins Spiel. Anstatt individuelle Softwarevarianten von Grund auf zu erstellen, konzentrieren wir bei der SPL-Entwicklung darauf, einen flexiblen Rahmen aus wiederverwendbaren Softwarekomponenten zu schaffen. Diese Komponenten sind so gestaltet, dass sie viele von Benutzerbedürfnissen, Produktkonfigurationen zu unterstützen. Ein Feature-Modell [18] ist eine abstrakte, strukturierte Darstellung, die im Kontext von SPL verwendet wird, um den gültigen Konfigurationsraum einer Software darzustellen. Diese Darstellung umfasst sämtliche Features, die in einer Software-Familie verfügbar sind, und spezifiziert, wie diese Features miteinander interagieren und in verschiedenen Produkten der SPL kombiniert werden können [14]. In der Praxis wird ein Feature-Modell durch ein Feature-Diagramm repräsentiert. Ein Feature-Diagramm zeigt die hierarchischen Beziehungen zwischen Features und deren mögliche Kombinationen. Dies ermöglicht eine leicht verständliche Visualisierung des Konfigurationsraums.

1.2 Problemstellung

Feature Modelle werden in der Regel nicht von Grund auf neu erstellt. Stattdessen werden Rekonstruktionsansätze verwendet, um aus einer Teilmenge von bekannten bzw. implementierten Produktvarianten ein Feature-Modell zu rekonstruieren, das den gesamten Konfigurationsraum für eine SPL formell beschreibt. Ein möglicher Rekonstruktionsansatz ist das sogenannte *Feature-Modell-Lernen* (FML). Beim FML kommen maschinelle Lernalgorithmen zum Einsatz, um den booleschen Konfigurationsraum zu lernen.

Das Anwenden von *maschinellem Lernen* (ML) erfordert jedoch die Kodierung des booleschen Konfigurationsraums in einem tabellarischen Datenformat. Aufgrund der hohen Anzahl der Features bei komplexen Systemen, wie beispielsweise im Fall des Linux-Kernels mit mehr als 15000 potenziellen Features [7], gestaltet sich der FML-Prozess äußerst anspruchsvoll.

Um das Problem zu verdeutlichen, betrachten wir ein Beispiel, bei dem ein Feature-Modell für eine Menge F von drei Features erstellt wird:

$$F = \{\text{Getriebe}, \text{Manuell}, \text{Automatisch}\}$$

Der gültige Konfigurationsraum wird als $C_{gültig}$ bezeichnet und ist eine Teilmenge des gesamten Konfigurationsraums 2^F:

$$C_{gültig} = \{\{Getriebe, Manuell\}, \{Getriebe, Automatisch\}\} \subseteq 2^F$$

Das Feature-Diagramm FM_1 in Abb. 1.1, was in FODA-Notation [12] dargestellt ist, modelliert genau die Menge der gültigen Konfigurationen $C_{gültig}$. Die naheliegendste Kodierung

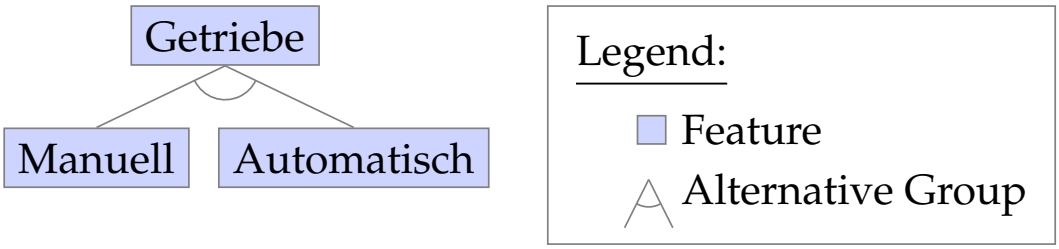


ABBILDUNG 1.1: Feature-Diagramm FM_1

Auto	Lenkslenker	Rechtslenker	Gültigkeit
0	0	0	ungültig
0	0	1	ungültig
0	1	0	ungültig
0	1	1	ungültig
1	0	0	ungültig
1	0	1	gültig
1	1	0	gültig
1	1	1	ungültig

TABELLE 1.1: Kodierung des booleschen Konfigurationsraums

des booleschen Konfigurationsraums ist die Kodierung jeder Konfiguration als einen booleschen Vektor, wobei für jedes Feature (Getriebe, Manuell, Automatisch) angegeben wird, ob es ausgewählt (z.B. 1) oder nicht ausgewählt (z.B. 0) ist. Die Tabelle 1.1 zeigt, wie diese Kodierung aussieht. Mit diesem Ansatz führt die hohe Anzahl von Features zu einer hohen Dimensionalität des resultierenden Datensatzes. Dies bedeutet, dass der resultierende Datensatz, der diese Features beschreibt, viele Dimensionen oder Spalten aufweist. In hochdimensionalen Räumen sind maschinelle Lernalgorithmen weniger effizient und können anfällig für den sogenannten *Curse-of-Dimensionality* [1] sein. Studien haben gezeigt, dass mit zunehmender Dimension des Datenraums eine erhöhte Daten-Spärlichkeit beobachtet wird. Diese Daten-Spärlichkeit stellt eine Herausforderung für maschinelle Lernalgorithmen dar, da sie diese daran hindert, aussagekräftige statistische Schlussfolgerungen aus dem Datensatz abzuleiten [1], was das Kernprinzip der meisten maschinellen Lernalgorithmen ist [15].

1.3 Lösungsansatz

In dieser Arbeit stellen wir einen Lösungsansatz vor, der die Herausforderungen des Curse-of-Dimensionality im Kontext von FML adressiert. Unser Ansatz konzentriert sich auf die Anwendung von zwei spezifischen Techniken zur Dimensionality-Reduction [21]: Die Linear Principal-Component-Analysis (Linear PCA) und die Logistic Principal-Component-Analysis (Logistic PCA). Diese Methoden dienen dazu, die Dimensionalität der Daten zu reduzieren und somit die Effektivität und Effizienz des maschinellen Lernprozesses zu verbessern. Die Linear PCA wird traditionell eingesetzt, um kontinuierliche Daten zu verarbeiten und die Hauptachsen (Principal-Components) zu identifizieren, die den größten Anteil an der Varianz im Datensatz tragen. Durch die Reduktion des Datenraums auf diese Principal-Components können wir das Problem von Curse-of-Dimensionality direkt adressieren, indem irrelevante oder redundante Informationen entfernt werden. Dies ermöglicht es, dass maschinelle Lernalgorithmen effizienter arbeiten können, da sie sich auf die wesentlichen Features in dem Datensatz konzentrieren [24]. In gleicher Weise adressiert die Logistic PCA speziell die Herausforderungen, die bei der Verarbeitung von booleschen Daten, wie diese in booleschen Konfigurationsräumen vorkommen, auftreten. Logistic PCA passt das Konzept der Linear PCA an, um mit der diskreten Natur dieser Daten umzugehen, indem sie eine logistische Funktion verwendet, um die Wahrscheinlichkeiten der binären Daten innerhalb des Datensatz zu modellieren. Dieser Ansatz ermöglicht es, die zugrundeliegenden Muster

und Beziehungen in booleschen Konfigurationsräumen effektiv zu erfassen und zu repräsentieren [3]. Wir vergleichen die Leistungsfähigkeit des Feature-Modell-Lernens in beiden Szenarien: einmal unter Einsatz von Techniken zur Dimensionality-Reduction mittels Linear und Logistic PCA und einmal ohne diese. Zudem analysieren wir, wie sich die Effektivität und Effizienz des Feature-Modell-Lernens durch die Reduktion der Dimensionalität in Abhängigkeit von der Größe der Feature-Sets des reduzierten Datensatzes verändert.

1.4 Zielsetzung

Im Rahmen dieser Arbeit verfolgen wir folgende Ziele:

- Wir wenden die beiden Algorithmen Linear PCA und Logistic PCA an, um die Dimensionalität des resultierenden Datensatzes zu reduzieren, ohne dabei wesentliche Informationen im Datensatzes zu verlieren.
- Wir evaluieren FML sowohl mit als auch ohne den Einsatz der beiden Dimensionality-Reduction-Techniken Linear und Logistic PCA. Die Vergleichsanalyse der beiden Ansätze fokussiert sich auf ihre Effektivität, bewertet anhand von Präzision und Recall, sowie ihre Effizienz, gemessen an der erforderlichen Rechenzeit.
- Abschließend diskutieren wir zukünftige Arbeiten, die auf der booleschen Kodierung von Konfigurationsräumen aufbauen. Dies umfasst das Entwerfen von maßgeschneiderte manuelle Kodierungsstrategie des booleschen Konfigurationsraums und die Implementierung von alternativer Rekonstruktionstechniken neben der PCA-Ansätze.

Kapitel 2

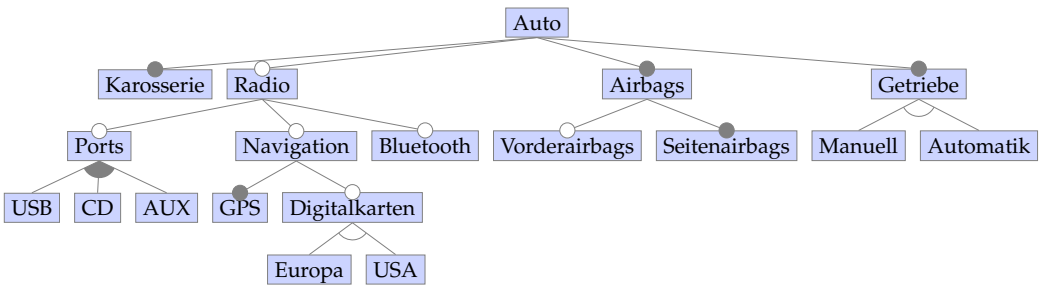
Grundlagen

In diesem Abschnitt erklären wir das Hintergrundwissen bezüglich des booleschen Konfigurationsraums und der Rekonstruktion von Feature-Modellen. In Abschnitt 2.1 führen wir ein anschauliches Beispiel für Feature-Modelle ein. Mit diesem Beispiel zeigen wir, wie Feature-Modelle in Form eines Feature-Diagramms festlegen, welche Auswahl von Features zu einer gültigen Konfiguration führt. In Abschnitt 2.2 erläutern wir die Technik der Rekonstruktion von Feature-Modelle durch Verwendung von Konzepten des maschinellen Lernens. Wir schließen dieses Kapitel ab, indem wir die Problemstellung der Implementierung und Evaluierung von Ansätzen zur Reduktion des booleschen Dimensionsraums darstellen.

2.1 Veranschaulichendes Beispiel

In diesem Abschnitt präsentieren wir ein Beispiel für ein Feature-Modell, das wir in unsere Arbeit verwenden. Das Beispiel veranschaulicht die grafische Repräsentation eines Feature-Modells in Form eines Feature-Diagramms. Auf Grundlage dieses Beispiels erläutern wir, wie Feature-Modelle verwendet werden, um den gültigen Konfigurationsraum der konfigurierbaren Softwareprodukten zu definieren.

Das Beispiel in Abb. 2.1 präsentiert ein Feature-Modell für die Konfiguration von *Software-Produktlinien* (SPL) im Bereich von Automobilsystemen. Das Modell ist in Form ein Feature-Diagramm in FODA-Notation [12] dargestellt und visualisiert die verschiedenen Funktionalitäten eines Autos. Das Feature-Diagramm zeigt eine Menge von eckigen blauen Knoten. Diese Knoten sind hierarchisch von oben nach unten angeordnet und bilden somit eine Baumstruktur. Im Kontext von SPL werden Knoten als *Features* bezeichnet. Ein zentrales Feature in diesem Modell ist das Feature *Auto*. *Auto* stellt das Root-Element in dem Diagramm dar und beinhaltet verschiedene untergeordnete Features wie *Karosserie*, *Airbags* und *Getriebe*. Um die Funktionalität eines Auto zu erweitern, präsentiert das Diagramm auch das Feature *Radio*. Das *Radio*-Feature stellt wiederum eine Komposition von weiteren Features dar, darunter *Ports*, *Navigation* und *Bluetooth*. Unser Feature-Diagramm besteht insgesamt aus 19 Features, diese Features können in verschiedenen Kombinationen ausgewählt oder



$$\texttt{Automatik} \Rightarrow \texttt{GPS}$$
$$\texttt{Karosserie} \Rightarrow \texttt{Automatik} \wedge \neg\, \texttt{Bluetooth}$$
$$\texttt{Vorderairbags} \Rightarrow \neg\, \texttt{CD}$$

ABBILDUNG 2.1: Feature-Modell eines Autos

nicht ausgewählt werden, wobei jede Kombination von Features als *Konfiguration* bezeichnet wird. Eine Konfiguration repräsentiert eine mögliche Autovariante. Beispielsweise könnte eine Konfiguration die Menge mit den Features *Auto*, *Karosserie*, *Airbags*, *Getriebe* und *Manuell* enthalten. Im Rest dieser Arbeit werden wir Konfigurationen in einer vektorbasierten Notation repräsentieren. In dieser Notation werden Konfigurationen als boolesche Vektoren dargestellt. Für eine Menge F mit n Features definieren wir ein boolescher Vektor durch $\vec{c_i} = (f_{i,1}, f_{i,2}, \ldots, f_{i,n})$. Jedes $f_{i,j}$ repräsentiert den Status eines bestimmten Features in einer Konfiguration. Die Werte $f_{i,j}$ sind boolesche Werte aus der Menge $\{0,1\}$, wobei 1 für *ausgewählt* und 0 für *nicht ausgewählt* steht. Die Menge aller Konfigurationen wird im Kontext von SPL als *Konfigurationsraum* bezeichnet. Ein Konfigurationsraum wird formal durch die Notation $\mathbb{B}^{|F|}$ dargestellt, wobei F die Menge der Features einer SPL und $\mathbb{B}$ die boolesche Menge {0, 1} ist. Insgesamt besteht einen Konfigurationsraum aus $2^{|F|}$ gültige und ungültige Konfigurationen. Die Zahl 2 kommt aus der Tatsache, dass wir jedes einzelne Feature entweder auswählen oder nicht auswählen können, und der Exponent $|F|$, weil diese Entscheidung $|F|$ Mal für jedes Feature getroffen wird. Somit besteht der Konfigurationsraum in unserem Szenario aus insgesamt 2^{19} möglichen Feature-Kombinationen. Formal definieren wir den booleschen Konfigurationsraum wie folgt:

Definition 1 (Vektorbasierte Notation des booleschen Konfigurationsraums).
Sei F eine Menge von booleschen Features.

- Der Konfigurationsraum ist definiert als $\mathbb{B}^{|F|}$
- Eine Konfiguration bezeichnen wir als $\vec{c_i} = (f_{i,1}, f_{i,2}, \ldots, f_{i,n}) \in \mathbb{B}^{|F|}$
  wobei $f_{i,j} \in \{0,1\}$ mit $n = |F|$, $i, j \in \mathbb{N}$ und $j \leq n$.

Allerdings repräsentiert nicht jede Konfiguration eine funktionsfähige Autovariante. Betrachten wir als Beispiel die drei Konfigurationen $\vec{c_1}$, $\vec{c_2}$ und $\vec{c_3}$ aus der Menge der Features *F*. Die drei Konfiguration sind in einer vektorbasierte Notation, gemäß Def. 1, dargestellt, wobei die Features in der folgenden Reihenfolge definiert sind:

$$\begin{aligned}\vec{c_{\mathrm{i}}} = (&c_{\mathrm{i,AUTO}}, c_{\mathrm{i,GET}}, c_{\mathrm{i,AU}}, c_{\mathrm{i,MAN}}, c_{\mathrm{i,AIR}}, c_{\mathrm{i,SAIR}},\\ &c_{\mathrm{i,VAIR}}, c_{\mathrm{i,RADIO}}, c_{\mathrm{i,BLUE}}, c_{\mathrm{i,NAVI}}, c_{\mathrm{i,DIGK}},\\ &c_{\mathrm{i,USA}}, c_{\mathrm{i,EUR}}, c_{\mathrm{i,GPS}}, c_{\mathrm{i,PORT}}, c_{\mathrm{i,CD}}, c_{\mathrm{i,USB}}, c_{\mathrm{i,KARO}})\end{aligned}$$

Daraus ergeben sich die folgenden Konfigurationsvektoren:

$$\vec{c_1} = (1,0,1,0,1,1,0,0,1,1,1,0,1,1,0,1,1,1,0)$$

$$\vec{c_2} = (1,1,1,1,1,0,0,0,0,1,0,1,0,0,1,0,0,0,1)$$

$$\vec{c_3} = (1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1)$$

Ein Auto ohne *Karosserie* und *Getriebe* kann nicht gebaut werden. Daher sollten diese beiden Features in jeder gültigen Autovariante vorhanden sein. Im Vektor $\vec{c_1}$ sind jedoch sowohl das Feature *Karosserie* ($c_{1,\mathrm{KARO}}$) als auch das Feature *Getriebe* ($c_{1,\mathrm{GET}}$) auf 0 gesetzt. Somit wird die Konfiguration $\vec{c_1}$ als ungültig betrachtet. Die Konfiguration $\vec{c_2}$ kombiniert die beiden Features *Automatik* ($c_{2,\mathrm{AU}}$) und *Manuell* ($c_{2,\mathrm{MAN}}$) in der selben Autovariante. Ein Auto mit zwei Getrieben ist jedoch nicht nur teuer, sondern auch für die meisten Kunden nicht gewollt. Daher gilt diese Variante aus wirtschaftlichen Gründen als ungültige Autovariante. Auf der anderen Seite stellt der Vektor $\vec{c_3}$ ein Beispiel für eine gültige Konfiguration dar. Diese Konfiguration enthält alle benötigten Features bei einem funktionsfähigen Auto, ohne dabei Features zu kombinieren, die miteinander inkompatibel sind.

Zusätzlich zu den verschiedenen Features visualisiert das Feature-Modell Beziehungen zwischen diesen Features. Beispielsweise sehen wir unter dem Feature-Diagramm in Abb. 2.1

eine Implikationsregel zwischen den Features *Automatik* und *GPS*, was bedeutet, dass wir das Feature *GPS* nur dann auswählen können, wenn das Feature *Automatik* im Auto konfiguriert ist.

Nachdem wir einige Beispiele für ausgewählte Constraints betrachtet haben, ist es offensichtlich, dass wir ein Modell benötigen, das alle Constraints einer SPL formalisiert. Hier kommt das Feature-Modell ins Spiel. Nun erklären wir die Constraints, die vom Feature-Diagramm in Abb. 2.1 definiert sind. Dabei fangen wir mit den booleschen Constraints zwischen genau zwei Features an:

- Eltern-Kind-Beziehung:
  - Beispiel: Die Relation zwischen den beiden Features *Airbags* und *Vorderairbags* ist ein Beispiel für eine Eltern-Kind-Beziehung. Die Logik dahinter besteht darin, dass bestimmte Sicherheit-Features, wie das Feature *Vorderairbags*, nur im Kontext des übergeordneten Feature *Airbags* sinnvoll sind.
  - Syntax: Die Eltern-Kind-Beziehung wird grafisch durch eine Kante zwischen dem Eltern- und dem Kind-Feature visualisiert. Das übergeordnete Eltern-Feature befindet sich hierarchisch über dem untergeordneten Kind-Feature.
  - Semantik: Die Eltern-Kind-Beziehung stellt die hierarchische Abhängigkeit zwischen den Features in einem Feature-Diagramm dar. Ein Feature kann nur ausgewählt werden, wenn sein übergeordnetes Feature ebenfalls ausgewählt ist. Das Root-Feature hat kein übergeordnetes Feature, daher muss es in jeder Konfiguration ausgewählt werden. Diese Beziehung kann als eine Art *Besteht aus*-Relation interpretiert werden. In diesem Zusammenhang kann das übergeordnete Feature (z. B. Airbags) als ein Container betrachtet werden, der verschiedene Teile enthält. Formal wird eine Eltern-Kind-Beziehung durch die Implikationsregel $f_j \rightarrow f_i$ beschrieben, wobei f_i (z.B. *Airbags*) das übergeordnete Feature und f_j (z.B. *Vorderairbags*) das untergeordnete Feature ist.
- Mandatory Feature:
  - Beispiel: In vielen modernen Autos ist ein GPS-System eine unverzichtbare Komponente der Navigationsdienste. Es ist also zwingend erforderlich, dass jedes Navigationsystem, als Teil seiner grundlegenden Zusammensetzung, immer ein GPS-Modul enthält. In unserem Feature-Modell wird dies dadurch dargestellt, dass das Feature *GPS* als obligatorisch (mandatory) unter dem Feature *Navigation* gekennzeichnet wird.
  - Syntax: Die FODA-Notation verwendet einen schwarzen ausgefüllten Kreis, um mandatory Features zu kennzeichnen. Dieser Kreis befindet sich am Ende der Kante, die zum untergeordneten mandatory Feature führt.
  - Semantik: Die Semantik hinter dem mandatory Feature lautet im Wesentlichen *Besteht aus*-Relation und kann durch die aussagenlogische Formel $f_i \rightarrow f_j$ ausgedrückt werden. f_i (z.B. *Navigation*) repräsentiert das übergeordnete Feature, unabhängig davon, ob es selbst als mandatory gekennzeichnet ist oder nicht, und f_j (z.B. *GPS*) stellt das mandatory untergeordnete Feature dar. Sobald f_i in einer Konfiguration ausgewählt wird, ist es zwingend erforderlich, dass auch f_j ausgewählt wird.
- Optionale Feature:
  - Beispiel: Ein Beispiel für ein optionales Feature ist das Feature *Radio*. Nicht jeder Kunde benötigt möglicherweise ein Radio, daher ist das Feature *Radio* optional. Die Kennzeichnung des Features *Radio* als optional ermöglicht es, das Feature nach Bedarf hinzuzufügen oder zu entfernen, je nach den individuellen Anforderungen für eine spezifische Konfiguration eines Autos.
  - Syntax: Die FODA-Notation verwendet einen weißen ausgefüllten Kreis, um optionale Features zu kennzeichnen. Dieser Kreis befindet sich am Ende der Kante, die zum untergeordneten optionalen Feature führt.

  - Semantik: Optionale Features im Feature-Modell sind solche, die in einer Konfiguration ausgewählt werden können, aber nicht zwingend ausgewählt werden müssen. Vorher haben wir bereits erwähnt, dass optionale Features einen speziellen Fall für die Eltern-Kind-Beziehung darstellen. Daher benötigt die Kennzeichnung eines Features als optional keine zusätzliche Semantik über die grundlegende Eltern-Kind-Beziehung hinaus.

Neben den hierarchischen booleschen Beziehungen zwischen Features enthält ein Feature-Modell auch sogenannte Cross-Tree-Constraints. Cross-Tree-Constraints ermöglichen die Definition von Abhängigkeiten zwischen Features, die nicht direkt in einer hierarchischen Beziehung zueinander stehen. In der FODA-Notation können Cross-Tree-Constraints zwei Formen annehmen:

- Require-Constraint:
  - Beispiel: Die erste Implikationsregel $Automatik \rightarrow GPS$ in Abbildung 2.1 veranschaulicht einen Require-Constraint zwischen den beiden Features *Automatik* und *GPS*. Moderne Automatikgetriebe verwenden GPS-Daten, um sich auf bevorstehende Straßenbedingungen einzustellen und das Schaltverhalten anzupassen. Aus diesem Grund erfordert das Feature *Automatik* das Vorhandenseins des Features *GPS*
  - Syntax: In der FODA-Notation lässt sich den Require-Constraint durch eine aussagenlogische Formel, wie beispielsweise $Automatik \rightarrow GPS$, ausdrücken.
  - Semantik: Die Require-Constraint drückt aus, dass das Vorhandensein eines Features die Auswahl eines hierarchisch nicht verbundenen anderen Features erfordert. Die Semantik hinter dem Require-Constraint wird durch die aussagenlogische Formel $f_i \rightarrow f_j$ ausgedrückt, wobei f_i (z.B. *Automatik*) das Feature ist, das das andere erfordert, und f_j (z.B. *GPS*) das erforderliche hierarchisch nicht verbundene Feature ist.
- Exclude-Constraint:
  - Beispiel: Die dritte Implikationsregel $Vorderairbags \rightarrow \neg CD$ in Abb. 2.1 ist ein Beispiel für einen Exclude-Constraint. In kleinen Autos kann die Anwesenheit von CD-Player den Einbau des Vorderairbags verhindern. Um dieses Inkompatibilitätsproblem zu vermeiden, setzt das Feature-Diagramm in Abbildung 2.1 mithilfe des Exclude-Contraints voraus, dass die beiden Features *Vorderairbags* und *CD* nicht gleichzeitig ausgewählt werden dürfen.
  - Syntax: Die FODA-Notation bietet die Möglichkeit, Exclude-Constraint durch aussagenlogische Formeln, wie beispielsweise $Vorderairbags \rightarrow \neg CD$, ausdrücken.
  - Semantik: Der Exclude-Constraint gibt an, dass die Auswahl eines Features die Auswahl eines hierarchisch nicht verbundenen anderen Features ausschließt. In aussagenlogischer Form wird dieser Constraint durch die Formel $f_i \rightarrow \neg f_j$ ausgedrückt. In SPL-Entwicklung bedeutet der Exclude-Constraint, dass die beiden konkurrierenden Features nicht gleichzeitig in einer gültigen Konfiguration ausgewählt werden dürfen.

Nachdem wir die booleschen Beziehungen zwischen genau zwei Features behandelt haben, erklären wir nun die Beziehungen, die zwei oder mehr Features miteinander verknüpfen:

- Oder-Gruppe:
  - Beispiel: Einige Fahrer bevorzugen das Abspielen von CDs, während andere lieber Musik von einem USB-Stick oder einem externen Gerät über den AUX-Eingang hören möchten. Deswegen besthet zwischen die drei Features *CD*, *USB* und *AUX* in der Abb 2.1 eine logische Oder-Beziehung. Alle drei Features sind unter dem Eltern-Feature *Ports* gruppiert. So ergeben sich insgesamt sieben verschiedene Möglichkeiten, wie das Radio mithilfe der Features *CD*, *USB* und *AUX* konfiguriert werden kann. Diese reichen von der Auswahl nur eines einzelnen Features,

entweder *USB*, *CD* oder *AUX*, bis hin zu verschiedenen Kombinationen dieser drei Features. Es ist wichtig zu beachten, dass die Konfiguration, bei der keines der drei Features ausgewählt wird, nicht zulässig ist. In diesem Fall wäre das Radio nicht funktionsfähig, da es keine Möglichkeit gäbe, Musik abzuspielen.

- Syntax: Oder-Gruppe wird durch einen schwarzen ausgefüllten Halbkreis visualisiert. Der Halbkreis enthält mehrere Kanten, die von dem Eltern-Feature (z.B *Ports*) zu den Kind-Features (z.B *CD*, *USB* und *AUX*) führen.
- Semantik: Die beteiligten Kind-Features innerhalb der Oder-Gruppe werden durch das logische Oder-Symbol *Or* verbunden. Dies bedeutet, dass mindestens ein Feature aus der Oder-Gruppe ausgewählt werden muss, um eine gültige Konfiguration zu bilden. Die Semantik von Oder-Gruppe können wir durch die aussagenlogische Formel $f_i \rightarrow (g_1 \vee g_2 \vee g_3 \vee ... \vee g_j)$ darstellen, wobei f_i das Eltern-Feature ist und $g_1...g_j$ die verschiedenen Kind-Features sind.

- Alternative-Gruppe:
  - Beispiel: Bei der Wahl des Getriebes in einem Auto besteht eine exklusive Entscheidung zwischen einem manuellen und einem automatischen Getriebe. Aus diesem Grund ist es sinnvoll, die Features *Manuell* und *Automatik* unter dem übergeordneten Feature *Getriebe* in einer Alternative-Gruppe zu gruppieren. Ähnlich wie die Oder-Gruppe muss bei der Alternative-Gruppe mindestens eines der beiden Features *Manuell* oder *Automatik* ausgewählt werden. Allerdings begrenzt die Alternative-Gruppe die Wahl auf maximal eine Feature. Somit ergeben sich genau zwei Möglichkeiten: Entweder wird das Feature *Manuell* oder das Feature *Automatik* ausgewählt.
  - Syntax: Alternative-Gruppe wird in der FODA-Notation durch einen weißen nicht ausgefüllten Halbkreis dargestellt. Der Halbkreis deckt mehrere Kanten ab, die vom dem Eltern-Feature zu den untergeordneten Kind-Features führen.
  - Semantik: Die Semantik hinter der Alternative-Gruppe wird durch die logische Entweder-Oder-Beziehung dargestellt. Wenn das Eltern-Feature der Alternative-Gruppe ausgewählt ist, muss genau ein Kind-Feature ausgewählt werden. Eine Alternative-Gruppe, bei der f_i das Eltern-Feature ist und g_j sowie g_k die Kind-Features sind, kann durch die folgende aussagenlogische Formel repräsentiert werden: $f_i \rightarrow (g_j \wedge \neg g_k) \vee (\neg g_j \wedge g_k)$

Um auch komplexere Beziehungen zwischen Features auszudrücken, erlaubt die FODA-Notation erweiterte aussagenlogische Formeln. Ein Beispiel dafür ist die zweite Implikationsregel in Abb. 2.1 ($Karosserie \Rightarrow Automatik \wedge \neg Bluetooth$). Diese Formel besagt, dass Wenn das Feature *Karosserie* ausgewählt wird, muss auch das Feature *Automatik* ausgewählt werden, aber das Feature *Bluetooth* darf nicht ausgewählt werden.

Alle diesen Constraints des Feature-Modells dienen dazu, die Auswahl der Features einzuschränken und sicherzustellen, dass nur eine Teilmenge von gültigen Konfigurationen erstellt wird. Diese Teilmenge bezeichnen wir als den gültigen Konfigurationsraum $C_{gültig}$. In diesem Kontext betrachten wir ein Feature-Modell als eine Abbildung $FM : \mathbb{B}^{|F|} \rightarrow \{1, 0\}$. Die Abbildung FM ordnet jedem booleschen Vektor im Konfigurationsraum einen Wert in der Menge {0,1} zu. Wenn $FM(\vec{c}) = 1$ bedeutet dies, dass die Konfiguration $\vec{c}$ im Feature-Modell als gültig betrachtet wird. Umgekehrt bedeutet $FM(\vec{c}) = 0$, dass die Konfiguration $\vec{c}$ eine ungültige Konfiguration ist. Als Teilmenge des Konfigurationsraum erhalten wir aus der Funktion FM die Menge $C_{gültig} = \{\vec{c} \mid \vec{c} \in \mathbb{B}^{|F|} \wedge FM(\vec{c}) = 1\}$. Diese Teilmenge enthält alle gültigen Konfigurationen aus dem Konfigurationsraum $\mathbb{B}^{|F|}$. Analog dazu repräsentiert die Teilmenge $C_{ungültig} = \{\vec{c} \mid \vec{c} \in \mathbb{B}^{|F|} \wedge FM(\vec{c}) = 0\}$ die Menge aller ungültigen Konfigurationen. Formal definieren wir ein Feature-Modell sowie die beiden Teilmengen $C_{gültig}$ und $C_{ungültig}$ wie folgt:

Definition 2 (Gültige und Ungültige Konfigurationen).
Sei F eine Menge von Features und $\mathbb{B}^{|F|}$ die Menge aller Konfigurationen.

- Ein Feature-Modell ist definiert als die Abbildung $FM : \mathbb{B}^{|F|} \rightarrow \{1,0\}$
- Der gültige Konfigurationsraum ist definiert als $C_{gültig} = \{\vec{c} \mid \vec{c} \in \mathbb{B}^{|F|} \wedge FM(\vec{c}) = 1\} \subseteq \mathbb{B}^{|F|}$
- Der ungültige Konfigurationsraum ist definiert als $C_{ungültig} = \{\vec{c} \mid \vec{c} \in \mathbb{B}^{|F|} \wedge FM(\vec{c}) = 0\} \subseteq \mathbb{B}^{|F|}$

Ein Feature-Modell kann Features enthalten, die in jeder gültigen Konfiguration vorhanden sein müssen. Ein Beispiel hierfür ist das Feature *Getriebe*, wie es in Abbildung 2.1 dargestellt ist. *Getriebe* ist ein Kind-Feature des Wurzel-Features *Auto* und wird im Diagramm als mandatory Feature gekennzeichnet. Wenn eine Konfiguration das Feature *Getriebe* nicht enthält, wird diese Konfiguration unmittelbar als ungültig betrachtet. Dieses Feature wird im Kontext von SPL als *Core-Features* bezeichnet. Core-Features sind die wichtigsten Features in einem Feature-Modell und beschreiben die Kernfunktionalität einer SPL. Im Gegensatz dazu gibt es Features, die bei keiner gültigen Konfiguration enthalten sein können. Diese Features bezeichnen wir als *Dead-Features*. Die zweite aussagenlogische Formel in Abb. 2.1 erklärt, warum das Feature *Manuell* ein Dead-Feature ist: Das Feature *Automatik* muss immer ausgewählt werden, wenn das Feature *Karosserie* gewählt wird, und da *Karosserie* ein mandatory Feature ist, wird das Feature *Manuell* niemals in einer gültigen Konfiguration erscheinen. Dadurch wird das Feature *Manuell* als Dead Feature betrachtet. Formal definieren wir Core- oder Dead-Features wie folgt:

Definition 3 (Core und Dead Features).
Sei C_{valid} der gültigen Konfigurationsraum für die Menge der Features F.

- $f_i \in F$ ist ein Core-Feature $\Leftrightarrow \forall \vec{c} \in C_{valid} : f_i = 1$
- $f_i \in F$ ist ein Dead-Feature $\Leftrightarrow \forall \vec{c} \in C_{valid} : f_i = 0$

In diesem Abschnitt haben wir das Konzept des Feature-Modells erklärt. Anhand eines Auto-Beispiels haben wir gezeigt, wie Feature-Modelle in Form eines Feature-Diagramms die logischen Beziehungen zwischen verschiedenen Features visualisieren. Durch die Erläuterung von Feature-Constraints innerhalb eines Feature-Modells haben wir zwischen dem ‚gültigen' und ‚ungültigen' Konfigurationsraum unterschieden. Ein Feature-Modell definiert die Menge der gültigen Features, die den gültigen Konfigurationsraum einer SPL bilden. Das Modell beschreibt also, welche Kombination von Features zu einem funktionsfähigen Produkt führen. In komplexen SPLs sind allerdings Feature-Modelle oft nicht vorhanden. Im folgenden Abschnitt beschreiben wir einen Ansatz, wie wir mithilfe von maschinellem Lernen ein Feature-Modell rekonstruieren können.

2.2 Feature-Modell-Lernen

Im letzten Abschnitt haben wir gezeigt, wie Feature-Modelle den gültigen Konfigurationsraum einer SPL repräsentieren. Anhand eines Beispiels haben wir erklärt, wie ein Feature-Modell in Form eines Feature-Diagramms sowohl die Funktionalität als auch die Variabilität von Software-Systemen darstellt. Bei hoch konfigurierbaren SPLs werden Feature-Modelle jedoch nicht von Grund auf neu erstellt, sondern müssen gegebenenfalls aus Teilmengen verschiedener bereits implementierter Produktvarianten rekonstruiert werden. Ein möglicher Ansatz zur Rekonstruktion von Feature-Modellen ist der Einsatz von maschinellem Lernen. In diesem Abschnitt erklären wir, wie die Technik des **F**eature-**M**odell-**L**ernens zur Rekonstruktion von Feature-Modellen funktioniert. Anschließend erläutern wir die Herausforderungen bei der Verwendung von FML zur Feature-Modell-Rekonstruktion.

In der Praxis existiert zu Beginn der Entwicklung einer SPL oft kein vollständiger Entwurf in Form eines Feature-Modells. Mit der kontinuierlichen Weiterentwicklung der SPL und der zunehmenden Anzahl implementierter Produktvarianten wird es schnell unpraktikabel, ein Feature-Modell für die gesamte SPL von Grund auf neu zu erstellen. Literaturen

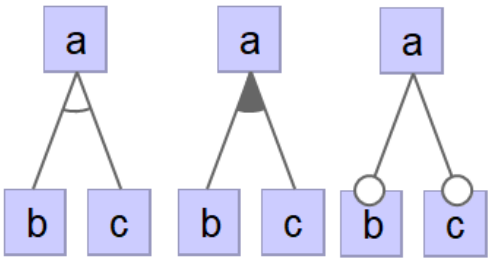


ABBILDUNG 2.2: Rekonstruierte Feature-Diagramme, von links nach rechts FM1, FM2, FM3.

haben die Idee untersucht, SPLs basierend auf einer bestehenden Produktfamilie zu entwickeln und ein Feature-Modell auf dieser Grundlage zu rekonstruieren. Wir erklären diesen Ansatz anhand eines Beispiels. Sei $F = \{a, b, c\}$ eine Menge von Features. Unser Ziel ist, ein Feature-Modell zu rekonstruieren, dass den gesamten gültigen Konfigurationsraum $C_{\text{gültig}}$, wie wir in der Def. 2 beschrieben haben, darstellt. Die Menge F besteht aus 3 Features. Somit haben wir $2^3 = 8$ mögliche Konfigurationen, aus dieser Konfigurationen haben wir insgesamt $2^8 = 256$ mögliche Zusammensetzungen von $C_{\text{gültig}}$. Wir beschränken aber die mögliche gültigen Konfigurationsräume durch die beiden Mengen U und V :

$$U = \{(1,1,0),(1,0,1)\} \subseteq C_{\text{gültig}}$$

$$V = \{(0,1,0),(0,0,1)\} \subseteq 2^F \setminus C_{\text{gültig}}$$

Die Menge U enthält die Konfigurationen von bereits implementierten, gültigen Produkten. U ist eine Teilmenge der Menge aller gültigen Konfigurationen $C_{\text{gültig}}$. Die Menge V besteht hingegen aus bekannten ungültigen Konfigurationen. Die Mengen U und V reduzieren die Anzahl der mögliche Kompositionen von $C_{\text{gültig}}$ auf insgesamt 16 Möglichkeiten. Dies bedeutet, dass die Rekonstruktion des Feature-Modells nicht eindeutig ist. Die Abb. 2.2 zeigt drei mögliche Feature-Modellen, die durch ein Feature-Diagramm in FODA-Notation repräsentiert sind. Das Feature-Diagramm FM_1 beschreibt genau die Teilmenge der gültigen Konfigurationen U, die zwei anderen Feature-Diagrammen FM_2 und FM_3 erweitern die Menge der gültigen Konfigurationen mit weiteren Feature-Kombinationen, ohne dabei die definierten Constraints der Menge V zu verletzen. Dies bedeutet, dass die erweiterten gültigen Konfigurationen in dem Feature-Diagrammen FM_2 und FM_3 mit keiner der bekannten ungültigen Konfigurationen aus $C_{\text{gültig}}$ übereinstimmen. Beispielsweise wird die Konfiguration ($\vec{c} = (1,1,1) \notin V$) im Feature-Diagramm FM_2 als eine weitere gültige Konfiguration betrachtet.

Eine der modernen Techniken, die auch auf die Rekonstruktion von Feature-Modellen basierend auf einer kleinen Menge von Konfigurationen abzielt, findet sich in der Masterarbeit von Mathis Weiß [23]. Weiß präsentiert die Technik des **F**eature-**M**odell-**L**ernens (FML), die es ermöglicht, ein Feature-Modell mithilfe von maschinellem Lernen zu rekonstruieren. Damit wir die Idee hinter FML beschreiben, betrachten wir einen Ausschnitt des gesamten Feature-Modells, wie in Abb. 2.3 dargestellt ist. Die Abbildung in 2.3 besteht insgesamt aus 8 Features und nur einer erweiterten aussagenlogischen Formel ($\neg$Karosserie $\vee$ Automatik). Das Feature-Diagramm enthält 4 mandatory Features, ein optionales Feature und eine Alternative-Gruppe. Die Oder-Gruppe, der Exclude-Constraint und der Require-Constraint sind bei diesem Feature-Diagramm nicht dargestellt. Im Kontext von FML wird eine Teilmenge des gesamten Konfigurationsraums in tabellarischer Form vorbereitet. Das Beispiel in Tabelle 2.1 besteht aus 8 Zeilen, wobei jede Zeile eine Konfiguration in vektorbasierter Notation repräsentiert. Die letzte Spalte wird im FML-Verfahren als *Label*, also Beschriftung, bezeichnet. Ein Label gibt an, ob eine Konfiguration gültig ist oder nicht. Da wir nur mit einem künstlichen Beispiel arbeiten und wir bereits wissen, dass ein Feature-Modell existiert und wie es definiert ist, haben wir die Labels auf Grundlage dieses Feature-Modell

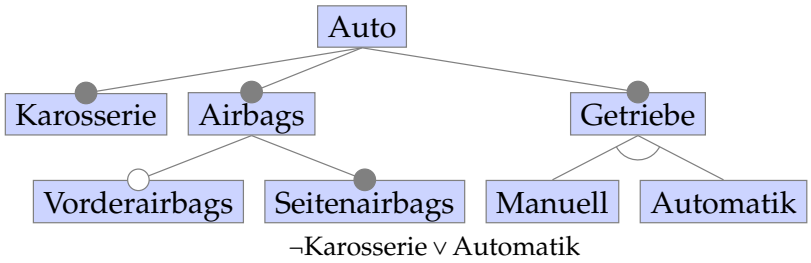


ABBILDUNG 2.3: Reduziertes Feature-Modell des Autos

AUTO	KARO	AIR	GET	VAIR	SAIR	MAN	AU	Label
1	1	1	1	0	1	0	1	1
1	1	1	1	1	1	0	1	1
0	0	1	1	0	0	1	1	0
0	1	0	0	1	0	1	0	0
1	0	1	1	0	1	1	1	0
0	1	0	1	1	0	0	1	0
0	1	1	0	1	1	1	0	0
1	1	0	0	0	0	0	1	0

TABELLE 2.1: Beschrifteter Datensatz für das Feature-Model-Lernen

festgelegt. Beispielsweise sind die ersten beiden Konfigurationen als *gültig* gekennzeichnet, da sie die Constraints des Feature-Modells in Abb. 2.3 erfüllen. Die letzten sechs Konfigurationen sind hingegen als *ungültig* markiert. In der Praxis sind allerdings Feature-Modelle nicht bekannt. In diesem Fall ist eine Umsetzung der Menge der Konfigurationen erforderlich, um die Gültigkeit jeder Konfiguration zu ermitteln. Die übrigen Spalten stellen die verschiedenen Features einer Konfiguration dar, wobei jede Konfiguration aus 8 Features besteht. Im Kontext des maschinellen Lernens werden die Daten in der Tabelle 2.1 als *Datensatz* bezeichnet. Jede Zeile in der Tabelle entspricht einen ML-*Instance* und die Spalten, welche die SPL-Features repräsentieren, werden in der ML-Strategie als *Attributen* bezeichnet. Im Verlauf unserer Arbeit verwenden wir die Begriffe *Attribute* und *Features* sowie *Konfigurationen* und *Instanzen* in gleicher Bedeutung. Basierend auf einem Datensatz versuchen die FML-Algorithmen das Feature-Modell zu lernen. Die Abb. 2.4 stellt grafisch dar, wie der Lernprozess von FML in der Praxis eingesetzt wird. In der Abbildung sehen wir zwei unterschiedliche Arten von gerichteten Kanten. Die gestrichelten Kanten repräsentieren die Eingabe und Ausgabe für die verschiedenen Phasen. Die ununterbrochenen, durchgehenden Kanten zeigen die Flussrichtung des gesamten FML-Prozesses. Der Fluss beginnt auf

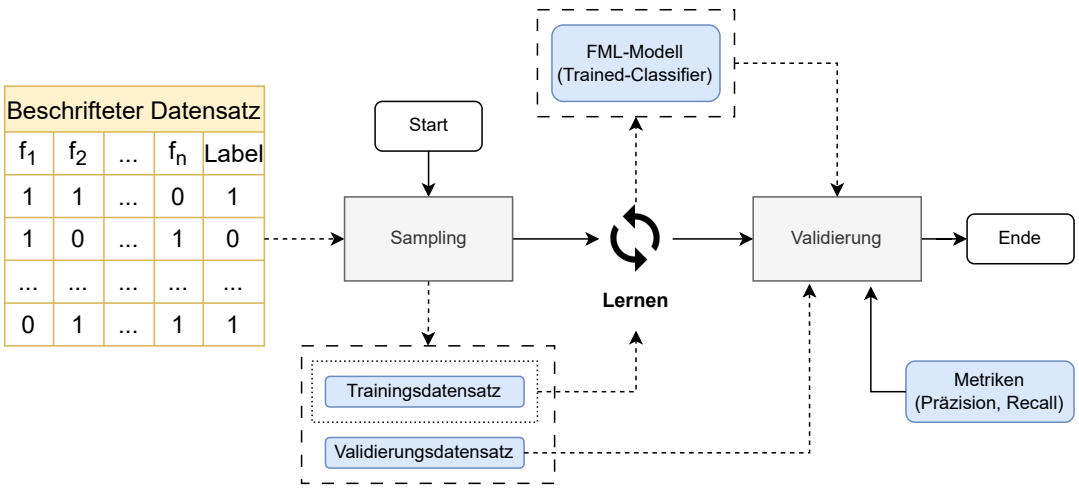


ABBILDUNG 2.4: Prozess des Feature-Modell-Lernens

der linken Seite mit einer Tabelle (Datensatz), die aus einer Auswahl von Instanzen (Konfigurationen) besteht, die jeweils aus Attributen (Features) und einem zugehörigen Label bestehen. Von dort aus führt der Prozess zum *Start*, der den Beginn des maschinellen Lernvorgangs darstellt. Die erste Phase ist das *Sampling*, ein Schritt, in dem der Datensatz in zwei Teile aufgeteilt wird: Den Trainingsdatensatz und den Validierungsdatensatz. Der Trainingsdatensatz wird bei dem Lernvorgang des FML-Algorithmus verwendet. Der Validierungsdatensatz dient jedoch dazu, das erlernte FML-Modell in der letzten Phase zu validieren. Der nächste Schritt ist das *Lernen*. Diese Phase stellt das Kernprinzip des FML-Prozesses dar. In der Praxis kann diese Phase sehr komplex sein, da es eine Vielzahl von ML-Algorithmen und Hyperparametereinstellungen gibt, die bei dieser Lernphase berücksichtigt werden müssen. Hier kommt die Automatisierung durch den AutoML-Ansatz [9] ins Spiel. Die Automatisierung des Lernprozesses im FML wird durch den Tool AutoGluon erreicht. Anders als die traditionelle Lernverfahren, bei denen ML-Algorithmen und Hyperparameter für jeden Lernvorgang manuell festgelegt werden, nutzt AutoGluon einen automatisierten zweistufigen Ansatz. In der erste Stufe führt AutoGluon ein automatisiertes *Tuning* durch. Beim Tuning werden verschiedene Kombinationen von Hyperparametern getestet, um möglichst optimale Hyperparameterkonfiguration zu finden, die die besten Ergebnisse für den vorliegenden Trainingsdatensatz liefern. Dies beinhaltet auch die Auswahl des geeignetsten ML-Algorithmus aus einer Auswahl von verfügbarer Algorithmen. Sobald die optimalen Hyperparameter sowie der Algorithmus identifiziert sind, beginnt die zweite Stufe, die den eigentlichen Lernvorgang darstellt. In dieser Stufe wird der Trainingsdatensatz dem ausgewählten ML-Algorithmus zugeführt. Das Algorithmus analysiert die verschiedenen Instanzen in dem Datensatz und lernt daraus, welche Kombinationen von Features zu gültigen oder ungültigen Konfigurationen führen. Am Ende des Lernprozesses erhalten wir ein Modell, welches im Kontext von maschinellem Lernen als *Trained-Classifier* bezeichnet wird. Der Begriff *Classifier* leitet sich davon ab, dass wir das FML-Modell verwenden, um neue unbeschriftete Konfigurationen als *gültig* oder *ungültig* zu klassifizieren. Nehmen wir an, wir verwenden die ersten drei gültigen und die zweite ungültige Instanz aus unserem Beispiel in Tab. 2.1 als Trainingsdatensatz und die übrigen Instanzen als Validierungsdatensatz. Der Classifier wird trainiert, um Muster und Regeln zu erkennen, die gültige von ungültigen Instanzen unterscheiden. Ein beobachtetes Muster ist zum Beispiel, dass das Attribut *AUTO* bei allen gültigen Instanzen den Wert 1 hat. Basierend auf diesem Muster erstellt der Classifier beispielsweise die Regel, dass eine Instanz unmittelbar als gültig klassifiziert wird, wenn das Attribut *AUTO* den Wert 1 hat. Die letzte Phase ist die *Validierung*. In der Validierungssphase wird der berechnete Trained-Classifier auf den Validierungsdatensatz angewendet. Hier wird untersucht, ob die vom Classifier gelernten Regeln die Validierungsdatensatz korrekt klassifizieren. Basierend auf der vorher gelernten Regel wird die letzte Instanz im Datensatz vom Trained-Classifier als gültig klassifiziert. Diese Klassifizierung ist allerdings nicht korrekt, da die letzte Konfiguration laut dem Validierungsdatensatz eine ungültig Konfiguration ist. Der Validierungsprozess erfolgt in gleicher Weise für alle Instanzen des Validierungsdatensatz. Die Qualität des Modells wird am Ende durch zwei Evaluierungsmetriken quantifiziert: Präzision und Recall. Generell wird ein FML-Modell als ideal betrachtet, wenn es sowohl eine hohe Präzision als auch einen hohen Recall aufweist. Die Berechnung von Präzision und Recall basiert auf drei wichtigen Werten: True-Positive (TP), False-Positive (FP) und False-Negative (FN). In unserem Beispiel repräsentiert *TP* die Anzahl der Konfigurationen, die vom Trained-Classifier korrekt als *gültig* identifiziert sind. *FP* ist die Anzahl der Instanzen, die fälschlicherweise als gültig klassifiziert sind, und *FN* ist die Anzahl der Konfigurationen, bei denen das Modell fälschlicherweise eine negative Vorhersage gemacht hat, obwohl sie gültig sind. Somit berechnet sich die Präzision durch das Teilen des Werts von TP durch die Summe von den beiden Werten TP und FP. Analog dazu können wir den Recall berechnen, indem wir den Wert von TP durch die Summe von den beiden Werten TP und FN teilen. Eine hohe Präzision bedeutet, dass die vorhergesagten gültigen Konfigurationen wahrscheinlich tatsächlich gültig sind. Auf der anderen Seite bedeutet ein hoher Recall, dass der Trainer-Classifier dazu neigt, die meisten der tatsächlich gültigen Konfigurationen zu erfassen, auch wenn dabei einige Konfigurationen fälschlicherweise als gültig klassifiziert werden. Formal definieren wir die beiden Metriken wie folgt:

Definition 4 (Präzision und Recall für Feature-Modell-Rekonstruktion).

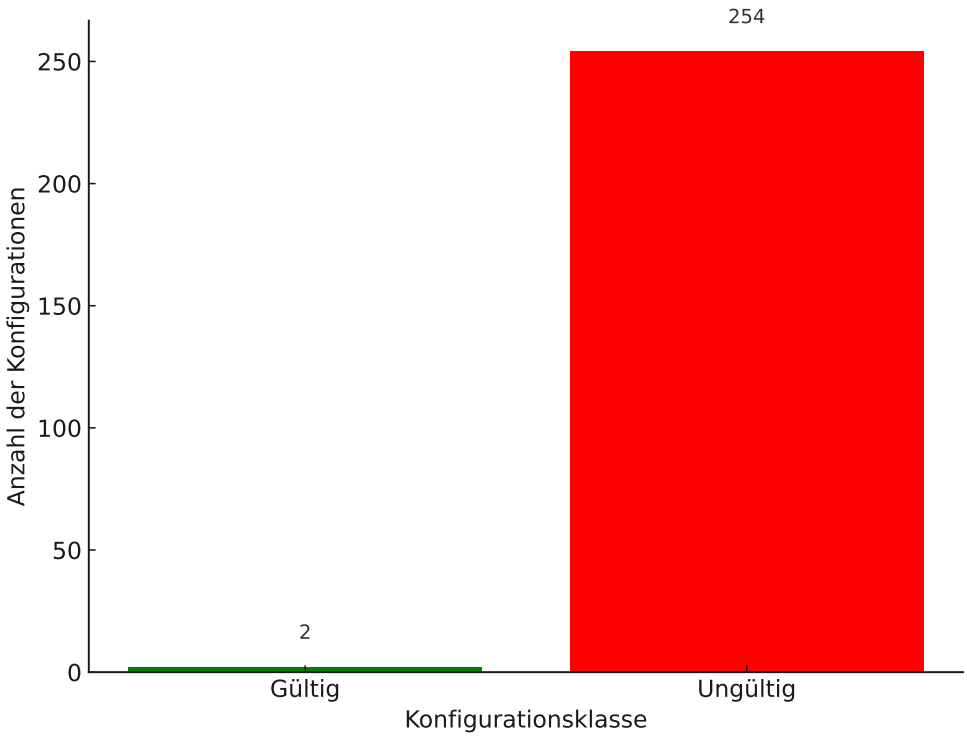


ABBILDUNG 2.5: Ungleichgewicht zwischen der Anzahl der gültigen und ungültigen Konfigurationen

Sei $FM_{rekonstruiert}$ das rekonstruierte Feature-Modell. Wir definieren die folgenden drei Werten:

- **True-Positives (TP):** Die Anzahl der Instanzen, bei denen $FM_{rekonstruiert}$ korrekt eine Konfiguration als gültig vorhergesagt hat.
- **False-Positives (FP):** Die Anzahl der Instanzen, bei denen $FM_{rekonstruiert}$ fälschlicherweise eine Konfiguration als gültig vorhergesagt hat.
- **False-Negatives (FN):** Die Anzahl der Instanzen, bei denen $FM_{rekonstruiert}$ fälschlicherweise eine Konfiguration als ungültig vorhergesagt hat.

Basierend auf diesen drei Werten können wir die Metriken Präzision und Recall definieren:

$$\text{Präzision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Im letzten Abschnitt haben wir gezeigt, dass der Konfigurationsraum sowohl gültige als auch ungültige Konfigurationen enthält. In der Praxis ist die Größe des gültigen Konfigurationsraums oft viel kleiner als die Größe des ungültigen Konfigurationsraums. Die Abb. 2.5 visualisiert die Differenz zwischen der Anzahl der gültigen und ungültigen Konfigurationen im Feature-Modell von Abb. 2.3. Insgesamt besteht der Konfigurationsraum aus 256 Konfigurationen. Der grüne Balken repräsentiert den gültigen Konfigurationsraum und zeigt 2 Konfigurationen. Im Gegensatz dazu stellt der rote Balken die Anzahl der ungültigen Konfigurationen dar. Insgesamt sind 254 Konfigurationen ungültig. In diesem Szenario besteht der Datensatz zu etwa 99% aus ungültigen Konfigurationen. Ein trivialer Trained-Classifier, der alle Konfigurationen als *ungültig* beschriftet, würde eine *Genauigkeit* von 99% erzielen. Genauigkeit im Kontext des FML bezieht sich darauf, wie oft die Vorhersagen des

Trained-Classifier korrekt sind, im Verhältnis zu allen getroffenen Vorhersagen. Die Genauigkeit wird als *Accuracy* bezeichnet und durch die folgende Formel berechnet:

$$\text{Accuracy} = \frac{\text{Anzahl der korrekten Vorhersagen}}{\text{Gesamtanzahl der Vorhersagen}}$$

ML-Algorithmen basieren auf der Annahme, dass eine hohe Accuracy auf ein gutes FML-Modell hinweist. Durch das konkrete Beispiel in Abb. 2.5 haben wir jedoch gezeigt, dass die Accuracy eine irreführende Metrik sein kann, wenn ML-Algorithmen auf unausgeglichene Datensätze trainiert werden. Um dieses Problem zu lösen, beschreibt Weiß die Technik des *Instance-Weighting*. Die Idee hinter Instance-Weighting besteht darin, den einzelnen Instanzen des Trainingsdatensatzes unterschiedliche Gewichte zuzuweisen. In unserem Szenario erhalten die gültigen Instanzen eine höhere Gewichtung als die ungültigen Konfigurationen. Die Evaluation dieser Lösungsstrategie basiert auf der sogenannten *balanced-accuracy* [2]. Im Gegensatz zur herkömmlichen Accuracy berücksichtigt die balanced-accuracy das Ungleichgewicht zwischen den verschiedenen Klassen im Trainingsdatensatz. Zunächst berechnen wir die Accuracy für jede einzelne Klasse und bilden dann den Durchschnitt der resultierenden Werte, um die gesamte balanced-accuracy zu bestimmen. Wenn die normale Accuracy von einem FML-Modell sehr hoch ist, weil sie durch die Ungleichverteilung des Datensatzes beeinflusst ist, kompensiert die balanced-accuracy die Auswirkungen des Klassenungleichgewichts und bleibt somit auf einem Zufallsniveau.

In diesem Abschnitt haben wir den Prozess der Rekonstruktion von Feature-Modellen unter Verwendung des Feature-Modell-Lernens behandelt. Anhand eines Beispiels haben wir gezeigt, wie FML basierend auf maschinellen Lernen-Strategien den booleschen Konfigurationsraum lernt. Anschließend haben Wir beschrieben, wie die Technik von FML mit unausgeglichener Datensätzen umgeht. Allerdings gibt es immer noch einige Schwachstellen bei diesem Ansatz. Im nächsten Abschnitt präsentieren wir die Problemstellung für die FML-Strategie.

2.3 Problemstellung: Curse-of-Dimensionality

In der letzten Abschnitt haben wir gezeigt, wie die Technik von FML basierend auf maschinelle Lernstrategien Feature-Modelle rekonstruiert. Ausgehend von einem unvollständigen Datensatz von gültigen und ungültigen Konfigurationen versuchen die ML-Algorithmen ein verallgemeinertes Modell zu lernen, das den gesamten gültigen Konfigurationsraum approximiert. In der Praxis, mit steigender Dimensionalität des Datenraums, also der Anzahl der Features bei dem Datensatz, die betrachtet werden, erhöht sich exponentiell die Anzahl der benötigten Instanzen für eine gute Leistung eines maschinellen Lernmodells [22]. Zum Beispiel nehmen wir an, dass für ein ML-Modell, um gute Ergebnisse zu liefern, mindestens 10 Instanzen für jede Kombination von Features erforderlich sind. In einem 1-dimensionalen binären Datensatz benötigen wir beispielsweise insgesamt 20 Instanzen ($2^1 \cdot 10 = 20$). Bei 2 Features würden wir 40 Instanzen benötigen ($2^2 \cdot 10 = 40$). Folglich würden wir für k Features insgesamt ($2^k \cdot 10$) Instanzen benötigen. Dieses exponentielle Wachstum stellt die Hauptauswirkung der sogenannten *Curse-of-Dimensionality* dar. Generell bedeutet der Curse-of-Dimensionality, dass mit zunehmender Dimensionalität die Daten spärlicher werden, was es den ML-Algorithmen erschwert, Muster und Strukturen im Datensatz zu erkennen [1].

Um das Problem der Curse-of-Dimensionality anzugehen, beschreibt Weiß in seiner Master-Arbeit [23] den Ansatz des manuellen *Preproccessing*. Dabei sind alle Core und Dead-Features, wie sie in Def. 3 beschrieben sind, zu einem Feature zusammengeführt. In Tab. 2.1 sind die Features *Auto*, *Karosserie*, *Airbags*, *Getriebe*, *Seitenairbags* und *Automatik* alle Core-Features, da sie in beiden gültigen Konfigurationen ausgewählt sind. Auf der anderen Seite hat das Attribut *Manuell* in beiden gültigen Konfigurationen den Wert 0, somit repräsentiert das Feature *Manuell* ein Dead-Feature. Das Attribut *Vorderairbags* ist in den gültigen Vektoren sowohl auf 0 als auch auf 1 gesetzt. Daher ist das Feature *Vorderairbags* weder ein Core-Feature noch ein Dead-Feature. Die gefundenen Gruppen der Core-Features und Dead-Features können

jeweils zu einem einzigen Feature zusammengeführt werden. Somit erhalten wir die neue

CoreMerged	DeadMerged	VAIR	Label
1	0	0	1
1	0	1	1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	1	0	0
1	1	1	0

TABELLE 2.2: Beschrifteter Datensatz nach Zusammenführung von Core-Features und Dead-Features

Datensatz in Tab. 2.2. Die Tabelle enthält insgesamt 4 Spalten. Die neu resultierten Spalten nach der Zusammenführung sind: *CoreMerged* und *DeadMerged*. *CoreMerged* repräsentiert die Zusammenführung der Core-Features. Analog dazu stellt die Spalte *DeadMerged* die Zusammenführung der Menge der Dead-Features. In unserem Szenario repräsentiert die Spalte *DeadMerged* genau das Dead-Feature *Manuell*. Der resultierte Datensatz enthält alle Informationen, die beschreiben, ob eine Konfiguration gültig ist oder nicht.

Die von Weiß beschriebene manuelle Preprocessing-Strategie hat allerdings einige Einschränkungen. Beispielsweise erfordert die Detektion der Core- und Dead-Features eine intensive Analyse des gesamten Datensatzes. Bei großen Datensätzen von komplexen SPL-Systemen ist diese Analyse zeit- und ressourcenintensiv. Auf der anderen Seite sind die Constraints des Feature-Modells, wie Oder-Gruppen und Alternative-Gruppen nicht behandelt. Daher untersuchen wir im Rahmen unserer Arbeit zwei Ansätzen zur *Dimensionality-Reduction* (DR), die den Dimensionsraum des Datensatzes automatisiert reduzieren. Wir betrachten dabei die herkömmliche Methode der Dimensionsreduktion, nämlich die Linear Principal-Component-Analysis (Linear PCA), und auch eine speziell für binäre Daten entwickelte Variante, die Logistic Principal-Component-Analysis (Logistic PCA).

Wir evaluieren FML sowohl mit als auch ohne den Einsatz von Linear bzw. Logistic PCA. Dabei vergleichen wir die Ansätze in Bezug auf ihre Effektivität, die anhand von Präzision und Recall gemessen wird, sowie hinsichtlich ihrer Effizienz, basierend auf der benötigten Rechenzeit für die Reduktion und das Training beim FML.

In diesem Kapitel haben wir die Grundlagen von Feature-Modellen und deren Rekonstruktion mittels maschinellem Lernen veranschaulicht. Anhand eines Beispiels aus der Automobilbranche haben wir gezeigt, welche Bedeutung Feature-Modelle für die Definition des gültigen Konfigurationsraums einer SPL repräsentieren. Nachdem wir die wichtige Rolle von Feature-Modellen in Kontext von SPL diskutiert haben, haben wir den automatisierten Prozess der Rekonstruktion von Feature-Modellen unter Verwendung der Feature-Modell-Lernen-Strategie erklärt. Außerdem haben wir kurz diskutiert, welche Techniken bei FML umgesetzt sind, um das Problem der unausgeglichenen Datensätze umzugehen. Wir haben den Abschnitt mit dem offenen Problem von Curse-of-Dimensionality abgeschlossen. Im nächsten Abschnitt präsentieren wir alternativen Kodierungsstrategien, um die Dimensionalität des booleschen Konfigurationsraum zu reduzieren.

Kapitel 3

Kodierungsstrategien für booleschen Konfigurationsräume

In diesem Kapitel diskutieren wir das Thema der Dimensionality-Reduction. Wir starten dieses Kapitel, indem wir die Grundlagen des Problems der Dimensionality-Reduction erklären. Anhand eines Beispiels aus Kapitel 2 zeigen wir, wie maschinelle Lern-Datensätze mathematisch durch Matrizen repräsentiert werden. Anschließend definieren wir das Problem der Dimensionality-Reduction. In der Praxis gibt es verschiedene Ansätze zur Umsetzung von Dimensionality-Reduction. Im Abschnitt 2 stellen wir die Technik der Principal-Component-Analysis vor, eine der bekanntesten Methoden zur Reduzierung der Dimensionalität von Datensätzen. Wir verwenden ein Beispiel eines zweidimensionalen Datensatzes, um den Ablauf der Principal-Component-Analysis zu beschreiben. Eine Herausforderung bei der Verwendung von Dimensionality-Reduction besteht darin, die optimale Dimensionalität des reduzierten Datensatzes zu bestimmen. Daher erläutern wir zwei bekannte empirische Heuristiken im Zusammenhang mit der Principal-Component-Analysis. Im Abschnitt 3 diskutieren wir einen anderen Ansatz zur Implementierung der Dimensionality-Reduction, der speziell für den Umgang mit booleschen Datensätzen entwickelt wurde, wie sie in booleschen Konfigurationsräumen vorkommen.

3.1 Dimensionality-Reduction

In diesem Abschnitt beschreiben wir die Grundidee hinter der *Dimensionality-Reduction*. Um einen besseren Einblick zu geben, wie maschinelle Lern-Datensätze strukturiert sind, werden wir anhand eines Beispiels aus Kapitel 2 veranschaulichen, wie diese Daten mathematisch in Form von Matrizen dargestellt werden. Anschließend beschreiben wir formal das Problem der *Dimensionality-Reduction*.

Wie wir in Kapitel 2 diskutiert haben, kann ein Datensatz D als eine Tabelle von n Instanzen ($n > 1$) betrachtet werden. Die Instanzen sind gegeben durch die Tupel $(x_1, y_1), \ldots, (x_n, y_n)$. In diesem Kontext ist x_i ein boolescher Vektor, der gemäß Definition 1 die i-te Konfiguration in einem booleschen Konfigurationsraum darstellt. Das Attribut y_i repräsentiert das Label. Wir können also den Datensatz D als eine Matrix darstellen:

$$X_{n\times p} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix} = [X_1, X_2, \cdots, X_p], \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{3.1}$$

Dabei repräsentiert $X_{n\times p}$ die Eingabematrix und besteht aus n Zeilen und p Spalten (Dimensionen). Y wird als eindimensionale Matrix (oder Vektor) mit n Einträgen beschrieben, wobei jeder Eintrag dem Label einer Instanz entspricht. Betrachten wir jetzt die Tabelle 2.1 in Kapitel 2. Diese Tabelle können wir beispielsweise als einen hochdimensionalen Datensatz D interpretieren. Aus diesem Datensatz D erhalten wir die Matrix $X_{8\times 8}$ und den Vektor $Y_{8\times 1}$

gemäß der Gleichung in 3.1:

$$X_{8\times 8} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad Y = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

In $X_{8\times 8}$ repräsentiert jede Zeile eine Instanz des Datensatzes, also eine spezifische Konfiguration der Features. Jede Spalte steht für ein Feature des Feature-Modells. Beispielsweise repräsentiert die erste Spalte das Feature *AUTO*, die zweite Spalte *KARO* usw. Die Werte in den Zellen der Matrix X sind boolesche Werte, die anzeigen, ob ein bestimmtes Feature in einer bestimmten Konfiguration ausgewählt ist (1) oder nicht (0). Der Vektor Y enthält die Labels für jede Instanz. Ein Label von 1 bedeutet, dass die entsprechende Konfiguration gültig ist, während ein Label von 0 darauf hinweist, dass die Konfiguration ungültig ist. Zum Beispiel ist die erste Konfiguration in der Matrix X gültig, da ihr Label in Y 1 ist, während die dritte Konfiguration in der Matrix X als ungültig gekennzeichnet ist, da ihr entsprechendes Label in Y eine 0 ist.

Ausgehend von der Herausforderung des *Curse of Dimensionality* in FML-Technik, die wir im Grundlagen-Kapitel erklärt haben, bietet sich die *Dimensionality-Reduction* (DR) [4] als einen intuitiven Lösungsansatz an. In der Praxis kann die Reduktion der Dimensionalität von Datensätzen auf verschiedene Weisen erreicht werden. Beispielsweise können wir den sogenannten Feature-Selection-Ansatz [10] verwenden. Bei diesem Ansatz wählen wir eine Teilmenge der ursprünglichen Features aus, die am relevantesten für die Vorhersage des Labels sind. Beispielsweise könnten einige Features in X redundant oder irrelevant für die Bestimmung der Gültigkeit einer Konfiguration sein und daher entfernt werden. Eine triviale Implementierung von Feature-Selection in unserem Kontext würde dann den Eingabematrix $X_{8\times 8}$ in einer niedrigdimensionalen Matrix $\tilde{X}_{8\times 3}$ reduzieren, indem beispielsweise 3 Spalten zufällig ausgewählt werden:

$$\tilde{X}_{8\times 3} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

Während diese Methode die Dimensionalität des Datensatzes reduziert, kann sie jedoch wesentliche Informationen aus dem Datensatz nicht berücksichtigen, die für die Gültigkeit der Konfigurationen notwendig sind. Als Ergebnis kann der Trained-Classifier aus der FML-Technik falsche Muster innerhalb der Daten lernen. In unserem konkreten Szenario haben wir beispielsweise das Feature *Karo* aus der Eingabematrix entfernt. Hier bedeutet das Entfernen dieses Features, dass wir eine der grundlegenden Eigenschaften eines Autos aus unserer Analyse ausschließen. Da das Feature *Karo* ein zentrales Element in der Definition der Konfigurationen ist, würde seine Entfernung dazu führen, dass wir möglicherweise nicht mehr zwischen gültigen und ungültigen Konfigurationen unterscheiden können. Ähnlich wäre das Entfernen des Features *Vorderairbags* problematisch, da wir dann die Information verlieren, dass ein Auto nicht nur Seitenairbags aber auch Vorderairbags haben könnte. Das willkürliche Weglassen solcher Features kann daher zu einem neuen Datensatz führen, welche nicht unbedingt die Eigenschaften des betrachteten SPL abbildet. Feature-Selection

behandeln wir in unserer Arbeit nicht, sondern beschäftigen wir uns mit einer anderen Überlegung von DR, die als *Feature-Transformation* bezeichnet wird. Bei Feature-Transformation geht es darum, eine Transformation T zu berechnen, die den hochdimensionalen ursprünglichen Datensatz ($X_{8\times8}$) in eine andere Matrix mit niedrigerer Dimensionalität zu komprimieren. Konkret erstellt die Funktion T einen neuen Feature-Raum aus der Menge der Features im ursprünglichen Datensatz, wobei die wesentlichen Informationen im neuen Unterraum erhalten bleiben [4]. Basierend auf der Idee von Feature-Transformation lässt sich das Problem der Dimensionality-Reduction durch zwei Hauptkomponenten definieren: Die erste Komponente ist die Eingabematrix $X_{n\times p}$. Unter Verwendung diese Matrix, das Problem der Dimensionality-Reduction ist somit definiert als die Berechnung der zweiten Komponente, eine Transformation T, die den originalen Datenraum $X_{n\times p}$ in der niedrigdimensionale Repräsentationen $\tilde{X}_{n\times k}$ komprimiert, wobei möglichst viele relevante Informationen aus $X_{n\times p}$ erhalten bleiben. In der Praxis ist die Dimensionalität des reduzierten Datensatz $\tilde{X}_{n\times k}$ viel kleiner als das originale Datensatz und wir schreiben $k \ll p$, wobei p die die Dimension der hochdimensionalne Datensatz und k steht für die Anzahl der Spalten in dem reduzierten Datensatz. Formal definieren wir das Problem der Dimensionality-Reduction wie folgt:

Definition 5 (Problem der Dimensionality-Reduction)
Sei $X_{n\times p}$ die Eingabematrix eines Datensatzes D.

$$T(X_{n\times p}) \equiv \tilde{X}_{n\times k} \equiv [\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_k] \qquad \text{mit } k \ll p.$$

In diesem Abschnitt haben wir das Prinzip der Dimensionality-Reduction erklärt. Wir haben gezeigt, wie Datensätze für maschinelles Lernen als Matrizen dargestellt werden. Anschießend haben wir gezeigt, wie das willkürliche manuelle Weglassen von Features zu Informationsverlust im Datensatz führt, was die Notwendigkeit einer automatischen Vorgehensweise wie die Feature-Transformation erfordert. Anschließend haben wir das Problem der Dimensionality-Reduction formal definiert, wobei das Ziel darin besteht, eine hochdimensionale Eingabematrix in eine niedriger dimensionierte Repräsentation zu transformieren, ohne dabei wesentliche Informationen aus der Daten zu verlieren. Im nächsten Abschnitt präsentieren wir die *Linear Principal-Component-Analysis*, eine der bekanntesten Methoden zur Implementierung von Feature-Transformation.

3.2 Principal-Component-Analysis

In diesem Abschnitt diskutieren wir die Technik der Linear Principal-Component-Analysis als einen Ansatz zur Dimensionality-Reduction. Zunächst beschreiben wir die mathematischen Grundlagen hinter der Linear Principal-Component-Analysis, indem wir einen zweidimensionalen Datensatz auf einem eindimensionalen Koordinatensystem projizieren. Danach erklären wir zwei verschiedene Ansätze zur Bestimmung der Dimensionalität des reduzierten Datensatzes im Kontext von Linear Principal-Component-Analysis.

In der Literatur existieren viele Methoden zur Dimensionality-Reduction. Die *Linear Principal-Component-Analysis* (Linear PCA) [24] ist dabei eine der beliebtesten Methoden davon. In dieser Arbeit werden wir die beiden Begriffe Linear PCA und PCA austauschbar verwenden. Dies erfolgt aus der Tatsache, dass die Linear PCA in der Praxis oft als die Standardvariante der PCA angesehen wird. Das Ziel von PCA ist, die Hauptachsen, bezeichnet als *Principal-Components* (PCs), zu finden, die die maximale Variabilität im ursprünglichen hochdimensionalen Datensatz erfassen. Durch die Projektion des originalen Datensatzes auf die gefundenen Principal-Components bewahrt die PCA so viel Informationen wie möglich aus dem ursprünglichen Datensatz im reduzierten niedrigdimensionalen Raum. Die Abbildung 3.2 zeigt beispielsweise eine grafische Darstellung einer PCA-Transformation in zwei Dimensionen. Der erste Graph von links repräsentiert eine Projektion eines Datensatzes im zweidimensionalen Raum. Der Datensatz besteht aus zwei Attributen, daher die zwei verschiedenen Achsen x_1 und x_2. Im zweiten Graphen wird die rote Ellipse verwendet, um die

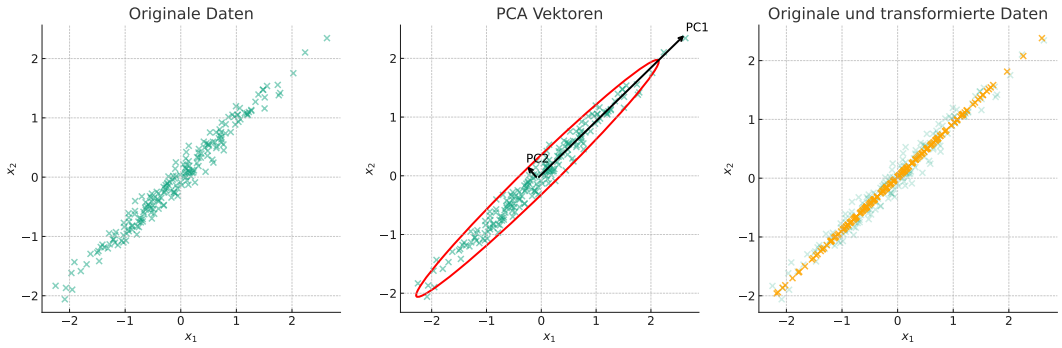


ABBILDUNG 3.1: Grafische Darstellung einer PCA-Transformation in nur zwei Dimensionen.

Streuung der Datenpunkte zu visualisieren, und die Pfeile zeigen die ersten zwei Principal-Components des Datensatzes, PC_1 und PC_2. Um diese beiden Vektoren zu finden, versucht PCA ein Optimierungsproblem zu lösen. Dieses Optimierungsproblem besteht darin, die richtigen Vektoren zu finden, die die Varianz innerhalb des Datensatzes maximieren. Mathematisch ausgedrückt sucht PCA nach einem Einheitsvektor u, der für eine Menge von Instanzen x_i die folgende Gleichung maximiert:

$$\max(\sum_i (x_i^\top \cdot u)^2) \quad \text{mit} \quad u^\top \cdot u = 1. \tag{3.2}$$

Mit einigen einfachen Umformung können wir jetzt diesen Ausdruck neu schreiben:

$$\max(u^\top \cdot C \cdot u) \qquad \text{mit} \quad u^\top \cdot u = 1. \tag{3.3}$$

Dabei repräsentiert C die Kovarianzmatrix des originalen Datensatzes. Konkret betrachtet C die Beziehung zwischen jedem Paar von Features in einem Datensatz. In unserem Szenario ist die Kovarianzmatrix eine 2×2 Matrix, die die Kovarianz zwischen x_1 und sich selbst, x_1 und x_2, x_2 und x_1, sowie x_2 und sich selbst enthält. Die Kovarianzmatrix wird dann in PCA verwendet, um die Struktur des Datensatzes in Bezug auf die Varianz der Daten zu erfassen. Die Diagonalelemente der Kovarianzmatrix repräsentieren die Varianz jedes einzelnen Features, also wie weit sich die Werte dieses Features im Durchschnitt vom Mittelwert entfernen. Die Nicht-Diagonalelemente geben die Kovarianz zwischen den Features an, also wie sich ein Feature verändert, wenn ein anderes Feature sich verändert. Ein hoher positiver Wert in der Kovarianzmatrix deutet darauf hin, dass die Features häufig gemeinsam auftreten; das heißt, die Auswahl eines Features macht die Auswahl des anderen Features wahrscheinlich. In einer SPL kann dies darauf hindeuten, dass zwei Features oft gemeinsam in Produktkonfigurationen vorkommen. Dies kann der Fall sein, weil zwischen den beiden Features eine Require-Beziehung besteht, oder sie könnten gemeinsam in einer Oder-Gruppe liegen, was bedeutet, dass ihre gemeinsame Präsenz auf eine komplementäre oder bedingt notwendige Beziehung innerhalb der Produktkonfigurationen hinweist.Umgekehrt deutet ein hoher negativer Wert darauf hin, dass die Auswahl eines Features tendenziell mit der Nicht-Auswahl des anderen Features korreliert. In der Praxis könnte dies bedeuten, dass, wenn ein bestimmtes Feature in einer Produktvariante enthalten ist, ein anderes Feature wahrscheinlich ausgelassen wird, was beispielsweise auf eine alternative oder ausschließende Beziehung zwischen den Features hindeutet. Aus der Gleichung 3.4 erstellen wir die *Lagrange*-Funktion, berechnen wir den Gradienten nach u und setzen wir ihn gleich null, und somit erhalten wir die Gleichung der Eigenvektoren und Eigenwerten von C:

$$C \cdot u = \lambda \cdot u. \tag{3.4}$$

Diese Gleichung bezeichnen wir als das *Eigenwertsproblem*. Die Lösung der Gleichung liefert uns die Haupteigenvektoren (Principal-Components) u und ihre entsprechenden Eigenwerte λ. Der optimale Principal-Component ist der Eigenvektor mit dem größten Eigenwert. In unserem konkreten Szenario repräsentiert PC1 den Eigenvektor mit dem größten Eigenwert.

Die zweite Komponente PC2 ist bei der Technik von PCA wieder ein Einheitsvektor, der aber keine Informationen enthält, die bereits im ersten Principal-Component PC1 enthalten sind. Aus geometrischer Sicht bedeutet dies, dass die zweite Komponente dem zu PC1 orthogonalen Unterraum angehört. Durch ähnliche Überlegungen wie zuvor finden wir, dass die PC2 durch den zweiten Eigenvektor der Kovarianzmatrix des Datensatzes gegeben ist. Nachdem wir jetzt den optimalen Principal-Component (PC1) gefunden haben, projizieren wir den Datensatz auf diesen gefundenen Principal-Component. Die Farbänderung von Blau zu Orange im dritten Diagramm illustriert die Projektion des originalen Datensatzes auf den ersten Principal-Component PC1. Die orange Datenpunkte repräsentieren den eindimensionalen Datensatz, der die maximale Varianz der Originaldaten beibehält.

In diesem illustrierten Beispiel des PCA-Prozesses wissen wir bereits im Voraus, dass der reduzierte Datensatz eindimensional sein soll. In der Praxis stellt jedoch die Bestimmung der geeigneten Dimensionalität des reduzierten Datensatzes eine bekannte Herausforderung für DR dar. Ein häufig verwendeter Ansatz in PCA ist die Berechnung des sogenannten kumulativen Prozentsatzes der erklärten Varianz (CPVE) [11]. Dieser Ansatz ermöglicht es, die Anzahl der Dimensionen, die für die gewünschte Informationserhaltung benötigt werden, empirisch festzulegen. Basierend auf CPVE wird die Anzahl der Principal-Components so festgelegt, dass ein gewählter Anteil γ erreicht oder überschritten wird. CPVE wird berechnet, indem wir die Eigenwerte der Kovarianzmatrix λ geordnet von der größten zur kleinsten Varianz summieren und durch die Gesamtsumme aller Eigenwerte teilen. Für jeden zusätzlichen Principal-Component wird dieser Prozess wiederholt, um den kumulativen Beitrag der erklärten Varianz zu bestimmen. Formal definieren wir CPVE für die ersten p Principal-Components in einem Datensatz mit D Dimensionen wie folgt:

Definition 5 (Kumulativen Prozentsatzes der erklärten Varianz (CPVE))
Sei λ_j die Eigenwerte der Kovarianzmatrix des Datensatzes D.

$$CPVE(p) = \left(\frac{\sum_{j=1}^{p} \lambda_j}{\sum_{j=1}^{D} \lambda_j} \right) \times 100. \tag{3.5}$$

In CPVE betrachten wir, wie viel Varianz jede Hauptkomponente des Datensatzes beinhaltet. Durch das iterative Aufsummieren der Varianz von der ersten bis zur aktuellen Hauptkomponente erhalten wir empirisch eine Zahl, die uns zeigt, wie viel Prozent der Gesamtinformation des Datensatzes wir bereits mit diesen ersten Hauptkomponenten erfassen können. In der Regel liegt der Wert von γ zwischen 75% und 95%. Die Anzahl der Principal-Components k wird dann so gewählt, dass der CPVE diesen Schwellenwert erreicht oder überschreitet.

Eine alternative Heuristik zur Bestimmung der optimalen Anzahl von Principal-Components in PCA besteht darin, den Anteil der erklärten Varianz für jeden Principal-Component zu visualisieren und anschließend nach dem sogenannten *Elbow*-Punkt im resultierenden Diagramm zu suchen. *Elbow*-Punkt ist der Punkt, an dem ein signifikanter Rückgang in der Grafik zu beobachten ist. Abbildung 3.2 visualisiert den Anteil der erklärten Varianz für die Tabelle 2.1 in Kapitel 2. Auf der X-Achse sind die Principal-Components dargestellt, während die Y-Achse den jeweiligen durch jeden Principal-Component erklärten Anteil der Datenvarianz anzeigt. Die Principal-Components sind durch eine durchgezogene Linie verbunden, wobei jeder Punkt einen Principal-Component repräsentiert. Die vertikale, gestrichelte rote Linie markiert den *Elbow*-Punkt bei dem fünften Principal-Component (PC5).

In diesem Abschnitt haben wir das grundlegende Konzept der Principal-Component-Analysis erläutert. Wir haben gezeigt, wie PCA einen neuen Unterraum innerhalb des Datensatzes findet, wobei möglichst viele Informationen des ursprünglichen Datensatzes erhalten bleiben. Da die Dimension des reduzierten Datensatzes oft nicht im Voraus bekannt ist, haben wir zwei unterschiedliche Ansätze vorgestellt, die im Kontext der PCA angewendet werden, um die optimale Dimensionalität des reduzierten Datensatzes zu bestimmen. Um die spezifischen Natur boolescher Konfigurationsräume zu berücksichtigen, betrachten

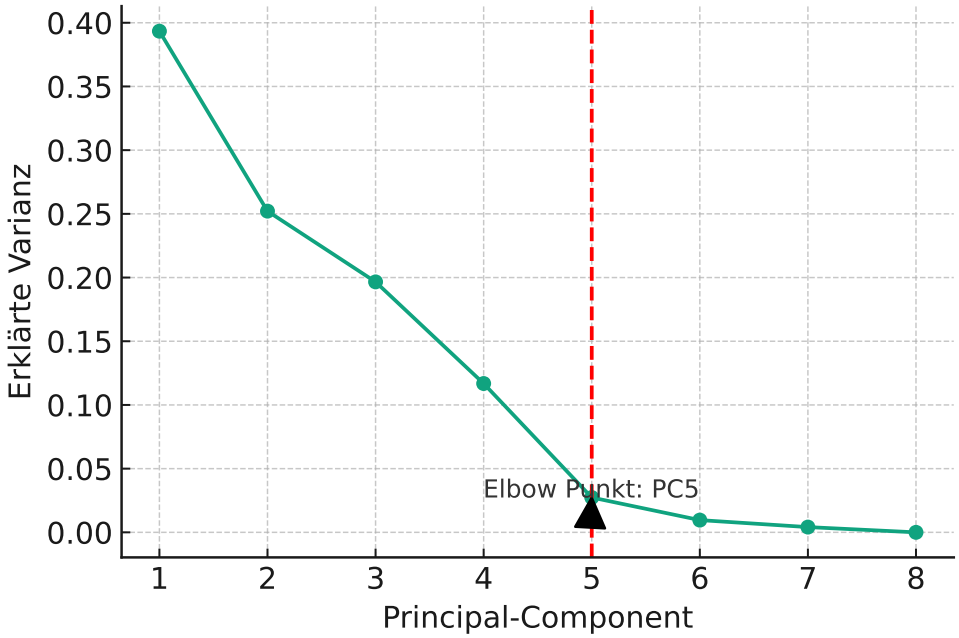


ABBILDUNG 3.2: Grafische Darstellung zur Bestimmung des Elbow-Punkts.

wir im nächsten Abschnitt eine angepasste Version von PCA, bekannt als Logistic Principal-Component-Analysis. Diese Variante wurde speziell für den Umgang mit diskreten Datensätzen, wie booleschen Daten, entwickelt.

3.3 PCA für boolesche Konfigurationsräume

In diesem Abschnitt diskutieren wir Logistic Principal-Component-Analysis, eine Erweiterung der klassischen PCA, die speziell für den Umgang mit nicht kontinuierlichen Datensätzen, insbesondere für boolesche Daten, entwickelt wurde. Wir erläutern das Logistic PCA-Modell aus einer probabilistischen Sichtweise. Anschließend definieren wir formal das Optimizationsproblem bei der Technik von Logistic Principal-Component-Analysis.

Logistic Principal-Component-Analysis (Logistic PCA) ist ein alternativer Ansatz zur Dimensionality-Reduction, der sich hauptsächlich auf die Wahrscheinlichkeitsverteilung der Daten konzentriert. Collins et al. [3] präsentierten die erste Version von Logistic PCA, die auf einer Generalisierung der Bernoulli-Verteilung der Daten basiert. Eine Bernoulli-Verteilung ist eine diskrete Verteilung mit nur zwei möglichen Ergebnissen, die im Kontext von booleschen Konfigurationsräumen als 1 oder 0 interpretiert werden. Betrachten wir erneut die Eingabematrix $X_{8\times 8}$ aus Abschnitt 1. Die Wahrscheinlichkeit für das Auswählen der ersten Spalte *Auto* wird mit p bezeichnet, während $1-p$ die Wahrscheinlichkeit für das Nicht-Auswählen dieses Features repräsentiert. Wir können diese Verteilung mithilfe des Log-Odds-Parameters $\theta_{\text{Auto}} = \log\left(\frac{1-p}{p}\right)$ und der logistischen Funktion $\sigma(\theta_{\text{Auto}}) = \left[1+e^{-\theta_{\text{Auto}}}\right]^{-1}$ beschreiben. Somit definieren wir die Bernoulli-Verteilung für eine boolesche Variable $x \in \{0,1\}$ wie folgt:

$$P(x|\theta_{Auto}) = \sigma(\theta_{Auto})^x \sigma(-\theta_{Auto})^{1-x}. \tag{3.6}$$

In unserem korrekten Szenario enthält die Spalte *Auto* dreimal eine 1 und zweimal eine 0. Daraus ergibt sich eine Wahrscheinlichkeit p von $\frac{2}{3}$ für das Auswählen von *Auto* und $\frac{1}{2}$ für das Nicht-Auswählen. Somit berechnen wir die Log-Odds-Parameter für das Feature *Auto*

als:

$$\theta_{Auto} = \log\left(\frac{1-p}{p}\right) = \log\left(\frac{1-\frac{3}{4}}{\frac{3}{4}}\right) = -0.48$$

Die Bernoulli-Verteilung für einem konkreten Eintrag $x = 0$ wird dann berechnet als:

$$P(x = 0|\theta_{Auto}) = \sigma(\theta_{Auto})^0 \sigma(-\theta_{Auto})^{1-0} = 1 \cdot (1 - \sigma(-0.48)) = 0.62$$

Eine Verallgemeinerung von Gleichung 3.6 für den gesamten Matrix $X_{8\times8}$ ergibt das Logistic PCA Modell:

$$P(X|\Theta_{8\times8}) = \prod_{8\times8} \sigma(\Theta_{8\times8})^{X_{8\times8}} \sigma(-\Theta_{8\times8})^{1-X_{8\times8}}, \tag{3.7}$$

wobei $\Theta_{8\times8}$ die Log-Odds der binären Matrix $X_{8\times8}$ darstellt. Die Log-Likelihood von binären Daten unter diesem Modell wird gegeben durch:

$$L = \sum_{8\times8} \left[X_{8\times8} \log \sigma(\Theta_{8\times8}) + (1 - X_{8\times8}) \log \sigma(-\Theta_{8\times8}) \right]. \tag{3.8}$$

Die grundlegende Idee hinter Logistic PCA besteht darin, die berechnete Log-Odds-Matrix $\Theta_{8\times8}$ in einer kompakten Form darzustellen, während gleichzeitig die wesentlichen Informationen der Daten erfasst werden. Dies wird erreicht, indem $\Theta_{8\times8}$ als Produkt zweier kleinerer Matrizen U und V dargestellt wird, welche die Log-Likelihood gemäß Gleichung 3.8 maximieren (zumindest lokal). Dabei ist U eine $8 \times k$-Matrix und V eine $k \times 8$-Matrix. Im Kontext der Dimensionality-Reduction repräsentiert U die reduzierte Form des ursprünglichen Datensatzes [20]. Formal definieren wir also das Optimierungsproblem von Logistic PCA wie folgt:

Definition 6 (Optimizationsproblem von Logistic PCA)
Gegeben sei ein binärer Datensatz X. Die Parameter U und V der Log-Odds-Matrix Θ in Bezug auf X werden so berechnet, dass sie die Log-Likelihood gemäß Gleichung 3.8 maximieren (zumindest lokal).

In diesem Abschnitt haben wir uns mit der Logistic Principal-Component-Analysis (Logistic PCA) beschäftigt, einer Erweiterung der klassischen PCA, die speziell für den Umgang mit nicht-kontinuierlichen, insbesondere booleschen Datensätzen entwickelt wurde. Zuerst haben wir das Logistic PCA-Modell aus einer probabilistischen Perspektive erläutert, anschließend haben wir das Optimierungsproblem bezüglich der Logistic PCA formal definiert.

In diesem Kapitel haben wir uns mit der Dimensionality-Reduction und deren Anwendung auf boolesche Konfigurationsräume auseinandergesetzt. Wir haben mit einer Einführung in die Grundlagen der Dimensionality-Reduction angefangen und erläutert, wie maschinelle Lern-Datensätze durch Matrizen repräsentiert werden. Anschließend haben wir formal das Problem der Dimensionality-Reduction definiert und zwei Ansätze zu dessen Umsetzung diskutiert. Ein Schwerpunkt lag auf der Principal-Component-Analysis als Methode zur Reduktion der Dimensionalität von Datensätzen. Wir haben den Prozess der PCA anhand eines zweidimensionalen Datensatzes illustriert. Anschießend haben wir zwei empirische Heuristiken zur Bestimmung der optimalen Dimensionalität des reduzierten Datensatzes erklärt. Im letzten Abschnitt haben wir die Logistic Principal-Component-Analysis beschrieben, eine speziell für den Umgang mit booleschen Datensätzen entwickelte Erweiterung der klassischen PCA. Wir haben das Logistic PCA-Modell aus einer probabilistischen Perspektive erklärt und danach das Optimierungsproblem von Logistic PCA definiert. Zusammenfassend haben wir in diesem Kapitel Methoden zur Bewältigung der Herausforderungen des Curse-of-Dimensionality in booleschen Konfigurationsräumen untersucht und gezeigt, wie Linear PCA und Logistic PCA dazu beitragen können, die Dimensionalität von Datensätzen zu reduzieren, während wesentliche Informationen erhalten bleiben. Im nächsten Kapitel beschreiben wir die Implementierungsdetails der Dimensionality-Reduction mittels Linear und Logistic PCA in Verbindung mit Feature-Modell-Lernen.

Kapitel 4

Implementierung

In diesem Kapitel diskutieren wir die praktische Umsetzung der Methoden und Konzepte zur Kodierung von booleschen Konfigurationsräumen im Kontext von Feature-Modell-Lernen. Zunächst beschreiben wir, wie wir mithilfe von SAT-Solvern Konfigurationsdaten generieren und diese in Form einer CSV-Tabelle für die Verwendung in AutoGluon vorbereiten. Anschließend gehen wir auf technische Details der Implementierung der Dimensionality-Reduction ein. Abschließend bieten wir einen Überblick darüber, wie wir AutoGluon nutzen, um maschinelle Lernmodelle auf automatisierte Weise zu trainieren und evaluieren.

Der erste Schritt in der Implementierung beinhaltet die Nutzung des externen Tools **FeatureIDE** [13]. FeatureIDE ist ein Open-Source-Framework, das in der Programmiersprache Java entwickelt wurde und speziell für die Entwicklung und Analyse von Feature-Modellen eingesetzt wird. Durch seine benutzerfreundliche Oberfläche erleichtert FeatureIDE das Erstellen von Feature-Modellen in Form von Feature-Diagrammen, die der FODA-Notation folgen. Nach der Modellierung eines Feature-Modells in FeatureIDE exportieren wir dieses Modell als DIMACS-Datei. Das DIMACS-Format ist ein standardisiertes Textformat, das primär zur Repräsentation von Booleschen SAT-Problemen in der konjunktiven Normalform (CNF) dient, wobei jede Zeile in der Datei eine Klausel repräsentiert. Die Abbildung 4.1 zeigt ein Beispiel für ein Dimacs-Datei. Die ersten drei Zeilen, die mit das Buchstabe 'c' beginnen, dienen als Kommentare, diese Zeilen werden vom SAT-Solver nicht verarbeitet. Die Zeile, die mit 'p' beginnt, bezeichnen wir als die Problemzeile. Durch diese Zeile definieren wir das Format und die Größe des SAT-Problems. In unserem konkreten Szenario besteht unser Problem aus eine CNF-Formel mit 3 Variablen und 3 Klauseln. Die folgenden Zeilen stellen die eigentlichen Klauseln der booleschen Formel dar. Jede Zahl repräsentiert eine Variable oder ihre Negation, beispielsweise repräsentiert die Zahl -1 die Negation der Variablen 1. Die Zahl 0 markiert das Ende einer Klausel. Im gegebenen Beispiel gibt es insgesamt drei Klauseln: $(1 \vee \neg 2), (\neg 3 \vee \neg 1)$ und $(1 \vee \neg 3)$. Nachdem das Feature-Modell in FeatureIDE erstellt und als Dimacs-Datei exportiert wird, beginnt der Parsing-Prozess, der in Abbildung 4.2 dargestellt ist. Die Kanten in der Abbildung zeigen die Reihenfolge der Operationen und die Richtung des Datenflusses zwischen den einzelnen Komponenten. Die größeren Boxen im Diagramm repräsentieren die Hauptphasen des Prozesses: *DimacsParser*, *SAT-Solver* und *CSV-Parser*. Der DimacsParser verwendet die Python-Bibliothek *PySat*[1] für den Import und die Validierung des DIMACS-Formats. Wir Betrachten nochmal das vorher

[1] PySat: A Python Toolkit for Prototyping with SAT Oracles, verfügbar unter `https://pysathq.github.io/`

```
 c  1  Auto
 c  2  Getriebe
 c  3  Airbags
 p cnf 3 3
 1 -2  0
-3 -1  0
 1  3  0
```

ABBILDUNG 4.1: Beispiel einer Dimacs-Datei

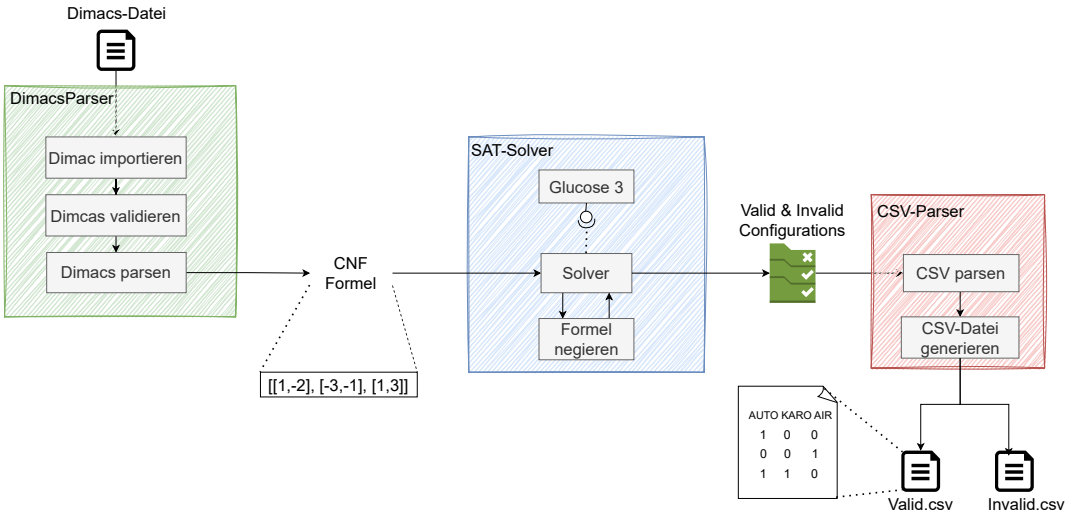


ABBILDUNG 4.2: Parsing und Generierung der Konfigurationsdaten

```
AUTO,KARO,AIR
1,0,0
0,0,1
1,1,0
```

ABBILDUNG 4.3: CSV-Datei für die gültigen Konfigurationen

vorgestellte Dimacs-Datei in Abb. 4.1, in diesem Beispiel liefert der DimacsParser eine Liste von Klauseln in CNF-Form, die wie folgt aussieht: $[[[1,-2],[-3,-1],[1,3]]]$. Der nächste Schritt im Prozess ist die Nutzung vom SAT-Solver. Der SAT-Solver ist ebenfalls in Python implementiert und verwendet den SAT-Solver *Glucose 3*[2] aus der PySAT-Framework. Der SAT-Solver verarbeitet die CNF-Formel und findet alle gültigen und ungültigen Konfigurationen, die die Bedingungen des Feature-Modells erfüllen oder verletzen. Für die Identifizierung ungültiger Konfigurationen wird die CNF-Formel negiert und erneut dem SAT-Solver übergeben. In der letzten Phase des Prozesses wird das Ergebnis des SAT-Solvers, also die Liste der gültigen und ungültigen Konfigurationen, in ein CSV-Format umgewandelt. Der CSV-Parser generiert daraus zwei CSV-Dateien: Eine für die gültigen und eine für die ungültigen Konfigurationen. Die Abbildungen 4.3 und 4.4 zeigen jeweils die CSV-Datei für den gültigen und ungültigen Konfigurationen aus dem illustrierten Beispiel in Abbildung 4.1. Die erste Zeile in der beiden Dateien enthält die Namen der Features der SPL, getrennt durch Kommas. In unserem konkreten Szenario haben wir insgesamt drei Features: Auto (AUTO), Karosserie (KARO) und Airbags (AIR). Die nachfolgenden Zeilen repräsentieren jeweils eine Konfiguration, wobei die Zahlen 0 und 1 festlegen, ob ein Feature ausgewählt ist oder nicht.

[2]Glucose 3: Ein SAT-Solver, verfügbar unter `https://www.labri.fr/perso/lsimon/glucose/`

```
AUTO,KARO,AIR
0,1,1
1,1,1
0,1,0
0,0,0
1,0,1
```

ABBILDUNG 4.4: CSV-Datei für die ungültigen Konfigurationen

Nach Abschluss des Parsing-Prozesses beginnt die Phase der Dimensionality-Reduction mittels Principal-Component-Analysis. Im Rahmen einem Preproccessing-Schritt werden die CSV-Dateien, die sowohl gültige als auch ungültige Konfigurationen enthalten, zusammengeführt und gemischt. Dies führt zu einem Datensatz, den wir als Basis für die Reduktion verwenden. Für die herkömmliche Linear PCA nutzen wir die *prcomp*-Funktion aus dem `stats`[3]-Paket der Programmiersprache R[4]. Diese Funktion akzeptiert verschiedene Parameter, wir beschränken uns aber in dieser Arbeit auf die Standardwerte und geben nur den Datensatz ein. Die *prcomp*-Funktion erzeugt ein PCA-Modell, welches die Principal-Components (Eigenvektoren) des Datensatzes enthält. Anschließend nutzen wir die *predict*-Funktion, um den ursprünglichen Datensatz auf den durch die PCA definierten niedrigerdimensionalen Raum zu projizieren. Für die Durchführung der Logistic PCA laden wir erneut die gültigen und ungültigen Konfigurationen aus den CSV-Dateien, führen wir eine Zusammenführung und Mischung der Daten durch und wenden wir anschließend die Funktion `logisticPCA`[5] aus dem gleichnamigen R-Paket an. Ähnlich wie bei der Linear PCA nutzen wir die Funktion *predict* zusammen mit dem resultierten Logistic PCA-Modell, um die Daten auf eine geringere Anzahl an Principal-Components zu reduzieren.

Nach Abschluss der Dimensionality-Reduction mittels der Linear PCA und Logistic PCA, verwenden wir AutoGluon[6] für das Training und die Evaluation. Diese Phase basiert auf der Implementierung, die ursprünglich von Weiß [23] in seiner Masterarbeit entwickelt wurde. Mit AutoGluon automatisieren wir den Prozess des Modelltrainings, der Modellauswahl und der Hyperparameter-Optimierung. Die Implementierung von Weiß liefert den Rahmen für die Nutzung von AutoGluon mit unseren Datensätzen. Diese umfasst beispielsweise die Sampling für die Datensätze, und die Evaluation durch manuell definierte Metriken.

In diesem Kapitel haben wir diskutiert, wie wir die Kodierung von booleschen Konfigurationsräumen praktisch umgesetzt haben. Zuerst haben wir den Ablauf der Generierung und Parsing von Konfigurationsdaten beschrieben. Anschießend haben wir gezeigt, wie Dimensionality-Reduction durch die R-Funktionen *prcomp* und `logisticPCA` implementiert wird. Abschließend haben diskutiert, wie AutoGluon zum Training und Evaluation verwendet wird. Im nächsten Kapitel präsentieren wir die Evaluierungsergebnisse zur Kodierungsstrategien des booleschen Konfigurationsraums.

[3] `stats`-Paket in R: Umfassende Sammlung statistischer Methoden und Modelle, verfügbar unter `https://stat.ethz.ch/R-manual/R-devel/library/stats/html/00Index.html`

[4] R: Eine Sprache und Umgebung für statistische Berechnungen und Grafiken, verfügbar unter `https://www.r-project.org/`

[5] `logisticPCA`: Eine Funktion für Logistic Principal-Component-Analysis in R, verfügbar im `logisticPCA`-Paket auf CRAN, siehe `https://CRAN.R-project.org/package=logisticPCA`

[6] AutoGluon: Automatisiertes maschinelles Lernen (AutoML) Toolkit, verfügbar unter `https://auto.gluon.ai`

Kapitel 5

Experimentelle Evaluierung

In diesem Kapitel präsentieren wir die Evaluierungsergebnisse zur Effizienz und Effektivität des Feature-Modell-Lernens mit und ohne Verwendung von Dimensionality-Reduction. Zunächst formulieren wir die Forschungsfragen, die wir untersuchen. Anschließend beschreiben wir die Fallstudien, die wir zur Beantwortung der Forschungsfragen verwenden. Danach erläutern wir das experimentelle Setup und wie wir die Evaluierungsdaten erhalten haben. Dann präsentieren wir die Evaluierungsergebnisse und diskutieren wir diese im Hinblick auf die Forschungsfragen. Abschließend gehen wir auf mögliche Gefährdungen der Validität ein.

5.1 Forschungsfragen

In diesem Abschnitt stellen wir die Forschungsfragen vor, die wir in diesem Kapitel untersuchen. Wir evaluieren die unterschiedlichen Ansätze des Feature-Modell-Lernens (FML) mit und ohne Einsatz von Dimensionality-Reduction (DR) anhand derselben Fallstudien. Wir vergleichen diese Ansätze hinsichtlich ihrer Effektivität, gemessen an Präzision und Recall, sowie ihrer Effizienz bezüglich der benötigten Rechenzeit. Beim Feature-Modell-Lernen mit Dimensionality-Reduction bewerten wir die Effizienz und Effektivität des Trained-Classifiers unter verschiedenen Werten für die Anzahl der Principal-Components. Die Forschung wird also durch die folgende Forschungsfragen geleitet:

- FF1: Was ist der optimale Wert für die Anzahl der Principal-Components bei dem Einsatz von Principal-Component-Analysis (PCA) in FML?
- FF2: Wie verändern sich Präzision und Recall beim Feature-Modell-Lernen mit DR im Vergleich zum Feature-Modell-Lernen ohne DR ?
- FF3: Wie verändert sich der Rechenzeit beim Feature-Modell-Lernen mit DR im Vergleich zum Feature-Modell-Lernen ohne DR ?

In diesem Abschnitt haben wir die Forschungsfragen vorgestellt, die wir in der Evaluierung untersuchen. Bevor wir zu den Ergebnissen der Evaluierung kommen, beschreiben wir die Fallstudien, die wir verwenden.

5.2 Fallstudien

In unserer experimentellen Evaluierung betrachten wir die gleichen drei Fallstudien, die Weiß [23] zum Evaluieren von FML verwendet hat. Unser Ziel ist es, die Effektivität und Effizienz des Feature-Modell-Lernens mit Dimensionality-Reduction mit den Ergebnissen ohne Dimensionality-Reduction zu vergleichen.

Die erste Fallstudie in unserer Evaluierung ist das *Body-Comfort-System* (BCS) [16]. Die Abbildung 5.1 zeigt dieses Modell in Form eines Feature-Diagramms. Die BCS Fallstudie besteht insgesamt aus 26 Features. Der Konfigurationsraum wird durch eine Oder-Gruppe, eine Alternative-Gruppe und verschiedene binäre Constraints geformt. Der Konfigurationsraum vom BCS hat 2^{26} mögliche Konfigurationen (d.h. 2^{26} verschiedene boolesche Vektoren

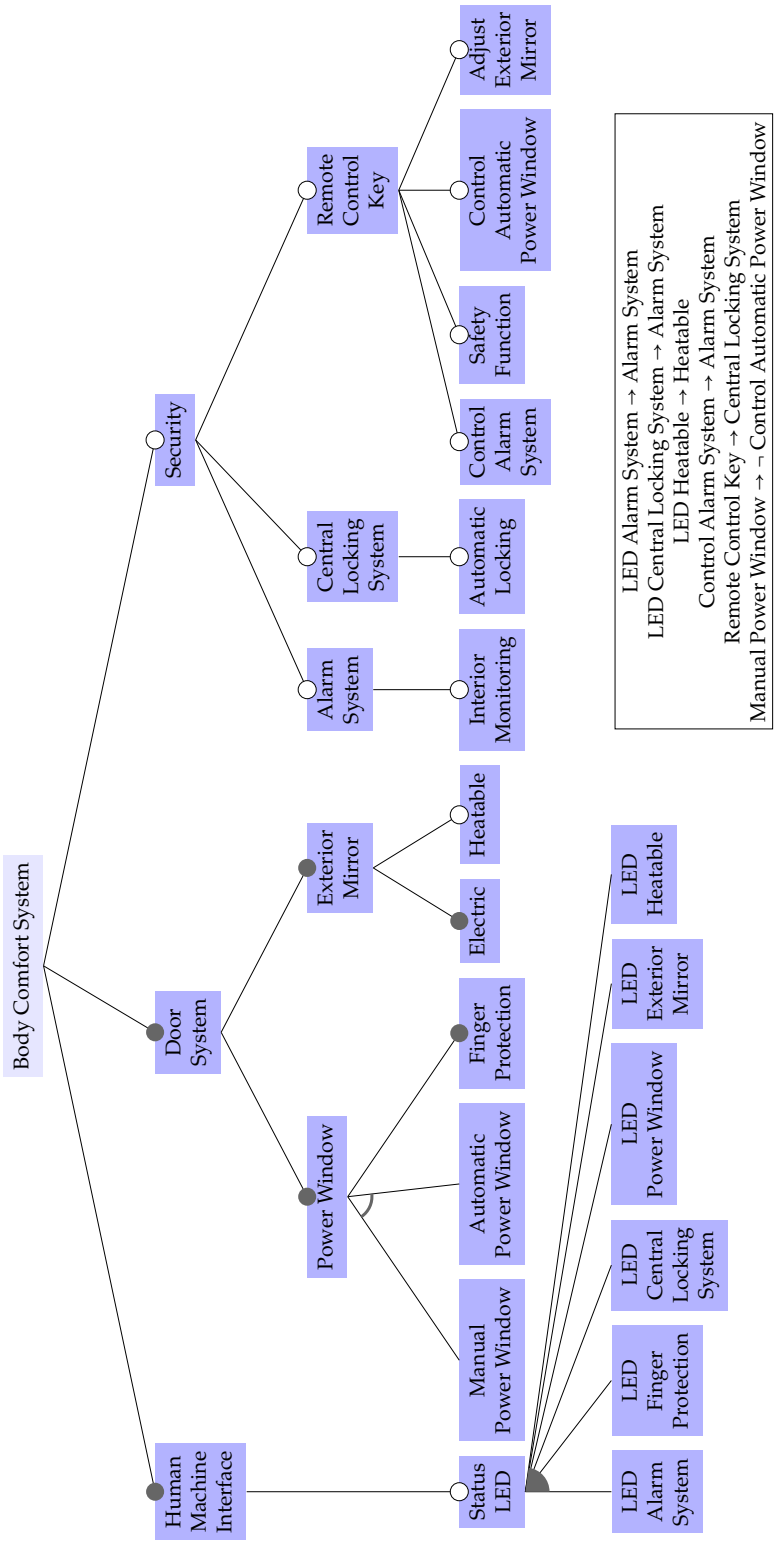


ABBILDUNG 5.1: Feature-Modell für das Body-Comfort-System

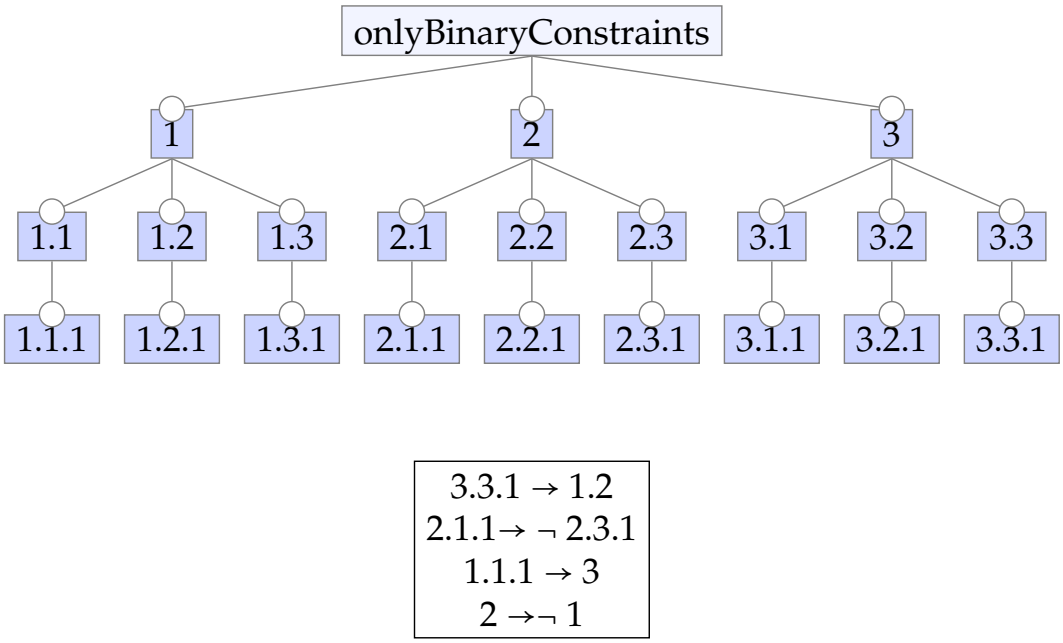


ABBILDUNG 5.2: Feature-Modell mit nur binären Constraints

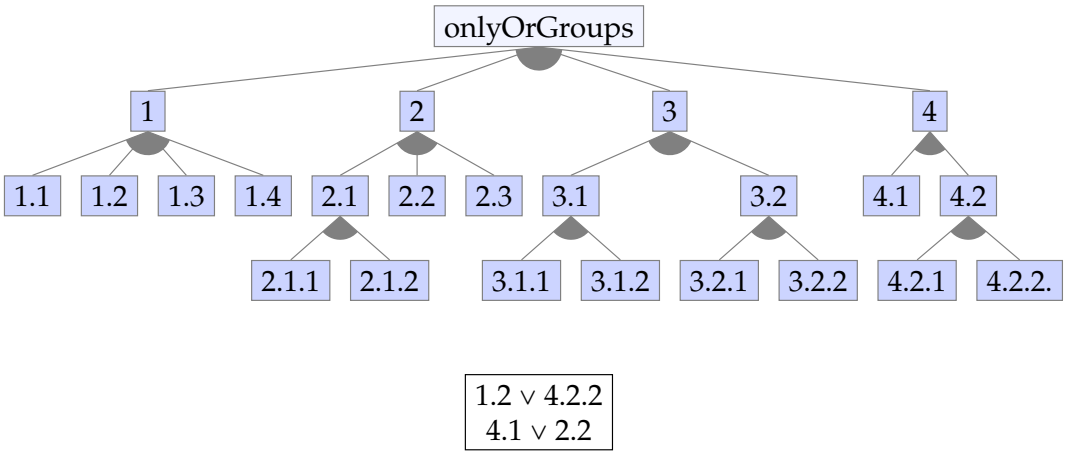


ABBILDUNG 5.3: Feature-Modell mit nur Oder-Gruppen

der Länge 26). Der gültige Konfigurationsraum des Feature-Diagramms umfasst 11616 gültige Konfigurationen. Das Verhältnis zwischen gültigem und ungültigem Konfigurationsraum liegt bei 1 zu 180, auf die nächste ganze Zahl gerundet.

Das Feature-Diagramm der zweiten Fallstudie, genannt *Only-Binary-Constraints*, ist in Abb. 5.2 dargestellt. Dieses System besteht aus 21 nicht-abstrakten Features, die alle optional sind. Das Feature-Diagramm wurde konstruiert, dass es nur binäre Constraints enthält. Der Konfigurationsraum wird durch zwei alternative Gruppen und viele zusätzliche binäre Constraints geformt. Der Konfigurationsraum besteht aus 2^{26} Konfigurationen. Der gültige Konfigurationsraum von Only-Binary-Constraints ist im Vergleich zur ersten Fallstudie kleiner und enthält insgesamt 1141 gültige Konfigurationen. Dies führt zu einem Verhältnis zwischen gültigem und ungültigem Konfigurationsraum von 1 zu 1.838, auf die nächste ganze Zahl gerundet.

Das Feature-Diagramm der dritten Fallstudie, genannt *Only-Or-Groups*, ist in Abb. 5.3 dargestellt. Das Feature-Diagramm enthält 23 nicht-abstrakte Features. Auch dieses Diagramm enthält nur optionale Features. Dieses Feature-Diagramm wurde so gestaltet, dass es nur

Oder-Gruppen enthält. Insgesamt wird der Konfigurationsraum von 11 Oder-Gruppen geformt. Die zwei Cross-Tree-Constraints führen zu teilweise überlappenden Oder-Gruppen, was bedeutet, dass ein Feature Teil mehrerer Oder-Gruppen ist. Der Konfigurationsraum hat eine Größe von 2^{23} Konfigurationen und enthält 18432 gültige Konfigurationen. Das Verhältnis zwischen gültigen und ungültigen Konfigurationen beträgt 1 zu 455, auf die nächste ganze Zahl gerundet.

In diesem Abschnitt haben wir die drei Fallstudien beschrieben, die wir zur Evaluierung des Feature-Modell-Lernens mit und ohne Dimensionality-Reduction verwenden. Im nächsten Abschnitt beschreiben wir unser experimentelles Setup und wie wir die Evaluierungsdaten für beide Ansätze entsprechend erfassen.

5.3 Datenerfassung

In diesem Abschnitt beschreiben wir, wie wir die Evaluierungsdaten gesammelt haben. Dies umfasst eine Beschreibung der Abfolge der Experimente und welche Parameter wir für die einzelnen Abläufe verwendet haben. Wir diskutieren dies für die einzelnen drei Forschungsfragen.

FF1: Effektivität von FML mit unterschiedlichen Anzahlen der Principal-Components. Wir haben untersucht, wie verschiedene Werte für die Anzahl der Principal-Components im PCA die Effektivität des resultierenden FML-Modells beeinflussen. Für die Evaluierung haben wir die Linear- sowie die Logistic Version von PCA eingesetzt. Dabei haben wir die drei unterschiedlichen Fallstudien, die wir in der letzten Abschnitt beschrieben haben, verwendet. Die Anzahl der gültigen Konfigurationen wurde konstant bei 50 gehalten, während die Anzahl der ungültigen Konfigurationen in Schritten von 2000 zwischen 2000 und 30000 Konfigurationen variiert wurde. Für jede Menge aus ungültigen Konfigurationen sowie für die Auswertungskriterien (Präzision und Recall) haben wir 5 Durchläufe für jeden Wert k (Anzahl der Principal-Components) zwischen 4 und 28 mit einem Schritt von 4 durchgeführt. Am Ende haben wir den Durchschnittswert der 5 Durchläufe berechnet, um eine robuste Schätzung der Modellleistung zu erhalten.

FF1: Effizienz von FML mit unterschiedlichen Anzahlen der Principal-Components. Wir haben die Effizienz von FML für unterschiedliche Anzahlen von Principal-Components für die drei Fallstudien bewertet, indem wir die Ausführungszeit der Reduktion sowie die Ausführungszeit des Feature-Modell-Lernprozesses untersucht haben. Sowohl für die Linear PCA als auch für Logistic PCA haben wir verschiedene Größen des ungültigen Konfigurationensraums berücksichtigt, während die Anzahl der gültigen Konfigurationen konstant bei 50 gehalten wurde. Die Anzahl der ungültigen Konfigurationen wurde in Schritten von 2000, beginnend bei 2000 Konfigurationen bis zu 30000 Konfigurationen, erhöht. Für jede Menge an Trainingsdaten pro Anzahl von Principal-Components haben wir 5 Durchläufe durchgeführt und haben wir anschießend die durchschnittliche Laufzeit berechnet. Damit ist die Datenerfassung bezüglich der Forschungsfrage 1 abgeschlossen. Lassen wir uns fortfahren, indem wir die Datenerfassung zur Beantwortung der Forschungsfrage 2 beschreiben, die sich auf das Feature-Modell-Lernen ohne den Einsatz von PCA bezieht.

FF2: Effektivität von FML ohne Verwendung von Dimensionality-Reduction. Wir haben die Verbesserung der Effektivität des Feature-Modell-Lernens ohne den Einsatz der beiden PCA-Strategien bewertet. Wir haben alle drei Bezugssysteme mit 50 gültigen und zwischen 2000 und 30000 ungültigen Konfigurationen evaluiert. Die Anzahl der ungültigen Konfigurationen haben wir in Schritten von 2000 erhöht. Jede Menge an ungültigen Konfigurationen für jede Fallstudie wurde 5 Mal bewertet. Bei jedem Durchlauf haben wir Präzision und Recall und anschließend den Durchschnittswert berechnet.

FF2: Effektivität von FML unter Verwendung von Dimensionality-Reduction. Wir haben die Verbesserung der Effektivität des Feature-Modell-Lernens unter Verwendung sowohl der Linear als auch der Logistic PCA-Technik bewertet. Wir haben den optimalen Wert der

Principal-Components k verwendet, den wir als Antwort für die erste Forschungsfrage gefunden haben. Mit dem optimalen k-Wert und den beiden PCA-Strategien haben wir eine Evaluierung über die drei Fallstudien mit 50 gültigen und zwischen 2000 und 30000 ungültigen Konfigurationen durchgeführt. Wie zuvor haben wir die Anzahl der ungültigen Konfigurationen in Schritten von 2000 erhöht. Jede Menge an ungültigen Konfigurationen für jede Fallstudie wurde 5 Mal evaluiert. Bei jedem Durchlauf haben wir Präzision und Recall berechnet und danach den Durchschnittswert ermittelt. Als nächstes beschreiben wir die Datenerfassung zur Beantwortung der Forschungsfrage 3.

FF3: Effizienz von FML mit und ohne Verwendung von Dimensionality-Reduction. Wir haben die Effizienz von FML mit und ohne Verwendung von PCA-Techniken bewertet, indem wir die Laufzeit des Feature-Modell-Lernprozesses in beiden Fällen untersucht haben. Im Falle der Verwendung von PCA haben wir den optimalen Wert der Principal-Components k benutzt, den wir als Antwort auf Frage 1 (FF1) ermittelt haben. Wir haben die Laufzeit der Dimensionality-Reduction gefolgt vom Feature-Modell-Lernen für verschiedene Trainingsdatengrößen jeder Fallstudie untersucht. Auch hier haben wir die Evaluierung mit 50 gültigen und zwischen 2000 und 30000 ungültigen Konfigurationen in Schritten von 2000 durchgeführt. Für jede Menge an Trainingsdaten pro Fallstudie haben wir 5 Durchläufe durchgeführt und anschließend die durchschnittliche Laufzeit berechnet. Im Fall, dass wir keine PCA eingesetzt haben, haben wir nur die Trainingszeit des Trained-Classifier berücksichtigt. Im anderen Fall haben wir die Summe der Zeiten für die PCA-Reduktion und das Training des Classifier berücksichtigt. Somit schließen wir den Abschnitt über die Datenerfassung ab. Im nächsten Abschnitt beschreiben wir die Ausführungsumgebungen, in denen wir FML mit und ohne die Verwendung von PCA-Techniken evaluiert haben.

5.4 Ausführungsumgebung

Für die experimentellen Evaluierungen unserer Forschungsarbeit haben wir die leistungsstarke Recheninfrastruktur der Universität Siegen, das *OMNI Cluster*, verwendet. Dieser Abschnitt gibt einen Überblick über die technischen Spezifikationen und die Softwareumgebung, die für die Durchführung unserer Experimente verwendet wurden.

Der OMNI-Cluster betreibt Rocky Linux, Version 8.6, und besteht aus 436 Rechenknoten. Jeder Knoten verfügt über zwei AMD EPYC 7452 CPUs mit 32 Kernen, die mit 2,35-3,35 GHz und einem 128 MB Cache betrieben werden, sowie über 256 GB DDR4 RAM mit einer Geschwindigkeit von 3200MHz. Jede PCA bzw. AutoGluon-Aufgabe wird auf einem Cluster-Knoten ausgeführt, wobei keine Parallelisierung über mehrere Knoten stattfindet. Für die verwendeten Software und Programmiersprachen haben wir die folgende Versionen genutzt:

- *Python 3.11.7*[1] in der Version 3.11.7
- R[2] in der Version 4.3.2
- *PySat*[3] in der Version 0.1.6.dev6
- AutoGluon[4] Tabular in der Version 0.8.0
- *Pandas*[5] in der Version 1.5.3
- `Logistic PCA`[6] in der Version 0.2

[1] Python 3.11.7: Programming language, Version 3.11.7, verfügbar unter `https://docs.python.org/3.11/`

[2] R: Eine Sprache und Umgebung für statistische Berechnungen und Grafiken, verfügbar unter `https://www.r-project.org/`

[3] PySat: A Python Toolkit for Prototyping with SAT Oracles, verfügbar unter `https://pysathq.github.io/`

[4] AutoGluon: Automatisiertes maschinelles Lernen (AutoML) Toolkit, verfügbar unter `https://auto.gluon.ai`

[5] pandas: Powerful data analysis toolkit for Python, Version 1.5.3, verfügbar unter `https://pandas.pydata.org/docs/`

[6] `logisticPCA`: Eine Funktion für Logistic Principal-Component-Analysis in R, verfügbar im `logisticPCA`-Paket auf CRAN, siehe `https://CRAN.R-project.org/package=logisticPCA`

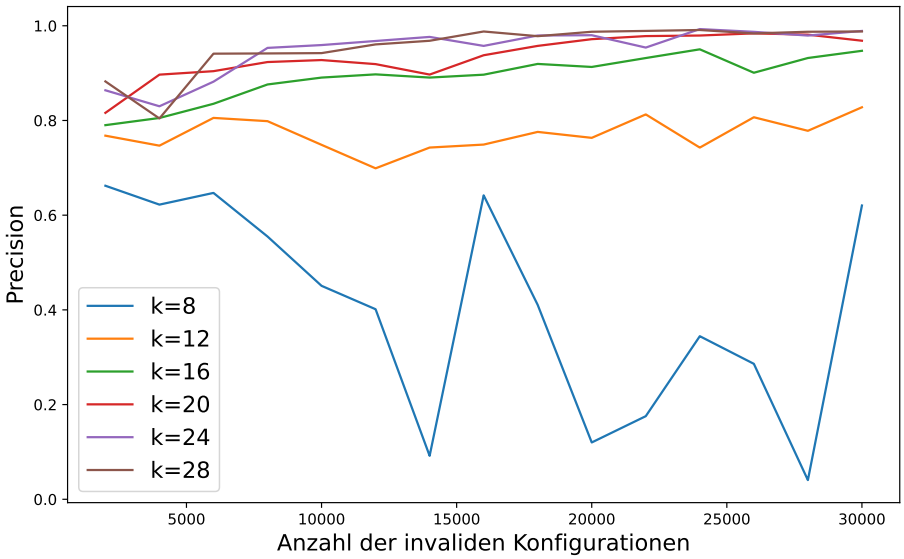


ABBILDUNG 5.4: Präzision des Feature-Modell-Lernens für das Body-Comfort-System (BCS) unter Einsatz von Linear PCA für verschiedenen k Werten

- SAT-Solver *Glucose 3*[7] in der Version 4.2.1

5.5 Ergebnisse

In diesem Abschnitt präsentieren und diskutieren wir die Ergebnisse unserer Evaluierung. Die Evaluierung ist in drei Teile gegliedert, die jeweils einer der drei Forschungsfragen entsprechen. Zunächst präsentieren wir die Ergebnisse hinsichtlich der Effektivität und Effizienz des Feature-Modell-Lernens unter Einsatz von Dimensionality-Reduction mit verschiedenen Werten für die Anzahl der Principal-Components (k). Basierend auf den optimalen Wert für k, stellen wir anschließend die Ergebnisse bezüglich der Effektivität des Feature-Modell-Lernens mit und ohne Verwendung von Dimensionality-Reduction vor. Abschließend präsentieren wir die Ergebnisse zur Bewertung der Effizienz des Feature-Modell-Lernens mit und ohne Einsatz von Dimensionality-Reduction.

5.5.1 FF1: Effektivität von FML unter Einsatz von PCA mit verschiedenen Werten für k.

In diesem Abschnitt präsentieren wir die Ergebnisse zur Effektivität von FML, wobei wir uns auf den Einsatz von PCA mit unterschiedlichen Principal-Components Anzahlen konzentrieren. Zuerst zeigen wir die Ergebnisse der Effektivität von FML in Kombination mit der Linear PCA auf. Danach gehen wir auf die Effektivität von FML unter Verwendung der Dimensionality-Reduction mit der Logistic PCA mit unterschiedlichen Anzahlen der Principal-Components ein. Wir halten dabei die folgende Reihenfolge ein: Zuerst werden wir die Evaluationsergebnisse für das Body-Comfort-System vorzustellen, dann die für das System mit ausschließlich binären Constraints und zuletzt die Ergebnisse für das System mit ausschließlich Oder-Gruppen. Für jede Fallstudie haben wir insgesamt 4 Graphen. Die ersten beiden Graphen beschreiben jeweils die Präzision und Recall unter Einsatz von der Linear PCA. Die anderen beiden Graphen beschäftigen sich jeweils mit der Präzisionwerten und Recallwerte unter Einsatz von Logistic PCA. Die Werten in allen Graphen präsentieren

[7] Glucose 3: Ein SAT-Solver, verfügbar unter `https://www.labri.fr/perso/lsimon/glucose/`

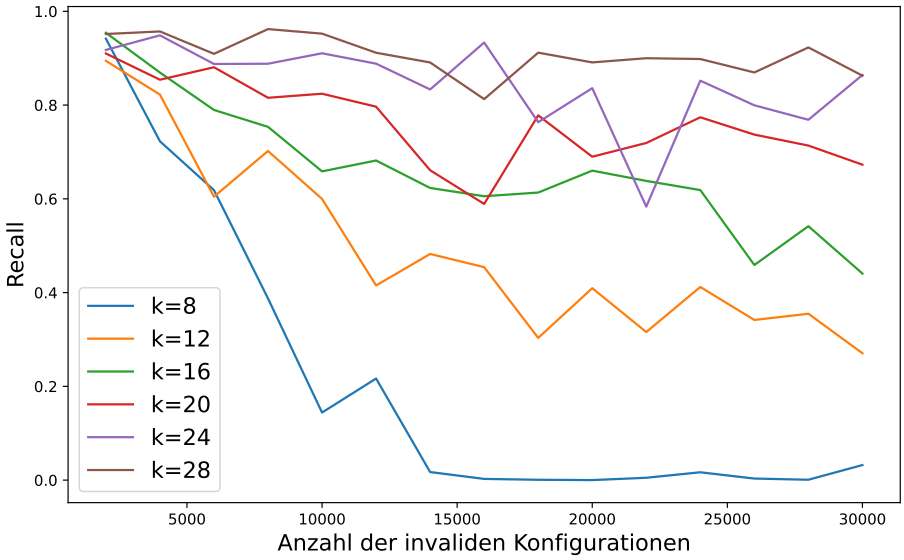


ABBILDUNG 5.5: Recall des Feature-Modell-Lernens für das Body-Comfort-System (BCS) unter Einsatz von Linear PCA für verschiedenen k Werten

den Durchschnitt von insgesamt 5 Messungen. Eine Messung repräsentiert die Durchführung von Dimensionality-Reduction auf dem Datensatz und anschließend das Trainieren des FML-Modells. Die X-Achse für alle Graphen repräsentiert die Anzahl der ungültigen Konfigurationen, die von 2000 bis 30000 in Schritten von 2000 reichen. Die Y-Achse in der Linear als auch in der Logistic PCA zeigt jeweils die gemessene Präzision und den Recall, die in einem Wertebereich von 0 bis 1 angegeben sind, wobei 1 die maximale Präzision bzw. den maximalen Recall darstellt. Die Linien repräsentieren die durchschnittliche Präzision und Recall für die unterschiedliche Werte von k, wobei jede Linie eine andere Anzahl von Principal-Components darstellt. Die Anzahl der Principal-Components variiert in Schritten von 4 von 8 bis der gesamten Anzahl der Features bei der betrachteten Fallstudie. Beispielsweise besteht die erste Fallstudie aus insgesamt 28 Features, und die dritte Fallstudie aus 24 Features. Das Only-Binary-Constraints-System hat insgesamt 22 Features, wir werden allerdings bei dieser Fallstudie nur die ersten 20 Principal-Components berücksichtigen, da 22 mit der Regel der Folge (konstante Schritten von 4) nicht einstimmt. Wir Beginnen zuerst mit der Präsentation der Evaluierungsergebnisse für das Feature-Modell-Lernen in Kombination mit der Linear PCA auf dem BCS-Feature-Modell.

Body-Comfort-System. Die beiden Abbildungen 5.4 und 5.5 zeigen jeweils die Ergebnisse der Präzision und des Recalls für das Body-Comfort-System (BCS) unter Einsatz der Linear PCA mit verschiedenen Anzahlen der Principal-Components. Für den Präzision-Graph beobachten wir, dass die Präzision für *k=8* eine sehr unterschiedliche Tendenz aufweist. Sie beginnt höher, fällt dann dramatisch ab und steigt erneut, bevor sie wieder sinkt und gegen Ende des Graphen erneut ansteigt. Für alle anderen k-Werten sehen wir tendenzielle Steigerung der Präzisionswerte mit der Erhöhung der Anzahl der ungültigen Konfigurationen. Die Linien für k=20 und k=24 scheinen insgesamt die höchste Präzision zu haben, da sie im oberen Bereich des Graphen verlaufen und selten unter eine Präzision von ca. 0.8 fallen. Die Linie für k=8 hat deutlich die niedrigste Präzision, mit einem signifikanten Abfall bis fast auf 0.1 und ist allgemein unterhalb der Präzisionswerte der anderen Linien.

Der Recall-Graph zeigt, dass die Linie für k=8 stark abfällt, wenn mehr ungültige Konfigurationen auftreten. Zum Beispiel sinkt der Recall-Wert bei ungefähr 14000 ungültigen Konfigurationen auf 0 und bleibt dann gleich, selbst wenn mehr ungültige Konfigurationen

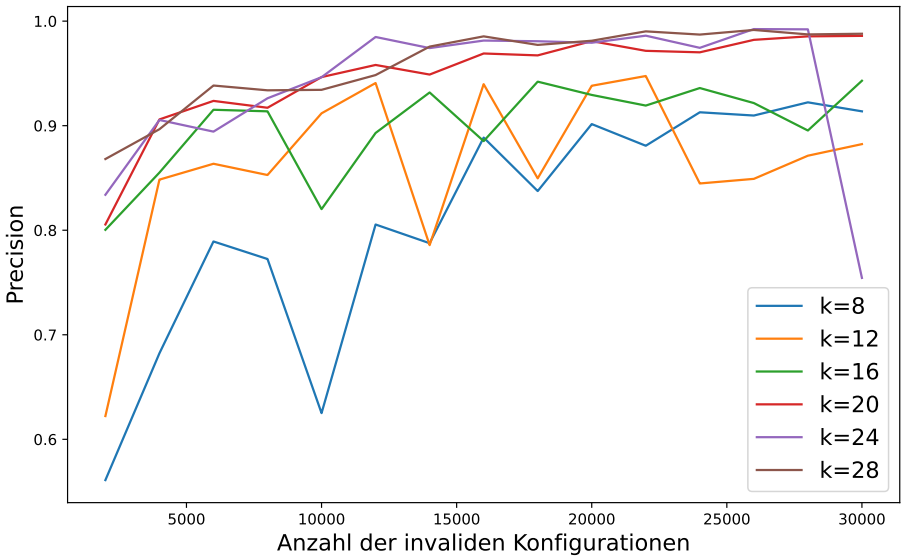


ABBILDUNG 5.6: Präzision des Feature-Modell-Lernens für das Body-Comfort-System (BCS) unter Einsatz von Logistic PCA für verschiedenen k Werten

hinzukommen. Die anderen Linien schwanken stark, sie sind allerdings meistens im mittleren Bereich (ungefähr 0.4) bis zum oberen Bereich (ungefähr 0.8) des Graphen. Die Linie mit den größten Wert k (k=28) hat deutlich die höchste Recall über den gesamten Bereich der ungültigen Konfigurationen hinweg.

Analog zur Linear PCA zeigen die beiden Abbildungen 5.6 und 5.7 die Präzisions- und Recall-Werte unter Einsatz von der Logistic PCA. Der Präzisionsgraph zeigt eine steigende Tendenz für alle k-Werte. Insbesondere erreicht die Linie für k=28 das höchste Niveau an Präzision und nähert sich dem Wert 1, wenn die Anzahl der ungültigen Konfigurationen zunimmt. Die Linien für niedrigere k-Werte wie k=8 und k=12 zeigen hingegen starke Schwankungen und liegen durchgehend auf einem niedrigeren Niveau.

Im Recall-Graph ist zu beobachten, dass die Tendenz für alle k-Werte abfallend ist. So sinkt beispielsweise der Recall für k=12 nach etwa 25000 ungültigen Konfigurationen auf den Wert 0. Für alle k-Werte gilt, dass der Recall zu Beginn oft höhere Werten aufweist (mehr als 0.8). Die Linie für k=28 zeigt wieder die höchsten Recall-Werte im Vergleich zu den anderen k-Werten. Wir präsentieren als nächstes die Ergebnisse bezüglich der zweiten Fallstudie, das nur binäre Constraints berücksichtigt.

Only-Binary-Constraints. Betrachten wir jetzt die Evaluationsergebnisse des zweiten Systems. Zuerst fangen wir mit der Linear PCA an. Die beiden Abbildungen 5.8 und 5.9 zeigen die Änderung für die Präzision bzw. den Recall mit der Erhöhung der Anzahl der ungültigen Konfigurationen. Im Präzisionsgraph ist es zu sehen, dass die Präzision bei verschiedenen k-Werten allgemein steigt, wenn mehr ungültige Konfigurationen vorhanden sind. Zum Beispiel startet die Linie für k=20 bei etwa 0.2 für ungefähr 2000 ungültige Konfigurationen und erreicht den höchsten Präzisionswert von ungefähr 0.7 bei über 30000 ungültigen Konfigurationen. Bei k=4 sinkt jedoch die Präzision, sie beginnt bei etwa 0.1 und fällt auf 0 ab.

Im Recallgraph zeigt sich ein ähnliches Muster wie in der ersten Fallstudie. Die Linie für $k = 20$ weist erneut die höchsten Recall-Werte auf, während die Recall-Werte für niedrigere k-Werte wie $k = 4$ und $k = 8$ deutlich geringer ausfallen. Die Linie für $k = 4$ ist die einzige Linie, die ab etwa 7000 ungültigen Konfigurationen einen Recall-Wert von weniger als ca. 0.2

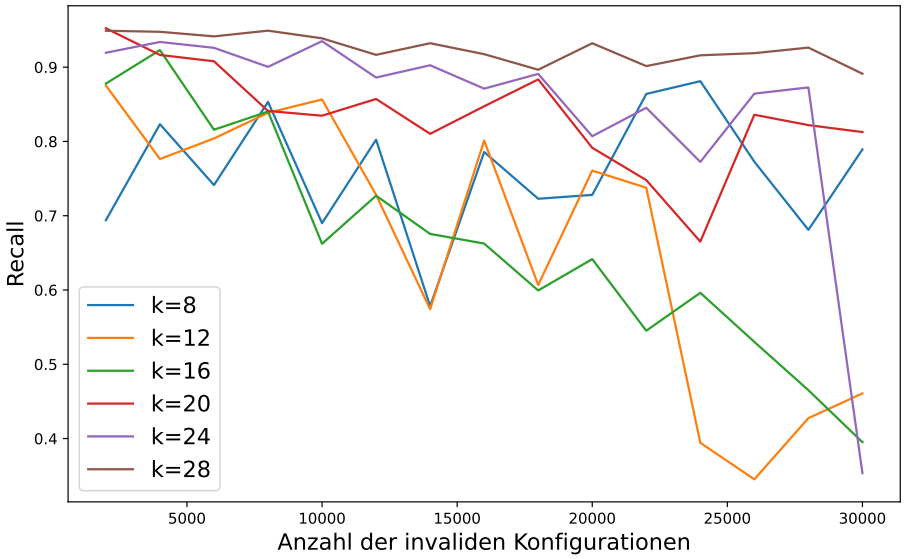


ABBILDUNG 5.7: Recall des Feature-Modell-Lernens für das Body-Comfort-System (BCS) unter Einsatz von Logistic PCA für verschiedenen k Werten

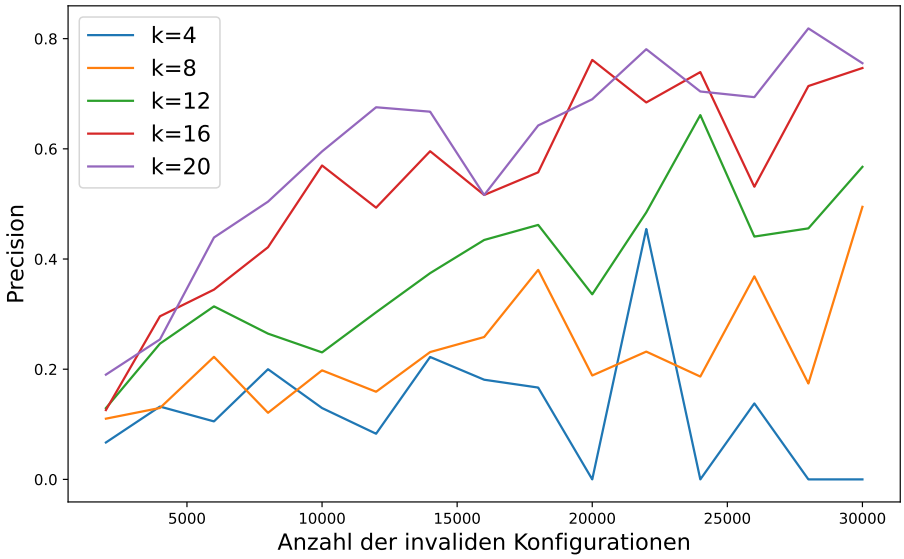


ABBILDUNG 5.8: Präzision des Feature-Modell-Lernens für das Feature-Modell mit nur binären Constraints (Only-Binary-Constraints) unter Einsatz von Linear PCA für verschiedenen k Werten

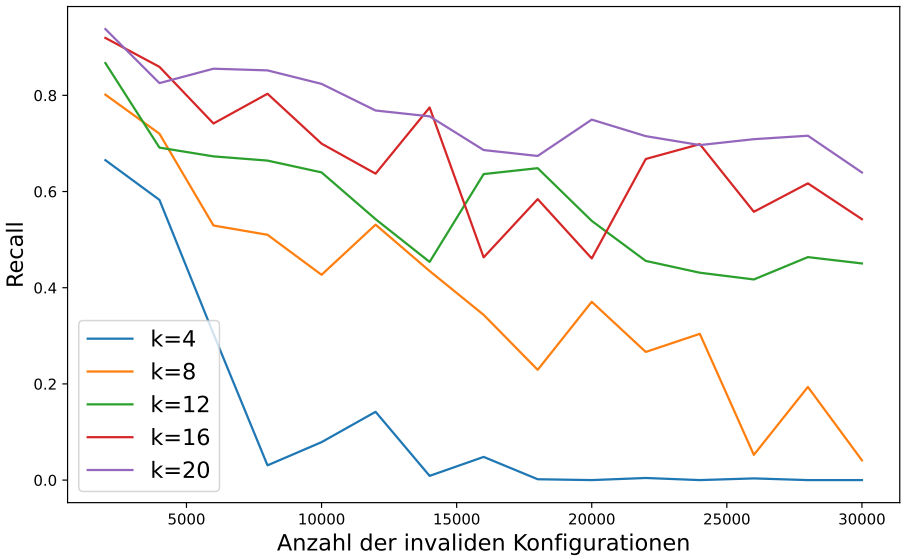


ABBILDUNG 5.9: Recall des Feature-Modell-Lernens für das Feature-Modell mit nur binären Constraints (Only-Binary-Constraints) unter Einsatz von Linear PCA für verschiedenen k Werten

erreicht und sich ab circa 18000 ungültigen Konfigurationen auf den Wert 0 abfällt. Der Recall bleibt in diesem Fall bei 0, obwohl die Anzahl der Konfigurationen weiter erhöht wird.

Nach der Analyse der Ergebnisse mit der Linear PCA für das *Only-Binary-Constraint*-Feature-Modell werden wir nun die Ergebnisse unter Verwendung der Logistic PCA betrachten. Die beiden Abbildungen 5.10 und 5.11 illustrieren jeweils die durchschnittliche Präzision und den durchschnittlichen Recall für verschiedene Anzahlen der Principal-Components. Für die Präzision sehen wir, dass die Werte für kleinere k-Werte (k=4 und k=8) generell niedriger beginnen und über den Bereich der ungültigen Konfigurationen hinweg eine sehr leicht aufsteigende Tendenz zeigen. Im Recall-Diagramm zeigt sich ein ähnliches Muster wie in den vorherigen Recall-Graphen. Die Linie für k=20 weist im Vergleich zu den anderen k-Werten die höchsten Recall-Werte auf, allerdings nimmt der Recall mit der Zunahme der Anzahl an ungültigen Konfigurationen ab. Für k=4 hingegen sind die Recall-Werte durchweg niedrig und schwanken in einem Bereich von 0 bis ca. 0.2. Wir betrachten als nächstes die Evaluierungsergebnisse für die dritte Fallstudie.

Only-Or-Groups. Wir analysieren nun die Ergebnisse der dritten Fallstudie, das *Only-Or-Groups*-System, welches ausschließlich Oder-Gruppen betrachtet. Zuerst stellen wir die Resultate hinsichtlich der Linear PCA vor und im Anschluss die Ergebnisse für die Logistic PCA. Die Präzisionsgrafik 5.12 zeigt, dass die Linien für k=16, k=20 und k=24 eine aufsteigende und stabile Tendenz aufweisen, die von ungefähr 0.4 auf nahezu 0.8 ansteigt. Bei den Linien für k=8 und k=12 sind die Trends nicht klar erkennbar und weisen starke Schwankungen auf. Bei niedrigeren k-Werten wie k=8 und k=12 ist die Präzision sehr gering und fällt für einige Mengen von ungültigen Konfigurationen auf 0. Die Linie für k=24 zeigt oft den höchsten Präzisionswert, unabhängig von der Anzahl der ungültigen Konfigurationen.

Auf der anderen Seite zeigt die Abbildung 5.13 die Recall-Werte für das Feature-Modell *Only-Or-Groups* unter Verwendung der Linear PCA für verschiedene k-Werte. Für k=8 beobachten wir eine signifikante Abnahme der Recall-Werte, welche ab etwa 6000 ungültigen Konfigurationen schnell auf circa 0 fällt. Die Linie für k=12 erreicht ebenfalls einen Recall von 0, allerdings erst bei einer höheren Anzahl von etwa 10000 ungültigen Konfigurationen. Die anderen dargestellten Linien für die höheren k-Werte von k=16, k=20 und k=24 zeigen

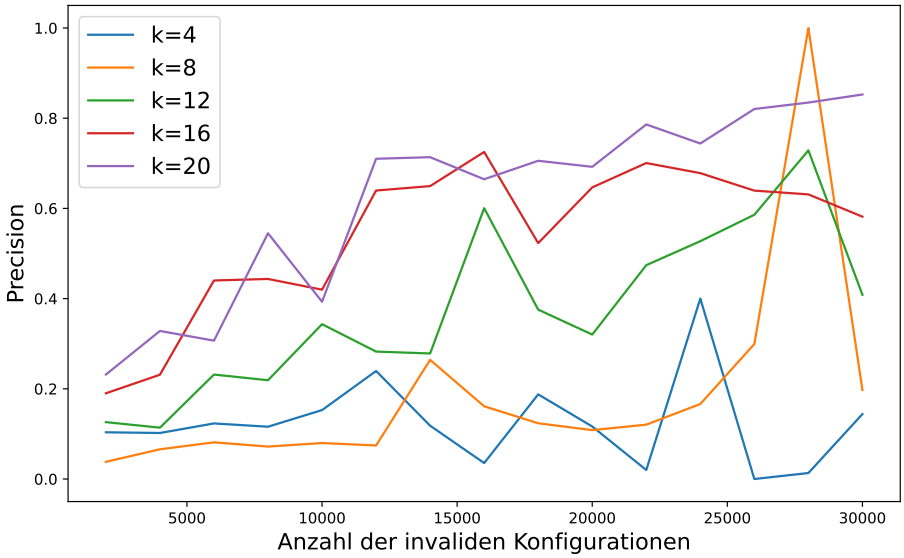


ABBILDUNG 5.10: Präzision des Feature-Modell-Lernens für das Feature-Modell mit nur binären Constraints (Only-Binary-Constraints) unter Einsatz von Logistic PCA für verschiedenen k Werten

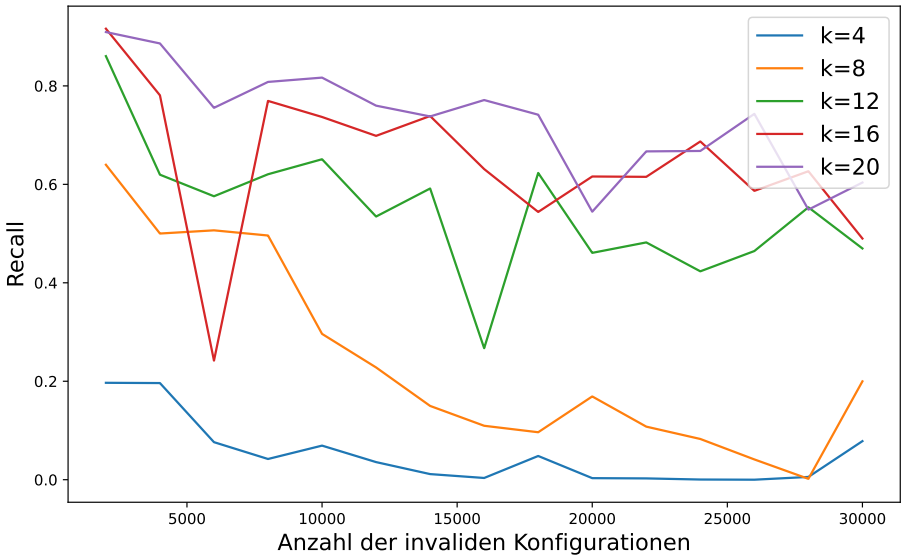


ABBILDUNG 5.11: Recall des Feature-Modell-Lernens für das Feature-Modell mit nur binären Constraints (Only-Binary-Constraints) unter Einsatz von Logistic PCA für verschiedenen k Werten

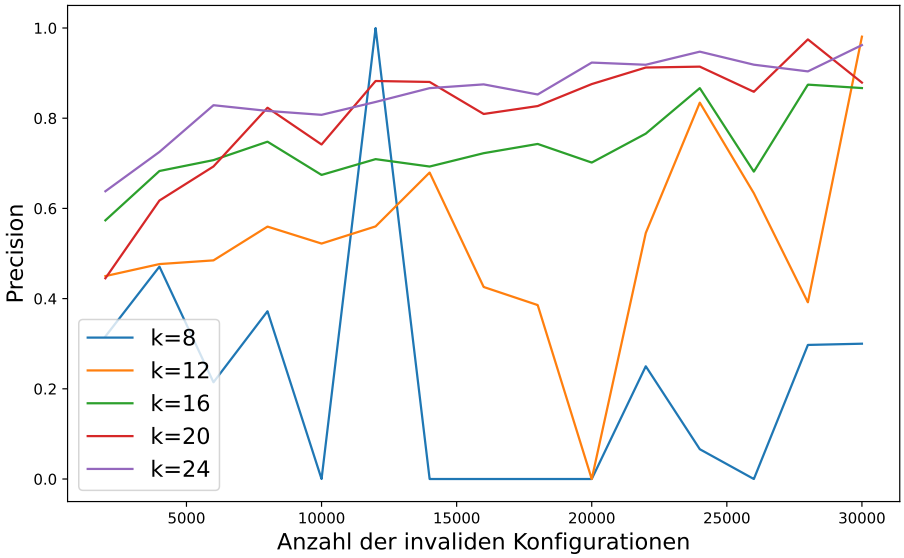


ABBILDUNG 5.12: Präzision des Feature-Modell-Lernens für das Feature-Modell mit nur Oder-Gruppen (Only-Or-Groups) unter Einsatz von Linear PCA für verschiedenen k Werten

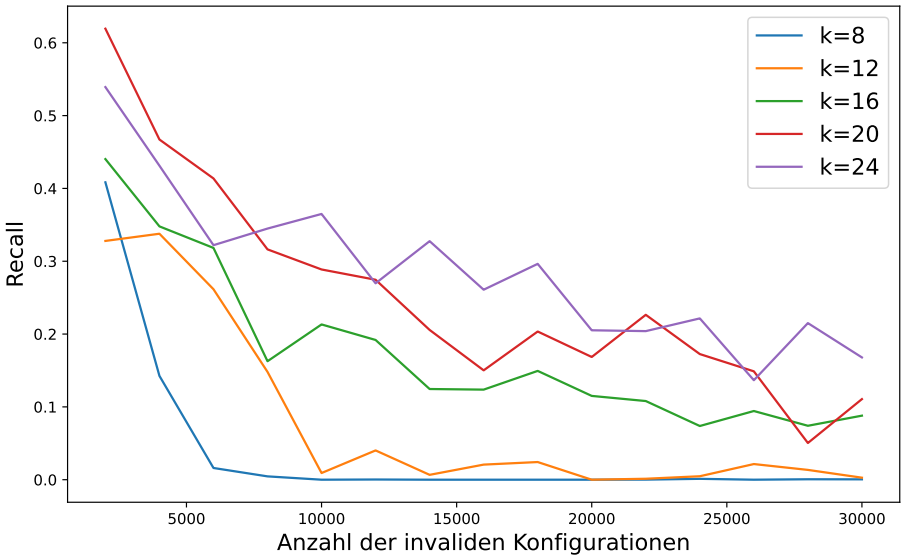


ABBILDUNG 5.13: Recall des Feature-Modell-Lernens für das Feature-Modell mit nur Oder-Gruppen (Only-Or-Groups) unter Einsatz von Linear PCA für verschiedenen k Werten

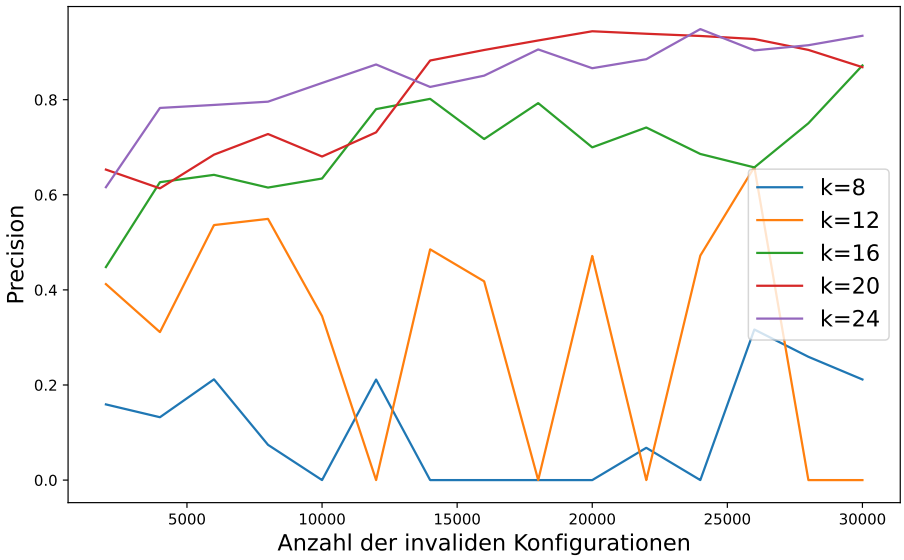


ABBILDUNG 5.14: Präzision des Feature-Modell-Lernens für das Feature-Modell mit nur Oder-Gruppen (Only-Or-Groups) unter Einsatz von Logistic PCA für verschiedenen k Werten

keinen Abfall auf 0, allerdings sinkt der Recall mit steigender Anzahl ungültiger Konfigurationen. Außerdem erkennen wir von dem Graph, dass die Linie für k=24 mit einem niedrigeren Recall im Vergleich zu k=20 beginnt, übertrifft diesen jedoch, sobald die Zahl der ungültigen Konfigurationen auf ungefähr 8000 ansteigt.

Wir betrachten als nächstes die Ergebnisse im Zusammenhang mit der Logistic PCA. Die Abbildung 5.14 zeigt ein ähnliches Muster wie bei der Linear PCA. Die Linien für niedrige Werte von k, wie k=8 oder k=12, weisen eine geringere Präzision auf, die bei circa 10000 auf den Wert 0 abfallen. Andererseits zeigen die Linien für höhere k-Werte wie k=24 eine höhere Präzision und steigen mit der Anzahl der ungültigen Konfigurationen weiter an, bis sie sich bei etwa 0.9 stabilisieren.

Hinsichtlich des Recalls zeigt die Abbildung 5.15 erneut die Veränderung des Recalls für verschiedene Werte von k. Für alle k-Werte ist eine abnehmende Tendenz zu beobachten. Bei niedrigen k-Werten wie k=8 oder k=12 erreicht der Recall sein Minimum von 0 und steigt nicht weiter an, obwohl die Anzahl der ungültigen Konfigurationen zunimmt. Die Linie für k=24 weist oft die höchste Präzision für die Anzahl der ungültigen Konfigurationen auf, mit der Ausnahme von etwa 23000 ungültigen Konfigurationen, bei denen die Linie mit k=20 die höchste Spitze erreicht. Als Nächstes präsentieren wir die Evaluationsergebnisse, um die Laufzeit für verschiedene k-Werte zu vergleichen.

5.5.2 FF1: Effizienz von FML unter Einsatz von PCA für verschiedenen k-Werten

Wir präsentieren die Bewertung der Laufzeit des Feature-Modell-Lernens unter Verwendung von Linear- und Logistic PCA mit verschiedenen k-Werten. Zuerst stellen wir die Ergebnisse für die BCS-Fallstudie, dann für das Only-Binary-Constraints-System und schließlich für das Only-Or-Group-System vor. Für jede Fallstudie betrachten wir zwei Graphen jeweils für Linear und Logistic PCA. Folgendes gilt für alle betrachteten Graphen: Die X-Achse repräsentiert die Anzahl der ungültigen Konfigurationen, die von 2000 bis 30000 reichen. Die Y-Achse in der Linear als auch in der Logistic PCA zeigt jeweils die gemessene

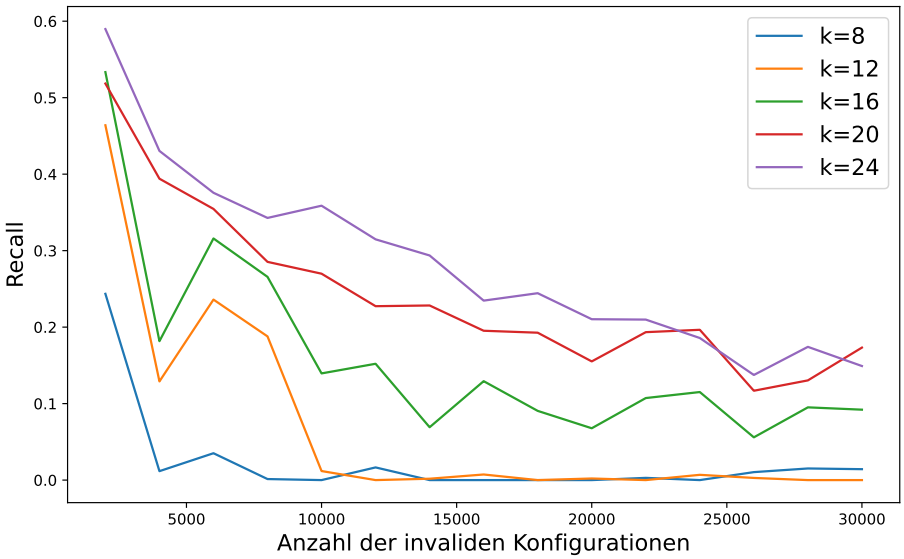


ABBILDUNG 5.15: Recall des Feature-Modell-Lernens für das Feature-Modell mit nur Oder-Gruppen (Only-Or-Groups) unter Einsatz von Logistic PCA für verschiedenen k Werten

Laufzeit, welche aus der Summe der AutoGluon-Trainingzeit Die PCA-Reduktionszeit berechnet wird. Die Linien repräsentieren die Laufzeit für die verschiedene k-Werten. Auch hier repräsentieren die Laufzeitwerten den Durchnittswert nach 5 Messungen. Betrachten wir als erstes die Ergebnisse hinsichtlich der ersten Fallstudie.

Body-Comfort-System. Die Abbildung 5.16 zeigt die Laufzeit für jeden k-Wert bei Verwendung von Linear PCA. Wir können sehen, dass die blaue Linie für k=12 eine längere Ausführungszeit im Vergleich zu den anderen k-Werten hat. Die braune Linie für k=8 zeigt jedoch ab ungefähr 15000 ungültigen Konfigurationen die niedrigste Laufzeit. Alle übrigen Linien verändern sich in gleicher Weise mit dem Anstieg der Anzahl ungültiger Konfigurationen.

Die Abbildung 5.17 zeigt die gleichen Evaluierungswerte, jedoch unter Verwendung von Logistic PCA. Es gibt eine aufsteigende Tendenz der Laufzeit mit der Erhöhung der Anzahl ungültiger Konfigurationen. Auch hier zeigen alle Linien eine steigende Tendenz, wobei die Linie für k=12 die höchsten Laufzeiten aufweist. Die anderen Linien verzeichnen eine ähnliche Entwicklung mit zunehmender Anzahl ungültiger Konfigurationen. Wir betrachten als nächstes die Ergebnisse für die zweite Fallstudie.

Only-Binary-Constraints. Wir betrachten die Bewertungsergebnisse für die zweite Fallstudie, das Only-Binary-Constraints-System. Die Bewertungsergebnisse für die Linear PCA sind in Abb. 5.18 dargestellt. Das Diagramm zeigt separate Bewertungsergebnisse für verschiedene k-Werte als Linien. Die Linie mit k=4 hat deutlich die niedrigste Laufzeit im Verlauf des gesamten Graphen. Die anderen Linien ändern sich in gleicher Weise, wobei die Linie k=12 im oberen Bereich liegt.

Analog dazu zeigt die Abbildung 5.19 die Ergebnisse für die Logistic PCA. Auch für Logistic PCA bleibt die Linie mit k=4 diejenige mit der geringsten Laufzeit. Die Linie für k=12 liegt weiterhin auf dem höchsten Niveau, während die anderen Linien sich zwischen den Linien k=4 und k=12 aufsteigend entwickeln. Wir präsentieren jetzt die Ergebnisse für die dritte Fallstudie.

Only-Or-Groups. Für die dritte Fallstudie haben wir zwei Abbildungen 5.20 und 5.21 für

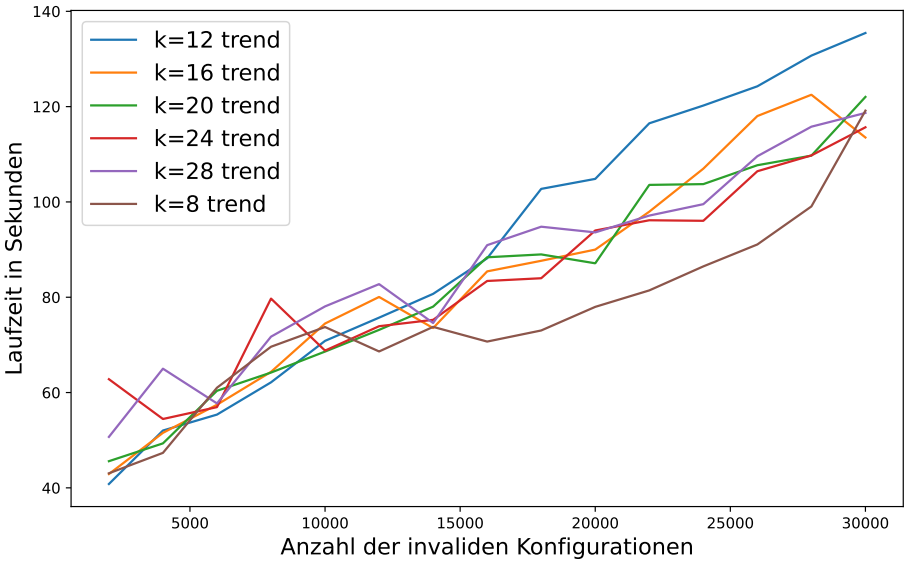


ABBILDUNG 5.16: Die Ausführungszeit des Feature-Modell-Lernens zusammen mit dem Reduktionsprozess für das Body-Comfort-System (BCS) unter Einsatz von Linear PCA für verschiedenen k Werten

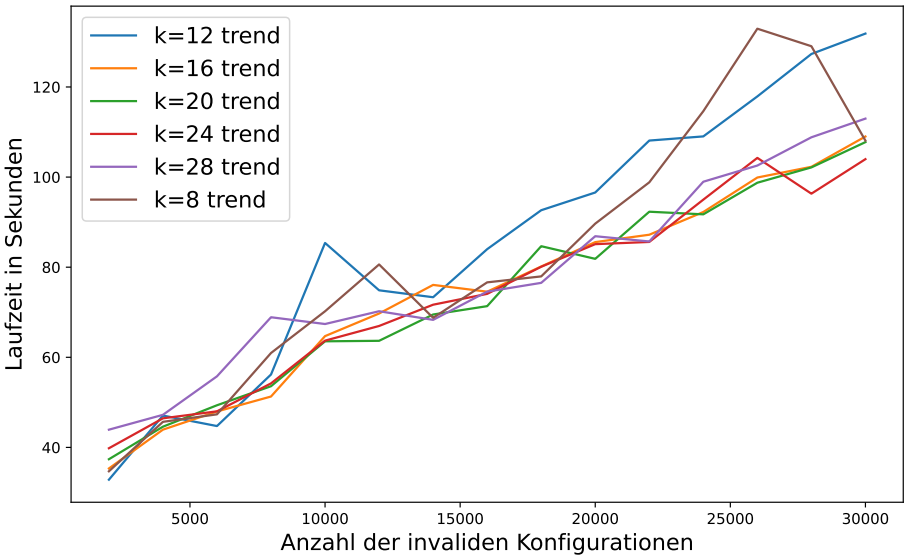


ABBILDUNG 5.17: Die Ausführungszeit des Feature-Modell-Lernens zusammen mit dem Reduktionsprozess für das Body-Comfort-System (BCS) unter Einsatz von Logistic PCA für verschiedenen k Werten

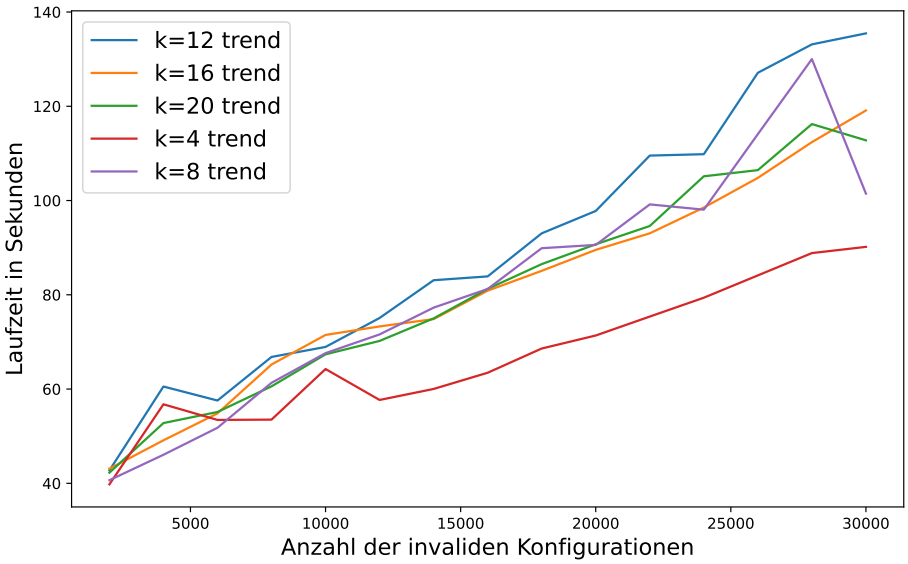


ABBILDUNG 5.18: Die Ausführungszeit des Feature-Modell-Lernens zusammen mit dem Reduktionsprozess für das System mit nur binären Constraints (Only-Binary-Constraints) unter Einsatz von Linear PCA für verschiedenen k Werten

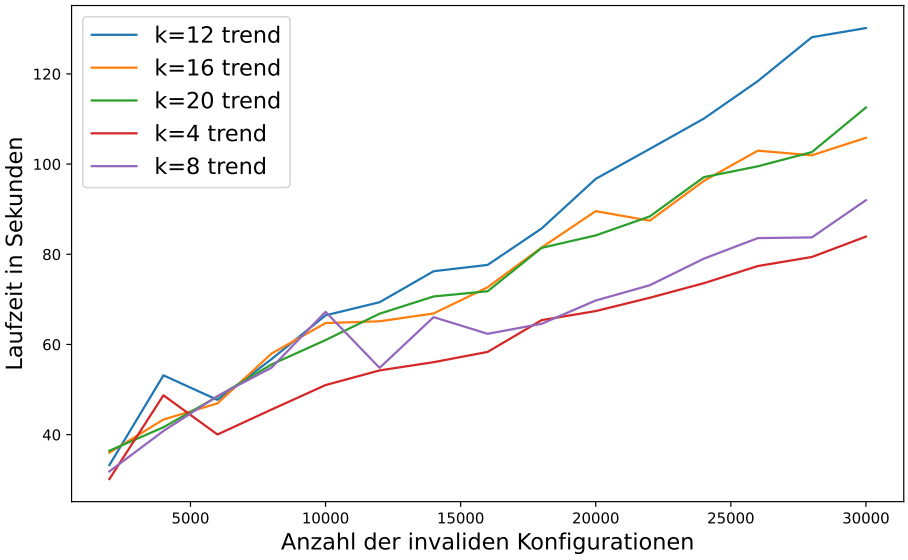


ABBILDUNG 5.19: Die Ausführungszeit des Feature-Modell-Lernens zusammen mit dem Reduktionsprozess für das System mit nur binären Constraints (Only-Binary-Constraints) unter Einsatz von Logistic PCA für verschiedenen k Werten

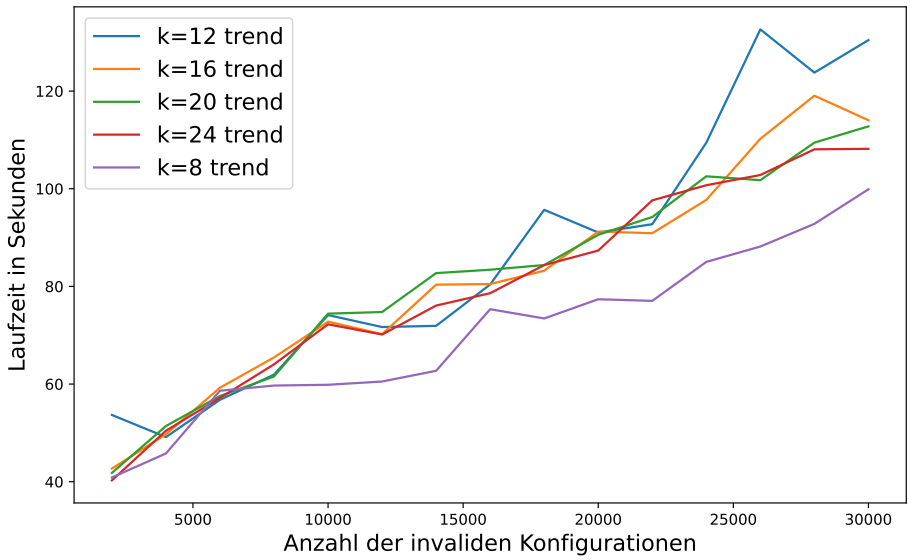


ABBILDUNG 5.20: Die Ausführungszeit des Feature-Modell-Lernens zusammen mit dem Reduktionsprozess für das System mit nur Oder-Gruppen (Only-Or-Groups) unter Einsatz von Linear PCA für verschiedenen k Werten

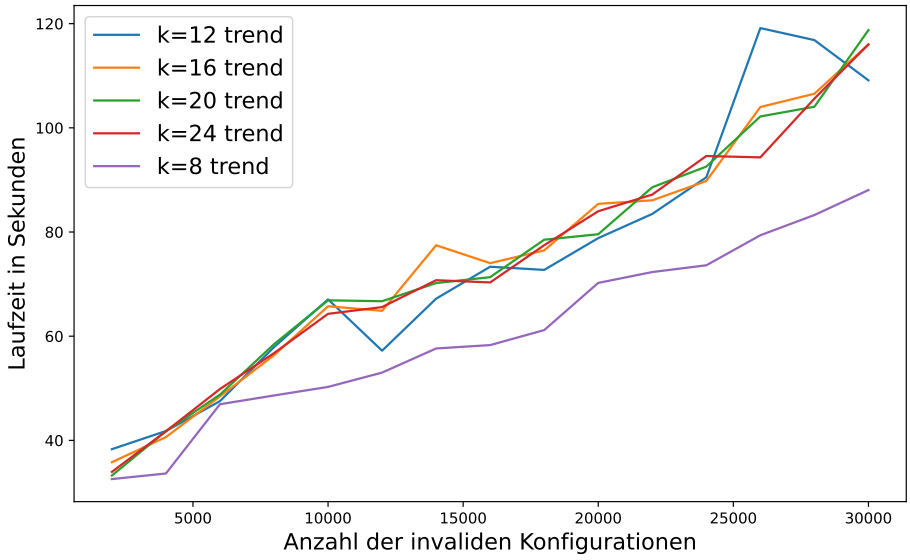


ABBILDUNG 5.21: Die Ausführungszeit des Feature-Modell-Lernens zusammen mit dem Reduktionsprozess für das System mit nur Oder-Gruppen (Only-Or-Groups) unter Einsatz von Logistic PCA für verschiedenen k Werten

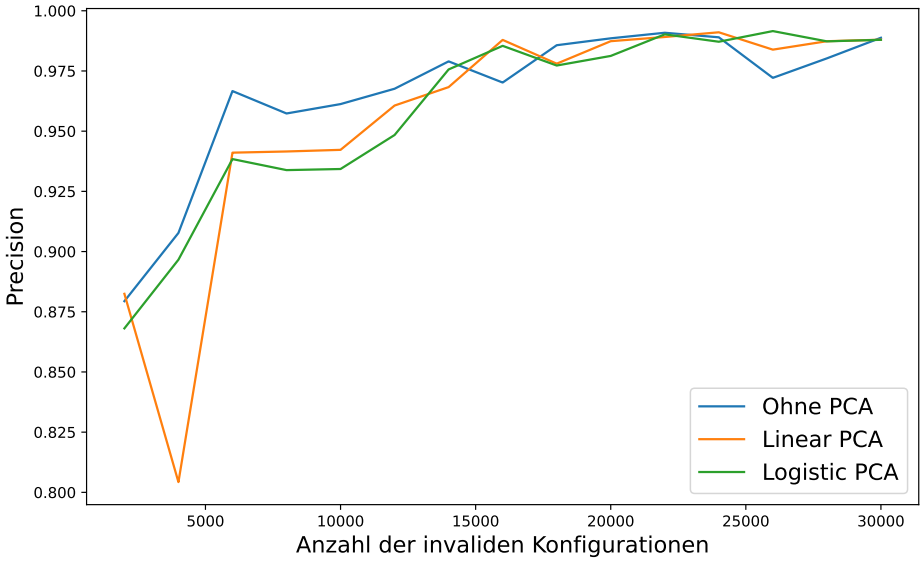


ABBILDUNG 5.22: Präzisionswerte des Feature-Modell-Lernens für das Body-Comfort-System (BCS)

jeweils Linear und Logistic PCA. Für die beiden Abbildungen gilt, dass die niedrigere k-Wert (k=8) tendenziell kürzere Laufzeiten über die Anzahl der invaliden Konfigurationen hat. Die Laufzeiten für die übrigen k Werten scheinen sich ähnlich zu verhalten, wobei die Laufzeit bei $k = 12$ geringfügig höher ist als bei der anderen k-Werten. Somit beenden wir die Ergebnisse für die erste Forschungsfrage. Wir betrachten jetzt die zweite Forschungsfrage.

5.5.3 FF2: Effektivität von FML mit und ohne Einsatz von Princpipal-Component-Analysis

Wir präsentieren nun die Evaluationsergebnisse des Feature-Modell-Lernens für beide Szenarien: FML Mit und ohne Einsatz von Principal-Component-Analysis. Die Ergebnisse für die drei Fallstudien werden in folgender Reihenfolge dargestellt: Zuerst das Body-Comfort-System (BCS), gefolgt vom Only-Binary-Constraints-System und abschließend die Only-Or-Groups-Fallstudie. Für jede Studie betrachten wir zwei Graphen: Ein Graph für die Präzision und ein für den Recall. Jeder Graph beinhaltet drei Linien: Die blaue Linie zeigt die Resultate im Fall von FML ohne PCA, die orange und grüne Linien stellen die Resultate im Fall von FML mit jeweils Linear und Logistic PCA. Für die beiden Linien für FML mit PCA gilt, dass die Anzahl der Principal-Components den optimalen k-Wert entspricht, der aus den Ergebnissen der ersten Forschungsfrage hervorgeht. Auf der X-Achse jedes Graphen wird die Anzahl der ungültigen Konfigurationen abgebildet, die im Bereich von 2000 bis 30000 in 2000er-Schritten variiert, während die Y-Achse die Werte der Präzision und des Recalls repräsentiert. Wir beginnen damit, die Evaluierungsergebnisse für die Effektivität des Feature-Modell-Lernens für das BCS zu beschreiben.

Body-Comfort-System. Wir betrachten zunächst die Ergebnisse der Evaluation bezüglich der Präzision, die in Abbildung 5.22 dargestellt sind. Im Diagramm ist zu erkennen, dass alle drei Linien auf einem ähnlichen Niveau mit einer Präzision knapp über 0,875 beginnen, wobei die blaue Linie für FML ohne PCA leicht führend ist. Bis etwa 5000 ungültige Konfigurationen gibt es einen deutlichen Abfall der Präzision für die Linear PCA, während die anderen beiden Linien eine deutliche Steigerung auf etwa 0,95 zeigen. Zwischen 5000 und 15000 ungültigen Konfigurationen stabilisieren sich die Präzisionswerte wieder. Die Linear und Logistic PCA verlaufen hier relativ dicht beieinander und überkreuzen sich gelegentlich. FML ohne PCA zeigt in diesem Bereich eine leicht höhere Präzision im Vergleich zu den

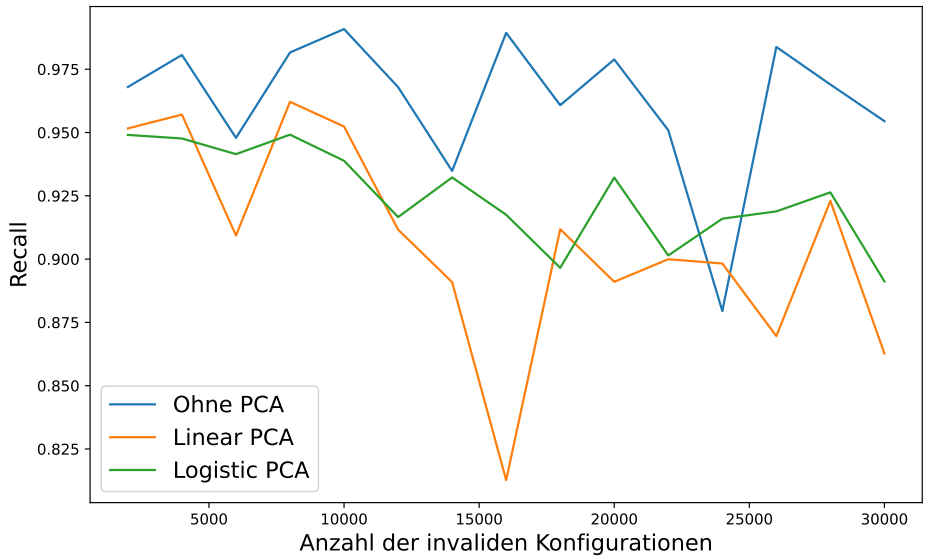


ABBILDUNG 5.23: Recallwerte des Feature-Modell-Lernens für das Body-Comfort-System (BCS)

anderen beiden Methoden. Ab etwa 20000 bis zum Ende des Diagramms zeigen alle drei Ansätze eine ähnliche Präzision mit leichten Schwankungen. FML mit Linear und Logistic PCA scheinen hier etwas konsistenter zu sein als ohne PCA, die etwas größere Schwankungen aufweist.

Wir berücksichtigen jetzt die Ergebnisse hinsichtlich des Recalls in der Abb. 5.23. Die blaue Linie für FML ohne PCA zeigt eine relativ hohe und stabile Recall-Rate, die über den gesamten Bereich der ungültigen Konfigurationen tendenziell über den anderen Methoden liegt. Es gibt einige Schwankungen, aber die Rate bleibt meistens über 0.925. Die Linear PCA zeigt mehr Variabilität im Recall. Besonders im mittleren Bereich der ungültigen Konfigurationen (um 15000) gibt es einen deutlichen Einbruch im Recall. Trotz einiger Erholungen bleibt die Recall-Rate für FML mit Linear PCA im Vergleich zu den anderen Methoden generell niedriger. Die Recall-Rate der Logistic PCA weist weniger Schwankungen auf als die Linear PCA und zeigt eine bessere Konsistenz als die Linear PCA, insbesondere im Bereich zwischen 5000 und 20000 ungültigen Konfigurationen. Jedoch bleibt diese Linie im gesamten Bereich des Graphen unter die blaue Linie für FML ohne PCA. Abgesehen von der starken Abfall in der Mitte des Graphen entwickeln die Kurven von Linear und Logistic PCA sehr eng nebeneinander. Als nächstes betrachten wir die Ergebnisse für die zweite Fallstudie.

Only-Binary-Constraints. Der Graph in Abbildung 5.24 zeigt die Präzision für drei verschiedene Szenarien: FML ohne PCA, FML mit linear PCA und FML mit Logistic PCA. Zu Beginn des Graphen, bei weniger als 10000 ungültigen Konfigurationen, beginnt die blaue Linie für FML ohne PCA mit einer Präzision knapp über 0,3 und zeigt einen allmählichen Anstieg bis zu einer Präzision von etwa 0,6 bei etwa 8000 ungültigen Konfigurationen. Die orange Linie für Linear PCA startet mit einer niedrigeren Präzision als das Modell ohne PCA. Mit zunehmender Anzahl ungültiger Konfigurationen zeigt die blaue Linie einen stabilen Anstieg, der von 0,2 bei 2000 ungültigen Konfigurationen auf etwa 0,6 bei 10000 ungültigen Konfigurationen steigt. In diesem Bereich entwickelt sich die grüne Linie für Logistic PCA ähnlich wie die Kurve für Linear PCA, jedoch mit leichten Schwankungen. Ab etwa 12000 zeigt die Kurve für FML ohne PCA einen leichten Anstieg, jedoch nicht über die anderen Linien für Linear und Logistic PCA hinweg. Diese beiden Linien erholen sich von ihrem Abfall und steigen so an, dass sie sich eng an die Linie für FML ohne PCA angleichen.

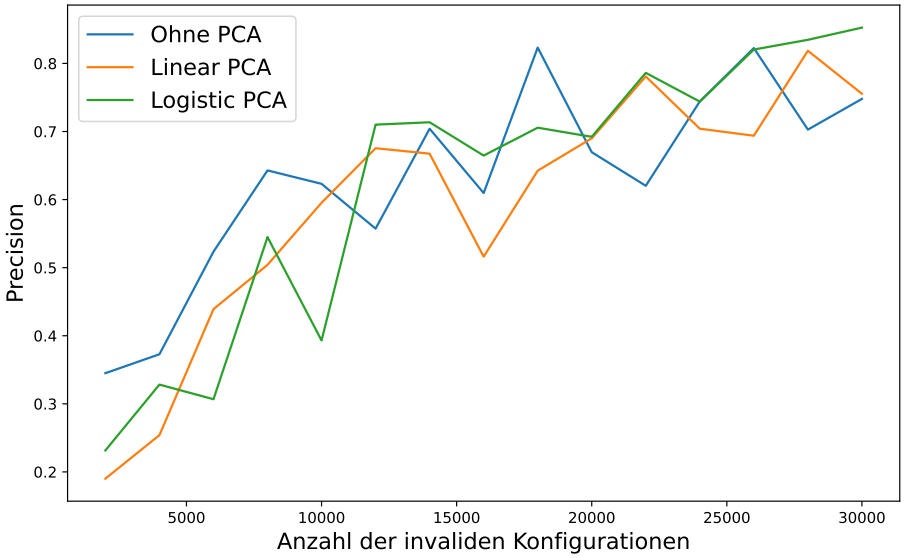


ABBILDUNG 5.24: Präzisionswerte des Feature-Modell-Lernens für das System mit nur binären Constraints (Only-Binary-Constraints)

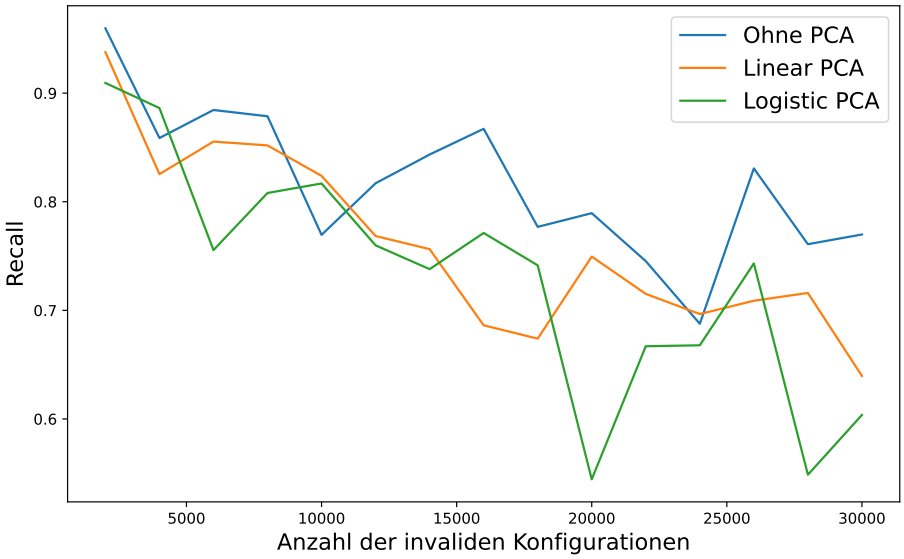


ABBILDUNG 5.25: Recallwerte des Feature-Modell-Lernens für das System mit nur binären Constraints (Only-Binary-Constraints)

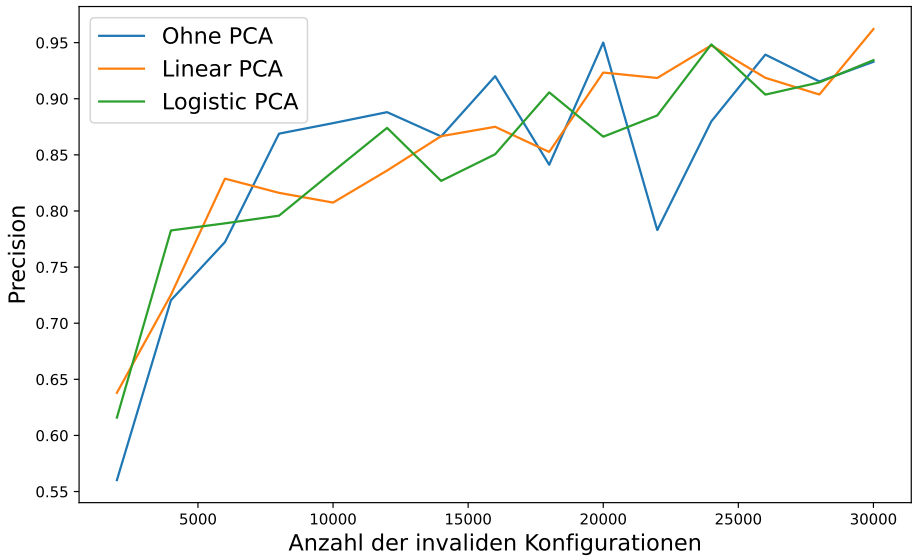


ABBILDUNG 5.26: Präzisionswerte des Feature-Modell-Lernens für das System mit nur Oder Gruppen (Only-Or-Group)

Der Graph in Abbildung 5.25 zeigt die Ergebnisse bezüglich des Recalls. Der Recall für alle drei Szenarien beginnt mit einem hohen Wert von nahezu 0,9 und zeigt über den gesamten Bereich hinweg einen absteigenden Trend. Die blaue Linie für FML ohne PCA zeigt im Verlauf des gesamten Graphen oft die höchsten Werte für den Recall. Die beiden Linien für Linear und Logistic PCA entwickeln sich eng nebeneinander, wobei die grüne Linie für Logistic PCA eine höhere Schwankung aufweist. Betrachten wir jetzt die dritte Fallstudie.

Only-Or-Group. Die Abbildung 5.26 zeigt die Präzisionswerte für das System mit nur Oder-Gruppen. Die drei Linien im Graph zeigen eine deutlich ansteigende Tendenz der Präzision bis zu etwa 10000 ungültigen Konfigurationen. Danach stabilisieren sich diese Kurven in einem Bereich von 0,85 bis 0,95. Die Linie für FML ohne PCA beginnt mit einer relativ niedrigeren Präzision (< 5000), überholt jedoch die anderen beiden Kurven für PCA bei etwa 10000 ungültigen Konfigurationen und zeigt dann eine recht stabile Leistung. Die orangefarbene und grüne Linie für Linear und Logistic PCA verlaufen im Verlauf des Graphen eng nebeneinander.

In Abbildung 5.27 sind die Evaluationsergebnisse bezüglich des Recalls dargestellt. Der Recall für FML ohne PCA bleibt auf den höchsten Niveau im Verlauf des gesamten Graphen. Die beiden Linien für Linear und Logistic PCA bleiben niedriger als das Modell ohne PCA und entwickeln sich sehr eng nebeneinander in einer absteigenden Tendenz. Betrachten Wir jetzt die Evaluationsergebnisse für die dritte Forschungsfrage.

5.5.4 FF3: Effizienz von FML mit und ohne Einsatz von DR

Wir beschreiben die Evaluationsergebnisse des Feature-Modell-Lernens mit und ohne den Einsatz von PCA und präsentieren diese Ergebnisse für drei verschiedene Fallstudien in der gleichen Reihenfolge wie zuvor. Zuerst präsentieren wir die Ergebnisse für das Body-Comfort-System, dann für die Only-Binary-Constraints und zuletzt die Ergebnisse für die Fallstudie Only-Or-Groups. Für jede Fallstudie werden wir einen Graph vorstellen. Für alle Graphen gilt dabei die folgende Beschreibung: Die X-Achse zeigt die Anzahl der invaliden Konfigurationen von 2000 bis 30000 in Schritten von 2000, und die Y-Achse repräsentiert die Laufzeit in Sekunden. Jeder Graph enthält drei verschiedene Linien: Die blaue Linie repräsentiert die Laufzeit von FML ohne PCA. Die orange und grüne Linien repräsentieren die

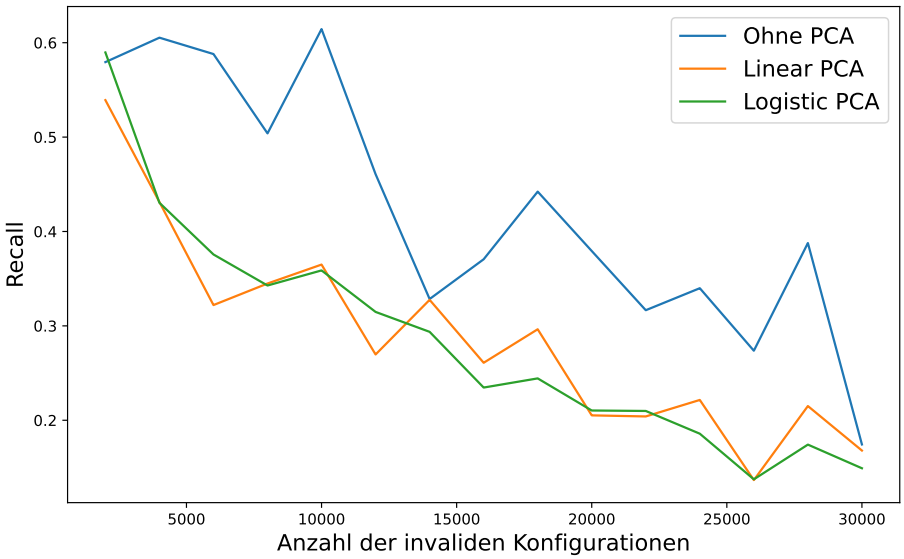


ABBILDUNG 5.27: Recallwerte des Feature-Modell-Lernens für das System mit nur binären Constraints (Only-Or-Group)

Laufzeit zur Durchführung von jeweils Linear und Logistic PCA zusammen mit dem Training für AutoGluon. Wir beginnen mit der Beschreibung der Evaluationsergebnisse für die Effektivität des Feature-Modell-Lernens für das BCS.

Body-Comfort-System. Das Diagramm 5.28 zeigt die Laufzeiten für die erste Fallstudie. Der Wert $k = 28$ zeigt an, wie viele Principal-Components benutzt wurden beim Training im Fall, wenn die Reduktion durch PCA ausgeführt wird. Alle drei Linien zeigen Schwankungen in der Laufzeit, aber tendenziell steigen die Laufzeiten mit der Zunahme der invaliden Konfigurationen. Die Laufzeit für FML ohne PCA ist an verschiedenen Punkten höher als die Laufzeiten für FML mit PCA. Die Laufzeiten für die Linear- und Logistic PCA entwickeln sich sehr ähnlich mit der Erhöhung der Anzahl der ungültigen Konfigurationen. Betrachten wir jetzt die Ergebnisse für die zweite Fallstudie.

Only-Binary-Constraints. Das Diagramm 5.29 veranschaulicht die Laufzeiten des Only-Binary-Constraints-Systems und zeigt ein ähnliches Muster wie beim Body-Comfort-System (BCS). Die Laufzeiten steigen im Allgemeinen mit der Zunahme der invaliden Konfigurationen an, allerdings mit einer Besonderheit: Im Bereich von etwa 18000 bis 22000 invaliden Konfigurationen gibt es einen signifikanten Abfall der Laufzeit, wenn keine PCA verwendet wird. Dieser Abfall geht bis auf etwa 100 Sekunden zurück und steigt dann wieder an, um am Ende einen Spitzenwert von ungefähr 170 Sekunden zu erreichen. Betrachten wir jetzt die Resultaten für die dritte Fallstudie.

Only-Or-Groups. Die Abbildung 5.30 zeigt die Ergebnisse für die dritte Fallstudie. Im Gegensatz zur Ergebnisse von der ersten und zweiten Fallstudie erkennen wir aus dieser Abbildung, dass die Laufzeit für FML ohne PCA niedriger ist als die Laufzeiten für FML mit PCA. Die Linien für Linear und Logistic PCA entwickeln sich in ähnlicher Weise mit einer steigenden Tendenz und verbleiben während des gesamten Verlaufs des Graphen über den Laufzeiten der Linie ohne PCA. Somit beenden wir die Evaluationsergebnisse zur dritten Forschungsfrage. Im nächsten Abschnitt diskutieren wir die Ergebnisse der Evaluation.

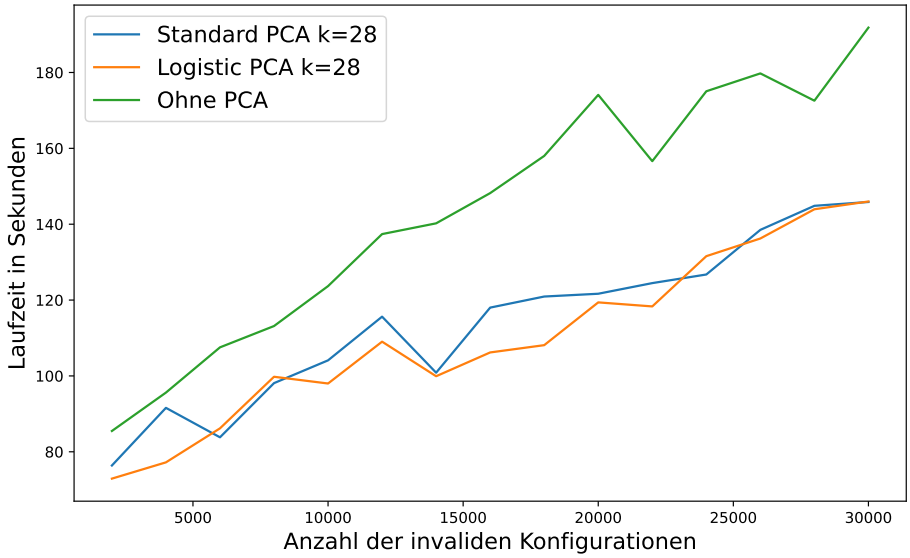


ABBILDUNG 5.28: Effizienz des Feature-Modell-Lernens für das Body-Comfort-System (BCS)

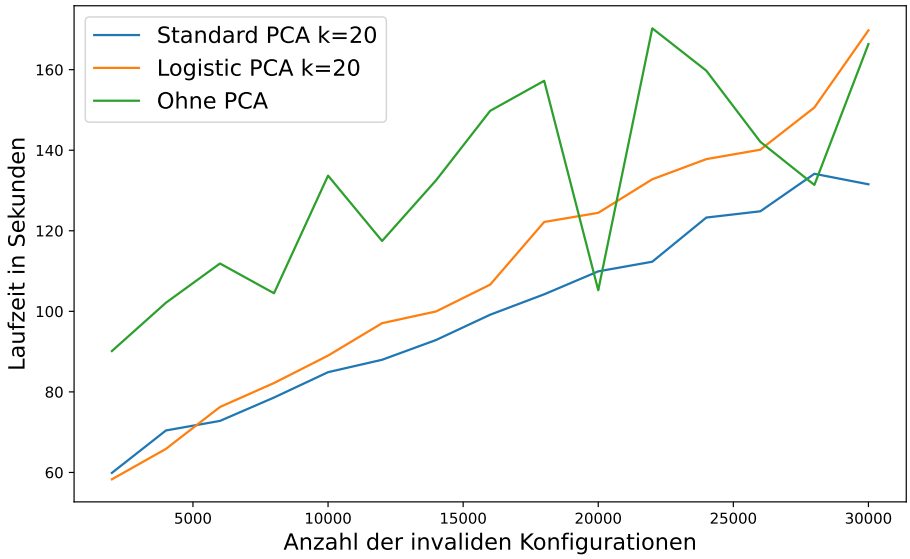


ABBILDUNG 5.29: Effizienz des Feature-Modell-Lernens für das Feature-Modell mit nur binären Constraints (Only-Binary-Constraints)

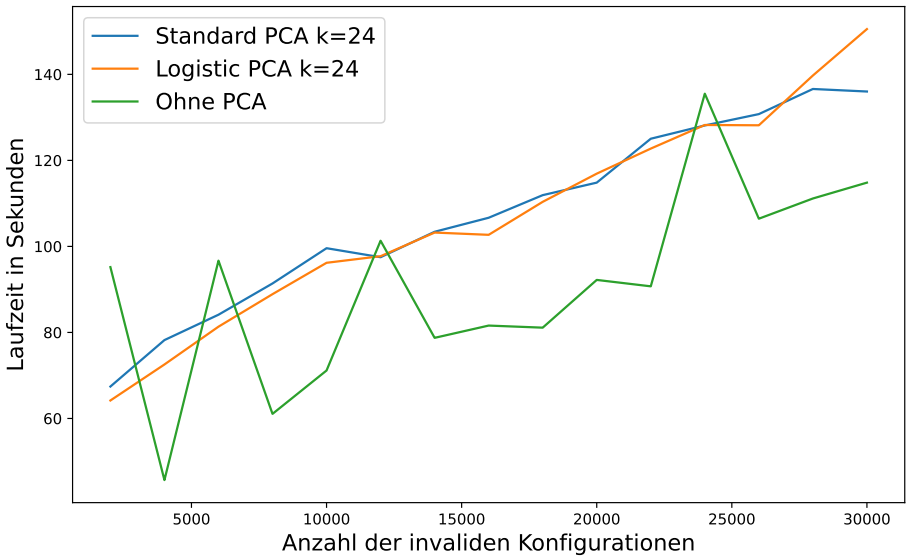


ABBILDUNG 5.30: Effizienz des Feature-Modell-Lernens für das Feature-Modell mit nur Oder-Gruppen (Only-Or-Groups)

5.6 Diskussion

In diesem Abschnitt diskutieren wir die Evaluierungsergebnisse. Wir beginnen mit der Diskussion der Evaluierungsergebnisse im Hinblick auf die erste Forschungsfrage. Wir besprechen die Effektivität und Effizienz des Feature-Modell-Lernens unter Verwendung von PCA mit verschiedenen Anzahlen für die Principal-Components (k). Anschließend diskutieren wir die Ergebnisse im Zusammenhang mit der zweiten Forschungsfrage, die sich mit der Effektivität des Feature-Modell-Lernens unter Verwendung von Dimensionality-Reduction befasst. Danach besprechen wir die Evaluierungsergebnisse zur Effizienz des Feature-Modell-Lernens unter Verwendung der Dimensionality-Reduction und vergleichen wir diese mit den Ergebnissen ohne Dimensionality-Reduction. Abschließend fassen wir die Evaluierungsergebnisse pro Forschungsfrage zusammen.

FF1: Effektivität von FML mit unterschiedlichen k-Werten. Für die Diskussion der Ergebnisse bezüglich der ersten Forschungsfrage betrachten wir die folgende Reihenfolge: Zuerst diskutieren wir die Ergebnisse hinsichtlich der Präzision für alle Fallstudien und anschließend die Ergebnisse bezüglich des Recalls. Betrachten wir zunächst die Präzisionsdiagramme für die verschiedenen Fallstudien unter Anwendung der beiden Techniken der Dimensionality-Reduction. Die Analyse des Body-Comfort-Systems mit Linear- und Logistic PCA in der beiden Abbildungen 5.4 und 5.6 zeigt, dass höhere k-Werte konstant bessere Präzision als bei niedrigen k-Werten wie $k = 8$ oder $k = 12$ liefern. Dies bedeutet, dass eine Reduzierung auf eine zu geringe Anzahl von Principal-Components zu einem signifikanten Informationsverlust bei dem Datensatz führt, was sich negativ auf die Präzision des gelernten Modells auswirkt. Für die Systeme mit ausschließlich binären Constraints in der Abbildungen 5.8 und 5.10 und ausschließlich Oder-Gruppen in Abbildungen 5.12 und 5.14 zeigen die Ergebnisse ähnliche Muster. Die Effektivität des Feature-Modell-Lernens verbessert sich generell mit der Erhöhung der Anzahl der Principal-Components, was die Schlussfolgerung unterstützt, dass eine höhere Anzahl von Principal-Components relevant ist, um die Leistungsfähigkeit des Lernprozesses zu optimieren.

Wir betrachten jetzt die Ergebnisse hinsichtlich des Recalls. Im Gegensatz zu den Präzisionsgraphen zeigt der Recall für alle k-Werte mit der Erhöhung der Anzahl der ungültigen Konfigurationen eine abfallende Tendenz. Beim Body-Comfort-System sowohl für Linear

als auch für Logistic PCA in der Abbildungen 5.5 und 5.7 stellen wir fest, dass die höchsten k-Werte, wie k=28, immer die besten Recall-Werte erreichen, was darauf hinweist, dass das Feature-Modell-Lernen effektiver komplexe Zusammenhänge erfassen kann, wenn es mit einer größeren Dimensionalität arbeitet. Dies bedeuten, dass das Modell besser in der Lage ist, relevante Muster zu erkennen und seltener relevante Konfigurationen fälschlicherweise als ungültig ausschließt, wenn es mehr Informationen zur Verfügung hat. Die Ergebnisse für niedrigere k-Werte wie k=8 zeigen bei der Linear PCA sowie bei der Logistic PCA oft die niedrigsten Recall-Werte. Speziell für die Linear PCA sinkt der Recall nach etwa 1000 ungültigen Konfigurationen dramatisch und bleibt dann auf einem Niveau nahe 0. Dies könnte bedeuten, dass das Feature-Modell-Lernen bei einer zu geringen Anzahl von Dimensionen nicht in der Lage sein könnte, die relevanten Muster zu erkennen. Ein zu niedrig gewähltes k könnte zur Folge haben, dass das Modell essentielle Informationen nicht erfasst, die für die korrekte Klassifizierung von Konfigurationen erforderlich sind. In der Fallstudie mit ausschließlich binären Constraints zeigen die Ergebnisse in Abb. 5.9 und 5.11, dass die Recall-Werte für höhere k-Werte wie $k = 20$ weniger stark abfallen als für niedrigere k-Werte. Dies legt nahe, dass die Bewahrung einer größeren Anzahl von Principal-Components dabei hilft, die Struktur der Daten besser zu erfassen und somit die Wahrscheinlichkeit erhöht, dass das Modell relevante Feature-Kombinationen korrekt erkennt. Für das System mit ausschließlich Oder-Gruppen zeigt sich, dass die Recall-Werte für höhere k-Werte generell stabiler sind und weniger stark abfallen, während die Recall-Werte für niedrige k-Werte schon ab circa 8000 ungültige Konfigurationen auf 0 abfallen. Dies bestätigt die getroffene Annahme in der ersten und zweiten Fallstudie, welche besagt, dass niedrigere k-Werte zu Informationsverlust führen und somit zur Verschlechterung der Effektivität des FML-Modells.

Zusammenfassend können wir sagen, dass höhere k-Werte konsequent bessere Präzisions- und Recall-Werte erzielen, was darauf hindeutet, dass Feature-Modell-Lernverfahren mit einer größeren Dimensionalität komplexere Datenbeziehungen effektiver erfassen können. Niedrigere k-Werte wie k=8 führen hingegen oft zu deutlich schlechteren Ergebnissen. Wir diskutieren als nächstes die Ergebnisse der Laufzeiten für unterschiedlichen k-Werten .

FF1: Effizienz von FML mit unterschiedlichen Anzahlen der Principal-Components. In dieser Diskussion untersuchen wir die Veränderung der Laufzeit mit zunehmender Anzahl ungültiger Konfigurationen für verschiedene Werte der Principal-Components. Sowohl bei der Linear- als auch bei der Logistic PCA und für alle drei Fallstudien erkennen wir das gleiche Muster: Mit steigender Anzahl ungültiger Konfigurationen nehmen die Laufzeiten für alle k-Werte tendenziell zu. Der Trend für die niedrigsten k-Werte, wie $k = 4$ bei der zweiten Fallstudie oder $k = 8$ bei der ersten und dritten Fallstudie, zeigt sich als die unterste Linie im Diagramm. Dies deutet darauf hin, dass die Reduktionszeit zusammen mit der Verarbeitungszeit von AutoGluon schneller bei geringerer Dimensionalität ist. Mit steigendem k-Wert, wie bei $k = 12$, $k = 16$ und $k = 20$, nehmen die Laufzeiten zu, was auf eine erhöhte Rechenzeit durch komplexere Dimensionsberechnungen oder auch das Training von AutoGluon schließen lässt. Die Kurve für $k = 28$, $k = 20$ und $k = 24$ für jeweils die erste, zweite und dritte Fallstudie zeigt eine konstante Steigerung, wobei sie über den niedrigsten k-Werten liegt, aber immer noch unter den mittleren k-Werten wie $k = 12$ steht. Da wir aus der letzten Frage bezüglich der Effektivität von FML gezeigt haben, dass diese höheren Werte von k oft die höchste Präzision und Recall erzielen, können wir schlussfolgern, dass die höchsten k-Werte ein besseres Gleichgewicht zwischen Laufzeit und Modellgenauigkeit bieten. wir betrachten uns als nächstes die Evaluationsergebnisse bezüglich der zweiten Frage.

FF2: Effektivität des Feature-Modell-Lernens mit und ohne Einsatz von PCA. Wir betrachten zunächst die Ergebnisse der ersten Fallstudie hinsichtlich der Präzision. In der Abbildung 5.22 ist zu sehen, dass alle Ansätze ähnlich hohe Anfangspräzisionen aufweisen, wobei FML ohne PCA leicht führend ist. Dies deutet darauf hin, dass der Einsatz von PCA mit einer geringeren Anzahl ungültiger Konfigurationen keine signifikanten Vorteile bietet. Die steigende Tendenz der drei Linien zeigt, dass die Effektivität der FML-Strategie unabhängig von der Nutzung der Dimensionality-Reduction mit zunehmender Anzahl ungültiger Konfigurationen zunimmt. Besonders deutlich wird dies zwischen 5000 und 15000 ungültigen Konfigurationen, wo die Präzisionswerte mit PCA von etwa 0.9 auf etwa 0.975 ansteigen.

FML ohne PCA weist tendenziell in bestimmten Abschnitten des Graphen etwas höhere Präzisionswerte auf als die PCA-Ansätze. Dies könnte darauf hindeuten, dass PCA möglicherweise nicht in allen Szenarien von Vorteil ist. Die Linear PCA und die Logistic PCA verlaufen oft dicht beieinander, was auf eine ähnliche Performance hinweist. Betrachten wir jetzt die Ergebnisse bezüglich der Recallwerten aus der Abb. 5.23. Die Beobachtungen zeigen, dass der Einsatz von PCA die Stabilität des Recall-Werts beeinflusst. Die blaue Linie ohne PCA zeigt eine tendenziell höhere und stabilere Recall-Rate über den gesamten Bereich der ungültigen Konfigurationen im Vergleich zu den PCA-Ansätzen. Dies legt nahe, dass die Dimensionality-Reduction durch PCA möglicherweise nicht immer vorteilhaft ist, insbesondere wenn eine hohe und stabile Recall-Rate angestrebt wird. Die Linear PCA zeigt eine höhere Variabilität im Recall, insbesondere mit einem deutlichen Einbruch im mittleren Bereich der ungültigen Konfigurationen. Obwohl die Logistic PCA weniger Schwankungen aufweist und eine bessere Konsistenz zeigt, bleibt ihre Recall-Rate im gesamten Bereich des Graphen unter derjenigen ohne PCA. Die Wahl zwischen Linear und Logistic PCA in diesem Szenario weniger entscheidend ist, da beide Ansätze ähnliche Muster aufweisen und sich im Recall nicht wesentlich unterscheiden.

Betrachten wir nun die Ergebnisse der zweiten Fallstudie. Wie wir bereits im Abschnitt der Ergebnisse beschrieben haben, sind die Präzisionswerte für FML ohne PCA in Abb. 5.24 höher als bei den anderen Ansätzen mit PCA, insbesondere wenn es nur eine geringe Anzahl von ungültigen Konfigurationen gibt (< 10000 ungültige Konfigurationen). Dies könnte darauf hinweisen, dass FML ohne PCA besser mit kleinen Datensätzen umgehen kann als mit implementierter Dimensionality-Reduction. Die Stabilisierung des Anstiegs für die blaue Linie zeigt, dass das FML-System ohne Dimensionality-Reduction in der Lage ist, sich an die Komplexität der Daten anzupassen und die Präzision der Vorhersage zu verbessern, wenn mehr Daten zur Verfügung stehen. Die beiden Linien für PCA zeigen ähnliche Entwicklungen, was darauf hindeutet, dass die Wahl zwischen Linear und Logistic PCA keinen großen Einfluss auf die Effektivität von FML hat. Insgesamt zeigen die Recall-Daten in Abb. 5.25 für das System Only-Binary-Constraints, dass alle drei betrachteten Methoden eine Herausforderung darin haben, die Recall-Rate konstant hoch zu halten, wenn die Anzahl der ungültigen Konfigurationen steigt. Dies könnte auf die zunehmende Schwierigkeit hinweisen, die Beziehungen zwischen Features in einem komplexeren Datenraum korrekt zu identifizieren. Die Linear PCA zeigt die größten Schwankungen im Recall, was darauf schließen lässt, dass die Methode der Dimensionality-Reduction möglicherweise nicht immer konsistent relevante Informationen beibehält. Das Modell ohne PCA zeigt eine allgemeine Abnahme des Recall, bleibt aber in einem ähnlichen Bereich wie die Linear PCA. Die Logistic PCA zeigt durchgehend niedrigere Recall-Werte, was darauf hindeutet, dass sie möglicherweise für dieses spezifische Fallstudie nicht die optimale Wahl ist.

Bei der Auswertung der dritten Fallstudie beobachten wir eine ähnliche Tendenz wie bei der ersten beiden Fallstudien. Die steigenden Tendenz der Präzisionswerte für FML ohne PCA in Abb. 5.26 könnte darauf hinweisen, dass das Modell ohne Dimensionality-Reduction konsistent relevante Instanzen identifizieren kann, selbst wenn die Anzahl der ungültigen Konfigurationen wächst. Die Linien für Linear PCA und Logistic PCA zeigen anfangs eine ähnliche Tendenz wie das Modell ohne PCA, mit einem anschließenden Anstieg der Precision. Jedoch zeigen diese Linien über den gesamten Bereich hinweg mehr Schwankungen, was auf eine gewisse Instabilität oder Variabilität in der Modellleistung hinweisen könnte. In der Abbildung 5.27 sehen wir, dass die Recall-Werte für FML ohne PCA oft auf einem höheren Niveau liegen. Dies deutet darauf hin, dass die Anwendung von Dimensionality-Reduction in dieser Fallstudie die Effektivität des Modells schlecht beeinflusst hat.

FF3: Effizienz des Feature-Modell-Lernens mit und ohne Einsatz von PCA. Die Ergebnisse der Evaluierung des Feature-Modell-Lernens mit und ohne Einsatz von Principal-Component-Analysis für die drei Fallstudien liefern Einblicke in die Effizienz des Lernprozesses. Zunächst ist festzustellen, dass die Laufzeiten tendenziell mit der Zunahme der

ungültigen Konfigurationen in allen Fallstudien steigen, was darauf hinweist, dass die Verarbeitung größerer Datensätze mehr Rechenressourcen erfordert. Für das Body-Comfort-System (BCS) zeigt bei der Abb. 5.28 sich, dass die Laufzeiten ohne PCA an allen Stellen höher sind als die Laufzeiten mit PCA, was darauf hinweist, dass die Dimensionality-Reduction durch PCA im Bezug auf der Rechenzeit vorteilhaft ist. Die Laufzeiten für die Linear- und Logistic PCA entwickeln sich ähnlich und zeigen eine tendenzielle Zunahme mit der Anzahl der ungültigen Konfigurationen. Beim Only-Binary-Constraints-System in Abb. 5.29 sind die Laufzeiten ebenfalls tendenziell steigend, jedoch mit einem interessanten Abfall der Laufzeit im Bereich von 18000 bis 22000 ungültigen Konfigurationen, wenn keine PCA verwendet wird. Dies könnte darauf hindeuten, dass in diesem Bereich bestimmte Eigenschaften der Daten vorliegt, die eine schnellere Verarbeitung ermöglichen. Für das Only-Or-Groups-System in Abb. 5.30 zeigen die Ergebnisse, dass die Laufzeiten ohne PCA im Allgemeinen höher sind als die Laufzeiten mit PCA. Die Laufzeiten entwickeln sich für PCA in ähnlicher Weise und verbleiben über den Verlauf des Graphen über den Laufzeiten ohne PCA. Insgesamt deutet dies darauf hin, dass PCA eine wichtige Rolle bei der Verbesserung der Effizienz des Feature-Modell-Lernens spielen kann, insbesondere wenn große Datensätze verarbeitet werden müssen. Die Wahl zwischen Linear PCA und Logistic PCA scheint keinen signifikanten Unterschied in Bezug auf die Laufzeiten zu machen, was darauf hinweist, dass beide Methoden ähnlich effektiv sind. Wir schließen diesen Abschnitt ab, indem wir unsere Ergebnisse zusammenfassen und die Forschungsfragen präzis beantworten:

- **Forschungsfrage 1:** Was ist der optimale Wert für die Anzahl der Principal-Components bei PCA für das Feature-Modell-Lernen?

  > Die Analyse der Ergebnisse zeigt, dass höhere k-Werten beim Feature-Modell-Lernen empfohlen wird. Diese k-Werte stellen einen optimalen Kompromiss zwischen der Erhöhung der Effektivität und der Handhabbarkeit der Laufzeiten dar.

- **Forschungsfrage 2:** Wie verändern sich Präzision und Recall beim Feature-Modell-Lernen mit DR im Vergleich zum Feature-Modell-Lernen ohne DR?

  > Feature-Modell-Lernen ohne PCA ist effektiver im Vergleich zu FML mit Einsatz von Linear oder Logistic PCA. Dies ist besonders deutlich für Datensätze mit geringeren Anzahl von ungültige Konfigurationen.

- **Forschungsfrage 3:** Wie verändert sich der Rechenzeit beim Feature-Modell-Lernen mit DR im Vergleich zum Feature-Modell-Lernen ohne DR?

  > Die Evaluierung des Feature-Modell-Lernens mit und ohne Principal-Component-Analysis zeigt, dass PCA dazu beiträgt, die Laufzeiten zu stabilisieren und die Effizienz insbesondere bei großen Datensätzen zu verbessern.

Die Untersuchung der Effektivität und Effizienz des Feature-Modell-Lernens mit und ohne den Einsatz von PCA zeigt einen signifikanten Trade-off zwischen diesen beiden Aspekten. Während Feature-Modell-Lernen ohne PCA eine höhere Effektivität zeigt, insbesondere bei Datensätzen mit einer geringeren Anzahl ungültiger Konfigurationen, trägt die Anwendung von PCA – sei es in der Linear oder Logistic Form – dazu bei, die Laufzeiten zu stabilisieren und die Effizienz des Prozesses zu verbessern, vor allem bei der Verarbeitung großer Datensätze. Diese Ergebnisse deuten darauf hin, dass der Einsatz von PCA eine wertvolle Strategie zur Optimierung der Laufzeiten sein kann, allerdings auf Kosten der maximalen Effektivität, die ohne Dimensionality-Reduction erzielt werden könnte. Der Kern des Tradeoffs liegt somit in der Abwägung zwischen der Notwendigkeit, schnellere Ergebnisse bei der Verarbeitung großer Datensätze zu erzielen, und dem Wunsch, die höchstmögliche Genauigkeit in der Modellvorhersage zu erreichen. Im nächsten Abschnitt besprechen wir die Gefährdung der Gültigkeit.

5.7 Gefährdung der Gültigkeit

Um dieses Kapitel abzuschließen, beschreiben wir mögliche Gefährdungen für die Gültigkeit. Wir unterscheiden zwischen internen und externen Gefährdungen, wie von Runeson et al. [19] vorgeschlagen. Wir beginnen mit der Beschreibung der internen Gefährdungen für die Gültigkeit.

5.7.1 Interne Gefährdung der Gültigkeit

Eine interne Gefährdung für die Gültigkeit könnte die Auswahl und Anzahl der Fallstudien sein. Die Tatsache, dass diese manuell konstruiert wurden, kann zu abweichenden Ergebnissen unter realen Bedingungen führen. Die Größe der Fallstudien ist auch untereinander ähnlich. Das Lernen kann sich für größere Fallstudien unterschiedlich verhalten. Eine weitere Gefährdung könnte die Auswahl der Lernmethoden sein, die durch die Untersuchungssysteme möglicherweise etwas voreingenommen ist. Es ist unklar, ob es andere Methoden gibt, die noch besser abschneiden, und die nicht in AutoGluon enthalten sind. Eine andere interne Gefährdung besteht darin, dass Die Wahl zwischen Linear- und Logistic PCA möglicherweise nicht die tatsächlich beste Methode für die spezifischen Daten und das Problem darstellen. Es besteht die Möglichkeit, dass eine der Methoden für bestimmte Datensätze besser geeignet ist, jedoch aufgrund von Einschränkungen in der Evaluation nicht ausgewählt wurde. Zusätzlich zur Methode und zur Wahl von k birgt auch das Fine-Tuning-Potenzial des Parameters m bei der Methode der logistic PCA ein weiteres Risiko. In unsere Arbeit haben wir nur mit der Linear-Werte für diesen Parameter gearbeitet, dies konnte möglicherweise die Leistungsfähigkeit der logistic PCA beeinträchtigen. Die Wahl von m ist entscheidend, da sie die Qualität der Dimensionality-Reduction und damit die Effektivität des Modells direkt beeinflusst. Eine unzureichende Exploration dieses Parameters könnte somit zu nicht optimalen Ergebnissen führen. Außerdem hängt die Leistung von PCA nicht nur von der Auswahl der Methode ab, sondern auch von anderen Faktoren wie der Anzahl der Principal Components, dem Datensatz und der Art des zu lösenden Problems. Die Interaktion dieser Variablen könnte zu unerwarteten Ergebnissen führen, die die Interpretation der Ergebnisse erschweren.

5.7.2 Externe Gefährdung der Gültigkeit

Externe Gefährdungen können hauptsächlich durch die Verwendung externer Programme verursacht werden. Die Bewertungsergebnisse für das Lernen hängen stark von AutoGluon [5] ab. AutoGluon wählt sowohl die verwendeten Maschinenalgorithmen aus als auch optimiert diese. Fehler in diesem Prozess würden die Bewertungsergebnisse beeinflussen, mit denen wir die Effizienz und Effektivität des FML-Modells bewertet haben. Da AutoGluon nur eine feste Auswahl von maschinellen Lernalgorithmen berücksichtigt, sind wir ausschließlich auf diese Algorithmen angewiesen. Andere Algorithmen, die nicht Teil von AutoGluon sind, können eine bessere Effizienz oder eine bessere Effektivität erreichen. Zusätzlich zur Abhängigkeit von AutoGluon unterliegen unsere Ergebnisse auch den Einschränkungen der Bibliotheken, die für die Durchführung der Principal-Component-Analysis verwendet wurden. Insbesondere wurde für die Linear PCA die Bibliothek stats aus der Programmiersprache R und für die logistic PCA das R-Paket logisticPCA verwendet. Ähnlich wie bei AutoGluon können Fehler, Limitationen oder die spezifische Auswahl und Implementierung der Algorithmen in diesen Bibliotheken die Gültigkeit unserer Ergebnisse beeinträchtigen. Die Einschränkung auf die Algorithmen und Methoden, die in diesen Bibliotheken verfügbar sind, könnte ebenfalls dazu führen, dass möglicherweise effektivere Techniken für die Dimensionality-Reduction, die außerhalb dieser Bibliotheken existieren, nicht berücksichtigt wurden.

In diesem Kapitel haben wir die Evaluierungsergebnisse des Feature-Modell-Lernens mit und ohne Verwendung von Dimensionality-Reduction beschrieben und diskutiert. Wir haben angefangen, indem wir die Forschungsfragen vorgestellt haben, die wir während der

Evaluation untersucht haben. Anschließend haben wir die drei verwendeten Fallstudien beschrieben und erläutert, wie wir die Daten gesammelt haben. Danach haben wir die Ausführungsplattformen beschrieben.

Dies wurde durch die Präsentation der Evaluierungsergebnisse pro Forschungsfrage fortgesetzt. Wir haben mit der Beschreibung der Evaluierungsergebnisse zur Effektivität und Effizienz des Feature-Modell-Lernens für jedes Fallstudie unter Verwendung von Linear- und Logistic PCA mit unterschiedlichen Anzahlen von Principal-Components angefangen. Anschließend haben wir die Effektivität des Feature-Modell-Lernens mit und ohne Verwendung von Dimensionality-Reduction diskutiert. Wir haben zunächst das Feature-Modell-Lernen mit PCA, strukturiert durch die Präsentation der Ergebnisse für die beiden Optimierungsmetriken Precision und Recall beschrieben. Danach haben wir die Evaluierungsergebnisse zur Effektivität des Feature-Modell-Lernens unter Verwendung sowohl der Linear- als auch der Logistic PCA präsentiert. Die Anzahl der Principal-Components wurde in diesem Fall auf Grundlage der Ergebnisse der ersten Forschungsfrage festgelegt. Anschließend haben wir die Evaluierungsergebnisse zur Effizienz mit und ohne Verwendung von PCA-Strategien betrachtet. Danach haben wir alle Evaluierungsergebnisse in einem separaten Abschnitt diskutiert und die Forschungsfragen beantwortet. Abschließend haben wir die internen und externen Gefährdungen für die Gültigkeit beschrieben.

Kapitel 6

Verwandte Arbeiten

In diesem Kapitel beschreiben wir verwandte Arbeiten zur Anwendung von Techniken zur Dimensionality-Reduction zur Verbesserung der Leistung von maschinellen Lernmodellen. Zunächst fassen wir eine kürzlich veröffentlichte Literaturübersicht zusammen, die die Anwendung der Principal-Component-Analysis zur Verwaltung von hochdimensionalen, unausgeglichenen Datensätzen hervorhebt. Abschließend diskutieren wir die Studie von George zur Untersuchung der Leistung von Klassifikationsmodelle unter Einsatz von PCA.

6.1 Literatur Review von Dimensionality-Reduction durch PCA

Narwane und Sawarkar untersuchen den Einsatz der Principal-Component-Analysis (PCA) zur Dimensionality-Reduction (DR) in unausgeglichenen Datensätzen, insbesondere im Gesundheitswesen. Durch die Anwendung von PCA zielten die Autoren darauf ab, die Leistung von maschinellen Lernmodellen durch die Behandlung des Unausgeglichenheitsproblems zu verbessern. Ihre Bewertung am Pima-Diabetes-Datensatz zeigte, dass PCA die Dimensionalität effektiv reduzieren und die Sensitivität der Modelle verbessern konnte. Dies wurde durch die verbesserten Klassifikationsergebnisse belegt, was darauf hindeutet, dass PCA eine praktikable Preproccessingstechnik zur Verwaltung von unausgeglichenen Datensätzen in maschinellen Lernanwendungen ist. Die Ergebnisse der Studie unterstreichen das Potenzial der PCA, die Datenanalyse und Modellgenauigkeit durch Fokussierung auf die wichtigsten Features zu verbessern und bieten somit einen robusten Ansatz zur Bewältigung der Herausforderungen, die durch unausgeglichene Datensätze entstehen [17].

Narwane und Sawarkar betrachten ebenfalls das Klassifikationsproblem für zwei Klassen. Die Unterschiede zu dieser Arbeit sind jedoch:

- Anwendungsdomäne: Unsere Arbeit zielt auf Software-Produktlinien ab, mit Schwerpunkt auf dem Lernen von Feature-Modellen in booleschen Konfigurationsräumen. Im Gegensatz dazu konzentrieren sich Narwane und Sawarkar auf Datensätze im Gesundheitswesen, die durch Features über kontinuierliche Werte geformt sind, und behandeln speziell das Problem unausgeglichener Daten, die Diagnosemodelle für Krankheiten beeinflussen.
- Zielsetzung: Sowohl unsere Arbeit als auch die Studie von Narwane und Sawarkar zielen darauf ab, die Leistung des Modells durch Dimensionality-Reduction zu verbessern. Unsere Arbeit erforscht jedoch die Wirksamkeit von FML mit DR zur Erstellung genauerer Feature-Modelle, während Narwane und Sawarkar sich auf die Verbesserung der Klassifikationssensitivität und -Genauigkeit in der Gesundheitsdiagnostik konzentrieren.
- Methodik: Unsere Arbeit bewertet den Einfluss von DR-Techniken, einschließlich PCA, auf die Effektivität und Effizienz von FML. Die Studie von Narwane und Sawarkar wendet jedoch PCA an, um das spezifische Problem der Datenungleichgewichte anzugehen, und demonstriert deren Nutzen zur Verbesserung der Modellleistung in einem anderen Kontext.

- Resultat: In unserer Arbeit haben wir gezeigt, dass der Einsatz von PCA nur zur leichten Verbesserung der Effektivität im Kontext des Feature-Modell-Lernens (FML) führt. Die Effizient hat sich allerdings deutlich durch die Anwendung von Dimensionality-Reduction im Kontext von FML verbessert. Auf der anderen Seite zeigen Narwane und Sawarkar die Vorteile von PCA bei der Steigerung der Sensitivität und Genauigkeit in der Anomalieerkennung und Klassifizierung innerhalb von Gesundheitsdatensätzen.

Wir betrachten als nächstes die zweite Literatur von Annie George.

Annie George erforscht die Effektivität von maschinellem Lernen bei der Anomalieerkennung und konzentriert sich dabei auf Anwendungen im Bereich der Netzwerksicherheit. Die Studie stellt einen Klassifizierungsansatz zur Anomalieerkennung unter Verwendung des KDD99-Datensatzes vor. Die PCA-Technik wird zur Dimensionality-Reduction genutzt, während Support-Vector-Machine (SVM) als Klassifizierungsalgorithmus eingesetzt wird. Die Ergebnisse zeigen eine Verringerung der Ausführungszeit für die Klassifizierung bei reduzierter Dimensionalität der Eingabedaten. Außerdem zeigen die Präzisions- und Recall der SVM mit PCA-Methode eine verbesserte accuracy durch eine Verringerung der Fehlklassifikationen, was darauf hindeutet, dass dieser kombinierte Ansatz effektiver für die Anomalieerkennung in Netzwerkdaten ist [8].

Annie George untersucht ebenfalls den Einfluss der Principal Component Analysis auf die Leistungsfähigkeit des erlernten maschinellen Lernmodells. Allerdings unterscheidet sich dieser Ansatz von unserer Arbeit in den folgenden Punkten:

- Anwendungsbereich: In unsere Arbeit beschäftigen wir uns mit der Rekonstruktion von Feature-Modelle im Kontext von SPL und die Klassifizierung der Gültigkeit von Konfigurationen, während das Paper von George auf Netzwerksicherheit und Anomalieerkennung fokussiert.
- Forschungsziel: Während unsere Arbeit die Effizienz und Effektivität von FML mit Dimensionality-Reduction untersucht, konzentriert sich die Studie von Annie George auf die Anomalieerkennung in Netzwerkdaten unter Verwendung von SVM für die Klassifikation.
- Methodik: Wir haben neben der linearen PCA auch die Logistic PCA für boolesche Konfigurationsräume untersucht. Außerdem haben wir AutoGluon für das Feature-Modell-Lernen genutzt, welches mehrere komplexere maschinelle Lernmodelle zu einem Gesamtmodell kombiniert. Im Gegensatz dazu verwendet der Artikel von George gezielt SVM zusammen mit PCA für die Aufgabe der Anomalieklassifikation.
- Resultat: Die Studie von Annie George belegt, dass der Einsatz von PCA nicht nur die Effizienz des Klassifikationsmodells im Hinblick auf Präzisions- und Recall-Werte steigert, sondern auch die Effektivität bezüglich der benötigten Rechenzeit erhöht. In unserer Arbeit haben wir jedoch festgestellt, dass PCA zwar die erforderliche Rechenzeit für das Feature-Modell-Lernen reduziert, die Effektivität in Bezug auf Präzision und Recall sich jedoch nicht signifikant verbessert hat, wie es in der betrachteten Literatur der Fall war.

In diesem Kapitel haben wir verwandte Arbeiten zur Anwendung von Dimensionality-Reduction, insbesondere der Principal-Component-Analysis, beschrieben. Zunächst haben ein Literaturüberblick vorgestellt, der den Einsatz von PCA zur Behandlung hochdimensionaler, unausgeglichener Datensätze hervorhebt, mit einem Fokus auf den Gesundheitssektor. Abschließend haben wir die Arbeit von Annie George beschrieben, welche die Effektivität von PCA in Kombination mit Support-Vector-Machines (SVM) für die Anomalieerkennung in Netzwerkdaten untersucht. Im nächsten Kapitel fassen wir die wichtigsten Aspekten unserer Arbeit zusammen.

Kapitel 7

Zusammenfassung und Ausblick

In diesem abschließenden Kapitel fassen wir die wichtigsten Aspekte unserer Arbeit zusammen. Wir betrachten auch die wesentlichen Erkenntnisse, die wir gewonnen haben. Abschließend geben wir einen Ausblick auf weitere Arbeiten.

7.1 Zusammenfassung

In dieser Arbeit haben wir die Effektivität und Effizient von Feature-Model-Lernen (FML) mit und ohne Einsatz der Dimensionlity-Reduction (DR) verglichen. Konkret wurden die Linear Principal-Component-Analysis (PCA) und die Logistic Principal-Component-Analysis (Logistic PCA) als Methoden zur DR untersucht. Wir haben die Technik von FML angewendet, um Feature-Modelle aus eine Menge von bereits implementierten Produktvarianten automatisch zu rekonstruieren. Feature-Modelle definieren, welche Kombinationen von Features zu gültigen Produktvarianten führen. Die manuelle Erstellung dieser Modelle ist jedoch zeitaufwendig, fehleranfällig und bei zunehmender Komplexität und Größe von Software-Produktlinien (SPLs) praktisch nicht umsetzbar. FML adressiert dieses Problem, indem es Techniken des maschinellen Lernens nutzt, um aus einer vorhandenen Menge von Produktkonfigurationen ein Feature-Modell zu lernen. Dieser Ansatz verspricht eine automatische Rekonstruktion des Konfigurationsraums, indem er die Beziehungen zwischen der SPL-Features identifiziert und daraus ein Feature-Modell in Form eines Black-Box-Classifiers berechnet. Ein zentrales Hindernis, das die Technik von FML jedoch bewältigen muss, ist der Curse-of-Dimensionlity. Dieses Problem resultiert aus der hohen Anzahl an Features innerhalb einer SPL und deren möglichen Kombinationen, was zu einem exponentiell wachsenden Konfigurationsraum führt. In solchen hochdimensionalen Räume werden die Daten zunehmend spärlicher verteilt, was maschinelle Lernalgorithmen vor Herausforderungen stellt.

Um dieses Problem anzugehen, haben wir in dieser Arbeit Strategien zur Dimensionality-Reduction des booleschen Konfigurationsraums eingesetzt. Zuerst haben wir die Grundlagen der DR eingeführt und illustriert wie maschinelle Lern-Datensätze mathematisch durch Matrizen repräsentiert werden können. Das Ziel der DR ist, den hochdimensionalen Eingabedatensatz in eine niedriger dimensionierte Repräsentation zu transformieren, ohne dabei wesentliche Informationen zu verlieren. Dieses Ziel adressiert direkt die Herausforderung des Curse-of-Dimensionality. Wir haben die PCA als eine populäre Methode zur DR beschrieben, die durch die Identifizierung der Hauptachsen (Principal Components) arbeitet, welche die maximale Varianz im Datensatz erfassen. Wir haben auch die Bedeutung der Wahl der richtigen Dimensionalität für den reduzierten Datensatz betont, wobei zwei empirische Heuristiken zur Bestimmung dieser Dimensionalität vorgestellt sind: Der kumulative Prozentsatz der erklärten Varianz (CPVE) und die Suche nach dem *Elbow*-Punkt in einem Scree-Plot. Im weiteren Verlauf haben wir die Anpassung der PCA für boolesche Konfigurationsräume durch die Einführung der Logistic Principal-Component-Analysis (Logistic PCA) erläutert. Logistic PCA nutzt eine wahrscheinlichkeitsbasierte Methode, um die Daten als Log-Odds-Matrix zu modellieren. Diese Matrix wird anschließend so zerlegt, dass eine Reduktion der Dimensionalität erfolgt, während gleichzeitig möglichst viele Informationen erhalten bleiben.

Für die Umsetzung des Feature-Modell-Lernens haben wir AutoGluon, ein AutoML-Toolkit, eingesetzt, um den gesamten Lernprozess zu automatisieren. Die Dimensionality-Reduction wurde separat in einer R-Programmierumgebung durchgeführt. Die Programmiersprache R bietet eine umfangreiche Sammlung von Paketen und Funktionen speziell für statistische Analysen und Datenverarbeitung, insbesondere für die Durchführung von Dimensionality-Reduction. Unsere experimentelle Analyse hat einen Vergleich der Ergebnisse des Feature-Modell-Lernens mit und ohne den Einsatz der Dimensionality-Reduction umfasst. Dieser Vergleich konzertiert sich auf die Bewertung sowohl der Effizienz (im Bezug auf den Rechenaufwand) als auch der Effektivität (bewertet anhand der Präzisions- und Recall-Metriken) jedes Ansatzes. Die Ergebnisse dieser Untersuchung werden wir im folgenden Abschnitt zusammenfassend darstellen.

7.2 Ergebnisse

Die Untersuchung zum Feature-Modell-Lernen unter Verwendung der Principal-Component-Analysis liefert Ergebnisse zu drei zentralen Forschungsfragen. Zunächst zeigt sich, dass eine höhere Anzahl von Principal-Components (k-Werte) die Präzisions- und Recall-Werte im FML verbessert. Allerdings führt dies auch zu längeren Laufzeiten, besonders bei einer hohen Anzahl ungültiger Konfigurationen, während sehr niedrigere k-Werte zwar kürzere Laufzeiten aufweisen, aber schlechtere Präzisions- und Recall-Werte liefern. Beim Vergleich von FML ohne PCA-Einsatz und dann mit PCA-Einsatz für sowohl Linear als auch Logistic PCA unter Verwendung von dem höchsten k-Wert, welcher wir als Antwort für die erste Forschungsfrage gefunden haben, wird deutlich, dass die Version ohne PCA tendenziell höhere Präzisions- und Recall-Werte erzielt, vor allem bei einer geringen Anzahl ungültiger Konfigurationen. Die Wahl zwischen Linear und Logistic PCA hat keinen signifikanten Einfluss auf die Effektivität und Effizient des FMLs. Hinsichtlich des Rechenaufwands zeigt sich, dass die Integration von DR, sowohl durch Linear PCA als auch durch Logistic PCA, zu einer Stabilisierung der Laufzeiten beiträgt und insbesondere bei großen Datensätzen die Effizienz des FML verbessert. Die Vorteile von PCA hinsichtlich der Laufzeiten variieren je nach Fallstudie, wobei komplexe Systeme wie das Body-Comfort-System und Systeme mit ausschließlich binären Constraints besonders profitieren.

Zusammenfassend können wir sagen, dass der Einsatz von PCA im Kontext des FMLs einen klaren Trade-off zwischen Effektivität und Effizienz darstellt. Die Entscheidung für oder gegen die Verwendung von DR erfordert, einen ausgewogenen Ansatz zwischen der Maximierung der Effektivität des Lernprozesses und der Minimierung des Rechenaufwands zu finden.

7.3 Ausblick

Um die Dimensionality-Reduction im Kontext des Feature-Modell-Lernens weiter zu verbessern, könnten wir in zukünftigen Arbeiten die folgenden Aspekte betrachten:

- Zukünftige Arbeiten könnten den Effekt von komplexere, manuell entworfene Kodierungsstrategien zu untersuchen, die die Dimensionen des resultierenden Datensatzes reduzieren.
- Durchführung umfangreicher Vergleichsstudien mit anderen Techniken außerhalb von AutoGluon und den in R verfügbaren PCA-Methoden, um ein breiteres Spektrum an Möglichkeiten für das Feature-Modell-Lernen zu evaluieren.
- Die Implementierung von weiterer PCA-Varianten oder anderer Techniken zur Reduktion von Datensätzen, die speziell für binären und diskreten Natur von Daten entwickelt sind.
- Die detaillierte Analyse des Fine-Tuning-Potenzials von Parametern der PCA, insbesondere des Parameters m in der Logistic PCA.

Literatur

[1] Oluseun Omotola Aremu, David Hyland-Wood und Peter Ross McAree. „A machine learning approach to circumventing the curse of dimensionality in discontinuous time series machine data“. In: *Reliability Engineering and System Safety* 195 (2020), S. 106706. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2019.106706. URL: https://www.sciencedirect.com/science/article/pii/S0951832019304752.

[2] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan und Joachim M Buhmann. „The balanced accuracy and its posterior distribution“. In: (2010), S. 3121–3124.

[3] Michael Collins, S. Dasgupta und Robert E Schapire. „A Generalization of Principal Components Analysis to the Exponential Family“. In: *Advances in Neural Information Processing Systems*. Hrsg. von T. Dietterich, S. Becker und Z. Ghahramani. Bd. 14. MIT Press, 2001. URL: https://proceedings.neurips.cc/paper_files/paper/2001/file/f410588e48dc83f2822a880a68f78923-Paper.pdf.

[4] Pádraig Cunningham. „Dimension reduction“. In: *Machine learning techniques for multimedia: Case studies on organization and retrieval*. Springer, 2008, S. 91–112.

[5] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li und Alexander Smola. „Autogluon-tabular: Robust and accurate automl for structured data“. In: *arXiv preprint arXiv:2003.06505* (2020).

[6] Eelco Rommes Frank Linden Klaus Schmid. „Software Product Lines in Action“. In: (10 June 2007). ISSN: 978-3-540-71437-8. DOI: https://doi.org/10.1007/978-3-540-71437-8. URL: https://link.springer.com/book/10.1007/978-3-540-71437-8.

[7] Patrick Franz, Thorsten Berger, Ibrahim Fayaz, Sarah Nadi und Evgeny Groshev. „ConfigFix: Interactive Configuration Conflict Resolution for the Linux Kernel“. In: *Software Engineering in Practice (ICSE-SEIP)*. 2021, S. 91–100. DOI: 10.1109/ICSE-SEIP52600.2021.00018.

[8] Annie George und A Vidyapeetham. „Anomaly detection based on machine learning: dimensionality reduction using PCA and classification using SVM“. In: *International Journal of Computer Applications* 47.21 (2012), S. 5–8.

[9] Xin He, Kaiyong Zhao und Xiaowen Chu. „AutoML: A survey of the state-of-the-art“. In: *Knowledge-Based Systems* 212 (2021), S. 106622.

[10] Zena M Hira und Duncan F Gillies. „A review of feature selection and feature extraction methods applied on microarray data“. In: *Advances in bioinformatics* 2015 (2015).

[11] Donald A Jackson. „Stopping rules in principal components analysis: a comparison of heuristical and statistical approaches“. In: *Ecology* 74.8 (1993), S. 2204–2214.

[12] Kyo Kang, Sholom Cohen, James Hess, William Novak und A. Peterson. In: CMU/SEI-90-TR-021 (1990). Accessed: 2023-Oct-17. URL: https://insights.sei.cmu.edu/library/feature-oriented-domain-analysis-foda-feasibility-study/.

[13] Christian Kastner, Thomas Thum, Gunter Saake, Janet Feigenspan, Thomas Leich, Fabian Wielgorz und Sven Apel. „FeatureIDE: A tool framework for feature-oriented software development“. In: *2009 ieee 31st international conference on software engineering*. IEEE. 2009, S. 611–614.

[14] Mohamed Lamine Kerdoudi, Tewfik Ziadi, Chouki Tibermacine und Salah Sadou. „A novel approach for Software Architecture Product Line Engineering“. In: *Journal of Systems and Software* 186 (2022), S. 111191. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2021.111191. URL: https://www.sciencedirect.com/science/article/pii/S0164121221002685.

[15] Yann LeCun, Yoshua Bengio und Geoffrey Hinton. „Deep Learning“. In: *Nature* 521.7553 (2015), S. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

[16] Sascha Lity, Remo Lachmann, Malte Lochau und Ina Schaefer. „Delta-oriented software product line test models-the body comfort system case study“. In: *Technical report, TU Braunschweig* (2013).

[17] Swati V Narwane und Sudhir D Sawarkar. „Dimensionality Reduction of Unbalanced Datasets: Principal Component Analysis“. In: *2021 Asian Conference on Innovation in Technology (ASIANCON)*. IEEE. 2021, S. 1–6.

[18] Damir Nešić, Jacob Krüger, undefinedtefan Stănciulescu und Thorsten Berger. „Principles of Feature Modeling“. In: ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery, 2019, 62–73. ISBN: 9781450355728. DOI: 10.1145/3338906.3338974. URL: https://doi.org/10.1145/3338906.3338974.

[19] Per Runeson und Martin Höst. „Guidelines for conducting and reporting case study research in software engineering“. In: *Empirical software engineering* 14 (2009), S. 131–164.

[20] Andrew I Schein, Lawrence K Saul und Lyle H Ungar. „A generalized linear model for principal component analysis of binary data“. In: *International Workshop on Artificial Intelligence and Statistics*. PMLR. 2003, S. 240–247.

[21] C. O. S. Sorzano, J. Vargas und A. Pascual Montano. „A survey of dimensionality reduction techniques“. In: (2014). arXiv: 1403.2877 [stat.ML].

[22] Michel Verleysen und Damien François. „The curse of dimensionality in data mining and time series prediction“. In: (2005), S. 758–770.

[23] Mathis Weiß. „Comparing Efficiency and Effectiveness of Feature Model Synthesis and Feature Model Learning“. In: (8September 2023).

[24] Laurenz Wiskott. „Lecture notes on principal component analysis“. In: (2013).