Hochschule Darmstadt

Fachbereiche Mathematik
und Naturwissenschaften
& Informatik

Einsatzmöglichkeiten der topologischen Datenanalyse zur Verbesserung von Modellen und des Modellverständnisses

Abschlussarbeit zur Erlangung des akademischen Grades
Master of Science (M. Sc.)
im Studiengang Data Science

vorgelegt von
Martin Alexander Wilk

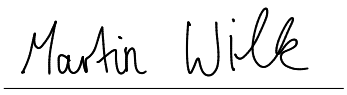
Referentin:	Prof. Dr. Inge Schestag
Korreferent:	Prof. Dr. Sebastian Döhler
Betreuer:	Dr. Jochen Papenbrock

Ausgabedatum:	16.07.2018
Abgabedatum:	31.12.2018

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, den 23. Dezember 2018



Zusammenfassung

Die korrekte Vorhersage von betrügerischen Kreditkartentransaktionen oder ausfallenden Krediten verringert die Verluste der Banken durch Kreditkartenbetrug oder nicht korrekt bezahlte Kredite. Aktuell erfreut sich Deep Learning großer Beliebtheit, welches zwar häufig eine gute Performance besitzt, aber keine Erklärung des Zustandekommens der Ergebnisse liefert. In einigen Bereichen wie im Finanzsektor ist die Erklärbarkeit der Modelle eine Voraussetzung für deren Einsatz.

In dieser Masterthesis wird ein Prozess vorgestellt, wie mithilfe der topologischen Datenanalyse in Form des Mapper Graphs die Performance der Klassifikation durch Entscheidungsbäume und die logistische Regression verbessert werden kann. Ein Ziel der Masterthesis ist das Design und die Implementierung des vorgestellten Prozesses.

Der vorgestellte Prozess wird auf den Kreditkartentransaktionsdatensatz von Kaggle, auf den German Credit Datensatz und auf einen Datensatz der Peer-to-Peer Kreditplattform Lending Club angewandt. Auf dem Kreditkartentransaktionsdatensatz ist die Performance der logistischen Regression um rund 6 Prozentpunkte angestiegen, die Performance des Entscheidungsbaumklassifikators erhöht sich um rund 5 Prozentpunkte. Der vorgestellte Prozess führt auf dem German Credit Datensatz zu einem Anstieg des AUPRC des Entscheidungsbaumklassifikators um etwa 14 Prozentpunkte. Auf dem Lending Club Datensatz ist ein Anstieg der Performance des Entscheidungsbaumklassifikators um 2 Prozentpunkte zu verzeichnen. Die Durchführung des Prozesses verursacht nicht in jedem Fall eine Zunahme des Performanceindexes AUPRC.

Insgesamt haben die Experimente gezeigt, dass der vorgestellte Prozess eine Möglichkeit darstellt, die Performance von einfacheren Klassifikatoren anzuheben. Daher lohnt es sich, die Einsatzmöglichkeiten der topologischen Datenanalyse auf dem eigenen Datenbestand zu evaluieren.

Abstract

The accurate prediction of credit card fraud or loan default helps banks to reduce the losses incurred by fraudulent transactions or loan default. Currently deep learning models are applied frequently due to their high accuracy. A major drawback of these models is that the outcome of these models are not explainable and interpretable. For some applications, including the finance industry, the explainability of models is key.

This master thesis presents a process based on the generation of the mapper graph to increase the performance of decision tree models and logistic regression. The mapper graph is an approach of topological data analysis and gives a summary of the shape of the data. The design and implementation of this process is also comprised in this master thesis.

The presented process is applied to a credit card fraud data set, the German credit data set and a dataset of all loans of Lending Club, which is an American peer to peer lending platform. On the credit card transactions dataset, the performance of the logistic regression increases by approximately 6 percentage points, the performance of the decision tree classifier is increased by approximately 5 percentage points. On the German credit data set, the presented process leads to an improvement of the AUPRC performance measure of the decision tree classifier by 14 percentage points. On the Lending Club dataset, the performance is increased by 2 percentage points on the decision tree classifier. But not all experiments show an increase of the performance by using the process presented in this thesis.

In total, the experiments are showing that the process, which is presented in this thesis, is able to increase classification performance on the data sets used. Therefore, it is worthwhile to check if topological data analysis leads to an improvement of the classifier on the data used.

Abkürzungsverzeichnis

AUPRC . . . . . area under precision recall curve

AUROC . . . . . area under receiver operating curve

CCID . . . . . Connected Component ID

LASSO . . . . . Least Absolute Shrinkage and Selection Operator

LDA . . . . . lineare Diskriminanzanalyse

ROC . . . . . Receiver Operating Curve

TSNE . . . . . t- distributed Stochastic Neighbor Embedding

UMAP . . . . . Uniform Manifold Approximation and Projection

Inhaltsverzeichnis

Teil I.

Grundlagen und aktuelle Forschung

1. Einleitung

Im Jahre 2017 bezifferte sich die Schadensumme durch Kreditkartenbetrug in Großbritannien auf rund 747 Millionen Euro. In Deutschland lag im gleichen Zeitraum die Höhe der Schäden, der den Banken, Händlern und Konsumenten durch Kreditkartenbetrug entstand, bei 94 Millionen Euro [50]. Weltweit betrug der Schaden durch Kreditkartenbetrug im Jahr 2016 22,8 Milliarden Dollar [38]. Zur Verminderung dieser hohen Schäden verwenden die Banken und Kreditkartenunternehmen Verfahren, um mögliche Betrugsfälle frühzeitig erkennen zu können. Eine weitere Fragestellung des Finanzbereichs ist die Vorhersage, ob ein Kredit ordnungsgemäß bezahlt werden kann, um daraufhin entscheiden zu können, ob dem jeweiligen Kredit anfragenden der Kredit gewährt werden soll.

Um durch die frühzeitige Erkennung von betrügerischen Kartentransaktionen oder ausfallenden Krediten den entstandenen Verlust für die Bank möglichst gering zu halten, sollte das Verfahren zur Klassifikation der Transaktionen beziehungsweise der Kreditanfragen eine hohe Prognosegüte aufweisen. Aktuell erfreut sich Deep Learning in Form von neuronalen Netzen großer Beliebtheit, da hier die Prognosegüte oftmals besser ist, als die herkömmlicher, weniger komplexer Verfahren. Der Hauptnachteil des Deep Learning besteht jedoch in der hohen Komplexität der generierten Modelle. Daher lässt sich oftmals das Zustandekommen des Outputs der neuronalen Netze nicht erklären und begründen, weshalb beispielsweise eine bestimmte Kreditkartentransaktion als betrügerische Transaktion angesehen wird. Besonders bei der Entscheidung über die Gewährung eines Kredites ist eine Begründung des Ergebnisses notwendig. Auch in anderen Themenbereichen ist die Nachvollziehbarkeit der Ergebnisse von Modellen essentiell. Daher fordert die Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin) in dem Thesenpapier [7], dass künftig die auf Basis von Algorithmen getroffenen Entscheidungen erklärt und plausibilisiert werden müssen. Dadurch kann ein besseres Verständnis des verwendeten Modells ermöglicht werden, welches die Erkennung von Fehlern im algorithmenbasierten Entscheidungsprozess vereinfacht.

Als Konsequenz der Forderung der BaFin besteht die Notwendigkeit nach Verfahren, die einerseits leicht erklärbar sind und gleichzeitig Prognosen mit einer hohen Qualität liefern.

Mit einer Verbesserung der Erklärbarkeit von Modellen wegen der geringeren Komplexität ist jedoch häufig eine Abnahme der Qualität des Klassifikators zu beobachten. Ein Mittel zur Verbesserung der Ergebnisse von Klassifikationsverfahren könnte die topologische Datenanalyse darstellen, welche eine Zusammenfassung der Form (engl. *shape*) des Datensatzes liefert und somit möglicherweise eine bessere topologische Trennbarkeit der Objekte hinsichtlich des Targets aufzeigt. Durch die Hinzunahme der aus der Zusammenfassung der Form des gegebenen Datensatzes gewonnenen Informationen in Form eines neuen Features ist eine Anhebung der Performance des Klassifikators naheliegend, da der Trainingsdatensatz ein neues Feature enthält, welches die Form des Datensatzes abbildet. In der vorliegenden Masterthesis soll untersucht werden, ob durch die topologische Datenanalyse ein Anstieg der Qualität von Modellen zur Klassifikation erzielt werden kann. Dies geschieht exemplarisch auf Basis von drei Datensätzen aus dem Finanzbereich verschiedener Größe. Das Ziel dieser Masterthesis ist die Konzeption und Implementierung eines Prototyps, welcher die durch die topologische Datenanalyse unterstützte Klassifikation durchführt. Dies ermöglicht das angenehme Testen verschiedener Parametereinstellungen. Im Rahmen der Untersuchungen in dieser Arbeit werden zur Klassifikation Entscheidungsbäume und die logistische Regression mit Least Absolute Shrinkage and Selection Operator (LASSO) Regularisierung verwendet.

In der Literatur finden sich zwei Ansätze der topologischen Datenanalyse. Ein Ansatz der topologischen Datenanalyse ist die persistente Homologie, welche sich mit der Analyse von „Löchern“ in den Datensätzen befasst. Die zweite Methode ist die Generierung eines Mapper Graphen als topologische Zusammenfassung des Datensatzes. Im Zusammenhang mit der topologischen Datenanalyse mithilfe des Ansatzes des Mapper Graphen ist das US-amerikanische Unternehmen Ayasdi zu nennen. Ayasdi ist ein Softwarehersteller, welcher Lösungen zur Datenanalyse und die Erstellung von Prädiktionsmodellen anbietet. Hierbei ist die topologische Datenanalyse in Form des Mapper Graphs eine Kerntechnologie des Unternehmens, die Verfahren aus dem Bereich des Machine Learning unterstützen soll. Diese Behauptung lässt sich auf die diversen von Ayasdi veröffentlichten Whitepaper stützen [3–6].

Der gegenwärtige Forschungsstand besteht hauptsächlich aus der Verwendung des Ansatzes der persistenten Homologie, um mithilfe der Features, die anhand der persistenten Homologie ermittelt werden, eine höhere Performance der Klassifikatoren zu erhalten. Der Einsatz des Mapper Graphen beschränkt sich oftmals auf eine Datenexploration der Struktur des Datensatzes, um relevante Features auszuwählen beziehungsweise durch statistische Methoden zu bestimmen. Ein weiteres Einsatzgebiet ist die Visualisierung der Schichten eines neuronalen

Netzwerks, um einen Eindruck zu bekommen, was in der jeweiligen Schicht des neuronalen Netzwerks passiert [18]. Eine Nutzung der Informationen des Mapper Graphs in Form eines zusätzlichen Features, welches die Performance des Klassifikators anhebt, wird in der aktuellen Forschung nicht durchgeführt. In dieser Masterthesis soll nun diese Lücke geschlossen und ein Prozess vorgestellt und implementiert werden, wie die Klassifikation von Daten durch ein zusätzliches topologisches Feature, welches aus den Informationen des Mapper Graphs stammt, verbessert werden kann. Zur Erstellung des Mapper Graphs kommt der Kepler Mapper, eine öffentlich zugängliche Implementierung in der Programmiersprache Python, zum Einsatz. Zunächst wird eine topologische Zusammenfassung des Datensatzes in Form eines Graphen generiert. Im nächsten Schritt erfolgt eine Analyse des generierten Graphen hinsichtlich der zusammenhängenden Komponenten, welche zur Bestimmung des zusätzlichen Features des Datensatzes benötigt werden. Als neues topologisches Feature wird jedem Datenobjekt die ID der Zusammenhangskomponente des Nodes zugewiesen, indem das jeweilige Datenobjekt liegt. Der in dieser Masterthesis vorgestellte Prozess wird auf drei Datensätze aus dem Finanzbereich angewandt und die erhaltenen Ergebnisse wiedergegeben und interpretiert.

Insgesamt umfasst der in dieser Thesis erarbeitete Prozess eine hohe Anzahl von Parametern, für die keine Methode zur Optimierung existiert. Somit müssen die Optimalwerte experimentell bestimmt werden. Um dies zu ermöglichen, ist das Design und die Implementierung eines Prototyps, der den gesamten Prozess von der Teilung der Daten in Trainings- und Testmenge bis zur Durchführung der topologische Datenanalyse auf dem Trainingsdatensatz bis zur Evaluation der Klassifikationsergebnisse auf dem Trainings- und Testdatensatz abdeckt, das Ziel dieser Masterthesis.

Im Rahmen dieser Masterthesis wurden drei Versionen mit unterschiedlichen Zielsetzungen dieses Prototypen geplant und implementiert. Zwei Versionen sind interaktiv und stellen dem Benutzer Zwischenergebnisse während des Programmlaufes grafisch dar. Der Benutzer kann daraufhin passend reagieren. Dieser interaktive Ansatz erlaubt die experimentelle Erforschung optimaler Werte für die zahlreichen Parameter des Gesamtprozesses der Arbeit.

Die Realisierung der interaktiven Versionen mit grafischer Benutzeroberfläche erfolgt zum Einen als Webanwendung und zum Anderen als Desktopanwendung. Zur Implementierung des Anwendungsprogramms kommt das Modul `tkinter`[1] zum Einsatz, welches zur Programmierung einer grafischen Benutzeroberfläche in Python geeignet ist. Die Implementierung der

[1] `https://docs.python.org/3/library/tk.html`

Webanwendung erfolgt mithilfe des Moduls `Dash`[2] in Python.

Eine weitere Variante ist eine *batch* Version, welche durch die Durchführung von Crossvalidierung geeignet ist, die Modellqualität auf unbekannten Daten zu evaluieren.

Struktur der Masterthesis

Zunächst wird im Kapitel 2 eine kurze Einführung in das Themengebiet Machine Learning gegeben. Diese umfasst eine kurze Beschreibung der Funktionsweise von Entscheidungsbäumen und der logistischen Regression mit einer LASSO Regularisierung. Zudem folgt eine Beschreibung der verwendeten Maße zur Beurteilung der Performance der genutzten Klassifikatoren. Anschließen wird sich ein Überblick über die Mathematik der topologischen Datenanalyse und eine Vorstellung der Eigenschaften der zur Generierung des Mapper Graphs verwendeten Linsen.

Das Kapitel 3 liefert einen Überblick über die aktuelle Forschung der Fragestellung, wie die topologische Datenanalyse zur Verbesserung von Klassifikationsverfahren genutzt werden kann. Hierbei werden ausgewählte Anwendungen der beiden Ansätze der topologischen Datenanalyse exemplarisch vorgestellt.

Im Kapitel 4 folgt eine Beschreibung des in dieser Arbeit erarbeiteten Prozesses zur Durchführung einer Klassifikation, die Informationen über die Form des Datensatzes nutzt, die mit der topologischen Datenanalyse extrahiert wurden. Die Untersuchungen zur Fragestellung, ob die topologische Datenanalyse eine Steigerung der Vorhersagegüte der Klassifikatoren bringt, erfolgen jeweils auf drei Datensätzen mit unterschiedlichem Umfang aus dem Finanzbereich. Einer dieser Datensätze enthält Kreditkartentransaktionen mit der Kennzeichnung, ob es sich bei einer Transaktion um Betrug handelt. Dieser Datensatz ist auf der Plattform Kaggle verfügbar. Aus diesem Grund sind alle Features bis auf den Zeitpunkt der Transaktion und deren Betrag durch eine Hauptkomponentenanalyse anonymisiert. Ebenfalls für diese Masterthesis genutzte Datensätze sind zum einen der bekannte German Credit Datensatz und ein Datensatz der US-amerikanischen Peer-to-Peer Kreditplattform Lending Club. Der German Credit Datensatz enthält Informationen von Krediten und ob diese bereits ausgefallen sind. Der Lending Club Datensatz besteht ebenfalls aus Informationen zu Krediten und dem Merkmal, ob es zu einem Zahlungsausfall gekommen ist. Eine genauere Beschreibung der Datensätze findet

[2] `https://dash.plot.ly`

sich im Kapitel 4.

Die Ergebnisse des Kreditkartentransaktionsdatensatzes der Data Science Plattform Kaggle findet sich in Kapitel 5. Die Ergebnisse auf dem German Credit Datensatz finden sich im Kapitel 6, die auf dem Datensatz der Peer-to-Peer Kreditplattform Lending Club sind im Kapitel 7 dargestellt und erläutert.

Eine Beschreibung der Implementierungen der bereitgestellten Versionen des Prototyps findet sich in Kapitel 8.

Im Kapitel 9 sollen neben einer Zusammenfassung der Erkenntnisse einen Ausblick über die mit dem Thema der Masterthesis zusammenhängenden Fragestellungen, welche nicht in dieser Arbeit thematisiert werden.

2. Mathematische Grundlagen

Dieses Kapitel soll die notwendigen mathematischen Grundlagen der Masterthesis vorstellen. Dies umfasst zunächst eine kurze Einführung in die Thematik der Klassifikation von Objekten mit einer genaueren Beschreibung der in dieser Arbeit genutzten Klassifikationsverfahren. Darüber hinaus werden Wege aufgezeigt, wie die Qualität der Klassifikatoren bewertet werden kann. Nach einer Einführung in das Themengebiet der topologischen Datenanalyse folgt eine kurze Beschreibung der zur Durchführung der topologischen Datenanalyse verwendeten Linsen. Dieses Kapitel schließt mit einer Kurzeinführung in die Analyse der Zusammenhangskomponenten von Graphen.

2.1. Einführung in die Klassifikation

Das Ziel einer Klassifikation ist es auf Basis von Daten $X = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \subset \mathbb{R}^d \times \{0, 1\}$ ein Modell $m\colon X \to \{0, 1\}$ zu trainieren, welches jedem d dimensionalen Featurevektor $\mathbf{x}_i \in \mathbb{R}^d$ eine geschätzte Klassifikation $\hat{y}_i \in \{0, 1\}$ zuweist. Zur Durchführung der Klassifikation wird der Datensatz X geteilt, sodass eine Evaluierung der Performance auf einer Datenmenge erfolgen kann, die nicht zum Training des Modells genutzt wurde. Der Datensatz, der zum Training des Modells m genutzt wird, wird als **Trainingsdatensatz** bezeichnet, der Datensatz der zur Vorhersage der Klassifikation genutzt wird heißt **Testdatensatz**. Da der Testdatensatz ebenfalls die korrekte Klassifikation enthält, kann die Qualität des Klassifikators durch den Vergleich von y_i mit $\hat{y}_i$ beurteilt werden. Der Teil 2.1.3 widmet sich den Performancemaßen, welche in dieser Masterthesis verwendet werden, um beurteilen zu können, ob eine Verbesserung der Klassifikation eingetreten ist.

Für die Durchführung der Klassifikation existieren unterschiedliche Algorithmen mit verschiedenen Konzepten. In dieser Arbeit kommt zur durch die topologische Datenanalyse unterstützten Klassifikation ein Entscheidungsbaumklassifikator und die logistische Regression mit LASSO Regularisierung zum Einsatz. Diese Verfahren sollen im Folgenden kurz vorgestellt werden.

2.1.1. Entscheidungsbäume

Entscheidungsbäume eignen sich zur Klassifikation und Regression von Daten und nehmen eine rekursive Partitionierung des Datenbestandes vor. In jeder Partition wird ein Modell trainiert welches jeweils einer Konstante entspricht. Bei der Klassifikation entspricht dies der häufigsten Klasse im jeweiligen Knoten, in den das jeweilige Objekt fällt. Ein wichtiger Vorteil des Entscheidungsbaumes ist die gute Erklärbarkeit des Verfahrens, da die vorgenommene Klassifikation aufgrund der Darstellung des Klassifikators als Baum direkt nachvollzogen werden kann.

Beginnend mit dem Wurzelknoten, in dem der gesamte Datensatz enthalten ist, erfolgt eine Verzweigung der Elternknoten in einen linken und einen rechten Teilbaum. Der in der Masterthesis eingesetzte Entscheidungsbaumalgorithmus nimmt immer binäre Splits vor. Es gibt allerdings Entscheidungsbaumalgorithmen, welche kategorielle Merkmale nach jeder vorhandenen Ausprägung verzweigen. Die Bestimmung des Splits erfolgt durch eine Bedingung vom Typ $feature \geqslant s$, wobei $feature$ ein Feature des Datensatzes und s den Splitwert bezeichnet. Das zum Splitten verwendete Feature wird auch als Splitattribut bezeichnet. Das Ziel des Entscheidungsbaumes ist es, dass die Knoten möglichst rein sind, das heißt, dass die Objekte in einem Knoten möglichst zur identischen Klasse gehören. Daher werden die für die Trennung der Daten benötigten Features und Vergleichswerte so bestimmt, dass die Kindknoten möglichst rein sind. Dies erfolgt durch die Berechnung eines Reinheitsmaßes der Kindknoten für alle zur Verfügung stehenden Features. Zur Berechnung des Reinheitsmaßes ist es notwendig den Anteil der Objekte mit der Klasse k im Knoten m zu kennen. Aus diesem Grund entspricht N_m der Gesamtzahl der Objekte in Knoten m und R_m bezeichnet die Menge aller Datensätze aus X, die in den Knoten m fallen. Die Funktion $\mathbb{1}(y_i = k)$ ist eine Indikatorfunktion und besitzt den Wert 1, wenn $y_i = k$ gilt und 0, wenn y_i und k ungleich sind. Mit diesem Vorbereitungen kann nun der gesuchte Anteil der Objekte der Klasse k in dem Knoten m durch die Gleichung 2.1.1 berechnet werden.

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} \mathbb{1}(y_i = k) \tag{2.1.1}$$

Mithilfe des Anteils $\hat{p}_{mk}$ lassen sich nun die Reinheitsmaße *Gini-Index* und *Entropie* sowie

die *Falschklassifikationsrate* im Knoten m berechnen (vgl. [23, S. 309]):

$$i_{MCR} = 1 - \hat{p}_{mk} \quad \textit{Falschklassifikationsrate im Knoten } m \tag{2.1.2}$$

$$i_{gini} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \quad \textit{Gini-Index} \tag{2.1.3}$$

$$i_{info} = \sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \quad \textit{Entropie oder Information-Gain} \tag{2.1.4}$$

Eines der Reinheitsmaße aus Gleichung 2.1.2 wird nun zur Auswahl des Splitattributs verwendet, indem für jedes Feature das Reinheitsmaß ermittelt wird. Das Feature mit der größten Reinheit wird für die Partitionierung des Baumes verwendet. Dieser Prozess wird solange wiederholt, bis eine Teilung der Datensätze nicht mehr möglich ist, da alle Blätter rein sind. Durch das hierarchische Vorgehen bei der Generierung des Baumes und der Teilung bis in den Blättern nur noch eine Klasse vertreten ist, neigen Entscheidungsbäume zur Überanpassung (*Overfitting*) [23, S. 312]. Um dieses Problem zu lösen, ist das Beschneiden (*pruning*) der Bäume sinnvoll. Das Pruning kann während des Trainingsprozesses (sog. *pre-pruning*) oder nach vollständig abgeschlossenem Trainingsprozess (*post-pruning*) erfolgen. Bei der verwendeten R-Implementierung des Entscheidungsbaumes wird mithilfe des Parameters cp ein *pre-pruning* durchgeführt. Durch das Pruning erfolgt eine Verzweigung des Baumes nur, wenn diese die Modellqualität des Baumes deutlich verbessert. Genauere Informationen zur Funktionsweise des Prunings mithilfe des Komplexitätsparameters cp finden sich in [48]. Der in dieser Arbeit verwendete Entscheidungsbaum splittet numerische und kategorielle Merkmale binär und verwendet als Reinheitsmaß zur Wahl des Splitattributs den Gini Index. Wie eingangs erwähnt, erfolgt die Vorhersage der Klasse eines Objektes beim Entscheidungsbaumbaum durch die häufigste Klassifikation im jeweiligen Blattknoten, in welchen das Objekt nach den Entscheidungsregeln der Baumes fällt.

2.1.2. Logistische Regression mit Lasso Shrinkage

Ein weiteres, in dieser Arbeit genutztes Klassifikationsverfahren ist die logistische Regression. Die Idee der logistischen Regression besteht darin, die Wahrscheinlichkeiten der Klassenzugehörigkeit einer Beobachtung x durch lineare Funktionen in x zu modellieren. Da es sich um Wahrscheinlichkeiten handelt, soll die Summe über alle Klassen der Werte 1 betragen und die Wahrscheinlichkeit für eine bestimmte Klasse im Wertebereich zwischen 0 und 1 liegen. Somit ergibt sich für den Fall, dass es K Klassen gibt, für die Wahrscheinlichkeit, dass x zur

Klasse k gehört, die Darstellung aus 2.1.5.

$$P(Y = k|X = \mathbf{x}) = \frac{e^{\beta_{k0}+{\boldsymbol{\beta}_k}^T\mathbf{x}}}{1+\sum_{l=1}^{K-1} e^{\beta_{l0}+{\boldsymbol{\beta}_l}^T\mathbf{x}}}, \quad k = 1, \ldots, K-1 \tag{2.1.5}$$

$$P(Y = K|X = \mathbf{x}) = \frac{1}{1+\sum_{l=1}^{K-1} e^{\beta_{l0}+{\boldsymbol{\beta}_l}^T\mathbf{x}}} \tag{2.1.6}$$

Die Wahrscheinlichkeiten aus der Gleichung 2.1.5 hängen von den Koeffizienten $\theta = \beta_{10}, \beta_1^T, \ldots, \beta_{(K-1)0}, \beta_{K-1}^T$ des logistischen Regressionsmodells ab. Wegen dieser Abhängigkeit setzen wir $P(Y = k|X = \mathbf{x}) = p_k(\mathbf{x}, \theta)$. Durch die Beschränkung auf den Fall $K = 2$ mit $Y \in \{0, 1\}$ beschränken sich die zu berechnenden Wahrscheinlichkeiten auf $P(Y = 1|X = \mathbf{x})$. Dies ist die Wahrscheinlichkeit, dass das Objekt x positiv klassifiziert wird. Für den gesamten Datenbestand X ergibt sich daher für das Objekt i des Datensatzes die Formel aus 2.1.7.

$$\mathrm{P}(Y = 1 \mid X = \mathbf{x}_i) = \mathrm{P}(Y_i = 1) = \frac{e^{\beta_0+\mathbf{x}_i^\top\boldsymbol{\beta}}}{1+e^{\beta_0+\mathbf{x}_i^\top\boldsymbol{\beta}}} = \frac{1}{1+e^{-(\beta_0+\mathbf{x}_i^\top\boldsymbol{\beta})}} \tag{2.1.7}$$

Die Schätzung der Parameter θ als Koeffizienten der Linearkombination mit den Werten des Features im Exponenten erfolgt über die Maximum Likelihood Methode. Die logarithmierte Likelihoodfunktion findet sich in Gleichung 2.1.8. Eine Bestimmung der Lösung dieser Funktion ist nur durch numerische Verfahren möglich.

$$l(\beta_0, \boldsymbol{\beta}) = \sum_{i=1}^{N} y_i(\beta_0 + \boldsymbol{\beta}^T x_i) - \log(1 + e^{\beta_0+\boldsymbol{\beta}^T\mathbf{x}_i}) \tag{2.1.8}$$

Wichtig für ein gutes Modell ist die Auswahl der richtigen Prädiktoren. Werden zu viele Prädiktoren verwendet, so besteht die Gefahr, dass man ein übertrainiertes Modell erhält. Dies weist eine gute Performance auf dem Trainingsdatensatz auf, liefert jedoch auf unbekannten Daten unbrauchbare, weil falsche Vorhersagen. Auch ein Modell mit zu wenigen Parametern ist problematisch, da hier eine zu schlechte Anpassung des Modells an die Daten vorliegt und daher die Ergebnisse der Klassifikation ebenfalls schlecht sein können. Des Weiteren ist ein einfaches Modell mit wenigen Prädiktoren gut erklärbar. Zur Reduktion der verwendeten Features haben sich bei der logistischen Regression verschiedene Techniken wie die **Best-Subset-Selection** etabliert [23, S. 61]. Hier wird das Modell schrittweise aufgebaut und jeweils das Feature zum Modell hinzugefügt, welches die höchste Signifikanz zeigt. Aufgrund dieser hierarchischen Strategie besitzt das resultierende Modell oft eine hohe Varianz, welches sich negativ auf den Fehler bei der Prädiktion auswirkt (vgl. [23, S. 61]). Die Lösung dieses Problems liefern Methoden, die die Werte der Parameter „stetig“ verkleinern. Diese

Methoden sind unter dem Namen **Shrinkage** oder **Regularisierung** bekannt. Methoden zur Verkleinerung der geschätzten Parameter der logistischen Regression sind Ridge oder Lasso. Beide Methoden zeichnen sich dadurch aus, dass ein Strafterm aus der Gleichung 2.1.9 zur Berechnung des Modellfehlers hinzugefügt wird, welcher die Größe der Parameter bestraft.

$$p_{ridge}(\boldsymbol{\beta}) = \lambda \sum_{j=1}^{p} \beta_j^2 \quad \textbf{Strafterm der Ridge-Regression} \tag{2.1.9}$$

$$p_{LASSO}(\boldsymbol{\beta}) = \lambda \sum_{j=1}^{p} |\beta_j| \quad \textbf{Strafterm der LASSO-Regression}$$

Der Parameter $\lambda \geqslant 0$ des Strafterms ist der Regularisierungsparameter, welcher die Stärke der Regularisierung einstellt. Ist λ 0, so erfolgt keine Bestrafung von hohen Werten des Parameterschätzers. Da zur Regularisierung nach LASSO jeweils die Betragsfunktion auf die Parameter des Regressionsmodells angewandt wird, ist es möglich ausgewählte Parameter des Modells bis auf Null zu schrumpfen, wie die Abbildung 2.1 zeigt. Dies entspricht einer Herausnahme des jeweiligen Features aus dem Modell. Eine Kombination dieser beiden Techniken

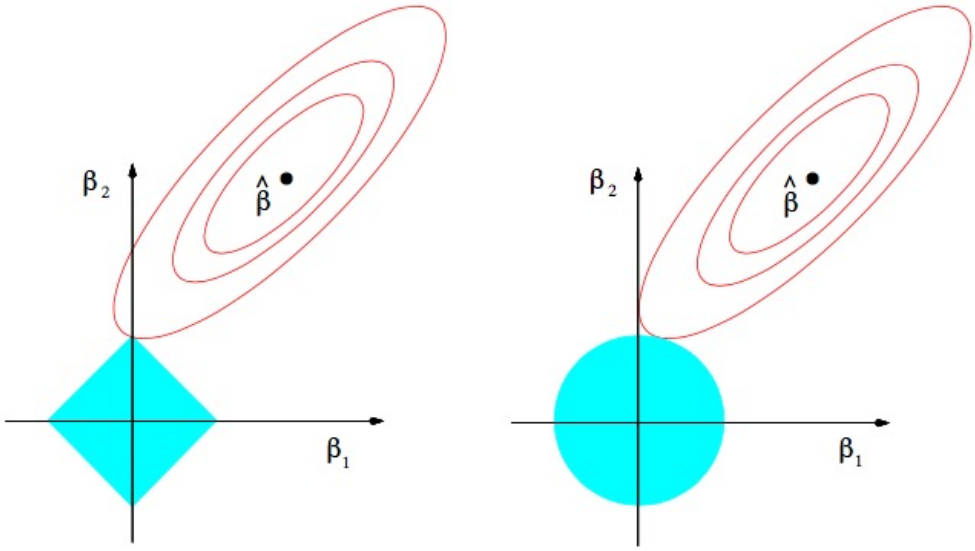


Abbildung 2.1.: Vergleich von Lasso (links) und ridge Regression (rechts). Dargestellt sind jeweils die Höhenlinien der Fehlerfunktion in roter Farbe und die Konturlinien $|\beta_1| + |\beta_2| = t$ der Lasso Regularisierung beziehungsweise $\beta_1^2 + \beta_2^2 = t^2$ der Ridge-Regression in hellblauer Farbe.

zum Schrumpfen der Regressionskoeffizienten stellt das **Elastic Net** dar, welches hier jedoch

nicht thematisiert wird [23, S. 73]. Zur gleichzeitigen Variablenselektion bei der Durchführung der Schrumpfung der Regressionskoeffizienten der logistischen Regression wird eine Regularisierung von Typ LASSO angewandt. Die Anwendung der Regularisierung durch Lasso auf die logistische Regression erfolgt durch die Penalisierung der Maximum-Likelihood Funktion durch die Funktion $p_{LASSO}(\boldsymbol{\beta})$. Die zu maximierende, logarithmierte Maximum Likelihood Funktion findet sich in Gleichung 2.1.10.

$$\max_{\beta_0,\boldsymbol{\beta}}\{\sum_{i=1}^{N}[y_i(\beta_0+\boldsymbol{\beta}^T\mathbf{x}_i)-\log(1+e^{\beta_0+\boldsymbol{\beta}^T\mathbf{x}_i})]-\lambda\sum_{j=1}^{p}|\beta_j|\} \tag{2.1.10}$$

Hier erfolgt die Berechnung der optimalen Parameter, welche die Maximum Likelihood Funktion maximieren ebenfalls über numerische Verfahren. Um anhand der Wahrscheinlichkeit $P(Y=1)$ die Klassifikation $\hat{Y}$ zu bestimmen, muss ein Schwellenwert ξ definiert werden. Liegt die Wahrscheinlichkeit $P(Y=1)$ oberhalb des Schwellenwerts ξ, so wird das Objekt als positiv (1) klassifiziert. Liegt die Wahrscheinlichkeit unterhalb von ξ, so wird die Klasse 0 vorhergesagt.

2.1.3. Performancemaße zur Beurteilung der Qualität der Klassifikation

Nachdem im vorangegangenen Abschnitt die Funktionsweise der genutzten Klassifikatoren vorgestellt wurde, soll hier die Bewertung der Ergebnisse eines Klassifikators thematisiert werden, um aussagen zu können, welcher Klassifikator die beste Performance hat. Hierzu wird zunächst die in Abbildung 2.1 dargestellte Confusion-Matrix bestimmt, welche die Anzahl der richtig positiv (tp), falsch negativ (fn), falsch positiv (fp) und richtig negativ (tn) klassifizierten Objekte angibt. In der Confusion Matrix steht $\hat{Y}$ für die durch das Modell vorhergesagte Klasse und Y für die wahre Klasse des Objektes. Anhand der Anzahlen in dieser Matrix lassen sich beispielsweise durch die Gleichungen 2.1.11 - 2.1.14 abgeleitete Maße zur Beurteilung der Performance berechnen. Beispielsweise kombiniert das F_β Maß aus 2.1.14 die Maße Precision und Recall zu einem weiteren Performancemaß, welches anhand des Parameters β justiert werden kann. Mithilfe des Parameters β kann der Benutzer wählen, ob bei dem gegenwärtigen Anwendungsfall der Klassifikation die Vermeidung von falsch negativen oder falsch positiven Ergebnissen Vorrang hat. Ist $\beta > 1$, so gewichtet das F_β-Maß den Recall höher, liegt β unterhalb von 1, so liegt der Fokus auf der Precision.

	$\hat{Y} = 1$	$\hat{Y} = 0$
$Y = 1$	tp	fn
$Y = 0$	fp	tn

Tabelle 2.1.: Confusion Matrix eines Klassifikators

$$recall = \frac{tp}{tp + fn} \tag{2.1.11}$$

$$fpr = \frac{fp}{fp + tn} \tag{2.1.12}$$

$$precision = \frac{tp}{tp + fp} \tag{2.1.13}$$

$$F_\beta = (1 + \beta^2)\frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}, \qquad \beta \geqslant 0 \tag{2.1.14}$$

Als visuelles Hilfsmittel zur Bewertung der Güte eines Klassifikators können anhand der Performancemaße aus den Gleichungen 2.1.11-2.1.14 ROC-Plots und Precision-Recall Plots generiert werden. In beiden Plots werden Gütemaße des Klassifikators in Abhängigkeit eines Schwellenwerts in einer Grafik darstellt. Dieser Schwellenwert gibt den Punkt an, ab dem ein Objekt auf Basis der Wahrscheinlichkeit, dass dieses Objekt zur positiven Klasse gehört, positiv klassifiziert wird. Positiv bedeutet in diesem Fall, dass der Klassifikator eine Transaktion als Betrug einstuft.

Im ROC Chart werden die Maße Recall und die falsch-positiv Rate in Abhängigkeit des Schwellenwerts in einem Diagramm dargestellt. Der ROC Chart eines zufälligen Klassifikators entspricht der Winkelhalbierenden. Die Berechnung der Gütemaße aus Gleichung 2.1.11 und 2.1.12 erfolgt für verschiedene Schwellenwerte für den Trainings- und den Testdatensatz. Durch Interpolation kann nun die Größe der Fläche unterhalb der Kurven ermittelt werden, welches das Maß area under receiver operating curve (AUROC) liefert. Ein Problem des ROC Charts ist, dass in die Berechnung der falsch positiv Rate die Gesamtzahl der Objekte eingeht, die zur negativen Klasse gehören. Bei stark unbalancierten Datensätzen führt dies zu einer Überschätzung der Qualität des Klassifikators.

Im Precision Recall Diagramm werden die Maße Precision und Recall in Abhängigkeit eines Schwellenwertes in einem Diagramm aufgetragen. Da in diesem Maßen die Anzahl der Objekte mit negativer Klasse nicht enthalten ist, ist das Precision Recall Diagramm zur Beurteilung der Qualität einer Klassifikation von Datensätzen mit ungleicher Häufigkeitsverteilung des Targets besser geeignet [12, S. 1]. Auf die gleiche Weise wie beim ROC Chart kann der Flächeinhalt

der Fläche unterhalb der Precision Recall Kurve ermittelt werden. Die Größe dieser Fläche wird nun durch den Wert area under precision recall curve (AUPRC) (*area under precision recall curve*) angegeben.

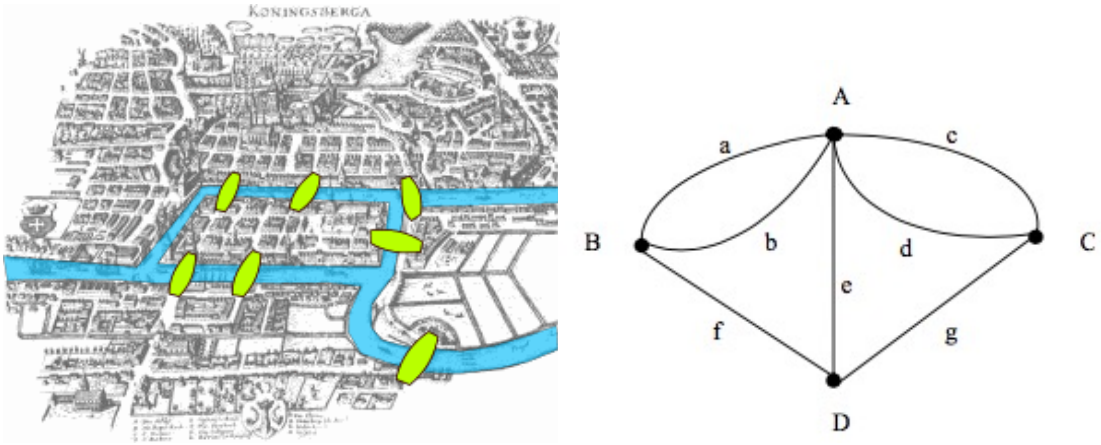
2.2. Kurzeinstieg in die topologische Datenanalyse

Die Topologie ist ein Teilbereich der Geometrie, welche sich mit den Eigenschaften mathematischer Objekte beschäftigt, deren Gestalt unter stetigen Verformungen unverändert ist. Das früheste Problem aus dem Bereich der Topologie ist das Königsberger Brückenproblem. Hierbei geht es darum einen Rundweg durch die russische Stadt Königsberg (heutiger Name *Kaliningrad*) zu finden, sodass jede der sieben Brücken jeweils genau einmal überquert wird und man schlussendlich am Startpunkt wieder ankommt . Auf der linken Seite der Abbildung 2.2 ist eine Skizze des Rätsels dargestellt, rechts daneben findet sich eine Darstellung als Graph, die Leonhard Euler 1736 verwendete, um zu beweisen, dass es keinen Weg gibt, der die geforderten Eigenschaften erfüllt. Dieses Rätsel gehört in den Bereich der Topologie, da es keine Rolle spielt, wo genau die Brücken liegen und wie lang diese sind. Die einzige relevante Information für das Rätsel ist die *Konnektivität*, also welche Teile der Stadt durch eine Brücke verbunden sind. Eine Topologie besitzt nach [10] folgende Eigenschaften:

- **Koordinateninvarianz**: Eine Verschiebung des Objektes in eine beliebige Richtung wirkt sich nicht auf die topologischen Eigenschaften des Objektes aus
- **Verformungsinvarianz**: Das Ziehen oder Zusammendrücken eines topologischen Objektes ändert nichts an dessen topologischen Eigenschaften
- **Komprimierte Darstellung**: Die Topologie liefert eine kompakte Repräsentation des topologischen Objektes.

Die Eigenschaft der Koordinateninvarianz des Königsberger Brücken Problems ergibt sich direkt aus der Tatsache, dass die genaue Lage der Brücken nicht relevant ist. Wie im rechten Teil der Abbildung 2.2 dargestellt, liefert der Graph eine komprimierte Darstellung des Sachverhaltes. Eine theoretisch mögliche Verformung der Brücken oder der Teile der Stadt wäre ebenfalls für die Problemstellung nicht relevant. Nun soll in Definition 1 eine formale Definition einer Topologie gegeben werden.

Das Ziel der topologischen Datenanalyse ist die Bereitstellung von Informationen über die Form des Datensatzes, welches dem Grundsatz der komprimierten Darstellung der Topologie entspricht. Für die Herstellung dieser komprimierten Darstellung des Datensatzes existieren auf dem Gebiet der topologischen Datenanalyse zwei unterschiedliche Ansätze. Diese sind die



(a) Skizze der Situation [51] (b) Darstellung als Graph [45, S. 318]

Abbildung 2.2.: Königsberger Brücken Problem

Untersuchung der persistenten Homologie des Datensatzes und die Generierung einer topologischen Zusammenfassung in Form des sogenannten Mapper Graphen. Die Einsatzgebiete der topologischen Datenanalyse in der Forschung sind vielfältig und reichen vom Finanzbereich über die Bilderkennung bis hin zur Erkennung einer besonderen Art von Brustkrebs [20],[24], [37].

Bevor die beiden Ansätze der topologischen Datenanalyse näher erläutert werden, sind zunächst folgende Definitionen hilfreich.

Definition 1 (Topologie). *Eine Topologie T auf der Menge X ist eine Untermenge $T \subseteq 2^X$ mit den Eigenschaften*

1. $i \in I, \forall i \in I,\ S_i \in T \Rightarrow \cap_{i \in I} \in T$
2. $\{S_j | j \in J\} \subseteq T \Rightarrow \cup_{j \in J} S_j \in T$
3. $\emptyset, X \in T$

Nach Definition 1 ist T eine Topologie der Menge X, wenn zum Einen die leere Menge und X in T enthalten sind, und dies auch für die Vereinigung und den Schnitt von endlich vielen Mengen aus T gilt. I in der Definition 1 ist eine endliche Indexmenge, J bezeichnet eine beliebige Indexmenge.

Definition 2 (Topologischer Raum). *Es sei X eine Menge und T eine Topologie gemäß Definition 1. Dann ist $\mathbb{X} = (X, T)$ ein topologischer Raum*

2.2.1. Persistente Homologie

$$\mathcal{X} = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d \tag{2.2.1}$$

Der erste Ansatz der topologischen Datenanalyse beschäftigt sich mit der sogenannten persistenten Homologie. Die Idee hierbei ist den Datensatz $\mathcal{X}$ als Punktwolke anzusehen und um jeden einzelnen Datenpunkt x_i eine Kugel mit Radius ε zu legen und Datenpunkte, bei denen sich die Kugeln überschneiden, durch Kanten oder Flächen zu verbinden [35]. Dadurch ergeben sich bei unterschiedlichen Radien verschiedene topologische Strukturen, die Simplizialkomplexe genannt werden. Eine formale Definition des Begriffes **Simplizialkomplex** liefert Definition 3.

Definition 3 (Simplizialkomplex). *Gegeben seien $k+1$ unabhängige Punkte der Menge $\mathcal{X} = \{x_0, \ldots, x_k\} \subset \mathbb{R}^d$ und sei der k-dimensionale Simplex $\sigma = [x_0, \ldots, x_k]$ die konvexe Hülle von $\mathcal{X}$. Die $x_i \in \mathcal{X}$ heißen Knoten (Vertices) von σ. Die durch die Untermengen von $\mathcal{X}$ aufgespannten Simplexe sind die Kanten (engl. faces) von σ. Ein Simplizialkomplex $K \subseteq \mathbb{R}^d$ ist eine endliche Vereinigung von Simplexen σ und besitzt die Eigenschaften:*

- *Jede Ecke eines Simplexes ist ein Simplex von K*
- *Der Schnitt von allen Paaren von Simplexen ist entweder leer oder eine gemeinsame Kante*

[47]

Die Generierung der Simplizialkomplexe erfolgt beispielsweise durch den Vietoris-Rips Komplex, der wie oben angedeutet um die Datenpunkte Kugeln mit dem Radius ε legt und die Datenpunkte verbindet, falls die Schnittmenge der Kugeln von benachbarten Punkten nicht leer ist. Genauer erfolgt die Verbindung der Punkte durch Kanten, wenn die Distanz zweier Punkte der Menge $\mathcal{X}' \subseteq \mathcal{X}$ kleiner als der Radius ε der Kugeln ist. Von diesen in Abhängigkeit des Radius ε entstehenden Simplizialkomplexen können die Bettizahlen β_0, β_1 und β_2 bestimmt, welche die topologische Struktur des Datensatzes beschreiben [35]. Die Anzahl der zusammenhängenden Simplizialkomplexe (engl. *connected components*) entspricht der Bettizahl β_0, die Bettizahl β_1 quantifiziert die Anzahl der eindimensionalen Löcher (engl. *loops*. β_2 wird definiert durch die Anzahl der zweidimensionalen Hohlräume (engl. *voids*). Eine Beispiel für die Bestimmung der Bettizahlen einiger Objekte findet sich in Abbildung 2.3. Die Radien ε_b und ε_d für jede topologische Struktur, bei der oben beschriebene i dimensionale Löcher erscheinen ε_b und verschwinden ε_d werden im Persistance Diagramm als Punkt graphisch dargestellt [39, S. 11]. Diese topologische Strukturen werden auch als *topologische Features* oder

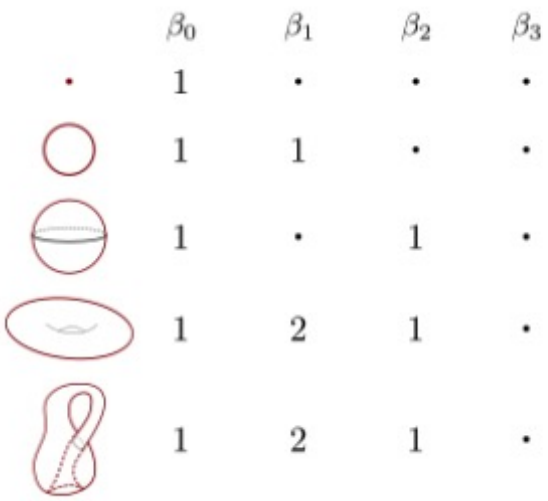


Abbildung 2.3.: Darstellung der Bettizahlen $\beta_0, \beta_1, \beta_2$ und β_3 eines Punkte, eines Kreises, einer Kugel, eines Torus und einer kleinschen Flasche [35, S. 6]

topologische Signaturen bezeichnet [24, S. 1]. Zu jeder der oben beschriebenen Dimension kann ein **Persistence Diagramm** erstellt werden, welches die topologischen Signaturen der Dimension i des Datensatzes visualisiert [24, S. 1]. Eine weitere Darstellungsmöglichkeit des Paares $(\varepsilon_b, \varepsilon_d)$ bietet der Barcode-Plot, der ε_b und ε_d in einem Diagramm darstellt, welches an einen Barcode erinnert [39, S. 11]. Ein Beispiel der persistenten Homologie findet sich in Abbildung 2.4. Im oberen Teil der Abbildung sind die Simplizialkomplexe verschiedener Radien dargestellt. Unten links befindet sich das Persistence Diagramm, welches die Persistenz der topologischen Features bei der Betrachtung der zweidimensionalen Löcher zeigt. Rechts daneben befindet sich eine grafische Darstellung des aus dem abgebildeten Persistence Diagramm ermittelte Persistence Landscapes.

Die aus einem der beiden Diagrammen gewonnenen Informationen können nun über die in [20] beschriebenen Persistence Landscapes oder die von [1] vorgestellten Persistence Images in Vektorform transformiert werden, damit die topologischen Informationen des Datensatzes als Input zum Training eines Klassifikators genutzt werden können.

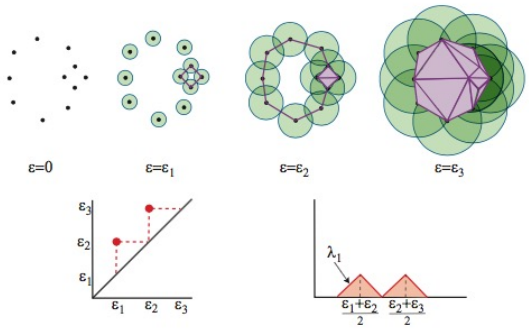


Abbildung 2.4.: Beispiel der persistenten Homologie bei der Betrachtung der zweidimensionalen Löcher β_1 [20, S. 6]

2.2.2. Mapper Graph

Die zweite, in der Literatur etwas weniger verbreitete, Möglichkeit der topologischen Datenanalyse ist die Generierung des sogenannten Mapper Graph, der eine graphische Repräsentation des Datensatzes liefert. Dieser Teil gibt nun einen Überblick über die Erstellung des Mapper Graphs. Eine genauere Beschreibung der Erstellung des Mapper Graphen liefert [46].

Definition 4 (Linse). *Eine stetige Funktion $f : D \to Z$, die von einem metrischen Raum $\mathcal{X}$ in einen metrischen Raum Z mit geringerer Dimension abbildet heißt **Linse** oder **Filter**.*

In Abbildung 2.5 findet sich eine schematische Darstellung des Prozesses zur Generierung eines Mapper Graphen. Ausgangspunkt für die Generierung des Mapper Graphs ist ein Datensatz $X \subset \mathbb{R}^d$. Vor Anwendung der Linse wird in einer Vielzahl der Publikationen durch die Wahl einer Metrik eine Distanzmatrix erstellt. Die Auswahl der genutzten Metrik hängt von der Anwendung ab [11, S. 1]. Der erste Schritt ist die Anwendung einer Linse auf die Distanzmatrix als Filterfunktion. Der Begriff einer Linse ist in der Definition 4 definiert. Die verwendete Linse kann der Benutzer frei auswählen, allerdings muss f stetig sein [46, S. 3]. Jede Linse liefert eine unterschiedliche Zusammenfassung der Form der Daten. Häufig eingesetzte Linsen sind beispielsweise Dichteschätzer oder Zentralitätsmaße des Datensatzes. Die Verwendung von Outputs einer Hauptkomponentenanalyse oder eines Dimensionsreduktionsalgorithmus als Linse ist ebenfalls möglich. Die im Rahmen dieser Masterthesis genutzten Linsen werden im Teilkapitel 2.3 vorgestellt.

Nun wird eine Überdeckung $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ des Raumes Z bestimmt, wobei A eine endliche Indexmenge ist. Dies entspricht dem Schritt A aus der Abbildung 2.5. Zur Generierung dieser Überdeckung werden in jeder Dimension des Raumes Z überlappende Intervalle ver-

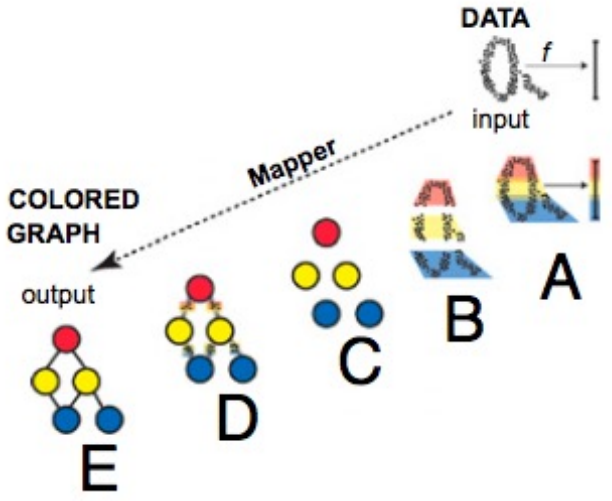


Abbildung 2.5.: Visualisierung des Ablaufes zur Erstellung eines Mapper Graphs [37, S. 2]

wendet. Hierzu ist die Vorgabe der Anzahl der gleich langen Intervalle n pro Dimension und die Stärke der Überlappung u notwendig [11, S. 8]. Eine Methode zur Bestimmung optimaler Werte für die Überdeckung der Linse existiert nicht. Wegen der Stetigkeit der Funktion f sind die Mengen $f^{-1}(U_\alpha), \forall U_\alpha \in \mathcal{U}$ ebenfalls Überdeckungen des Raumes $\mathcal{X}$. Die Bestimmung von $f^{-1}(U_\alpha), \forall U_\alpha \in \mathcal{U}$ erfolgt als Schritt B der Abbildung 2.5 durch ein Clustering der Datensätze aus $\mathcal{X}$, welche in der Überdeckung U_α liegen. Der hierbei eingesetzte Clusteringalgorithmus ist durch den Benutzer frei wählbar [11, S. 9]. Dadurch entstehen die Cluster C_α, welche die Datensätze enthalten deren Linsenergebnis in der jeweiligen Überdeckung U_α liegt.

Der letzte Schritt ist Konstruktion eines Graphen aus diesen Ergebnissen. Im Schritt C der Abbildung 2.5 stellt die Cluster $\mathcal{C} = \{C_\alpha\}, \alpha \in A$ jeweils als Nodes des Graphen dar. Eine Verbindung zwischen zwei Nodes durch eine Kante wird hergestellt, wenn die Bedingung $f^{-1}(U_\alpha) \cap f^{-1}(U_{\alpha'}) \neq \emptyset$ gilt, wobei $\alpha \neq \alpha'$ gelten muss [11, S. 8]. Die dargestellte Bedingung bedeutet, dass die Nodes verbunden werden, wenn die Cluster C_α und $C_{\alpha'}$ gemeinsame Datensätze enthalten. Die Einfärbung der Nodes des Graphen kann beispielsweise nach dem Durchschnittswert der Funktion f oder auf Basis der häufigsten Klasse in diesem Node erfolgen. Die Erzeugung des Graphs erfolgt in der Abbildung 2.5 durch die Schritte D und E [37].

Der erstellte Graph bietet nun eine Zusammenfassung der Form des Datensatzes durch die topologische Datenanalyse. Die Informationen dieser Zusammenfassung können nun von weiteren Algorithmen genutzt werden, um Informationen über die Form des Datensatzes beispielsweise bei einer Klassifikation zu nutzen.

2.3. Kurzvorstellung der Linsen

Wie in Teil 2.2.2 beschrieben, erfordert die Generierung eines Mapper Graphens die Vorgabe einer sogenannten Linse. Jede stetige Funktion kann als Linse verwendet werden, daher kommen beispielsweise Algorithmen zur Dimensionsreduktion als Linsen in Frage. Bei der Durchführung der Experimente zur Beantwortung der Frage, ob durch die topologische Datenanalyse eine Verbesserung von Klassifikatoren möglich ist, wird als Linse die lineare Diskriminanzanalyse, eine Kombination aus einem Isolation Forest und der L_2-Norm der Features, sowie die Dimensionsreduktionsverfahren TSNE und UMAP als Linsen für die Erstellung des Mapper Graphen verwendet. Die Eigenschaften und Funktionsweise dieser Linsen wird in den folgenden Abschnitten vorgestellt.

2.3.1. Lineare Diskriminanzanalyse

Die lineare Diskriminanzanalyse wird hier als überwachtes Dimensionsreduktionsverfahren genutzt, das heißt das Target, geht in die Dimensionsreduktion mit ein. Das Ziel dieses Verfahrens ist es, durch eine Transformation in einen eindimensionalen Raum Objekte unterschiedlicher Klassen mit größtmöglichem Abstand zu separieren.

Unter der Annahme, das die Objekte aller Klassen eine gemeinsame Kovarianzmatrix Σ aufweisen, lassen sich die Daten wie in Gleichung 2.3.1 angegeben skalieren. Durch diese Skalierung entspricht die Kovarianzmatrix der Daten X^* der Einheitsmatrix. Die Klassifikation der Objekte erfolgt nun über die Bestimmung des nächstgelegenen Klassenzentroids auf den transformierten Daten. Durch diese lineare Projektion verringert sich die Dimension auf $d-1$, wobei d für die Anzahl der Klassen steht. Somit führt die lineare Diskriminanzanalyse implizit eine Dimensionsreduktion auf $d-1$ durch [23, S. 113]. Die Dimension der linear projizierten Daten muss immer um eins geringer als die Anzahl der Ausprägungen des Klassenzugehörigkeitsattribut. Demnach entsprechen die projizierten Daten bei einem binären Target einem Vektor der Länge N, wobei N die Anzahl der Objekte im Datensatz kennzeichnet.

$$X^* = D^{-\frac{1}{2}} U^T X \quad \text{mit } \Sigma = UDU^T \tag{2.3.1}$$

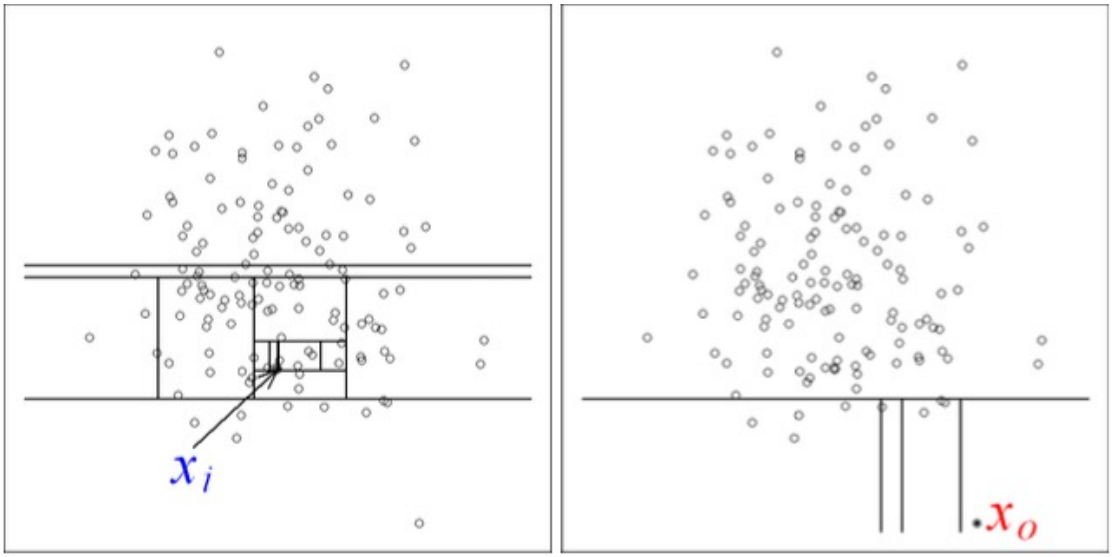
Die sich auf Basis dieser Berechnungen ergebenden Unterräume führen zu einer Reduktion der Dimensionalität. Es ist möglich, durch Wiederholung dieser Überlegungen die Dimension des Unterraumes auf L zu senken, und die Klassifizierung der Daten durch die Zuordnung zu L Klassenzentroide durchzuführen [23, S. 117]. So ergibt sich aus der Anwendung der linearen Diskriminanzanalyse eine Reduktion der Dimension der Daten.

2.3.2. Isolation Forest & L_2 Norm

Eine weitere, unter `https://github.com/MLWave/kepler-mapper/issues/35` im Forum zur Implementierung des KeplerMappers diskutierte, für die Erkennung von Betrug geeignete Linse ist eine Kombination aus dem Abnormalitätsscore eines Isolation Forests und der L2-Norm der Features im Datensatz.

Ein Isolation Forest ist ein Verfahren zur Erkennung von Ausreißern. Der Isolation Forest basiert auf Entscheidungsbäumen und ist konzeptionell dem Random Forest ähnlich. Isolierung bedeutet hier die Trennung eines Punktes von allen anderen Punkten des Datensatzes. Auf einer zufälligen Auswahl von n Objekten des Gesamtdatensatzes wird ein Baum generiert [28]. Dies erfolgt durch die zufällige Bestimmung eines Splitattributs und eines Splitwertes zwischen dem Minimum und dem Maximum des zufällig gewählten Features [28]. Der Aufbau des Baumes erfolgt bis zu einer vorgegebenen maximalen Tiefe erreicht ist. Die maximale Tiefe entspricht in der verwendeten Implementierung $\lceil \log_2(n) \rceil$, wobei n der Anzahl der Beobachtungen des Trainingsdatensatzes entspricht [41]. Der beschriebene Prozess wird wiederholt, sodass sich mehrere Bäume ergeben, welche die Entscheidung im Verbund treffen, ob es sich um einen Ausreißer handelt [28]. Nun wird für jeden Datensatz die Pfadlänge in jedem Baum von der Wurzel bis zum Knoten bestimmt. Dieser Wert, gemittelt über alle generierten Bäume gibt einen Hinweis darauf, ob der jeweilige Datensatz ein Ausreißer ist. Je kleiner dieser Wert ist, desto eher handelt es sich bei der jeweiligen Beobachtung um einen Ausreißer, da abnorme Beobachtungen früher isoliert werden [28]. Dies ist in Abbildung 2.6 exemplarisch dargestellt. Weitergehende Details zum Isolation Forest liefert [28]. Das beschriebene Prozedere wird für alle Objekte im Datensatz wiederholt und bildet so die erste Komponente der Linse. Ein Vorteil des Isolation Forest ist, dass er in der Lage ist durch einen globalen Ansatz ohne die Verwendung eines Distanzmaßes Ausreißer in Datensätzen zu erkennen.

Die zweite Komponente der Linse ist die L_2 Norm jedes Objektes $\vec{t}^{(i)}$ im Trainingsdatensatz mit N Objekten. Die Dimension des Featureraumes entspricht p. Die Berechnung der Norm



(a) Isolating x_i (b) Isolating x_o

Abbildung 2.6.: Beispiel der Isolierung von x_0 und x_i durch den Isolation Forest. [28]

erfolgt nach Gleichung 2.3.2.

$$\|\vec{t}^{(i)}\|_2 = \sum_{j=1}^{p} (\vec{t}_j^{(i)})^2, \vec{t}^{(i)} \in \mathbb{R}^p, \quad \text{für } \forall i \in 1, \ldots, N \tag{2.3.2}$$

Durch die Verbindung der beiden Komponenten entsteht eine zweidimensionale Linse, deren erste Komponente aus einem Isolation Forest und deren zweite Komponente aus der L_2 Norm des Vektors aller Features des Datensatzes besteht .

2.3.3. TSNE

Als Dimensionsreduktionsverfahren transformiert TSNE Daten in einem höherdimensionalen Raum $\mathcal{X} = \{x_1, \ldots, x_n\}, x_i \in \mathbb{R}^p \forall i$ in einen Raum $\mathcal{Y} = \{y_1, \ldots, y_n\}, y_i \in \mathbb{R}^d, d < p$ mit einer niedrigeren Dimension, sodass die lokale Struktur der Daten erhalten bleibt [30]. Häufig besitzt der Raum $\mathcal{Y}$ zur Darstellung der dimensionsreduzierten Punkte in einem Scatterplot die Dimension 2, weswegen die verwendete `MulticoreTSNE` Implementierung nur Transformationen in einen zweidimensionalen Raum erlaubt. Die Dimensionsreduktion durch TSNE kann nur in einen zwei oder drei dimensionalen Raum erfolgen [30, S. 2598].

Die Idee der TSNE ist die Optimierung der dimensionsreduzierten Daten durch das Gradientenabstiegsverfahren unter Minimierung der Kullback-Leibler Divergenz aus Gleichung 2.3.6

[30]. Die Werte im zweidimensionalen Raum, die später der Dimensionsreduktion entsprechen, werden zu Beginn des Verfahrens zufällig erzeugt und im Verlaufe des Verfahrens optimiert.

Zu Beginn des Verfahrens wird durch Gleichung 2.3.3 die paarweise Ähnlichkeit der Ausgangsdaten berechnet [30]. Die $p_{j|i}$ geben die Wahrscheinlichkeit an, dass der Punkt x_i den Punkt x_j als Nachbarn auswählt, wenn hierzu die um x_i zentrierte Wahrscheinlichkeitsverteilung $P \sim \mathcal{N}(0, \sigma^2)$ Entscheidungskriterium ist. Die Zahl n in Gleichung 2.3.3 entspricht der Anzahl der Objekte im Datensatz. Die Berechnung der p_{ij} erfolgt auf die in Gleichung 2.3.4 dargestellte Weise zur schnelleren Berechnung des Gradienten der Kostenfunktion. Der wichtigste Parameter der TSNE Dimensionsreduktion ist die Perplexität und spielt eine Rolle bei der Bestimmung der Varianz der Normalverteilung P durch binäre Suche [30]. Bei einer hohen Perplexität ist die Lösung eher global, da hier viele Nachbarn der Punkte bei der Berechnung der Dimensionsreduktion in Betracht gezogen werden. Kleinere Werte für Perplexität liefern eine lokalere Lösung, da überwiegend naheliegende Punkte die Dimensionsreduktion beeinflussen. [30]

$$p_{j|i} = \frac{e^{-\|x_i - x_j\|^2 / 2\sigma_i^2}}{\sum_{k \neq l} e^{-\|x_k - x_l\|^2 / 2\sigma_i^2}} \tag{2.3.3}$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{2.3.4}$$

Für den Output der Dimensionsreduktion wird ebenfalls aus den zufällig bestimmten Werten das paarweise Ähnlichkeitsmaß aus Gleichung 2.3.5 bestimmt, allerdings wird hier als Wahrscheinlichkeitsverteilung eine t-Verteilung mit einem Freiheitsgrad verwendet.

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_i - y_j\|^2)^{-1}} \tag{2.3.5}$$

$$C = KL(\mathcal{P} \| \mathcal{Q}) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{2.3.6}$$

Durch Optimierung der Kostenfunktion 2.3.6 mithilfe des Gradientenverfahrens werden die dimensionsreduzierten Werte bestimmt. Detailliertere Informationen zur Funktionsweise des Dimensionsreduktionsalgorithmus finden sich in [30].

2.3.4. UMAP

UMAP ist ein weiteres Verfahren zur Dimensionsreduktion auf Basis eines topologischen Abbildes des hochdimensionalen Datensatzes[33]. Die Funktionsweise von UMAP soll hier in den

Grundzügen erklärt werden. Für die Reduktion der Dimension durch UMAP sind drei Schritte notwendig. Zunächst wird jeder Datenpunkt des Datensatzes in einen diskreten metrischen Raum übersetzt. Die Herstellung einer globalen Struktur aller Datenpunkte erfolgt durch eine approximierte topologische Repräsentation dieser metrischen Räume durch eine Approximation durch simpliziale Mengen (vgl. [33]). Durch die Vereinigung dieser simplizialen Mengen aller Datenpunkte entsteht eine globale topologische Form der Daten [33]. Für die dimensionsreduzierten Daten wird ebenfalls eine topologische Darstellung ermittelt. Im Gegensatz zu den Ausgangsdaten ist die Mannigfaltigkeit $\mathbb{R}^d$ der dimensionsreduzierten Daten bekannt. Im Gegensatz zu TSNE ist UMAP in der Lage, die Ausgangsdaten auf eine beliebige Dimension d zu reduzieren [33].

Wie bei dem Dimensionsreduktionsalgorithmus TSNE erfolgt die Berechnung der Dimensionsreduktion durch die Minimierung einer Kostenfunktion. Eine genauere Beschreibung der Funktionsweise von UMAP liefert [33].

Wichtige Parameter der UMAP Dimensionsreduktion sind die Anzahl der Nachbarn *n_neighbors*, die minimale Distanz zwischen Punkten in der niedrigdimensionalen Repräsentation der Daten *min_dist*, die Dimension des Raumes, in den die Daten projiziert werden und die verwendete Metrik (vgl. [32]). Da sich der Output des Dimensionsreduktionsverfahrens in einem 2D Scatterplot darstellen lassen soll, besitzt der Raum, in den die Daten projiziert werden sollen die Dimension 2. Analog zur Perplexität bei TSNE steuert der Parameter *n_neighbors* von UMAP, ob eher die globale oder die lokale Struktur der Daten wiedergegeben werden soll [32]. Durch den Parameter *min_dist* wird definiert, wie dicht die dimensionsreduzierten Punkte beieinander liegen sollen [32].

2.4. Kurzeinführung in die Graphentheorie

Zur Generierung des neuen Features erfolgt eine Analyse des Mapper Graphs durch Methoden der Graphanalyse. Daher soll in diesem Teil eine kurze Definition der relevanten Begriffe aus dem Bereich der Analyse von Graphen gegeben werden. Eine Definition eines Graphs liefert die Definition 5 (vgl. [45, S. 319]). Die Menge E enthält die Kanten des Graphs, welche jeweils zwischen zwei Knoten aus der Menge V liegen. Auf einem Graphen lassen sich nun Wege von einem Knoten A zu einem anderen Knoten B definieren.

Definition 5. ***Graph*** *Ein Graph G ist das Paar (V, E) aus den Menge der Knoten V und der Menge der Kanten E. Es müssen die folgenden Voraussetzungen erfüllt sein:*

- $V \neq \emptyset$

- $V \cap E = \emptyset$

Es existieren gerichtete und ungerichtete Graphen. Die Kanten der gerichteten Graphen besitzen eine Richtung, die der ungerichteten Graphen nicht. Der in dieser Thesis thematisierte Mapper Graph gehört zu den ungerichteten Graphen. Ein wichtiges Mittel der Analyse von Graphen ist die Bestimmung der Zusammenhangskomponenten eines Graphens.

Hierzu definiert man zunächst in Definition 6 den Begriff der Zusammenhangskomponente. Unter diesem Begriff versteht man nach der Definition 6 einen Teilgraphen des Graphens G, bei dem die Knoten untereinander verbunden sind und es keine Kanten zwischen Knoten aus dem Teilgraphen und dem restlichen Teil des Graphen gibt. Die Bestimmung der Zusammenhangskomponenten des Mapper Graphs ist für die Bestimmung des topologischen Features notwendig.

Definition 6. ***Zusammenhangskomponente*** *Es sei $G = (V, E)$ ein Graph gemäß Definition 5. Der Teilgraph Z heißt Zusammenhangskomponente des Graphen G, falls*

- *alle Knoten des Teilgraphen Z durch Wege erreichbar sind.*
- *es keinen Weg zwischen den Knoten des Teilgraphen Z und den Knoten des Graphs G gibt.*

Mit den beiden Definitionen und Erläuterungen sind die notwendigen Grundlagen der Analyse von Graphen gelegt, die in dieser Masterthesis verwendet werden.

3. Aktueller Forschungsstand

Dieses Kapitel soll einen Überblick über den aktuellen Forschungsstand über die Anwendung der topologischen Datenanalyse in Verbindung mit Verfahren zur Klassifikation geben. Wie im Kapitel 2 beschrieben, erfolgt die topologische Datenanalyse entweder über die Analyse der persistenten Homologie oder über die Erstellung einer Zusammenfassung des Datensatzes in Form eines Graphen. Die Recherche zeigt, dass überwiegend der Weg über die persistente Homologie eingeschlagen wird, um Verfahren aus dem Gebiet des Machine Learning mit der topologischen Datenanalyse zur Verbesserung der Performance von Klassifikatoren zu kombinieren. Nur wenige Publikationen thematisieren die Anwendung von Verfahren aus dem Gebiet des Machine Learning zusammen mit der Nutzung von Informationen aus dem Mapper Graphen mit dem Ziel eine Verbesserung der Performance zu erhalten.

Die Anwendung des Mapper Graphens als alternative Herangehensweise an die topologische Datenanalyse erfolgt vorrangig zur **Feature Selection** oder zur Exploration der Daten. Anhand der Ergebnisse aus diesen Untersuchungen können jedoch wertvolle Informationen gewonnen werden, welche bei der Verbesserung von Modellen hilfreich sind. Durch ein besseres Datenverständnis können Modelle implementiert werden, welche besser an die Daten angepasst sind.

Wegen der zwei unterschiedlichen Ansätze der Durchführung der topologischen Datenanalyse zur Bereitstellung von topologischen Informationen über den Datensatz folgt eine Beschreibung der aktuellen Forschung getrennt nach dem Ansatz der topologischen Datenanalyse. Die Domänen der Anwendung der persistenten Homologie sind vielfältig und umfassen beispielsweise den Finanzbereich [20], die Klassifikation von Bildern [24], [14], [22] oder die Klassifikation von Fingerabdrücken [19]. Doch auch zur Klimaforschung, der Modellierung von starken Schwingungen von Fertigungsmaschinen oder der Klassifikation von Bewegungsmustern von Menschen wie beispielsweise für Fitnesstracker wird die topologische Datenanalyse in Form der persistenten Homologie als Hilfsmittel eingesetzt ([36], [25], [16]).

Die topologische Datenanalyse in Form der Erstellung des Mapper Graphs erfolgt beispielsweise bei der Analyse von Inhalten in sozialen Netzwerken, bei der prädiktiven Analyse von Produktionsprozessen der chemischen Industrie [21] oder im medizinischen Bereich zur Diagnose von Lungenembolien [44]. In allen dieser wissenschaftlichen Publikationen wird das Klassifikationsverfahren durch Informationen aus dem Mapper Graph unterstützt. Allerdings erfolgt dies nicht durch die Umwandlung von Informationen aus der topologischen Datenanalyse in eine Vektordarstellung, wie im Falle der persistenten Homologie. Eine Möglichkeit der Ableitung von Informationen aus dem Mapper Graph ist die Identifizierung von Subgruppen im Graphen und die Anwendung von statistischen Verfahren, welche die Features identifizieren, welche die Zugehörigkeit der Objekte zu ihrer Subgruppe möglichst gut erklären. Diese Features werden schlussendlich zur Klassifikation herangezogen.

3.1. Forschungsstand auf Basis der persistenten Homologie

Eine Anwendung der persistenten Homologie zur Bildung eines Frühwarnsystems vor Finanzkrisen bietet das Paper von Gidea und Katz mit dem Titel „Topological Data Analysis of Financial Time Series: Landscapes of Crashes“ [20]. In diesem Paper stellen die Autoren eine auf die topologische Datenanalyse basierende Lösung zur Erkennung von Warnsignalen vor Finanzkrisen vor. Die Funktionsweise und die Ergebnisse werden zum Einen auf einem simulierten Datensatz und zum Anderen auf realen Daten vorgeführt.

Die vorgestellte Methode zur Früherkennung von Finanzkrisen begründet sich auf die persistente Homologie. Auf Basis des Persistenzdiagrammes zum Entstehen und Verschwinden der zweidimensionalen Löcher der Simplizialkomplexe, werden sogenannte Persistence Landscapes berechnet. Die Persistence Landscapes können nun wegen ihrer mathematischen Eigenschaften als Input für statistische Verfahren dienen und wurden durch Bubenik im Paper mit dem Titel „Statistical Topological Data Analysis using Persistence Landscapes“ entwickelt [9]. Im seinem Paper weist Bubenik die Stabilität gegenüber dem Rauschen der Daten dieser Repräsentation der Informationen aus der persistenten Homologie nach.

Ausgehend vom Persistence Diagramm mit den Tupeln $\{(b_i, d_i)\}$, $\forall i \in 1, \ldots, m$ werden Persistence Landscapes durch die Gleichung 3.1.1 definiert. In der Gleichung 3.1.1 steht b_i für den Radius, bei dem die i. topologische Struktur auftaucht und d_i quantifiziert den Radius bei dem die topologische Struktur verschwindet. Als Funktion $f_{(b_i,d_i)}$ verwenden die Autoren

die Funktion aus Gleichung 3.1.2 [20, S. 7].

$$\lambda_k(t) = k - \max(f_{(b_i,d_i)})(x) \tag{3.1.1}$$

$$f_{(b_i,d_i)}(x) = \begin{cases} x - b_i, & \text{wenn } x \in (b_i, \frac{b_i+d_i}{2}] \\ -x + d_i, & \text{wenn } x \in (\frac{b_i+d_i}{2}, d_i) \\ 0, & \text{sonst} \end{cases} \tag{3.1.2}$$

Im Anschluss werden die L_p Normen der Persistence Landscapes berechnet, welche einen Hinweis auf eine finanzielle Instabliltät geben sollen, indem diese im Vorlauf einer Krise ansteigen. Die Anwendung des von Gidea et. al. vorgestellten Prozesses erfolgt jeweils auf einem Sliding Window mit der Breite von einem Tag. Als Anwendung der Methode auf reale Daten wählen die Autoren die täglichen Renditen der US-amerikanischen Aktienindexe S&P 500, DJIA, NASDAQ und Russel 2000 auf einem Sliding Window mit der Länge von 50 oder 100 Tagen und einer Überlappung von einem Tag [20, S. 16].

Es zeigt sich bei den Untersuchungen der Autoren, dass sowohl L_1 als auch L_2 Norm der Persistence Landscapes bereits 250 Tage vor der Dotcom Blase am 10.03.2000 und der Bankrotterklärung von Lehman am 15.09.2008 stark ansteigen [20, S.22]. Der Mann-Kendall Test, welcher zur Erkennung von Trends bei Zeitreihen dient, weist einen statistisch signifikanten Anstieg der Normen der Persistence Landscapes aus [20, S. 22]. Zusammengefasst haben die Autoren in einem Paper eine neue ökonometrische Methode der Erkennung von Frühwarnsignalen von Instablilitäten des Finanzsystems vorgestellt, welche auf die topologische Datenanalyse zurückgreift und zumindest auf den verwendeten Daten zuverlässig funktioniert [20, S. 24].

Hofer et. al. stellen in ihrem Paper „Deep Learning with Topological Signatures“ dar, wie die Informationen aus der persistenten Homologie zur Verbesserung der Bildklassifikation genutzt werden können [24]. Der Input Layer des neuronalen Netzes erhält als Eingabe ein Persistence Diagramm, welches durch drei Schritte in eine Darstellung übersetzt wird, welche zum Trainieren eines neuronalen Netzes geeignet ist [24, S. 2].

Hierzu erfolgt werden die Punkte des Persistence Diagrammes zunächst um 90 Grad gedreht [24, S. 2]. Die gedrehten Punkte werden nun mithilfe von Exponentialfunktionen mit den Parametern μ und σ transformiert und durch Bildung einer Summe auf eine Dimension projiziert. Für diesen Schritt ist, wie in der wissenschaftlichen Ausarbeitung von den Autoren

genannt, die Verwendung anderer Funktionen möglich [24, S. 4]. Die vorgestellte Methode wird auf Datensätze im Bereich der Objekterkennung und der Analyse von Graphen aus sozialen Netzwerken angewandt. Obwohl der von Hofer et. al. vorgestellte Ansatz keine Extraktion der Konturen der Objekte der zum Training beziehungsweise Test verwendeten Bilder erfordert, sind die Ergebnisse mit Methoden anderer Autoren vergleichbar.

Insgesamt zeigen die Experimente von Hofer et. al., dass die Nutzung von Informationen aus der topologischen Datenanalyse zu einer Verbesserung der Performance der Machine Learning Methoden führen kann (vgl. [24, S. 8] .

Dey et. al. verwenden in ihrem Paper „Improved Image Classification using Topological Persistence“ zur Anhebung der Performance von Klassifikatoren zur Klassifikation von Bildern ebenfalls das Persistence Diagramm zur Generierung eines neuen topologischen Features [14]. Diese neuen topologischen Features entstehen durch die Subtraktion der Radien $\varepsilon_d - \varepsilon_b$, welches die Persistenz der jeweiligen topologischen Struktur misst. Hierbei entspricht ε_b dem Radius, bei dem eine topologische Struktur erscheint und ε_d dem Radius bei dem diese Struktur verschwindet.

Der Grund für die Entscheidung der Autoren zur Vektorisierung der Persistence Diagramme nicht die etablierten Verfahren wie Persistence Landscapes zu nutzen, besteht in der Einfachheit und einer höheren Geschwindigkeit des angewendeten Verfahrens im Vergleich zu den Alternativen (vgl. [14]). Das neue topologische Feature wird als zusätzliches Feature zur Klassifikation der Bilder verwendet. Die Anwendung der vorgestellten Methode zur Nutzung topologischer Informationen aus der persistenten Homologie erfolgt auf den Datensätzen CIFAR-10, Caltech-256 und MNIST. In den Datensätzen CIFAR-10 und Caltech-256 befinden sich Bilder mit zugehöriger Bezeichnung. Der MNIST Datensatz enthält Abbildungen von handgeschriebenen Ziffern. Durch die zusätzliche Verwendung des topologischen Features konnte eine Verbesserung der Precision und Recall Maße beobachtet werden. Auch die Genauigkeit ist durch die Unterstützung durch die topologische Datenanalyse leicht angestiegen (vgl. [14]).

Eine Anwendung der persistenten Homologie aus dem Bereich der Klimaforschung liefern Muszynski et. al. [36]. Die Autoren stellen eine alternative Methode der Erkennung von sogenannten atmosphärischen Flüssen vor, welche die persistente Homologie verwendet.

Atmosphärische Flüsse sind Bänder mit sehr feuchter Luft in der Atmosphäre und können ursächlich für extreme Niederschläge sein [36, S. 2]. Das Ziel des Papers ist die Klassifikation atmosphärischer Flüsse durch eine Support Vektor Machine unter Zuhilfenahme der topologischen Datenanalyse. Bislang erfolgt die Identifikation eines atmosphärischen Flusses durch die Durchführung von Messungen, welche mit einem zu definierenden Schwellenwert verglichen werden [36, S. 4]. Durch die von den Autoren präsentierte Methode zur Klassifikation der atmosphärischen Flüsse entfällt die Notwendigkeit der Definition eines Schwellenwertes zur Klassifikation der atmosphärischen Flüsse.

Als Datenbestand verwenden die Autoren die Daten des Klimamodells Community Atmosphere Model in der Version 5.1 (CAM5.1) und des Modells der Modern-Era Retrospective Analysis for Reasearch & Applications (MERRA-2). Das Klimamodell CAM5.1 besteht aus täglichen und 3-stündlichen Messungen in den Auflösungen 25 Kilometern, 100 Kilometern und 200 Kilometern im Zeitraum von Januar 1979 bis Dezember 2005 [36, S. 4]. MERRA-2 enthält 3-stündliche Messungen mit der Auflösung von 50 Kilometern im Zeitraum vom Januar 1980 bis zum Juni 2017 [36, S. 4]. Die für die Analyse verwendete Messgröße ist die Gesamtmenge des Wasserdampfs (engl. Integrated Water Vapor, IWV).

Die Anwendung der topologischen Datenanalyse erfolgt durch die Extraktion von zusammenhängenden Regionen anhand der Snapshots der Eingabedaten (vgl. [36, S. 6]). Die Bildung der zusammenhängenden Regionen erfolgt in ähnlicher Weise über die Variation des Radius der Überdeckung mit dem Unterschied, dass die Autoren hier die Größe IWV variieren. Diese zusammenhängenden Regionen sind topologische Deskriptoren, welche Wettereffekte wie atmosphärische Flüsse charakterisieren (vgl. [36, S. 6]). Je Zeitpunkt ergibt sich durch Transformation eine Darstellung dieser zusammenhängenden Komponenten als Vektoren der Länge k. Bei der von den Autoren vorgestellten Anwendung hat dieser Vektor 60 Komponenten.

Zusammen mit dem Label, ob es sich bei der Messung um einen atmosphärischen Fluss handelt, werden diese Vektoren genutzt, um mithilfe einer Support Vektor Maschine die Klassifikation durchzuführen. Insgesamt funktioniert die vorgestellte Methode der Erkennung der atmosphärischen Flüsse auf dem Datensatz mit einer Genauigkeit zwischen 77% und 91% [36, S. 15].

Die Autoren Dirafzoon, Lokare und Lobaton stellen in ihrem Paper „Action Classification

From Motion Capture Data Using Topological Data Analysis“ die Anwendung der topologischen Datenanalyse zur Klassifikation von menschlichen Aktionen wie Sitzen, Stehen, Laufen und Radfahren vor [16]. Dieser Anwendungsfall ist besonders bei Fitnesstrackern interessant, welche die Bewegungen ihrer Träger aufzeichnen und zur Analyse bereitstellen. In der im Paper vorgestellten Situation erfolgt die Registrierung der Bewegungen der Testpersonen anhand von verschiedenen Sensoren, welche an den Gelenken der Personen positioniert sind.

Zur Entfernung von Rauschen werden die n dimensionalen Messungen $\{s_{1j}(t), \ldots, s_{nj}(t)\}$ von jedem Gelenk j des Körpers an dem die Bewegung gemessen wird, eine Filterung bezüglich des Medians der Messungen unterzogen. Des Weiteren wird zur Reduktion der Dimension eine Hauptkomponentenanalyse durchgeführt. Die zur Klassifikation herangezogenen Features ergeben sich aus der persistenten Homologie auf die nun beschriebene Weise. Aus den Paaren (b_i,d_i), welche aus den Persistence Diagrammen entnommen werden können, wird $l_m = max_{(b_i,d_i)\in PD_1}|d_i - b_i|$ berechnet [16, S. 1262]. Diese Werte für alle vermessenen Gelenke und Hauptkomponenten des Datensatzes, welche mindestens p Prozent der Varianz erklären ergeben den Vektor der Features, welche der Klassifikation zugeführt werden. Die Gestalt des Featurevektors F_w ist in Gleichung 3.1.3 wiedergegeben [16, S. 1262].

$$F_w = l_m^{1,1}, \ldots, l_m^{1,n_p}, \ldots, l_m^{n_j,1}, \ldots, l_m^{n_j,n_p} \tag{3.1.3}$$

Als Klassifikator verwenden die Autoren den nächste Nachbarn Klassifikator. Die Messung des Abstandes der Punkte erfolgt durch die euklidische Metrik, die Entscheidung über die Klassifikation erfolgt durch alle Nachbarn des Punktes durch eine Mehrheitswahl. Mithilfe des vorgestellten Verfahrens werden hohe Genauigkeiten der Klassifikationen von Aktionen wie Radfahren, Golf spielen, Laufen und Sitzen von mindestens 97% erzielt [16, S. 1263]. Für diese Klassifikationen wurden die ersten zwei Hauptkomponenten der Messungen am rechten Handgelenk und dem linken Fußgelenk durchgeführt (vgl. [16, S. 1263]).

Anstatt der Nutzung von Persistence Landscapes ist eine Nutzung anderer Methoden denkbar, welche die topologischen Informationen aus der persistenten Homologie in Vektorform übersetzen. Eine Präsentation einer weiteren Methode inklusive Anwendung auf einen Datensatz zur Klassifikation von Proteinen ist im Paper „Persistence weighted Gaussian kernel for topological data analysis“ von Kusano et. al. zu finden [26]. Die Autoren vergleichen in ihrem Paper ihren Vorschlag der Repräsentation der Informationen aus der persistenten Homologie über Persistente gewichtete Gaußkernfunktionen (engl. *Persistent Weighted Gaussian kernels*, PWGK) mit anderen Methoden der Bereitstellung einer topologischen Zusammenfassung wie

dem molekularen topologischen Fingerabruck (MTF) von Cang et. al. oder dem positiv definiten Kernel namens Persistence scale space kernel (PSSK).

Die Autoren berechnen die Persistenzdiagramme unter Berücksichtigung der zweidimensionalen Hohlräume. Anhand des Persistenzdiagrammes wird eine Darstellung als Vektor erstellt, welche im Anschluss von statistischen Verfahren beispielsweise zur Klassifikation genutzt werden kann. Zur Umwandlung des Persistenzdiagramm in eine Vektorform ist die Wahl eines Kernels und einer Gewichtung entscheidend. Als Kernel verwenden die Autoren die mehrdimensionale Normalverteilung [26]. Die Begründung für die Gewichtung der Generatoren $x \in D$ aus dem Persistenzdiagramm D ist, dass Punkte des Persistenzdiagrammes häufig durch zufälliges Rauschen verursacht wurden. Um deren Beitrag zum Gesamtmodell herabzusetzen, ist eine Gewichtung dieser Punkte notwendig.

Mithilfe der Vektorisierung der Informationen durch Persistence Weighted Gaussian Kernels, ist zunächst auf 100 zufällig generierten Datensätzen im Vergleich zu den anderen Methoden mit diesem Ziel ein deutlicher Performancegewinn sichtbar [26]. Die Autoren zeigen in ihrem Paper, dass die Berechnung der PWGK deutlich effizienter ist als die Berechnung der PSSK. Die Qualität der vorgestellten Methode zur Bereitstellung von Informationen aus der persistenten Homologie wird ebenfalls auf realen Datensätzen mit der anderer Verfahren verglichen. Hierbei stellt sich heraus, dass das von den Autoren vorgestellte Verfahrens zur Quantifizierung der Informationen aus der persistenten Homologie zu besseren Klassifikationsergebnissen führt, als die Vergleichsverfahren. Ein Vorteil des Verfahrens ist die Möglichkeit der Kontrolle über die Spezifikation einer Kernfunktion beziehungsweise der verwendeten Gewichte (vgl. [26]).

Eine Alternative zum Persistence Diagramm ist der Persistence Barcode, welcher zur Transformation der aus der persistenten Homologie gewonnenen Informationen in eine für die Anwendung statistischer Verfahren notwendige Darstellung dient. Adcock et. al. stellen in ihrem Paper mit dem Titel "The Ring of Algebraic Functions on Persistence Bar Codes" den Persistence Barcode vor, welcher zur Übersetzung von topologischen Informationen in eine Vektordarstellung geeignet ist. Diese Persistence Barcodes finden Anwendung im Paper „Movie Genre Detection Using Topological Data Analysis" von Doshi und Zadrozny, welches die Klassifikation von Filmen der Filmdatenbank IMDB mithilfe von Informationen aus der persistenten Homologie durchführt [17]. Obwohl das vorgestellte Verfahren auf einer wesentlich geringeren Datenmenge trainiert wurde, ist die Performance des Klassifikators, den die

Autoren im Paper vorgestellt haben und die persistente Homologie nutzt, höher als die der Verfahren anderer Autoren[17].

Eine Alternative zu den Persistent Landscapes bieten die von Adams et.al. vorgestellten Persistence Images, welche ebenfalls die Informationen aus der persistenten Homologie für Verfahren aus dem Machine Learning nutzbar machen [1]. Die Entwicklung des Persistence Images geschieht als Repräsentation des Persistenzdiagrammes D als Vektor im $\mathbb{R}^n$. Darüber hinaus soll die Repräsentation eine Stabilität gegenüber leichten Verzerrungen der Daten aufweisen und mit wenig Aufwand berechnet werden können. Zudem soll der Anwender eine Verbindung zwischen der Vektordarstellung und dem Persistenzdiagramm herstellen können. Eine letzte Anforderung ist die Möglichkeit einzelne Punkte des Persistenzdiagramms unterschiedlich zu gewichten, um deren Einfluss auf die Vektordarstellung des Persistenzdiagrammes zu justieren.

Der erste Schritt der Generierung einer Vektordarstellung des Outputs aus der persistenten Homologie ist zunächst die Herstellung einer Persistence Surface. Hierzu werden zunächst die Paare (b_i, d_i) aus dem Persistenzdiagramm mithilfe der Abbildung $T(b_i, d_i) = (b_i, d_i - b_i)$ transformiert [1, S. 7]. Durch Anwendung der Dichtefunktion der zweidimensionalen Normalverteilung mit Erwartungswert $u = (b_i, d_i - b_i)^T$ und Varianz σ^2 entsteht durch Summation über alle im Persistenzdiagramm dargestellten Paare mit einer Gewichtsfunktion $f(u)$ die Persistence Surface ρ_D. Mithilfe der Definition 7 erhält man das Persistence Image (vgl. [1, S. 7]).

Definition 7 (Persistence Image). *Ist D ein Persistenzdiagramm, so entsteht das Persistence Image durch die Diskretisierung p von ρ_D und die Berechnung des Integrals der Persistence Surface in allen Intervallen der Diskretisierung. Somit ergibt sich das Persistence Image als $I(\rho_D)_p = \int \int_p \rho_D dydx$.*

Die Diskretisierung p erfolgt durch ein Gitter mit vorgegebener Auflösung, mit dem das Persistence Diagramm überdeckt wird. Von den Autoren durchgeführte Experimente zeigen, dass die Methode robust gegenüber der Wahl der Auflösung dieses Gitters ist [1, S. 8]. Als Gewichtsfunktion wird die in Gleichung 3.1.4 darstellte stückweise definierte Funktion verwendet. Hierbei kennzeichnet $b = \varepsilon_{b,min} - \varepsilon_{d,max}$ die Persistenz der topologischen Struktur,

bei der die Differenz von ε_b und ε_d am größten ist.

$$w_b(t) = \begin{cases} 0, & \text{falls } t \leqslant 0 \\ \frac{t}{b}, & \text{falls } 0 < t < b \\ 1, & \text{falls } t \geqslant b \end{cases} \tag{3.1.4}$$

Die Autoren des Papers weisen die Stabilität der Persistence Surfaces und der Persistence Images gegenüber kleineren Änderungen durch Rauschen der Daten nach. Zudem zeigt sich bei Experimenten auf verschiedenen Datenbeständen, dass die vorgestellten Persistent Images in Bezug auf die Genauigkeit der Vorhersage den Persistence Landscapes und den Persistence Diagrammen überlegen sind [1, S. 22].

Alle vorgestellten wissenschaftlichen Publikationen verwenden als Basis das Persistence Diagramm oder den Barcode Plot. Diese Diagramme entstehen durch die Bestimmung der persistenten Homologie. Die Informationen der Radien des Auftretens und des Verschwindens der topologischen Strukturen, welche im Persistence Diagramm oder im Barcode Plot dargestellt sind, werden auf mehr oder weniger komplizierte Weise transformiert, sodass eine Repräsentation dieser Informationen entsteht, welche ein Machine Learning als Input oder zusätzlichen Input verwenden kann. Insgesamt kann bei allen Publikationen, die eine Vorhersage durchführen, eine Verbesserung der Performance der Vorhersage durch die ausschließliche oder zusätzliche Nutzung der Informationen aus der persistenten Homologie beobachtet werden. Allerdings ist der Effekt auf die Vorhersagegüte der Nutzung von Features aus der topologischen Datenanalyse in den verschiedenen Anwendungsfällen unterschiedlich stark ausgeprägt.

3.2. Forschungsstand auf Basis des Mapper Graphs

Eine Einsatzmöglichkeit des Mapper Graphs zeigt das Paper „Extracting Knowledge from The Geometric Shape of Social Network Data Using Topological Data Analysis“ von den Autoren Khaled Almgren, Minkyu Kim und Jeongkyu Lee ist die Klassifikation von Bildern und deren Beschreibung des sozialen Netzwerks Instagram hinsichtlich der Popularität der Inhalte (vgl. [2]). Die Autoren nutzen als Hilfsmittel zur Klassifikation die topologische Datenanalyse in Form des Mapper Graphs. Der Datenbestand umfasst rund 49.000 Bilder mit Bildunterschriften. Als Maßzahl, ob es sich um ein populäres Bild handelt, wird eine normalisierte Anzahl der Followers genutzt. Dies trägt der Beobachtungen der Autoren Rechnung, dass 20% der Bilder 99% aller Likes erhalten [2, S. 5]. Diese Maßzahl wird durch die Autoren als sozialer Kontext bezeichnet. Ein Bild gilt in einer Zeitspanne von einer Stunde als populär, wenn es in dieser

Zeit 45 Likes bekommen hat. Populär nach einem Tag ist ein Bild, am ersten Tag 69 Likes bekommt. Als Schwellenwert für die Popularität eines Bildes bezogen auf das Zeitintervall einer Woche gelten 75 Likes [2, S. 9]. Die Autoren evaluieren in getrennten Untersuchungen die Vorhersage der Popularität basierend auf die Bildunterschriften der hochgeladenen Bilder und auf Basis der normalisierten Anzahl der Followers. Wegen der notwendigen Transformation der Bildunterschriften in eine Vektordarstellung mithilfe von *word2vec* ergibt sich hierbei ein hochdimensionaler Datensatz. Anhand der 300-dimensionalen Vektordarstellung der Bildunterschriften wird jeweils die paarweise Cosinus-Ähnlichkeit der Bildunterschriften berechnet. Für die Erstellung einer Distanzmatrix des sozialen Kontextes wird die euklidische Metrik verwendet.

Das Ziel des Papers ist eine Vorhersage der Popularität der Bilder durch ein Clustering der Daten. In diesen Clustern erfolgt die Bestimmung der durchschnittlichen Popularität. Die Vorhersage der Popularität erfolgt durch die Zuordnung der Inhalte zu den berechneten Clustern. Für das Clustering verwenden die Autoren des Papers den Mapper Graph, den k-Means Algorithmus und das hierarchische Clustering. Als Linse zur Erstellung des Mapper Graphen nutzen die Autoren eine Dichteschätzung. Für die Überdeckung der Linse werden 5 Intervalle gewählt. Eine Aussage zur Überlappung dieser Intervalle treffen die Autoren nicht. Sie zeigen in ihrem Paper, dass bei der Verwendung des Mapper Graphs bei hochdimensionalen Datensätzen höhere Genauigkeiten der Vorhersage der Popularität erzielt werden als bei der Verwendung des k-Means oder des hierarchischen Clusterings (vgl. [2, S. 15]). Bei der Nutzung des Features, welches den sozialen Kontext des Bildes bewertet und somit eindimensional ist, kann die topologische Datenanalyse die Genauigkeit der Klassifikation leicht anheben, allerdings zeigt sich dieser Anhub nicht auf den Bildern mit einer niedrigen Popularität. Zusammenfassend lässt sich daher sagen, dass die topologische Datenanalyse besonders für Datensätze mit einer hohen Anzahl von Features geeignet ist. Für die in der wissenschaftlichen Ausarbeitung vorgestellten Ergebnisse wurde eine Teilung des Datenbestandes in 70% Trainings- und 30% Testdaten vorgenommen.

Eine weitere, in der wissenschaftlichen Literatur thematisierte Einsatzmöglichkeit des Mapper Graphens ist die **Feature Selection**. Diese Variante der Kombination der topologischen Datenanalyse mit Machine Learning Verfahren erfolgt in den Papers von Rucco et. al. und Guo und Banerjee [44], [31], [21]. Die Autoren der Paper erstellen zunächst den Mapper Graph und leiten daraus auf verschiedene Weisen Gruppierungen der Daten ab. Für die Erstellung

des Mapper Graphs wird als Metrik die Funktion aus Gleichung 3.2.1 verwendet.

$$D(x_1, x_2) = \sqrt{\sum_{i=1}^{d} \frac{(x_{1i} - x_{2i})^2}{\sigma^2}} \tag{3.2.1}$$

Im Anschluss daran werden die Features ermittelt, welche einen großen Einfluss auf die Trennung der Beobachtungen in diese separate Gruppierungen besitzen. Bei der Publikation von Rucco et. al. werden als Subgruppen die durch *Ayasdi Iris* ermittelten Gruppierungen der Daten verwendet [44, S. 50]. *Ayasdi Iris* ist eine Software, welche die Durchführung der topologischen Datenanalyse ermöglicht und durch Ayasdi vertrieben wird. Thematisch beschäftigt sich das Paper von Rucco et. al. mit der Diagnose von Lungenembolien.

Zur Klassifikation wird ein von den Autoren trainiertes künstliches neuronales Netz mit alternativen Verfahren wie dem *Revised Geneva Score* oder dem *Wells Score* verglichen. Beim *Revised Geneva Score* oder dem *Wells Score* erfolgt die Diagnose einer Lungenembolie auf Basis kategorieller Variablen, bei denen jede Ausprägung eine bestimmte Punktzahl zugewiesen bekommt. Durch die Summierung der Punkte und Vergleich mit einem Vergleichswert erfolgt die Diagnose oder der Ausschluss einer Lungenembolie. Als Filterfunktion zur Berechnung des Mapper Graphs wird die L_∞ *Zentralität* gewählt, welche jedem Punkt die Distanz des am Weitesten entfernten Punktes zuordnet. Zur Überdeckung werden 60 überlappende Intervalle mit einer Überlappung von $\frac{2}{3}$ verwendet [44, S. 48]. Die Autoren konnten zeigen, dass die Anwendung der topologischen Datenanalyse in Form des Mapper Graphens einen deutlichen Anstieg des AUROC der Klassifikatoren verursacht (siehe [44, S. 50]). Somit zeigt sich, dass die **Feature Selection** für das Trainieren eines künstlichen neuronalen Netzwerkes mithilfe des Mapper Graphens funktioniert.

Die Autoren Guo und Banerjee verwenden in ihrem Paper „Towards Automated Prediction of Manufacturing Productivity Based on Feature Selection Using Topological Data Analysis“ als Datenbestand einen Datensatz aus dem R Paket *AppliedPredictiveModeling*, welcher Daten zu Produktionsprozessen der chemischen Industrie enthält [21]. Zur Erstellung einer Distanzmatrix der Daten kommt die euklidische Metrik zum Einsatz. Als Linse wird die multidimensionale Skalierung verwendet. Zur Erstellung des Graphen ist die Überdeckung der Linse in jeder der zwei Dimensionen durch 14 Intervalle mit einer Stärke der Überlappung von 80% überdeckt worden. Aus diesem werden vier Subgruppen extrahiert. Mithilfe des Kolmogorov-Smirnov für zwei Stichproblem werden im Anschluss die Gruppen verglichen und Features identifiziert, die für die Trennung der Daten in diese Gruppen am ausschlaggebendsten sind.

Diese signifikantesten Features werden im Anschluss zur Klassifikation genutzt. Zur Vorhersage der durch den chemischen Herstellungsprozess hergestellten Menge wird eine Hauptkomponentenregression, ein Random Forest und *cubist* als regelbasiertes Klassifikationsverfahren genutzt. Alle drei Klassifikationsverfahren werden zunächst mit allen zur Verfügung stehenden Features trainiert. Im Vergleich hierzu erfolgt auf allen drei Klassifikatoren ein Trainingsprozess auf den Features, welche mithilfe des Mapper Graphs selektiert wurden. Durch die Reduktion der Features auf die Features, welche die Zugehörigkeit zu den Subgruppen des Mapper Graphs gut erklären, nimmt der Fehler auf Trainings- und Testdatensatz ab (siehe [21, S. 35]). Somit haben die Autoren eine neue Art der **Feature Selection** vorgestellt.

Eine weitere sehr interessante Anwendungsmöglichkeit der topologischen Datenanalyse mithilfe des Mapper Graphens ist die Visualisierung der Gewichte eines künstlichen neuronalen Netzes. Künstliche neuronale Netze erfreuen sich zurzeit größter Beliebtheit, da sie oftmals eine gute Performance haben. Jedoch gehören künstliche neuronale Netze zu den Blackbox Modellen und damit ist das Zustandekommen der Outputs dieser Klassifikatoren nicht transparent. In dem Paper mit dem Titel „A look at the topology of convolutional neural networks“ stellen Gabrielsson und Carlsson eine Methode vor, welche die Gewichtsmatrizen der Layer von neuronalen Netzwerken einer topologischen Datenanalyse unterzieht, um die dort durchgeführten Berechnungen durch eine Visualisierung nachvollziehen zu können [18]. Anhand dieser Visualisierungen kann der Anwender nun die durchgeführten Berechnungen und deren Auswirkungen leichter nachvollziehen. Zur Vorführung der Methode zur Visualisierung der Gewichtsmatrizen von neuronalen Netzen verwenden die Autoren den MNIST Datensatz, den CIFAR-10 Datensatz und dem ImageNet Datensatz. Diese Datensätze entstammen alle dem Anwendungsbereich der Klassifikation von Bildern. Der MNIST Datensatz enthält Bilder von handgeschriebenen Ziffern in Graustufen. Der zweite Datensatz besteht aus Farbbildern von 10 unterschiedlichen Objekten wie Flugzeugen, Katzen, Hunden und Schiffen. Einen weiteren von den Autoren verwendeter Datensatz ist der ImageNet Datensatz, welcher zum Trainieren von Convolutional Neural Networks dient [13]. Auf jedem Datensatz wird eine unterschiedliche Zahl von künstlichen neuronalen Netzen trainiert. Die von den Autoren Gabrielsson und Carlsson präsentierte Methode zur Herstellung eines Verständnisses der Berechnungen in einer Schicht des Netzwerks beruht auf die Visualisierung der Gewichte in verschiedenen Schichten des neuronalen Netzwerkes durch den Mapper Graph. Die notwendige Distanzmatrix, welche die paarweisen Abstände zwischen den Gewichtswerten enthält, wird ein k-nächste Nachbar Dichteschätzer angewandt [18, S. 3]. Als Linse zur Erstellung des Mapper Graphen wird für alle Datensätze die ersten zwei Hauptkomponenten verwendet. Zur Überdeckung der Linse werden einheitlich 30 Intervalle mit einer Überlappung von zwei Dritteln verwendet. Durch die

erhaltenen Visualisierungen können Zwischenergebnisse des neuronalen Netzes besser nachvollzogen werden.

Skycak zeigt in seiner Einführung zur topologischen Datenanalyse mit dem Titel „Topological Data Analysis - Theory Practice, Software, and Potential“ eine Anwendung des Mapper Graphens als Methode zur Identifikation der Fehlerquellen von Modellen [47]. Hierzu werden die Nodes des bereits erstellten Mapper Graphs etwa nach dem Anteil der falsch positiven Klassifikationen im jeweiligen Node eingefärbt [47, S. 21]. Auf diese Weise ist leicht erkennbar, in welchen Gebieten das Modell hohe Fehlerraten besitzt. Sind die Bereiche mit einer schlechten Performance bekannt, so können zum Beispiel mithilfe statistischer Tests die Features identifiziert werden, welche ursächlich für die schlechte Performance sind. Dies erfolgt durch den Vergleich des Bereiches mit der hohen Falsch-Positiv-Rate und dem „Rest“ des Graphen [47, S. 21]. Sind die Bereiche bekannt, in denen das Modell eine schlechte Performance aufweist, kann dies bei der Modellierung berücksichtigt und gegengesteuert werden.

Im Unterschied zur persistenten Homologie sind keine wissenschaftlichen Publikationen auffindbar, welche Methoden vorstellen, um die Performance von Machine Learning Verfahren durch Hinzunahme oder Verwendung von topologischen Features, welche direkt anhand des Mapper Graphs bestimmt werden, zu verbessern. Diese Masterthesis nimmt sich in den folgenden Kapiteln dieser Lücke an und stellt eine Methode vor, welche den Datensätzen ein topologisches Feature hinzufügt, welches aus dem Mapper Graph generiert wird.

Teil II.

Experimente

4. Durchführung der Experimente

Nachdem die Grundlagen im Kapitel 2 beschrieben und im Kapitel 3 die aktueller Forschungsgegenstand wiedergegeben wurde, soll dieses Kapitel das Vorgehen der im Rahmen dieser Masterthesis durchgeführten Experimente zur Verbesserung eines Klassifikators beschrieben werden.

Für die in dieser Masterarbeit angestellte Forschung wird der Mapper Graph als Resultat des zweiten Ansatzes der topologischen Datenanalyse verwendet. Zur Durchführung der Experimente wurde im Rahmen der Masterthesis ein Prozess der die durch die topologische Datenanalyse unterstützte Klassifikation implementiert, erarbeitet. Dieser Prozess und die Durchführung der Experimente werden im weiteren Textverlauf beschrieben.

Ein weiteres Unterkapitel widmet sich der Vorstellung der zur Untersuchung der Fragestellung, ob die topologische Datenanalyse eine Möglichkeit bietet, die Qualität der Klassifikation zu verbessern, verwendeten Datenbasis. Die verwendeten Datensätze entstammen dem Finanzbereich. Ein Datensatz enthält die Daten von Transaktionen mit Kreditkarten und die Kennzeichnung, ob es sich bei einer Transaktion um Betrug handelt oder nicht. Zwei weitere, ebenfalls zur Untersuchung der Fragestellung, verwendete Datensätze beschäftigen sich mit dem Ausfall von Krediten.

4.1. Der Prozess von den Rohdaten zur mithilfe der topologischen Datenanalyse unterstützten Klassifikation

Es soll untersucht werden, ob sich die Qualität eines Klassifikators verbessert, wenn Informationen aus der topologischen Datenanalyse als zusätzlicher Input beim Training des Klassifikators zur Verfügung stehen.

Zur Beantwortung dieser Frage wurden die Schritte aus dem Prozessdiagramm 4.1 erarbeitet. Im Diagramm nicht dargestellt ist der Prozess des Trainings eines Klassifikators auf den Ursprungsdaten ohne topologische Informationen, um die Performance des Klassifikators mit

und ohne topologische Features vergleichen zu können.

Im ersten Schritt des Gesamtprozesses in Abbildung 4.1 werden die Daten aus einer *csv* Datei eingelesen. Der eingelesene Datensatz wird nun, wie in (2) des Prozessdiagramms in Abbildung 4.1 in Trainings- und Testdatensatz geteilt. Hierbei wird darauf geachtet, dass die Verteilung von betrügerischen und normalen Transaktionen im Trainings- als auch im Testdatensatz gleich der im Gesamtdatensatz ist. Dies geschieht durch stratifiziertes, zufälliges Auswählen der Datensätze im Trainings- und Testdatensatz. Das Verhältnis der Datensätze in Trainings- und Testdatensatz ist der Parameter dieses Schrittes.

Der nächste Schritt ist die Erzeugung des topologischen Features *connectedComponentId*, welche unter anderem durch den in Python implementierten KeplerMapper durchgeführt wird. Zum Aufrufen von Methoden der Klasse `KeplerMapper` und zur Evaluation des Klassifikators kommt daher ebenfalls die Programmiersprache Python zum Einsatz. Die Implementierung der in Abbildung 4.1 beschriebenen Schritte und die Darstellung der Ergebnisse erfolgt für die Durchführung der Experimente in einem Jupyter Notebook. In Kapitel 8 wird ein Prototyp designt und entwickelt, welcher den hier beschriebenen Prozess abbildet.

Der Ablauf der Erzeugung des topologischen Features *connectedComponentId* ist im Prozessdiagramm in Abbildung 4.2 graphisch dargestellt. Im Folgenden werden die einzelnen Bestandteile des Prozesses in Abbildung 4.2 erläutert. Als Eingabe des Prozesses zur Erzeugung des topologischen Features dient der Trainingsdatensatz ohne Targetvariable.

Zur Berechnung des Mapper Graphen als Output der topologischen Datenanalyse werden im Schritt (2) die Trainingsdaten zunächst in einen Raum mit geringerer Dimension projiziert. Diese Projektion erfolgt, nicht wie in den Grundlagen beschrieben auf Basis einer Distanzmatrix. Stattdessen wird die Linse direkt auf dem Datensatz berechnet, da die Berechnung von Distanzmatrizen für größere Datensätze wegen des enormen Speicherplatzbedarfs nicht möglich ist. Die hierbei verwendete Projektionsfunktion ist in der Literatur unter dem Namen Linse (engl. lens) bekannt. Die Parametrisierung dieses Schrittes wird durch die verwendete Projektionsfunktion bestimmt. Insgesamt soll diese Dimensionsreduktion eine Trennung der normalen und der betrügerischen Transaktionen in Bezug auf den dimensionsreduzierten Datensatz herbeiführen. Aus diesem Grund wird beispielsweise als Linse die lineare Diskriminanzanalyse genutzt, weil diese das Ziel hat, die Daten in einen Raum mit maximaler Trennung der Gruppen zu transformieren. Ob Trennung der betrügerischen Transaktionen von den normalen Transaktionen gelungen ist, wird anhand eines Scatterplots, in dem die auf ein oder

zwei Dimensionen reduzierten Daten in der Farbe rot für betrügerische Transaktionen und blau für normale Transaktionen visualisiert sind, optisch überprüft. Eine Beschreibung der Eigenschaften der verwendeten Linsen liefert das Teilkapitel 2.3.

In Schritt (3) der Abbildung 4.2 erfolgt eine Überdeckung der Linse mit überlappenden Intervallen. Die Parameter für diesen Schritt ist zum Einen die Anzahl der Intervalle pro Dimension *n_cubes* und zum Anderen die Stärke der Überlappung in Prozent der Intervallbreite *overlap_perc*. Gibt man für den Parameter *n_cubes* eine Liste von ganzzahligen, positiven Zahlen an, so können für jede Dimension unterschiedliche Anzahlen von Intervallen je Dimension für die Überdeckung verwendet werden. Entspricht der Parameter *n_cubes* einer Zahl, so wird die Linse in allen Dimensionen durch die gleiche Anzahl von Intervallen überdeckt.

Der Schritt (4) der Erzeugung des topologischen Features *connectedComponentId* führt je überlappendem Intervall ein Clustering der Daten aus dem Trainingsdatensatzes durch, die innerhalb des jeweiligen Intervalles liegen. Für diesen Vorgang können verschiedene Clusteringalgorithmen verwendet werden. Die Spezifizierung des genutzten Clusteringalgorithmus geschieht über den Parameter *clusterer*, dem eine Instanz einer Klasse übergeben wird, die den gewählten Clusteringalgorithmus implementiert. Die Parametrisierung ergibt sich hier aus den Parametern des gewählten Clusteringalgorithmus. Als Clusteralgorithmus verwende ich den in der Bibliothek `sklearn` implementierten `KMeans++` mit einem Cluster pro Intervall. Somit handelt es sich in diesem Fall beim Clustering um eine Optimierung der Clustermittelpunkte innerhalb jedes Intervalles.

Im Anschluss an das Clustering der Ursprungsdaten stellt der 5.Schritt in Abbildung 4.2 die Cluster in jedem Intervall in einem Graph dar. Jedes Cluster im überlappenden Gitter entspricht einem Node im Graph. Sollten Datenpunkte sowohl in einem als auch in einem benachbarten Cluster liegen, so werden die nebeneinander liegenden Nodes mit einer Kante verbunden. Die Farbe der Nodes ergibt sich aus der häufigsten Klasse der Datenpunkte, die in diesem Cluster liegen. Um eine möglichst gute Trennung der Nodes mit überwiegend betrügerischen Transaktionen von denen mit mehrheitlich unauffälligen Kreditkartentransaktionen zu erhalten, muss die Anzahl der Intervalle pro Dimension hoch angesetzt werden. Gute Ergebnisse zeigen sich beim Kreditkartentransaktionsdatensatz für Werte von circa 50, wobei dies mit der verwendeten Linse zusammenhängt. Da sich wegen dieser Einstellung eine sehr hohe Anzahl von Nodes ergibt und die Nodes mit überwiegender Anzahl von Betrugsfällen im Graph häufig eine zusammenhängende Komponente bilden, kommt als *connectedComponentId* die

ID der zusammenhängenden Komponente des Nodes, in dem der jeweilige Trainingsdatensatz liegt, infrage.

Die Bestimmung der zusammenhängenden Komponenten im Graph und das Mapping der Datensätze zu den ermittelten ID der zusammenhängenden Komponenten erfolgt im Schritt (6) der Abbildung 4.2. Die Ausgabe der *map* Funktion des Kepler Mappers ist ein Python Dictionary, welches Angaben über die Knoten und Kanten des Mapper Graphen enthält. Daher kann dieses Zwischenergebnis zur Bestimmung der zusammenhängenden Komponenten des Mapper Graphs verwendet werden. Die Ermittlung der Gruppierungen von Nodes ohne Verbindungen geschieht mithilfe des Paket `networkx`, welches zur Analyse von Graphen genutzt werden kann. Zunächst wird der durch den `KeplerMapper` erzeugten Mapper Graph in einen networkx Graph übersetzt, indem die Namen der Knoten und die Kanten aus dem Output der *map* Funktion des Kepler Mappers extrahiert werden. Basierend auf diesen Graphen wird durch die Funktion *compute_connected_components* des networkx Pakets die zusammenhängenden Komponenten bestimmt. Die Ergebnisse dieses Teilschritts werden in einem Python Dictionary gespeichert, welches jedem Knoten im Graphen die ID seiner zusammenhängenden Komponente zuordnet. Der Output der zuvor beschriebenen sechs Schritte aus dem Prozessdiagramm 4.2 ist eine Liste der *connectedComponentIds* jedes Objektes im Trainingsdatensatz.

Nun wird in Schritt (4) der Abbildung 4.1 auf dem durch das topologische Feature *connectedComponentId* angereicherten Trainingsdatensatz ein Klassifikator trainiert. Das Training eines Entscheidungsbaumes erfolgt in der Programmiersprache R, deren Nutzung durch das Modul `rpy2` direkt innerhalb Python ermöglicht wird. Da durch das topologische Feature *connectedComponentId* in den Datensatz, der nur aus stetigen Merkmalen besteht, ein kategorisches Merkmal hinzukommt, sollte der Algorithmus der zum Training des Einscheidungsbaumes genutzt wird in der Lage sein mit kategorischen Merkmalen zu arbeiten. Dies ist bei der in der in der Bibliothek `sklearn` implementierten Klasse `DecisionTreeClassifier` nicht der Fall, weswegen das Training des Entscheidungsbaum in der Programmiersprache R durchgeführt wird. Nichtsdestotrotz splittet der Entscheidungsbaum des R Pakets `rpart` binär. Für das Feature *connectedComponentId* ergibt sich daher eine Zusammenfassung der Ausprägungen dieses Merkmals in zwei Mengen. Das Training eines logistischen Regressionsmodells mit Regularisierung und die Vorhersage der Klassifikation auf Basis dieses Modells erfolgt durch das Modul `statsmodels`. Die verwendete Funktion erfordert die Durchführung eines One Hot Encodings der *connectedComponentId*, da die verwendete Funktion nur nu-

merische Daten verarbeiten kann.

Im 5. Schritt des in Abbildung 4.1 dargestellen Flussdiagramms muss das topologische Feature *connectedComponentId* für jede Instanz im Testdatensatz vorhergesagt werden, da die topologische Datenanalyse nur auf dem Trainingsdatensatz durchgeführt wird. Um die Zuordnung von Objekten des Testdatensatzes zu einem Node im Graphen zu ermöglichen wurde die Klasse KeplerMapperSplitted implementiert, die von der Klasse KeplerMapper erbt. In der Klasse `KeplerMapperSplitted` wird die Methode *map* dahingehend geändert, sodass die Namen eines jeden Nodes zusammen mit dem Clusterzentroid in einem Python Dictionary als Variable der Klasse gespeichert werden. Des Weiteren ist die Programmierung einer neuen Methode von Nöten, welche die Vorhersage der Werte des topologischen Features auf dem Testdatensatz ermöglicht. Für die Vorhersage der connectedComponentId werden zwei unterschiedliche Methoden programmiert. Bei der ersten Methode erfolgt die Vorhersage in zwei Schritten. Zunächst werden die Datensätze des Testdatensatzes einem Node im Graphen zugeordnet. Im Anschluss daran wird anhand des vorhergesagten Nodes, wie beim Trainingsdatensatz auch, die ID der zusammenhängenden Komponente ermittelt, in die der vorhergesagte Node fällt. Die Zuweisung einer Instanz des Testdatensatzes zu einem Node geschieht über die Distanz von einem Objekt des Testdatensatzes zu einem Clustermittelpunkt eines bestehenden Clusters. Die Messung der Distanz erfolgt über eine Metrik, die diesen Schritt parametrisiert. Analog zum Trainingsdatensatz wird jedem vorhergesagten Node die ID der zusammenhängenden Komponente zugeordnet, welche als neues topologisches Feature *connectedComponentId* in den Testdatensatz eingeht. Eine weitere Methode zur Vorhersage des topologischen Features auf dem Testdatensatz ist es jedem Objekt im Testdatensatz die connectedComponentId des nächstgelegenen Objektes des Trainingsdatensatzes zuzuweisen. Zur Messung dieser Distanz kommt ebenfalls eine Metrik zum Einsatz, die den Parameter dieses Schrittes darstellt. Für große Datensätze ergibt sich bei der zweiten Methode ein enorm hoher Berechnungsaufwand, sodass diese Weise der Vorhersage der connectedComponentId für große Datensätze nicht in Betracht gezogen werden sollte. Durchgeführte Experimente anhand der TSNE Linse zeigten, dass durch die Verwendung dieser algorithmisch aufwendigeren Methode zur Vorhersage der connectedComponentId auf dem Kreditkartentransaktionsdatensatz keine weitere Erhöhung der Performance erzielt wird.

Der letzte Schritt umfasst die Vorhesage der Klassifikation auf dem Testdatensatz und die Evaluation der Klassifkationsergebnisse durch verschiedene Performancemaße. Da der Kreditkartendatensatz, wie bereits erwähnt sehr unbalanciert ist, empfiehlt sich die Verwendung der

Fläche unter der Precision Recall Kurve als Bewertungskriterium des Klassifikators.

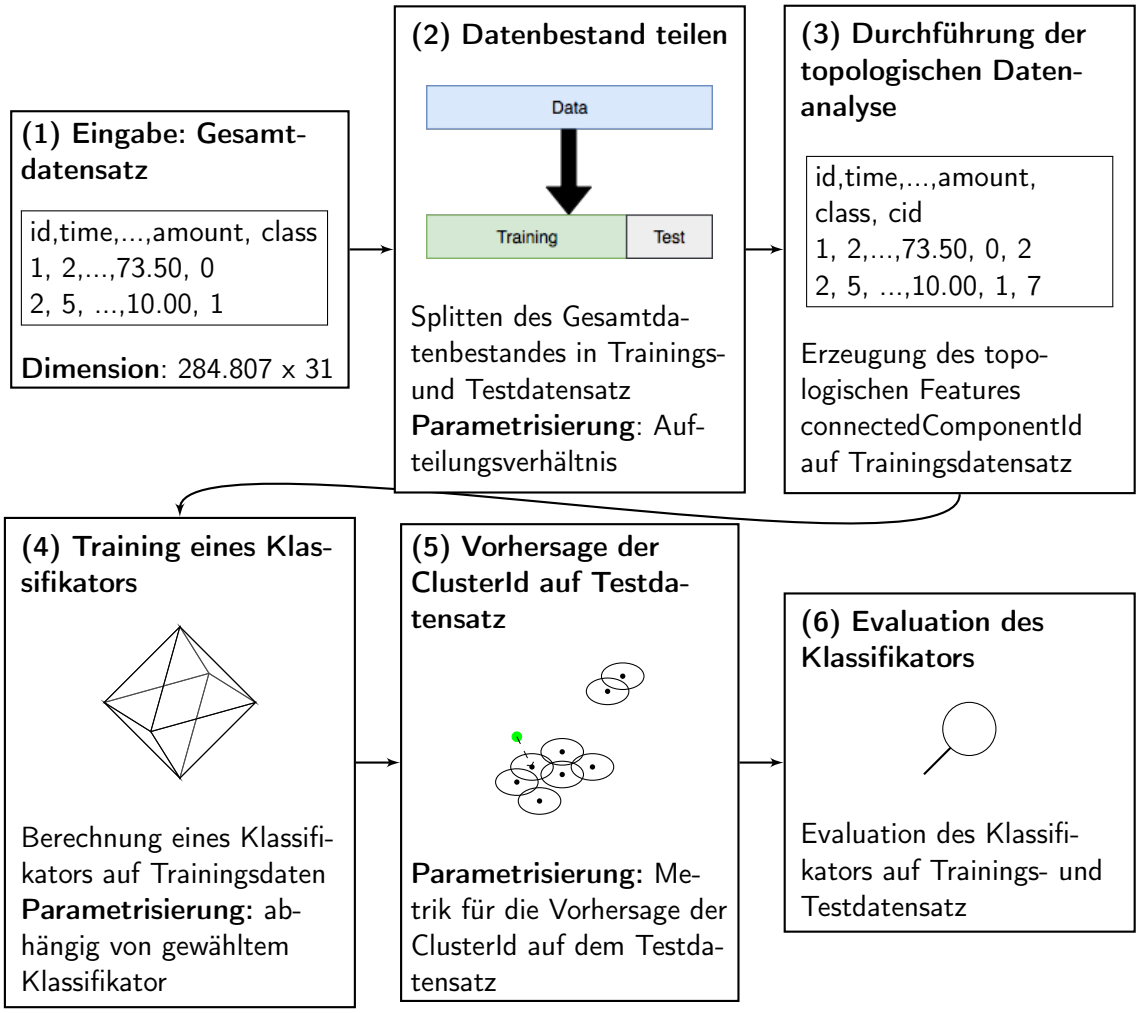


Abbildung 4.1.: Flussdiagramm des Gesamtprozesses am Beispiel des Kaggle Kreditkartentransaktionsdatensatzes

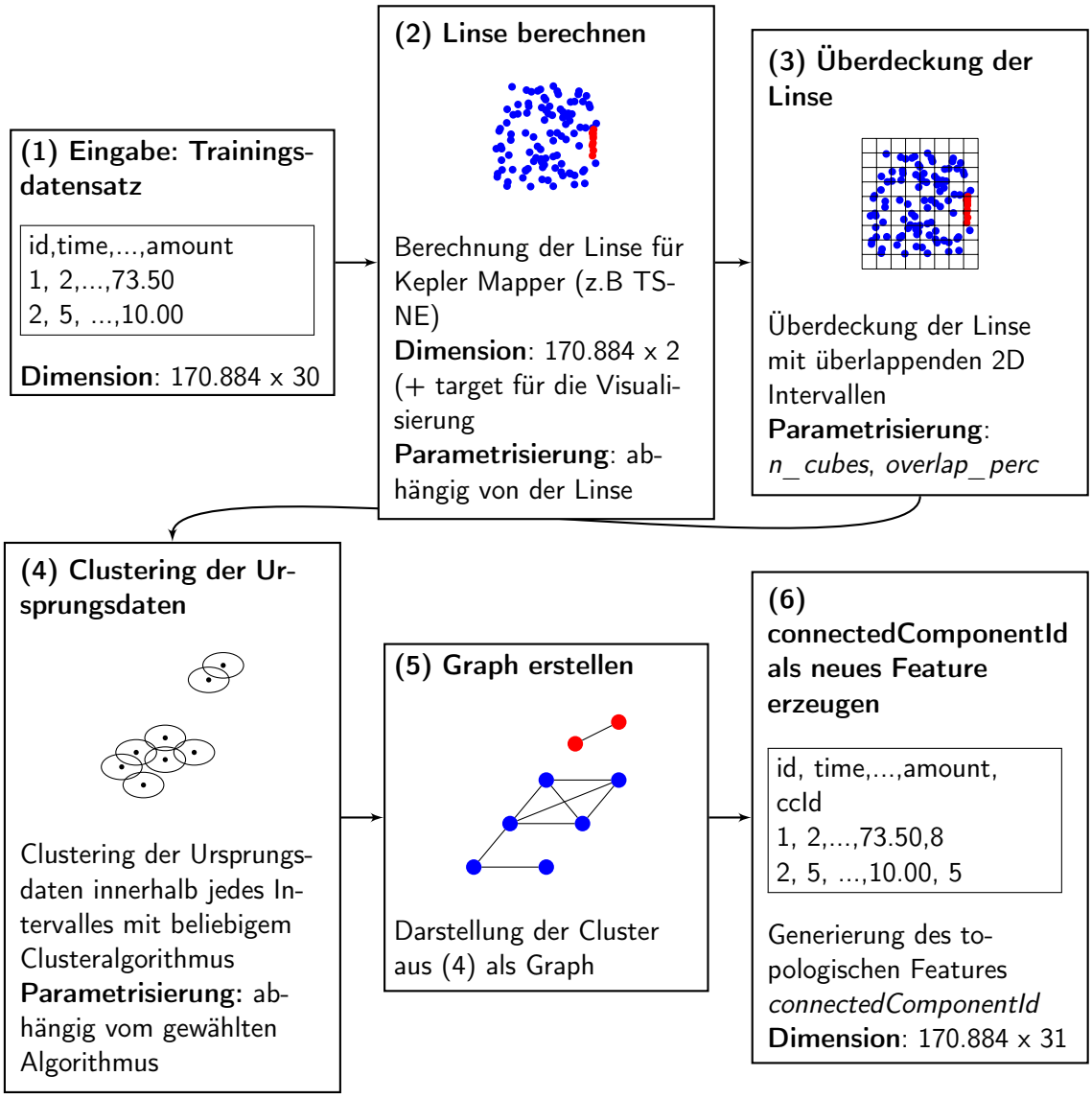


Abbildung 4.2.: Flussdiagramm des Prozesses der Erzeugung des topologischen Features *connectedComponentId* am Beispiel des Kaggle Kreditkartentransaktionsdatensatzes

4.2. Vorstellung der Datenbasis

In diesem Teilkapitel sollen die drei Datensätze, die für die Experimente genutzt werden, vorgestellt werden. Die Vorstellung der Datensätze beginnt mit einer Beschreibung der **Semantik des Datensatzes und des Targets**. Der Abschnitt **Informationen zur Quantität des Datensatzes** informiert über die Anzahl der Beobachtungen im Datensatz und die Dimension

des Featureraumes. Der nächste Aspekt der Vorstellung eines Datensatzes ist die Beschreibung der notwendigen Transformationen der Daten. Dies geschieht im Abschnitt **Notwendige Transformationen des Datensatzes**. Die Vorstellung eines Datensatzes umfasst neben der Darstellung der **Häufigkeitsverteilung des Targets** eine Beschreibung der **Semantik der Features** des Datensatzes. Anschließend wird im Abschnitt **Fehlende Werte und 0-Werte** die Behandlung von fehlenden Werten und Nullwerten behandelt. Ein Abschnitt zur **Datenexploration** des jeweiligen Datensatzes schließt mit einer kurzen explorativen Datenanalyse die Vorstellung eines Datensatzes ab.

4.2.1. Kreditkartentransaktionsdatensatz von Kaggle

Als erste Datengrundlage zur Untersuchung der eingangs erwähnten Fragestellung wird der Kreditkartentransaktionsdatensatz der Plattform Kaggle genutzt.

Semantik des Datensatzes und des Targets

Der Kaggle Kreditkartentransaktionsdatensatz enthält europäische Kreditkartentransaktionen im Internet zweier Tage im September 2013 [43, S. 164]. Der Datensatz enthält das Target *class*, welches angibt, ob eine jeweilige Transaktion Betrug ist oder nicht. Handelt es sich um eine betrügerische Transaktion so ist der Wert des Targets bei dieser Transaktion 1, ist die Transaktion normal, so ist die Targetvariable dieser Transaktion 0.

Informationen zur Quantität des Datensatzes

Der Kreditkartendatensatz besteht aus 284807 Kreditkartentransaktionen. Zu jeder Transaktion wurden 30 Features erhoben. Mit dem Target besitzt der Datensatz die Dimension 284807 x 31.

Notwendige Transformationen des Datensatzes

Aufgrund der Tatsache, dass alle Features des Datensatzes des Datensatzes numerisch sind, ist keine Transformation der Daten notwendig, um die geplanten Algorithmen anwenden zu können.

Häufigkeitsverteilung des Targets

Wie bei einem Kreditkartenbetrugsdatensatz wenig verwunderlich, weist dieser Datensatzes eine extrem starke Unbalanciertheit des Klassenattributs auf, wie der Abbildung 4.1 zu entnehmen ist.

class	Anzahl
normale Transaktionen	284315
betrügerische Transaktionen	492

Tabelle 4.1.: Ausprägungen und Häufigkeit des Targets *class* beim Kreditkartendatensatz

Semantik der Features

Aus Gründen des Datenschutzes ist der Datensatz mit Ausnahme der Features *time* und *amount*, mithilfe einer Hauptkomponentenanalyse transformiert worden. Deshalb ist die Bedeutung der restlichen Features nicht bekannt. *time* gibt der Anzahl der Sekunden zwischen der jeweiligen Transaktion und dem ersten Zahlungsvorgang an. Die Beträge der Kreditkartentransaktionen in US-Dollar finden sich in der Variable *amount*. Ein Data Dictionary zum Datensatz, in dem eine Beschreibung der Features zu finden ist, existiert nicht.

Fehlende Werte und 0-Werte

Im Datensatz sind keine fehlenden Werte vorhanden, weswegen eine Ersetzung derselben hier nicht notwendig ist. Allerdings weisen 1825 Transaktionen des Datensatzes einen Transaktionsbetrag von 0,00$ auf. Der Transaktionsbetrag von 0,00$ begründet sich vermutlich darauf, dass es sich bei diesen Transaktionen nicht um eine Bezahlung einer Ware oder Dienstleistung sondern lediglich um eine Reservierung eines Betrages auf dem Kreditkartenkonto handelt, die mit einem Wert von 0,00$ in den Datensatz eingeflossen ist, da keine Zahlung an den Anbieter erfolgte. Für die folgenden Experimente werden diese Transaktionen zunächst nicht ausgeschlossen. Ob diese Transaktionen mit einem Transaktionsbetrag ausgeschlossen werden sollten ist noch zu diskutieren.

Datenexploration

Es soll zunächst untersucht werden, ob normale Transaktionen im Vergleich zu betrügerischen Transaktionen einen höheren oder niedrigeren Transaktionsbetrag aufweisen. Eine Antwort auf diese Fragestellung liefert beispielsweise ein Boxplot, in dem die wichtigsten Lagemaße wie 1. und 3. Quartil, der Median und Minimum und Maximum dargestellt sind. Zusätzlich ist im Boxplot aus Abbildung 4.3 das arithmetische Mittel als grüne, gestrichelte Linie einbeschrieben. Wegen des im Gegensatz zu den im Vergleich eher geringen Transaktionsbeträgen hohen Maximum sind die Transaktionsbeträge im Boxplot in logarithmischer Skalierung dargestellt. Die in Abbildung 4.3 dargestellten Punkte oberhalb der Box gelten als Ausreißer, da sie um mehr als den 1,5-fachen Interquartilsabstand vom 1. beziehungsweise 3.Quartil abweichen.

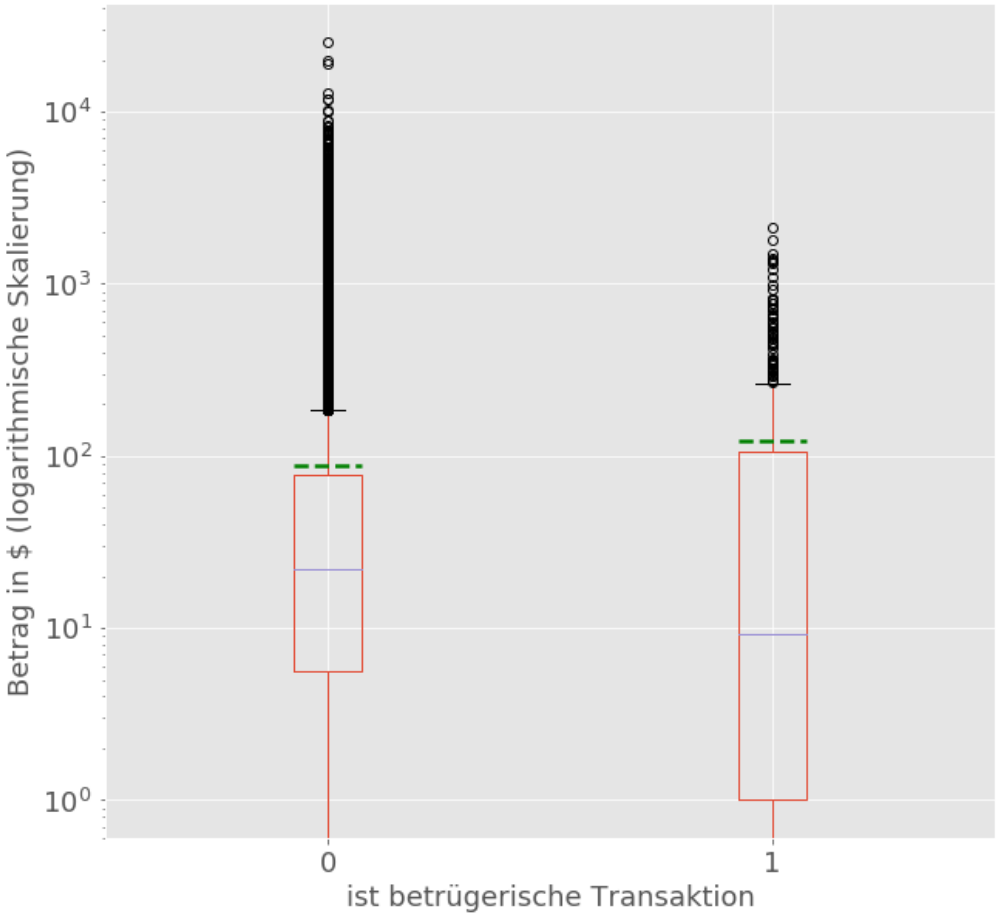


Abbildung 4.3.: Boxplot der Transaktionsbeträge in Abhängigkeit des Targets class

Bei der Betrachtung der Boxplots in Abbildung 4.3 ist deutlich zu erkennen, dass der Median und das 1.Quartil der betrügerischen Transaktionen unterhalb dem der normalen Transaktionen liegt. Zahlenmäßig ergibt sich bei den betrügerischen Transaktionen ein Median von 9,25$, bei den normalen Transaktionen weisen 50% der Transaktionen einen Transaktionsbetrug unterhalb von 22,00$ auf. Beim 1.Quartil stehen 1,00$ bei den auffälligen Transaktionen 5,65$ bei den unauffälligen Transaktionen gegenüber. Beim arithmetischen Mittel zeigt sich das entgegengesetzte Bild, da der Mittelwert der Betrugstransaktionen mit 122,21$ höher liegt als der der normalen Transaktionen mit 88,29$. Somit treten bei den 492 betrügerischen

Transaktionen deutliche Ausreißer in Bezug auf den Betrag der Transaktion auf.

Nun sollen die Unterschiede zwischen den betrügerischen und normalen Transaktionen bei einer Auswahl der Features gezeigt werden, die aus der Hauptkomponentenanalyse stammen. Es wurden in der Abbildung 4.4 für diejenigen Features Kerndichteschätzer in Abhängigkeit der Klassenzugehörigkeit berechnet, die im Entscheidungsbaum aus Abbildung 5.1 auf Seite 75 als Splitkriterium genutzt werden. Es ist bei der Betrachtung der Plots der Kerndichteschätzer offensichtlich, dass sich die Lage der Werte der ausgewählten Features innerhalb der Gruppe der betrügerischen und der Gruppe der unauffälligen Transaktionen unterscheiden.

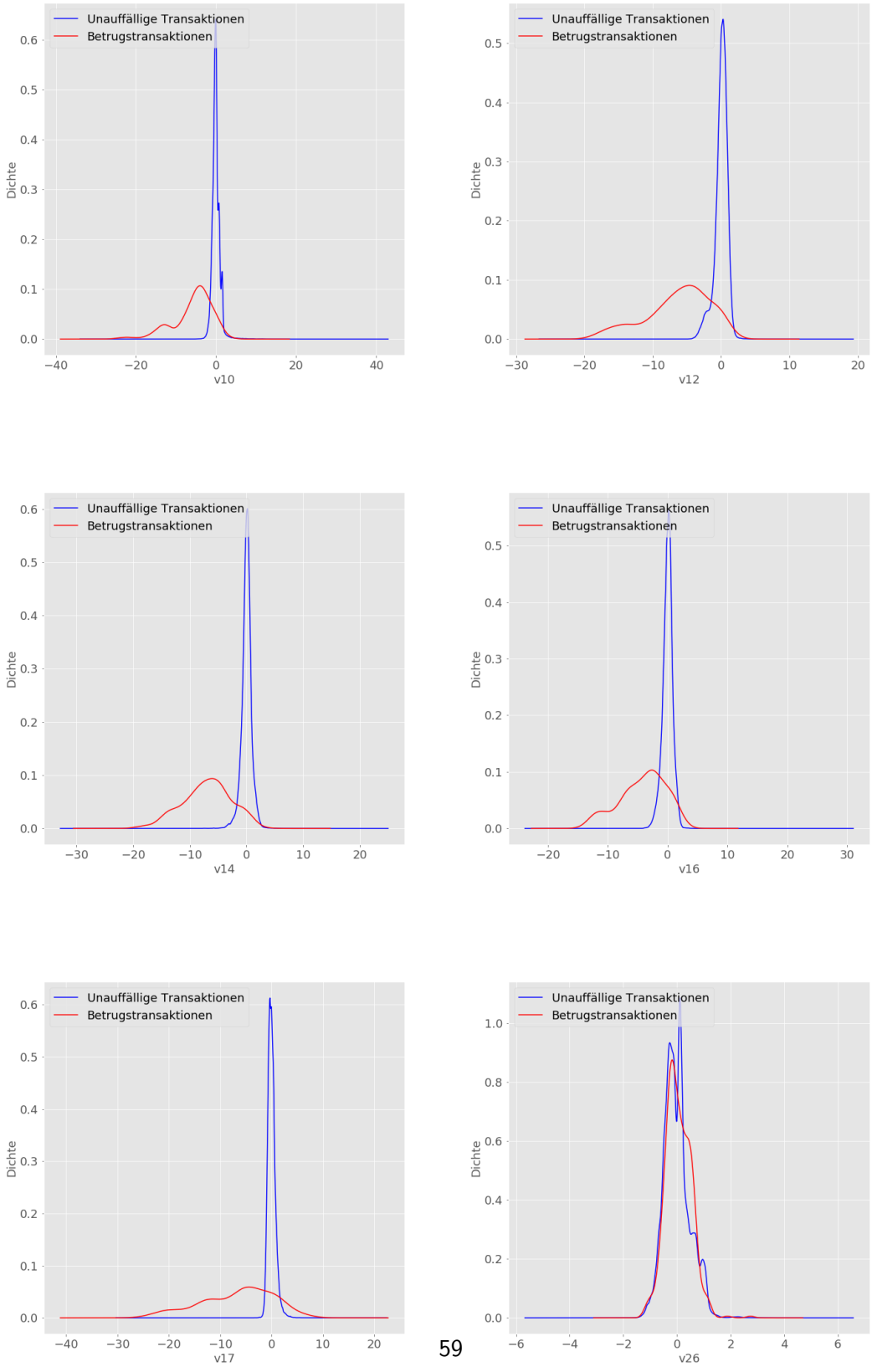


Abbildung 4.4.: Kerndichteschätzer für Features, die im Entscheidungsbaum als Splitattribut genutzt werden, in Abhängigkeit der Klasse

4.2.2. German Credit Data Set des UCI Machine Learning Repository

Als zweite Datenbasis dient der German Credit Datensatz des UCI Machine Learning Repository [15]. Der German Credit Datensatz ist im Gebiet des Machine Learning sehr bekannt.

Semantik des Datensatzes und des Targets

Der German Credit Datensatz enthält Informationen von Kreditnehmern einer Bank wie zum Beispiel den aktuellen Kontostand, den Zweck des Kredits und die Angabe ob ein Kredit zurückgezahlt wird oder nicht. Das Ziel ist es anhand von gegebenen Daten vorherzusagen, ob der Kredit ordnungsgemäß bedient werden kann.

Informationen zur Quantität des Datensatzes

Der Datensatz besteht aus 1000 Beobachtungen mit 20 Features und einer Angabe, ob der jeweilige Kredit ausgefallen ist. Somit ist der German Credit Datensatz deutlich kleiner als die beiden anderen Datensätze, die als Datenbasis dieser Masterthesis genutzt werden.

Notwendige Transformationen des Datensatzes

Im Unterschied zum Datensatz von Kaggle enthält der German Credit Data Set kategorische Merkmale wie *purpose*, welches den Grund des Kredites angibt. Darüber hinaus werden unter anderem die berufliche Situation, die frühere Kreditgeschichte des Kunden, in Form von kategorische Merkmalen erhoben. Bevor die topologische Datenanalyse auf den Datensatz angewandt werden kann, müssen daher alle kategorischen Merkmale des Datensatz in eine numerische Repräsentation übersetzt werden.

Da zu dem Datensatz ein Data Dictionary existiert, welches Beschreibungen zu den Merkmalen liefert, ist die Codierung dieser Merkmale ohne große Probleme durchzuführen. Um die kategorischen Features des Datensatzes für die topologische Datenanalyse und möglicherweise für den anschließenden Klassifikationsalgorithmus nutzbar zu machen, existieren zwei Strategien. Bei ordinalen Merkmalen ist es möglich, jeder Ausprägung eine Zahl zuzuordnen. Diese Methode wird hier **Label Encoding** genannt. Ein Beispiel für ein ordinales Merkmal wäre das Feature *savings*, welches das verfügbare Sparguthaben der Kunden in Klassen einteilt. Nominale Merkmale wie zum Beispiel das Geschlecht und die persönliche Situation der Kunden werden binär encodiert. Hier entsteht aus einem kategorischen Merkmal mit k Ausprägungen $k-1$ neue Features, die jeweils den Wert 1 aufweisen, wenn das Feature die jeweilige Ausprägung aufweist. Diese Strategie zur Encodierung nomialer Merkmale heißt daher **One Hot**

Encoding. Durch die notwendige Encodierung kategorischer Merkmale wächst die Dimension des Datensatzes von 20 Features auf 35 Features an.

Häufigkeitsverteilung des Targets

In Tabelle 4.2 ist die Verteilung des Targets, ob ein Kredit ordnungsgemäß bedient wurde oder nicht, abzulesen. Es stehen 700 erfolgreich bezahlten Krediten 300 Krediten gegenüber, bei denen es Probleme bei der Rückzahlung gab. Damit ist die Unbalanciertheit der Klassen bei dem Kreditdatensatz sehr viel geringer ausgeprägt.

class	Anzahl
Kredit wurde ordnungsgemäß bezahlt	700
Rückzahlung nicht erfolgreich	300

Tabelle 4.2.: Ausprägungen und Häufigkeit des Targets *class* beim German Credit Dataset

Semantik der Features

Der German Credit Datensatz enthält Angaben zum Kreditnehmenden wie die Art und Dauer der Berufsausübung, des Familienstandes und Geschlechts sowie des Alters und Angaben zum Kredit, wie die Kredithöhe, Kreditlaufzeit und dem Grund für den Kredit. Eine weitere Gruppe bilden Features, die das finanzielle Verhalten des Kunden in der Vergangenheit wiedergeben wie der Kontostand des Kunden oder seinem Sparverhalten. Ein Dokument, welches zu jedem Feature eine Beschreibung dieses Features enthält, ist vorhanden.

Fehlende Werte und 0-Werte

Der German Credit Datensatz enthält keine fehlenden Werte beziehungsweise Nullwerte. Daher kann der Schritt der Behandlung dieser Auffälligkeiten entfallen.

Datenexploration

Die Abbildung 4.5 enthält eine Zusammenfassung des German Credit Datensatzes, welche für numerische Features jeweils das Minimum und Maximum, das 1. und 3. Quartil sowie Median und arithmetisches Mittel der Features abdruckt. Für kategorische Merkmale wird jeweils die im Datensatz beobachteten Häufigkeiten der Ausprägungen dieser Merkmale im Datensatz abgebildet.

Abbildung 4.5.: Statistische Zusammenfassung des German Credit Datensatz

```
                                                account_status
above 200DM or salary assignments for at least 1 year: 63
below 0DM                                                :274
between 0DM and 200DM                                    :269
no checking account                                      :394


                                                              credit_history
all credits at this bank paid back duly                              : 49
critical account/ other credits existing (not at this bank):293
delay in paying off in the past                                      : 88
existing credits paid back duly till now                             :530
no credits taken / all credit paid back duly                         : 40


              purpose                            savings
radio/television   :280   above 1000DM             : 48
new car            :234   below 100DM              :603
furniture/equipment:181   between 100DM and 500DM  :103
used car           :103   between 500DM and 1000 DM: 63
business           : 97   unknown/no savings       :183
education          : 50
(Other)            : 55
               employment                                personal_status
above 7 years        :253   female: divorced/separated/married:310
below 1 year         :172   male: divorced/separated          : 50
between 1 and 4 years:339   male: married/widowed             : 92
between 4 and 7 years:174   male: single                      :548
unemployed           : 62


     other_debtors                                                 property
co-applicant: 41   building society savings agreement/life insurance:232
guarantor   : 52   car or other not in attribute savings            :332
None        :907   real estate                                      :282
                   unknown / no property                            :154


other_installments     housing    number_of_credits
bank  :139         for free:108   1:633
none  :814         own     :713   2:333
stores: 47         rent    :179   3: 28
                                  4:  6
```

```
                                                        job
management/ self-employed/ highly qualified employee/officer:148
skilled employee / official                                  :630
unemployed/unskilled - non-resident                          : 22
unskilled - resident                                         :200


number_of_people_being_liable                                    telephone
1:845                         none                                   :596
2:155                         yes, registered under the customers name:404



foreign_worker
no : 37
yes:963

duration_of_loan     amount      installment_rate present_residence_since
Min.   : 4.0     Min.   :  250   Min.   :1.000    Min.   :1.000
1st Qu.:12.0     1st Qu.: 1366   1st Qu.:2.000    1st Qu.:2.000
Median :18.0     Median : 2320   Median :3.000    Median :3.000
Mean   :20.9     Mean   : 3271   Mean   :2.973    Mean   :2.845
3rd Qu.:24.0     3rd Qu.: 3972   3rd Qu.:4.000    3rd Qu.:4.000
Max.   :72.0     Max.   :18424   Max.   :4.000    Max.   :4.000
     age            class
Min.   :19.00   Min.   :0.0
1st Qu.:27.00   1st Qu.:0.0
Median :33.00   Median :0.0
Mean   :35.55   Mean   :0.3
3rd Qu.:42.00   3rd Qu.:1.0
Max.   :75.00   Max.   :1.0
```

Statistische Zusammenfassung des German Credit Datensatz (fortgesetzt)

4.2.3. Lending Club Datensatz

Eine weitere Datenquelle für die in dieser Masterthesis durchgeführten Untersuchungen ist der Lending Club Datensatz. Dieser Datensatz kann von der Seite `https://www.kaggle.com/wendykan/lending-club-loan-data/home` herunter geladen werden.

Semantik des Datensatzes und des Targets

Der Datensatz enthält alle auf US-amerikanischen Plattform Lending Club vermittelten Kredite im Zeitraum von 2007-2015. Auf Lending Club können Privatpersonen sich gegenseitig Geld verleihen oder leihen. Zu jedem vermitteltem Kreditinserat in diesem Zeitraum ist der Status der Rückzahlung im Datensatz enthalten.

Im Gegensatz zu den beiden zuvor beschriebenen Datensätzen besitzt das Target loan_status die in Abbildung 4.6 dargestellten Ausprägungen. Die Höhe der Balken entspricht der Anzahl der Kredite mit demjenigen Status. Die Bedeutung der einzelnen Stati können der Tabelle 4.3 entnommen werden. Für die weitere Verarbeitung wird durch Zusammenfassung von Ausprägungen von loan_status ein binäres Target gebildet, welches die Ausprägungen *good* für einen ordnungsgemäß bezahlten Kredit und *bad* für einen Kredit bei dem es zu Zahlungsausfällen gekommen ist, besitzt.

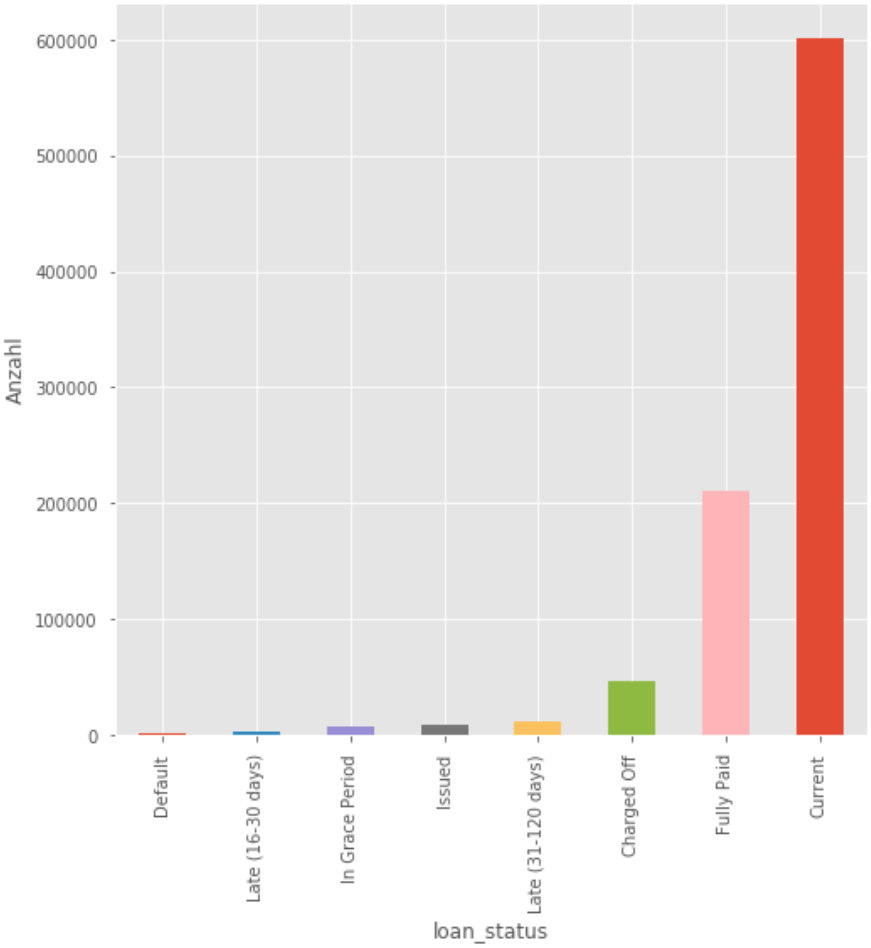


Abbildung 4.6.: Verteilung des Targets loan_status im Ursprungsdatensatz Lending Club

Status	Beschreibung
Charged Off	weitere Zahlungen werden nicht mehr erwartet, typischerweise bei Ausfall von Zahlungen über 120 Tage
Default	Kredit ausgefallen, längere Zeit keine Zahlung
Fully Paid	Kredit wurde voll zurückgezahlt
Late (16-30)	Zahlungsverzug zwischen 16 und 30 Tagen
In Grace Period	Zahlungsverzug unter 15 Tagen
Issued	Neue akzeptierte Kreditanfrage
Late (31-120)	Zahlungsverzug von 31 bis 120 Tagen
Current	Kredit wird ordnungsgemäß bedient

Tabelle 4.3.: Beschreibung der Stati von Krediten bei Lending Club (vgl. [27])

Informationen zur Quantität des Datensatzes

Der Lending Club Datensatz besitzt einen Beobachtungsumfang von 887.379 Objekten mit ursprünglich 74 Features. Damit ist der Lending Club der größte Datensatz, der in dieser Thesis untersucht wird.

Notwendige Transformationen des Datensatzes

Da der Lending Club Datensatz kategorielle Features enthält, wie beispielsweise eine Bewertung der Kreditwürdigkeit des Kreditnehmers in den Stufen von A bis F, und zumindest die topologische Datenanalyse einen rein numerischen Datensatz erfordern, ist die Enkodierung der kategoriellen Features in eine numerische Darstellung notwendig. Hierzu werden die Verfahren **Label Encoding** und **One Hot Encoding** verwendet. Besitzen die Ausprägungen eines kategoriellen Features eine Ordnung wie beispielsweise das Feature *grade*, welches die durch Lending Club vorgenommene Bewertung der Kreditwürdigkeit enthält, so werden die Ausprägungen dieses Features mittels **Label Encoding** mit fortlaufenden, natürlichen Zahlen kodiert. Bei kategoriellen Features ohne Ordnung wie der Grund für den Kredit erfolgt die Enkodierung der kategoriellen Features in eine numerische Darstellung durch das **One Hot Encoding**.

Durch die aus der explorativen Analyse erworbenen Kenntnis des Datensatzes werden die

ID-Variablen *id* und *member_id*, das Feature *url*, welches die URL der Kreditannonce enthält, für das Training des Klassifikators und die Durchführung der topologischen Datenanalyse ausgeschlossen. Zudem werden die Features *title* und *emp_title* für das Training des Klassifikators nicht verwendet. Das Feature *emp_title* enthält die Berufsbeschreibungen der Kreditnehmer, das Feature *title* entspricht dem Titel des Kreditinserates. Diese beiden kategoriellen Features besitzen sehr viele Ausprägungen und kommen daher nicht als Input für die durch die topologische Datenanalyse unterstützte Klassifikation in Frage. Die Angabe des Bundesstaats *addr_state* wird im Entscheidungsbaum verwendet, wird jedoch zur Durchführung der topologischen Datenanalyse nicht verwendet, da ein **One Hot Encoding** auf allen Bundesstaaten der USA erfolgen müsste.

Weitere Features, die nicht zur Klassifikation herangezogen werden sind Datumsangaben wie die Features *earliest_cr_line*, *issue_d* und *last_pymnt_d* welche den Monat des frühesten Kredites, die Monat der Ausgabe des aktuellen Kredites und Angaben zum Zeitpunkt der letzten Zahlung des laufenden Kredites angeben und Features wie der Bezeichnung des Arbeitsplatzes der Kreditbewerber *emp_title*, bei denen eine Vielzahl von unterschiedlichen Werten zu beobachten ist.

Ein Ausschluss dieser Features geschieht aus zwei Gründen. Zum Einen ist die Behandlung dieser Datumsangaben in einer späteren Dimensionsreduktion, dem Training von Klassifikatoren und der Durchführung der topologischen Datenanalyse schwierig. Darüber hinaus kann durch die Kombination des Ausgabedatums des Kredites und dem Datum der letzten Zahlung des Kredits ein Rückschluss auf den Status des Kredites erfolgen. Darüber hinaus können Features mit einer Vielzahl von unterschiedlichen Werten bei Verwendung bei einer Klassifikation zu Overfitting führen.

Häufigkeitsverteilung des Targets

In der Tabelle 4.4 ist abgebildet, wie viele Kredite einen deutlichen Zahlungsverzug aufweisen. Kredite, deren Zahlungsverzug unterhalb von 15 Tagen liegt, fallen hier in die Kategorie *good*. Insgesamt werden rund 93% der Kredite pünklich bedient, die restlichen 7%, weisen Zahlungsverzögerungen auf. Die Unbalanciertheit des Targets ist damit extremer als beim German Credit Datensatz, allerdings ist das Verhältnis von positiver Klasse zu negativer Klasse hier ausgeglichener als beim Kreditkartentransaktionsdatensatz.

	loan_status	loan_status(%)
good	826203	93,11%
bad	61176	6,89%

Tabelle 4.4.: Verteilung des Targets unterteilt in good und bad

Semantik der Features

Der Lending Club Datensatz enthält ursprünglich 74 Features, deren Bedeutungen in einem Data Dictionary zu finden sind, welches mit dem Datensatz herunter geladen werden kann. Der Datensatz enthält, wie der German Credit Datensatz Informationen zum Kreditnehmenden, zum Kredit. Persönliche Daten des Kreditnehmers sind beispielsweise das Jahreseinkommen oder die Anzahl der Jahre, die der Kreditnehmer berufstätig ist oder die Wohnsituation des Kreditnehmenden. Informationen die den Kredit betreffen sind zum Beispiel Volumen und Zinssatz des Kredits und die Höhe der letzten Zahlung. Eine weiteres Beispiel für eine Information zu dem Kredit ist das Feature *purpose*, welches eine Angabe des Kreditnehmers enthält, aus welchem Grund er oder sie den Kredit benötigt.

Zudem enthält der Lending Club Datensatz aus anderen Informationen berechnete Größen wie die Variable *dti*. Die Größe *dti* ist das Verhältnis der Summe aller monatlichen Raten des Kreditnehmenden zum Einkommen des Kreditnehmenden im gleichen Turnus. Der Datensatz enthält zudem Informationen im Datumsformat, wie zum Beispiel das Datum des letzten Zahlungseingangs oder der nächsten fälligen Zahlung.

Fehlende Werte und 0-Werte

Im Gegensatz zu den beiden anderen Datensätzen, enthält dieser Datensatz fehlende Werte. Während bei einigen Features nur wenige Werte fehlen, gibt es Variablen des Datensatzes, die fast ausschließlich fehlende Werte enthalten. Eine Aufstellung der 25 Features mit dem größten Anteil an fehlenden Werten enthält Abbildung 4.7. Features mit einer sehr großen Anzahl an fehlenden Werten eignen sich nicht zur Verwendung bei der Klassifikation. Aus diesem Grund werden Features, in denen bei mehr als 10% der Beobachtungen Werte fehlen, zum Training des Klassifikators nicht herangezogen. Hierdurch verringert sich die Anzahl der Variablen des Datensatz mit der in Abschnitt 4.2.3 durchgeführten Feature Selection auf 42.

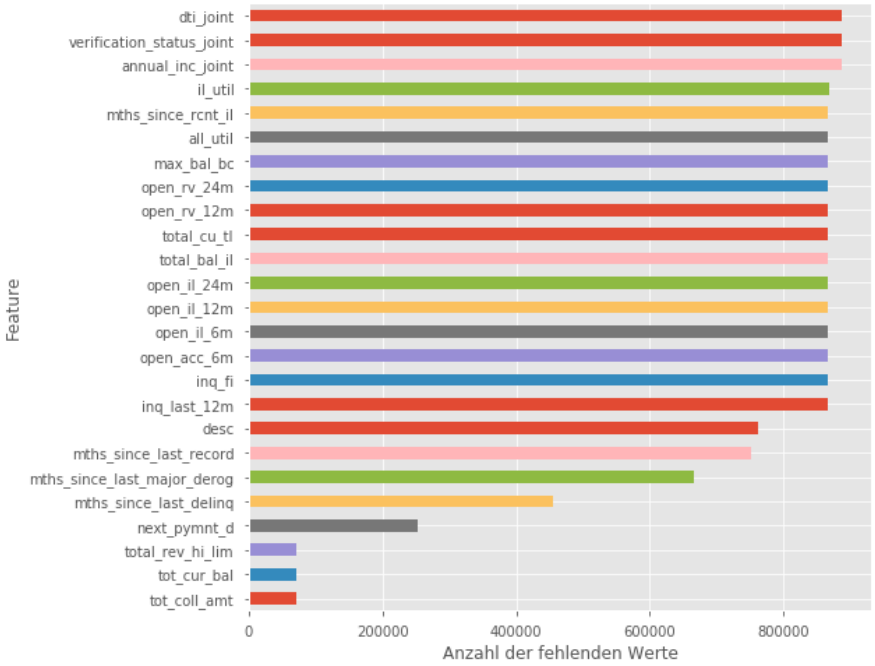


Abbildung 4.7.: Darstellung der 25 Features mit der größten Anzahl an fehlenden Werten.

Ist die Dauer der Beschäftigung unbekannt und entspricht damit einem fehlenden Wert *NaN*, so wird dieser fehlende Wert durch die neue Ausprägung *unknown* (unbekannt) ersetzt. Bei allen anderen Features, die numerischen Datentyp haben, werden fehlende Werte jeweils durch den Median der vorhandenen Werte ersetzt. Dieser als Imputation bezeichnete Vorgang erfolgt mithilfe des in 4.1 angegebenen Codeteils.

```
1  imputation_dict = {'emp_length': 'unknown',
2                     'annual_inc': np.nanmedian(clean_lending_data['annual_inc']),
3                     'collections_12_mths_ex_med': np.nanmedian(clean_lending_data['collections_12_mths_ex_med'
                          ]),
4                     'delinq_2yrs': np.nanmedian(clean_lending_data['delinq_2yrs']),
5                     'dti': np.nanmedian(clean_lending_data['dti']),
6                     'inq_last_6mths': np.nanmedian(clean_lending_data['inq_last_6mths']),
7                     'open_acc': np.nanmedian(clean_lending_data['open_acc']),
8                     'pub_rec': np.nanmedian(clean_lending_data['pub_rec']),
9                     'revol_util': np.nanmedian(clean_lending_data['revol_util']),
10                    'total_acc': np.nanmedian(clean_lending_data['total_acc']),
11                    'total_rev_hi_lim': np.nanmedian(clean_lending_data['total_rev_hi_lim']),
12                    'tot_coll_amt': np.nanmedian(clean_lending_data['tot_coll_amt']),
13                    'tot_cur_bal': np.nanmedian(clean_lending_data['tot_cur_bal']),
14                    'acc_now_delinq': np.nanmedian(clean_lending_data['acc_now_delinq'])}
15
16 imputed_lending_data = clean_lending_data.fillna(value=imputation_dict)
```

Programm 4.1: Imputation des Lending Club Datensatzes

Die Ersetzung der fehlenden Werte erfolgt durch die Methode `fillna` des Pandas Da-

taframe `clean_lending_data`. Als Parameter *value* wird ein Python Dictionary übergeben, welches als Schlüssel den Namen des Features und als Wert den Wert enthält, der zur Ersetzung der fehlenden Werte des jeweiligen Features verwendet werden soll. Die Funktion des numpy Modules `nanmedian` berechnet den Median unter Ausschluss der fehlenden Werte.

Datenexploration

In der Tabelle 4.5 ist eine statistische Zusammenfassung der Kredithöhe, des Zinssatzes, der Ratenhöhe sowie des jährlichen Einkommens abgedruckt. Der über Lending Club verliehene Betrag liegt zwischen 500 und 35000 US-Dollar. Die mittlere Kredithöhe liegt bei etwa 14755 US Dollar, der Median liegt bei 13000 US-Dollar. Der durchschnittliche Zinssatz liegt im Mittel bei 13,24%. Der geringste Zinssatz liegt bei 5,32 %, die höchste hingegen bei 28,99%. Insgesamt liegen die Zinssätze in einem sehr hohen Bereich, was vermutlich der Tatsache geschuldet ist, dass Geld über Lending Club geliehen wird, wenn die Kreditwürdigkeit für einen Kredit bei einer Bank nicht ausreichend ist. Das Feature *installment* gibt die Höhe der Rate an, die vom Kreditnehmenden bezahlt werden muss. Der Wertebereich der Ratenhöhen liegt zwischen 15,67 US-Dollar bis 1445,46 US-Dollar. Das jährliche Durchschnittseinkommen *annual_inc* der Kreditnehmenden liegt im Bereich von 0 US-Dollar bis zu 9,5 Millionen US-Dollar. Es fällt auf, dass das arithmetische Mittel des Jahreseinkommen bei rund 75000 US-Dollar liegt, der Median des Jahreseinkommens jedoch bei 65000 US-Dollar liegt. Diese Diskrepanz ist bei Einkommensverteilungen üblich. Die Begründung hierfür ist, dass das arithmetische Mittel durch hohe Einkommen verzerrt wird, der Median jedoch ein gegenüber Ausreißern unempfindliches Lagemaß darstellt.

	loan_amnt	int_rate	installment	annual_inc
count	887379,000000	887379,000000	887379,000000	8,873750e+05
mean	14755,264605	13,246740	436,717127	7,502759e+04
std	8435,455601	4,381867	244,186593	6,469830e+04
min	500,000000	5,320000	15,670000	0,000000e+00
25%	8000,000000	9,990000	260,705000	4,500000e+04
50%	13000,000000	12,990000	382,550000	6,500000e+04
75%	20000,000000	16,200000	572,600000	9,000000e+04
max	35000,000000	28,990000	1445,460000	9,500000e+06

Tabelle 4.5.: Statistische Zusammenfassung der Merkmale Kredithöhe *loan_amnt*, Zinssatz *int_rate*, Ratenzahlungshöhe *installment* und jährliches Einkommen *annual_inc* auf dem Lending Club Datensatz

Im Folgenden werden weitere explorative Untersuchungen zum Datensatz durchgeführt. Das Balkendiagramm in Abbildung 4.8 zeigt, dass der Großteil der Kreditnehmer in einem 10 Jahre oder länger dauernden Angestelltenverhältnis sind. Bei rund 40000 Kreditbewerbern ist die Dauer einer Anstellung unbekannt beziehungsweise nicht vorhanden.

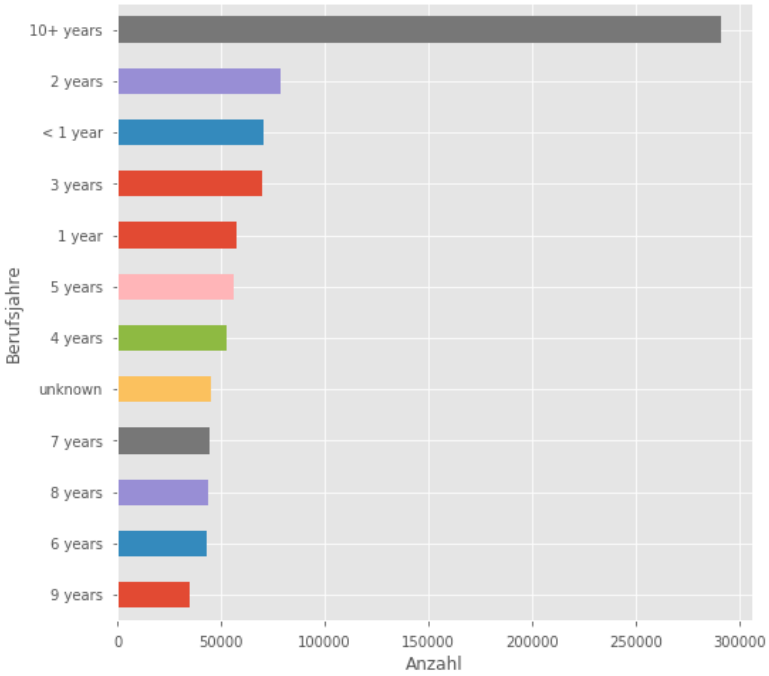


Abbildung 4.8.: Balkendiagramm der Beschäftigungsdauer in Jahren der Kreditnehmenden als kategorisches Merkmal

Interessant ist weiterhin, aus welchen Gründen Personen Kredite bei Lending Club in Anspruch nehmen. Der am Häufigsten genannte Grund für einen Kredit ist nach Abbildung 4.9 *debt_consolidation*, also die Aufnahme eines größeren Kredites, um mehrere kurzfristige vom Volumen kleine Kredite tilgen zu können. Hierdurch ergibt sich eine Ersparnis durch geringere Zinszahlungen. Die Aufnahme von Kapital zum Ausgleichen von Kreditkartenzahlungen stellt einen weiteren, häufigen Grund für die Aufnahme eines Lending Club Kredites dar. Wenige auf Lending Club inserierte Kredite dienen zur Finanzierung von Urlaub, einer Hochzeit oder zu Bildungszwecken.

Nun soll die Erwartung überprüft werden, dass das Risiko des Zahlungsausfalles von der Bewertung der Bonität *grade* abhängt. Das Feature *grade* ist eine durch Lending Club vorgenommene Beurteilung der Kreditwürdigkeit einer Person, ähnlich einer Bewertung eines Unternehmens durch Ratingagenturen. Diese Bewertung erfolgt bei Lending Club auf einer Skala von A (höchstmögliche Kreditwürdigkeit) bis G (schlechteste Kreditwürdigkeit). Diese

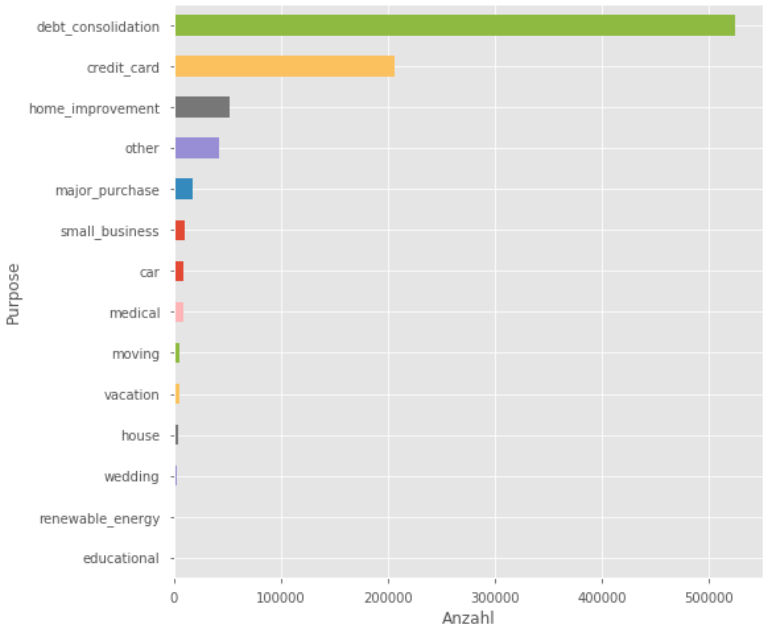


Abbildung 4.9.: Häufigkeitsverteilung des Features *purpose*

Bewertung ist durch das Features *sub_grade* feiner unterteilt. Für oben beschriebene Untersuchung ist diese feinere Unterteilung nicht relevant. Zur Messung des Risikos wird die Wahrscheinlichkeit genutzt, dass ein Kredit einer Person mit jeweiliger Bewertung ausfällt. Zur Schätzung dieser Wahrscheinlichkeit wird pro Risikoklasse ausgezählt, wie viele Kredite mit dem jeweiligen Rating ausgefallen sind und die Anzahl durch die Gesamtzahl der Personen in dieser Risikoklasse geteilt. Das Ergebnis dieser Fragestellung findet sich in der Abbildung 4.10. Durch die Betrachtung dieser Abbildung ist offenkundig, dass eine schlechteres Rating der Kreditwürdigkeit mit einer Zunahme der Kreditausfallrate einher geht. Fällt die Bonität eines Kunden in die Klasse A, so treten Zahlungsausfälle nur mit einer Wahrscheinlichkeit von unter 2,5% auf. In der Gruppe von Personen mit der Bonitätsklasse G fallen hingegen über ein Fünftel der Kredite aus. Dadurch bestätigt sich die Erwartung, dass Kredite von Kunden mit einer schlechteren Bewertung ihrer Bonität mit höherer Wahrscheinlichkeit nicht vollständig getilgt werden.

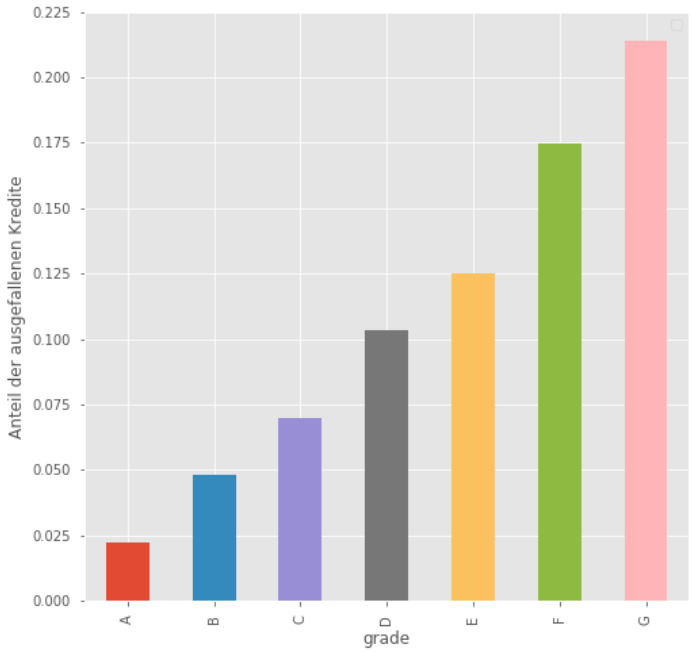


Abbildung 4.10.: Kreditausfallrate in Abhängigkeit der Bonitätsklasse *grade*

5. Ergebnisse auf Kaggle Kreditkartendatensatz

Wie in Kapitel 4 erläutert, ist die Wahl der Projektionsfunktion (auch Linse genannt) ein wichtiger Einflussfaktor der Generierung des Mapper Graphen. Daher sollen in diesem Kapitel die Ergebnisse anhand des in Teil 4.2 beschriebenen Datenbestands für unterschiedliche Projektionsfunktionen und die zwei Klassifikationsverfahren Entscheidungsbaum und logistische Regression mit Regularisierung vorgestellt werden. Eine auf den vorliegenden Datensatz angewandte Projektionsfunktion ist die lineare Diskriminanzanalyse. Darüber hinaus werden die Verfahren zur Dimensionsreduktion TSNE und UMAP ebenfalls als Linse benutzt. Eine weitere hier eingesetzte Projektionsfunktion ist eine Kombination aus dem Abnormalitätsscore eines Isolation Forest und der L_2-Norm aller Features.

Um bewerten zu können, ob mit der Hinzunahme des topologischen Features *connectedComponentId*, die die Zugehörigkeit eines Objektes zu einer zusammenhängenden Komponente im Mapper Graphen repräsentiert, eine Verbesserung der Performance der Klassifikatoren einhergeht, müssen die Klassifikatoren zunächst auf dem Trainingsdatensatz ohne topologische Informationen trainiert werden, die im Anschluss auf dem Testdatensatz hinsichtlich der Performance der Klassifikation evaluiert werden. Für das Training des Entscheidungsbaumklassifikators kommt R zum Einsatz. In diesem Kapitel werden die zunächst die Defaults der Funktion *rpart* verwendet, die den Trainingsprozess durchführt. Diese Defaults enthalten ein Pruning des Baumes mit dem Komplexitätsparameter von 0,01. Pruning bezeichnet die Entfernung von Splits, die in diesem Fall keine Erhöhung der Genauigkeit um mindestens einen Prozentpunkt verursachen. Die Abbildung 5.1 zeigt den auf dem Ursprungsdatensatz ohne topologische Informationen mit der Trainingsmenge trainierten Entscheidungsbaum. In Abbildung 5.1 und allen anderen Bäumen sind in den Knoten jeweils die vorherrschende Klasse, der Anteil der betrügerischen Transaktionen im jeweiligen Node, sowie der Anteil der Transaktionen im jeweiligen Knoten an der Gesamtzahl der Objekte im Trainingsdatensatz angegeben. Die Auswertung der Ergebnisse erfolgt, wie in Kapitel 4 bereits angeklungen ist, sowohl auf dem Trainings- und dem Testdatensatz. Der Datensatz wird stratifiziert in 60 % Trainingsdaten

und 40 % Testdaten geteilt.

Für das Training der logistischen Regressionsmodelle wird als Regularisierungsparameter $\alpha = 50$ verwendet. Als Toleranz für die Optimierung des Regressionsschätzers wurde 0,01 gewählt, die maximale Anzahl von Iterationen soll 1.000 betragen. Diese Einstellungen werden für alle Experimente auf dem Kreditkartentransaktionsdatensatz beibehalten. Der Trainings- und Vorhersageprozess der logistischen Regression erfolgt mithilfe des Moduls `statmodels` in Python.

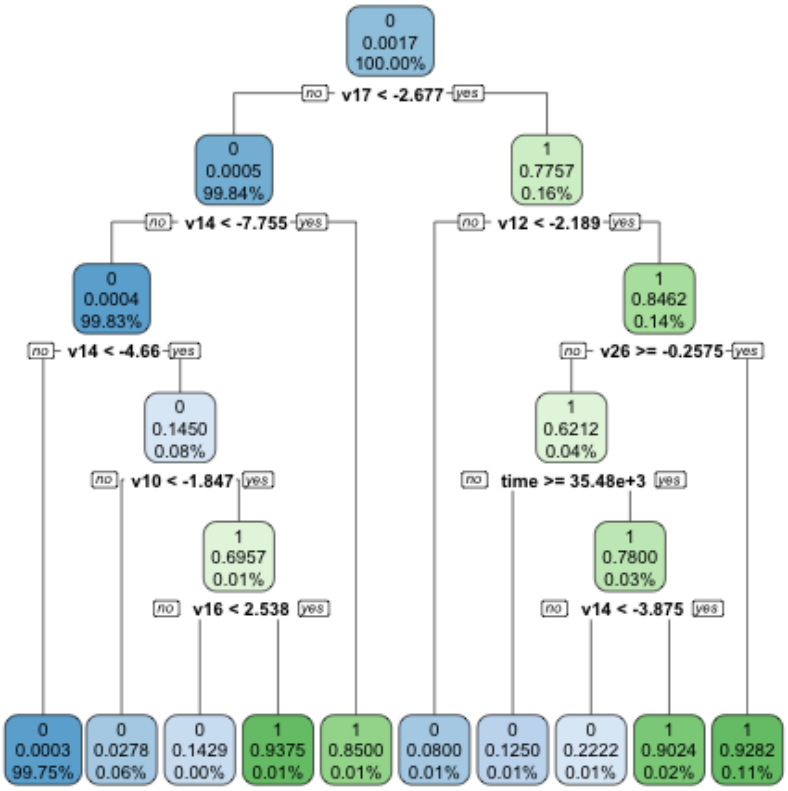


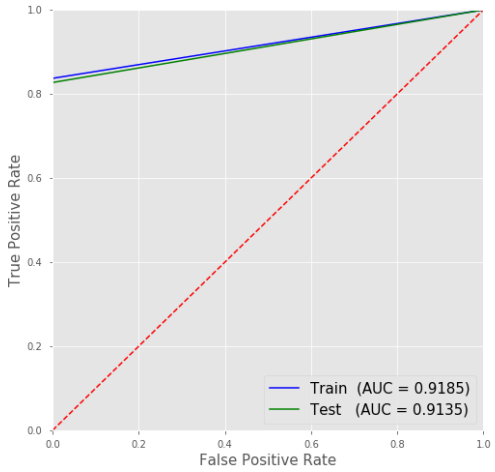
Abbildung 5.1.: Entscheidungsbaum auf dem Datensatz creditcard ohne topologische Informationen mit Defaultparametern.

Da die Standardperformancemaße wie Accuracy, f_1 Score und f_2-Score aufgrund der Unbalanciertheit des Datensatzes zur Beurteilung der Qualität des Klassifikators eine reduzierte Aussagekraft besitzen, wird zur Bewertung der Performance der Klassifikatoren die Fläche unter der Precision-Recall Kurve (AUPRC) herangezogen. Dennoch werden diese Performan-

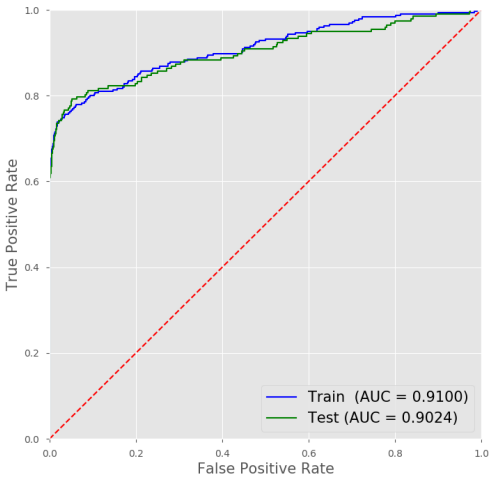
cemaße zwischen den verschiedenen Klassifikatoren, die sich wegen der unterschiedlichen Projektionsfunktionen ergeben, wiedergegeben.

Für die Klassifikatoren, welche auf dem Trainingsdatensatz ohne Kenntnis des neuen topologischen Features trainiert wurde, ist der ROC Chart in Abbildung 5.2a beziehungsweise 5.2b und das Precision Recall Diagramm in Abbildung 5.3a und 5.3b zu finden. In beiden ROC Charts findet sich als Teil der Legende die Ausgabe der Fläche unterhalb der ROC Kurve AUROC auf dem Trainings- und Testdatensatz, die durch die Funktion *auc* durch eine Approximation durch Trapeze ermittelt wurde. In den Precision-Recall Diagrammen ist an selbiger Position sowohl für den Trainings- als auch für den Testdatensatz die Fläche unterhalb der Precision-Recall Kurve AUPRC angegeben, die analog zum ROC Chart ermittelt wird. In beiden Grafiken liegt die Fläche unterhalb den Kurven zwischen 0 und 1. Das optimale Klassifikationsverfahren hat einen AUROC und AUPRC Wert von 1. Mit einem AUROC Wert von 91,35% und einem AUPRC Wert 78,06 % auf dem Testdatensatz performt der Klassifikator, der keine topologischen Informationen des Datensatzes nutzt, gut. Allerdings treten bei f_1, f_2 und AUPRC erhebliche Abweichungen zwischen Trainings- und Testdatensatz auf, die auf eine leichtere Form von Overfitting hindeuten können. In den folgenden Teilen soll nun durch die im Kapitel 4 beschriebenen Schritte für alle oben erwähnten Linsen ermittelt werden, ob sich der Klassifikator durch die Hinzunahme des topologischen Features *connectedComponentId* verbessert.

Durch den Vergleich der Precision Recall Diagramm der beiden Klassifikatoren aus den Abbildungen 5.3a und 5.3b lässt sich entnehmen, dass der Entscheidungsbaum einen sehr viel besseren Klassifikator liefert, als die logistische Regression.

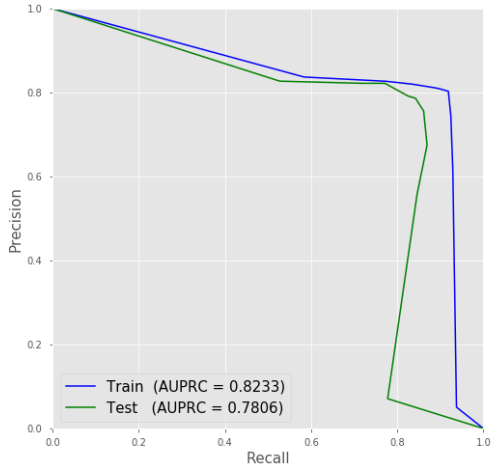


(a) Entscheidungsbaumklassifikator mit $cp = 0.01$

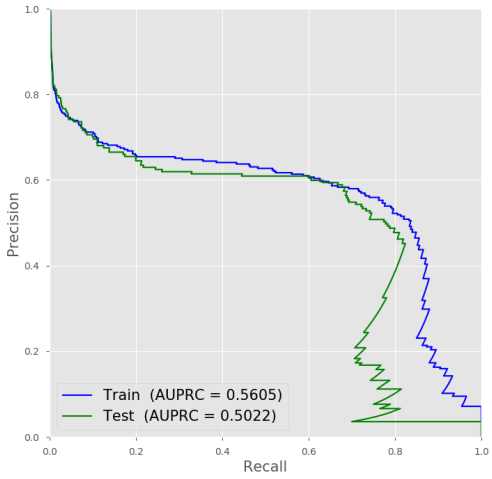


(b) Logistische Regression mit $\alpha = 50$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 5.2.: ROC Curves der Klassifikatoren



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$



(b) Logistische Regression mit $\alpha = 50$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 5.3.: Precision-Recall Diagramm der Klassifikatoren

5.1. Lineare Diskriminanzanalyse als Projektionsfunktion

In diesem Teil werden die Ergebnisse der Klassifikationen dargestellt, wenn als Linse für die topologische Datenanalyse die lineare Diskriminanzanalyse verwendet wird, die die Daten in einen Raum transformiert, in dem unterschiedliche Gruppen möglichst weit separiert sind. Es besteht die Erwartung, dass sich diese Separation im Mapper Graph niederschlägt und Nodes mit überwiegend betrügerischen Transaktionen isoliert von den Nodes mit mehrheitlich unauffälligen Zahlungen sind. Zur Berechnung der Dimensionsreduktion durch die lineare Diskriminanzanalyse kommt die Klasse `LinearDiscriminantAnalysis` des Paketes `sklearn` zum Einsatz. Als Linse werden die durch die *transform* Methode berechneten, auf eine Dimension reduzierten Daten, genutzt, die in Abbildung 5.4 dargestellt sind.

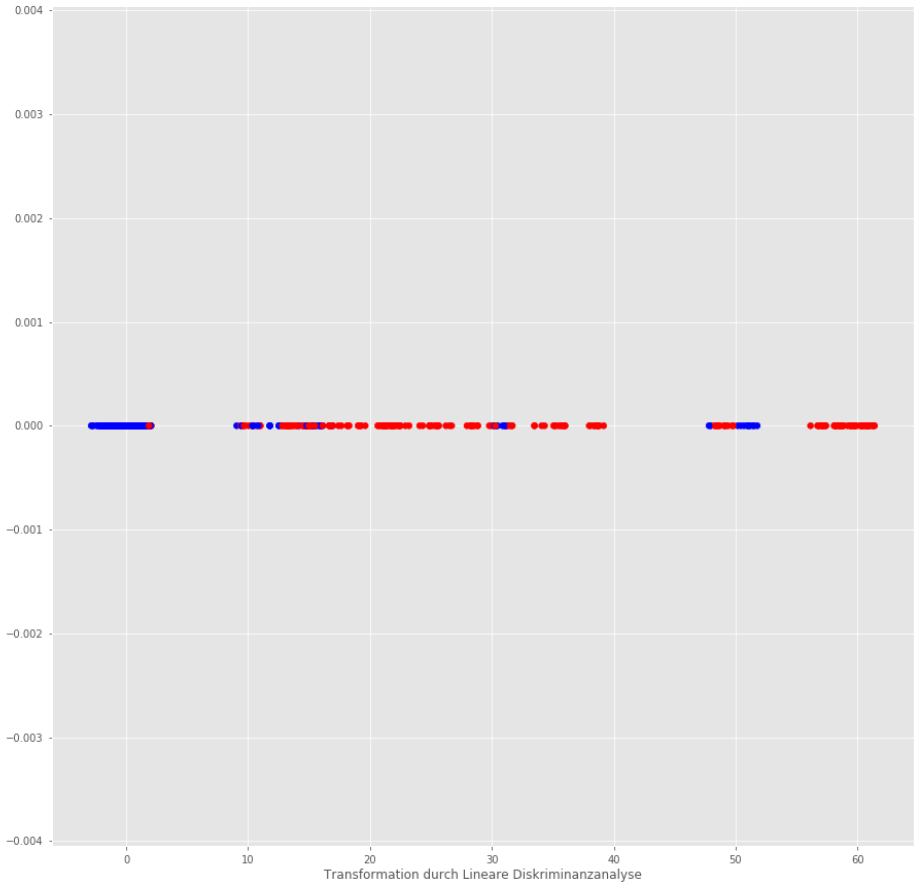


Abbildung 5.4.: Plot der Linearen Diskriminanzanalyse als Linse

In der Abbildung 5.4 lassen sich mehrere eindimensionale Inseln von normalen Transaktionen (blau) und auffälligen Transaktionen (rot) erkennen. Eine Standardisierung der Daten vor der Linsenberechnung erfolgt nicht. Wie in Kapitel 4 bereits beschrieben, wird die in 5.4 dargestellte Linse mit überlappenden Intervallen überdeckt. In diesem Fall werden dafür 50 Intervalle mit einer Überlappung von 20% verwendet. Schlussendlich wird das topologische Feature *connectedComponentId* auf dem Trainingsdatensatz erzeugt. Mit dem durch das topologische Feature angereicherten Trainingsdatensatz erfolgt das Training des in 5.5 grafisch dargestellten Entscheidungsbaumes.

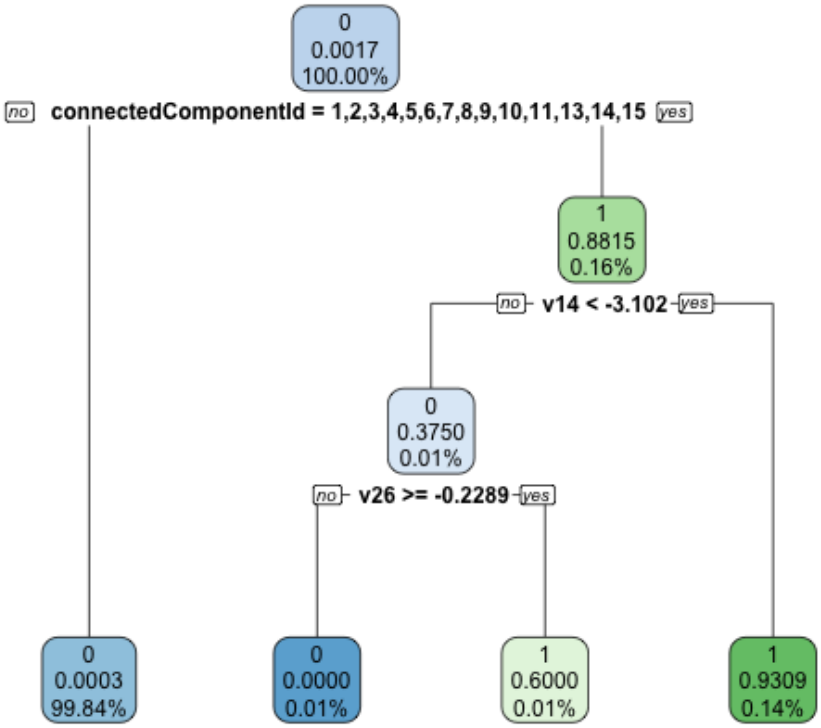
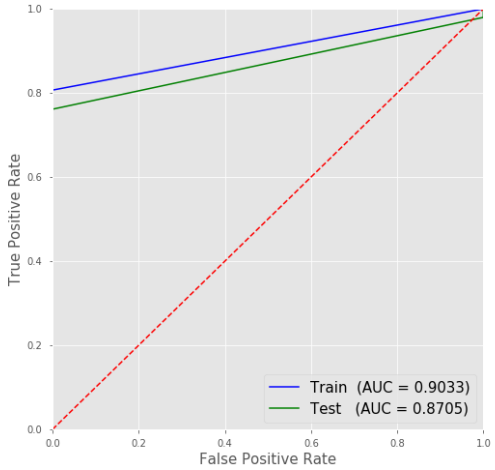


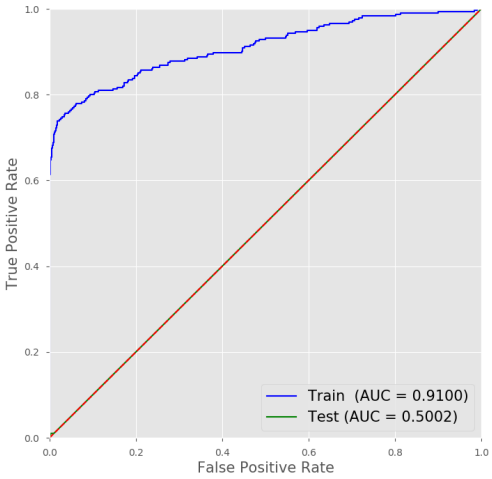
Abbildung 5.5.: Entscheidungsbaum auf dem Datensatz creditcard mit LDA Linse mit Defaultparametern.

Der Entscheidungsbaum verwendet das Feature *connectedComponentId* als Splitattribut der Wurzel. Fällt eine Transaktion in die zusammenhängende Komponente 12, so wird diese Transaktion als normal eingestuft. Liegt eine Zahlung in einer anderen zusammenhängenden Komponente ungleich 12, so erfolgt überwiegend eine Klassifikation als betrügerische Transaktion, wobei noch die Features v14 und v26 als Splitattribute verwendet werden, welche dennoch zu einer Vorhersage als normale Transaktion führen können. Bei der Betrachtung der ROC Curve aus Abbildung 5.6a fällt auf, dass sich im Vergleich zum Entscheidungsbaum ohne topologische Information der AUROC auf dem Trainingsdatensatz verschlechtert hat. Auf dem Testdatensatz ist die Verschlechterung des AUROC mit 87,05 % gegenüber 90,35% beim Entscheidungsbaum ohne Kenntnis der topologischen Struktur deutlicher.

Trotz der geringen Komplexität des Baumes wegen seiner Tiefe von 4, ist eine Zunahme der AUPRC bei Trainings- und Testdatensatz gegenüber dem Entscheidungsbaum aus 5.1 zu



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$

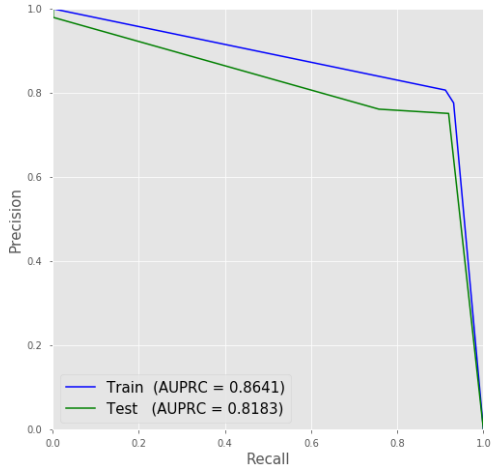


(b) Logistische Regression mit $\alpha = 50$, auto_trim_tol = 0.01, max_iter=1000

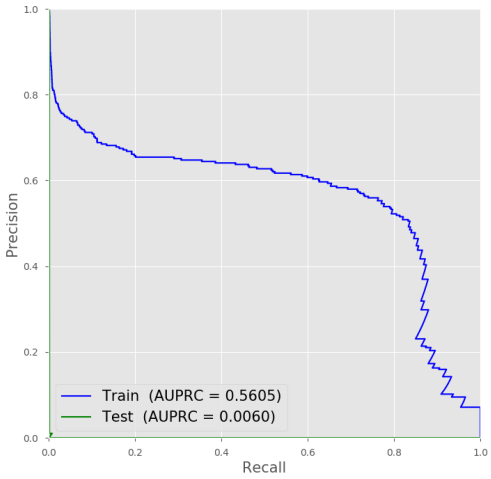
Abbildung 5.6.: ROC Curve des Entscheidungsbaumes auf Trainings- und Testdatensatz bei Verwendung der LDA als Linse

verzeichnen, wie anhand des Precision Recall Diagrammes 5.7a ersichtlich ist.

Für die logistische Regression ergibt sich die ROC Kurve aus Abbildung 5.6b. Auffällig ist hier, dass die logistische Regression auf dem Trainingsdatensatz mit einem AUROC von 91% sehr viel besser performt als auf dem Testdatensatz mit etwa 50 %. Noch deutlicher ist die Diskrepanz der Modellqualität der logistischen Regression auf Trainings- und Testdatensatz bei der Betrachtung des Precision Recall Diagrammes aus Abbildung 5.7b. Aufgrund dieser Beobachtung ist das trainierte logistische Regressionsmodell zweifelsfrei durch die Hinzunahme der topologischen Information übertrainiert.



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$



(b) Logistische Regression mit $\alpha = 50$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 5.7.: ROC Curve des Entscheidungsbaumes auf Trainings- und Testdatensatz bei Verwendung der LDA als Linse

Zusammenfassend lässt sich sagen, dass durch die Hinzunahme des mithilfe der Dimensionsreduktion nach der linearen Diskriminanzanalyse erzeugten topologischen Features eine Verbesserung der Performance des Entscheidungsbaumklassifikators verursacht wird. Bei der logistischen Regression verschlechtert sich die Performance auf dem Testdatensatz durch die topologische Datenanalyse mit der lineare Diskriminanzanalyse (LDA) Linse deutlich. Eine Begründung hierfür liefert das vorhandene Overfitting.

5.2. Kombination aus Isolation Forest und L_2-Norm als Projektionsfunktion

Als Implementierung des Isolation Forest wird die Klasse `Isolation Forest` des `sklearn` Moduls verwendet. Die Methode *decision_function* dieser Klasse ermittelt für jedes Trainingsobjekt einen Abnormalitätsscore. Dieser bildet die erste Dimension der Linse. Die zweite Dimension der Linse entsteht durch die Berechnung der L_2-Norm der Features im Trainingsdatensatz..

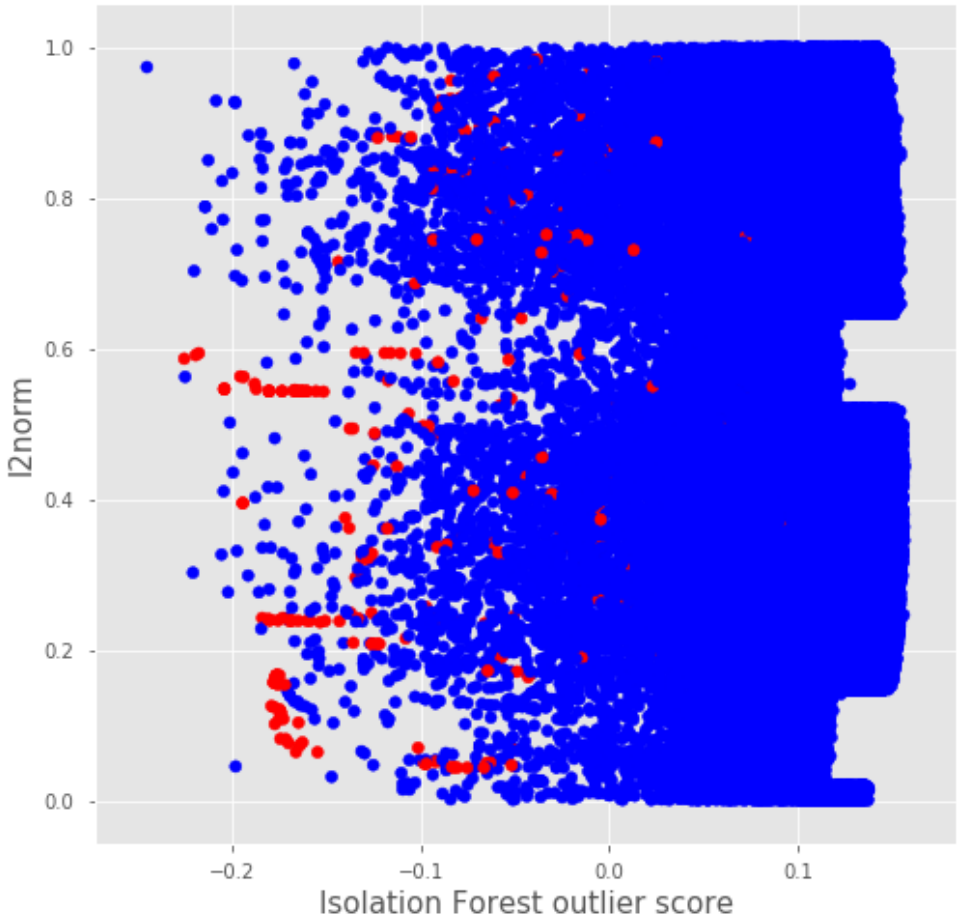


Abbildung 5.8.: Plot der Linse bestehend aus Isolation Forest Abnormalitätscore und L_2-Norm

Die auf die beschriebene Weise zusammengesetzte Linse ist in Abbildung 5.8 dargestellt. Auch hier kennzeichnen blaue Punkte normale Transaktionen und rote Punkte betrügerischen Zahlungsverkehr. Es fällt in der Abbildung 5.8 auf, dass die Punkte der auffälligen Transaktionen links gehäuft auftreten und die normalen Transaktionen überwiegend eher rechts zu finden sind. Die Überdeckung dieser Linse erfolgt hier mit Rechtecken, da für die erste Dimension 20 und für die zweite Dimension 50 Intervalle gewählt werden. Diese Einstellung erfolgt durch die Übergabe einer Liste dem Parameter *n_cubes*, deren Länge mit der Anzahl

Dimensionen übereinstimmt und in der jeder Eintrag der Anzahl der überlappenden Intervalle in dieser Dimension entspricht. Die Stärke der Überlappung liegt hier ebenfalls bei 20%.

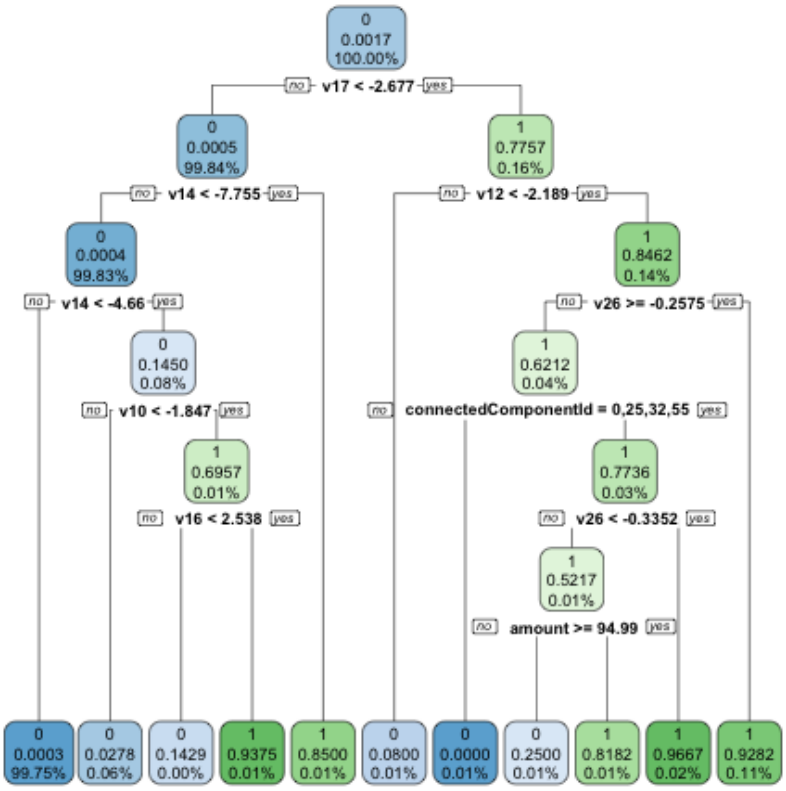
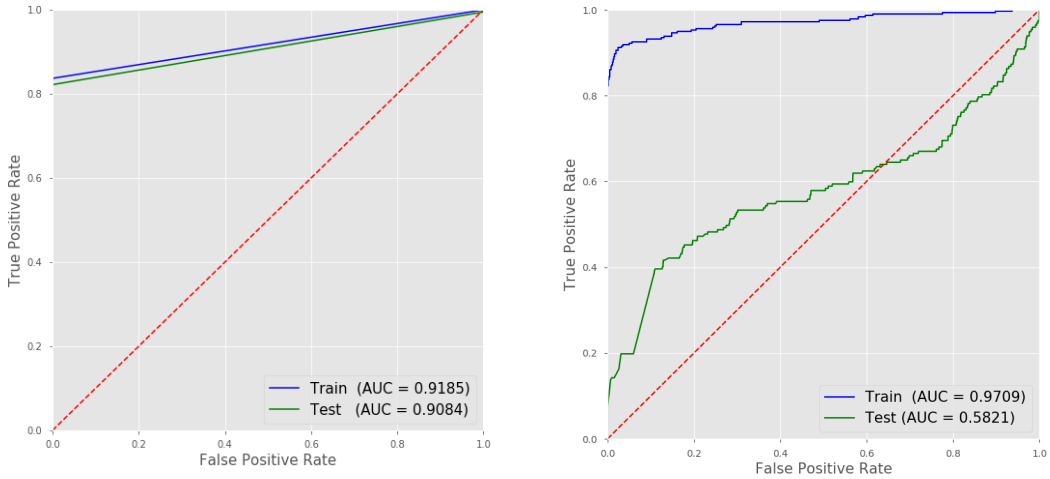


Abbildung 5.9.: Entscheidungsbaum auf dem Datensatz creditcard mit einer Kombination aus Isolation Forest und L_2-Norm als Linse mit Defaultparametern bei 20 Intervallen in der ersten Dimension und 50 Intervallen in der zweiten Dimension bei 20% Überlappung

Der auf Basis des durch die Kombination aus Isolation Forest und L_2-Norm erzeugten topologischen Features trainierte Entscheidungsbaum ist in Abbildung 5.9 visualisiert. Hier wird der Baum nicht zuerst nach dem Feature *connectedComponentId* getrennt, sondern das Feature wird erst in einer tieferen Ebene des Baumes als Splitattribut verwendet. Es fallen jedoch nur 0,04% der Transaktionen in den jeweiligen Knoten. Geprüft wird in diesem Knoten, ob Transaktionen entweder in die zusammenhängende Komponente 0,25,32 oder 55 fallen. Ist dies nicht der Fall erfolgt eine Klassifikation als normale Transaktion. Liegt ein Objekt innerhalb dieser zusammenhängenden Komponenten des Graphen, so scheint es sich um Betrug zu handeln. Ohne einen Blick auf die Visualisierung des Mapper Graph zu werfen,

kann anhand der Werte für das topologische Feature *connectedComponentId* jenseits der 50 geschlossen werden, dass der Graph in viele Komponenten zerfallen ist, welches sich bei der Betrachtung des Graphen bestätigt.



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$

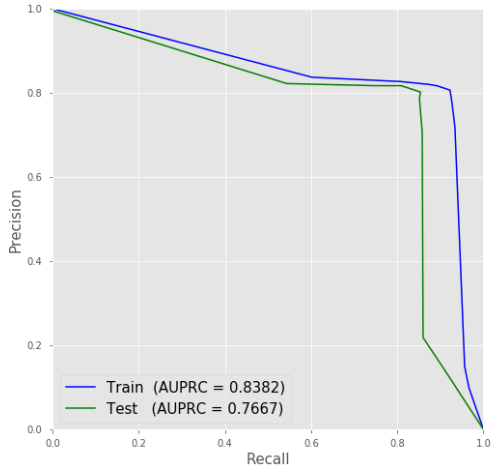
(b) Logistische Regression mit $\alpha = 50$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 5.10.: ROC Curve des Entscheidungsbaumes auf Trainings- und Testdatensatz bei Verwendung einer kombinierten Linse aus Isolation Forest und L_2-Norm

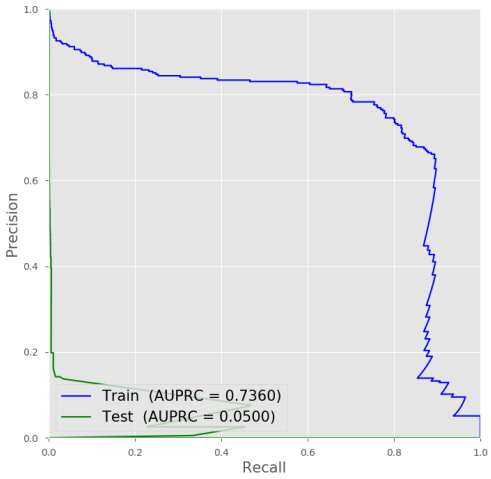
Die Güte des Entscheidungsbaumes ist anhand des ROC-Charts aus Abbildung 5.10a und der Precision Recall Kurve in Abbildung 5.11a zu beurteilen. Während die Werte für die Fläche des ROC Charts auf dem Trainings- und Testdatensatz hier näher an dem AUROC des Klassifikators ohne Verwendung topologischer Informationen liegt, ist in Abbildung 5.11a auf dem Testdatensatz eine deutliche Verschlechterung des AUPRC zu erkennen. Durch den hohen Abstand zwischen des AUPRC auf dem Trainings- und dem Testdatensatz von über 7 % ist mit Overfitting zu rechnen.

Auch die Durchführung der topologischen Datenanalyse mit der Linsenkombination aus Isolation Forest und L_2-Norm führt nicht zu einer Zunahme der Qualität der logistischen Regression, wie die Abbildungen 5.6b und 5.11b zeigen. Auch hier fällt ein deutliches Overfitting der logistischen Regression auf.

Durch den Vergleich der Fläche unterhalb der Precision Recall Kurve AUPRC, welche in Abbildung 5.11a ausgegeben ist, stellt sich heraus, dass die Verwendung einer Kombinati-



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$



(b) Logistische Regression mit $\alpha = 50$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 5.11.: Precision-Recall Diagramm auf Trainings- und Testdatensatz bei Verwendung einer kombinierten Linse aus Isolation Forest und L_2-Norm

on von Abnormalitätsscores in Verbindung mit der L_2-Norm beim vorliegenden Datensatz nicht geeignet ist, um topologische Informationen aus dem Datensatz zu extrahieren, die eine Verbesserung der Klassifikation erzielen.

5.3. TSNE als Projektionsfunktion

In diesem Teil wird nun untersucht, welche Güte die Klassifikation eines Entscheidungsbaum besitzt, wenn für die Generierung des topologischen Features connectedComponentId in Schritt (2) des Prozesses für die Generierung des topologischen Features in Abbildung 4.2 auf Seite 54 als Linse das Dimensionreduktionsverfahren TSNE verwendet wird. Alternativ ist die Vorgabe anderer Startwerte möglich, dies ist jedoch bei der in Abbildung 5.12 erfolgten Dimensionsreduktion nicht erfolgt.

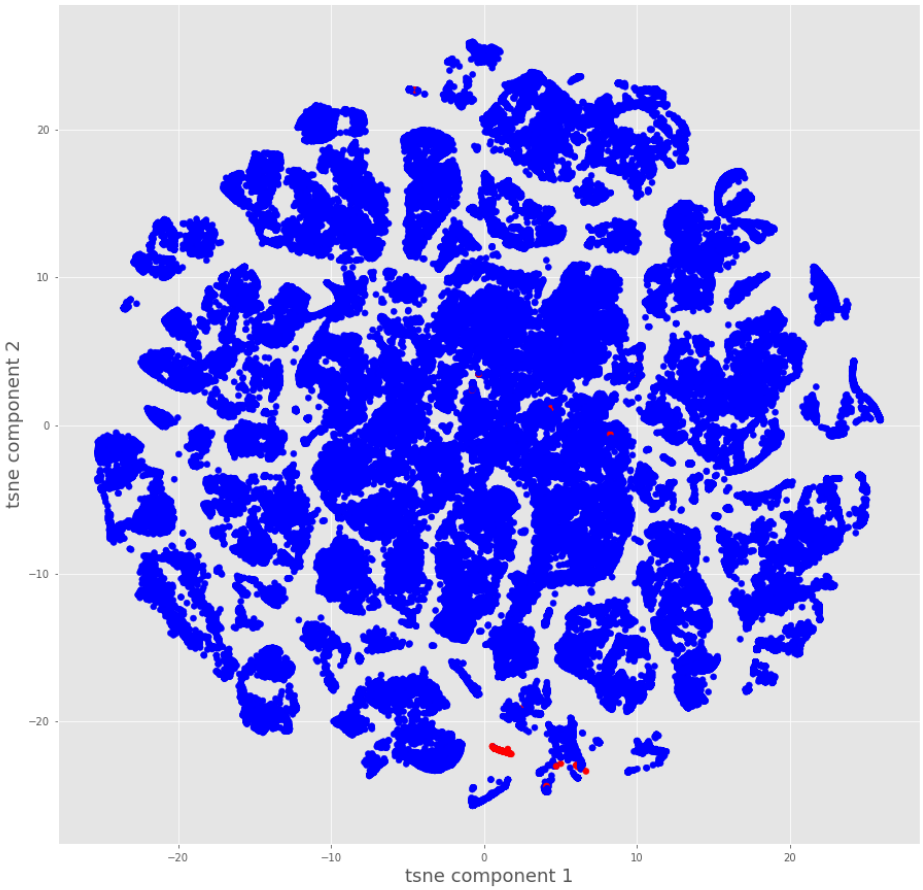


Abbildung 5.12.: Plot einer TSNE Dimensionsreduktion auf zwei Dimensionen als Linse mit Perplexität 100

Für die Grafik in Abbildung 5.12 wurde eine Perplexität von 100 gewählt. Im Scatterplot in Abbildung 5.12 sind ebenfalls normale Transaktionen als blaue Punkte und betrügerische Transaktionen als rote Punkte dargestellt. Die Zuordnung des Targets zu den Punkten im Scatterplot der Dimensionsreduktion erfolgt über die ID des Objektes im Trainingsdatensatz. In der unteren Mitte des Scatterplots befinden sich in einem Loch zwischen normalen Transaktionen eine „Insel“ mit auffälligen, in rot dargestellten Transaktionen. Diese weisen in der ersten Komponente einen Wert von etwas größer als 0 und in der zweiten Komponente einen

Wert von etwa -20 auf. Rechts daneben sind weitere Betrugsfälle zu finden, die jedoch sehr nah an normalen Transaktionen liegen.

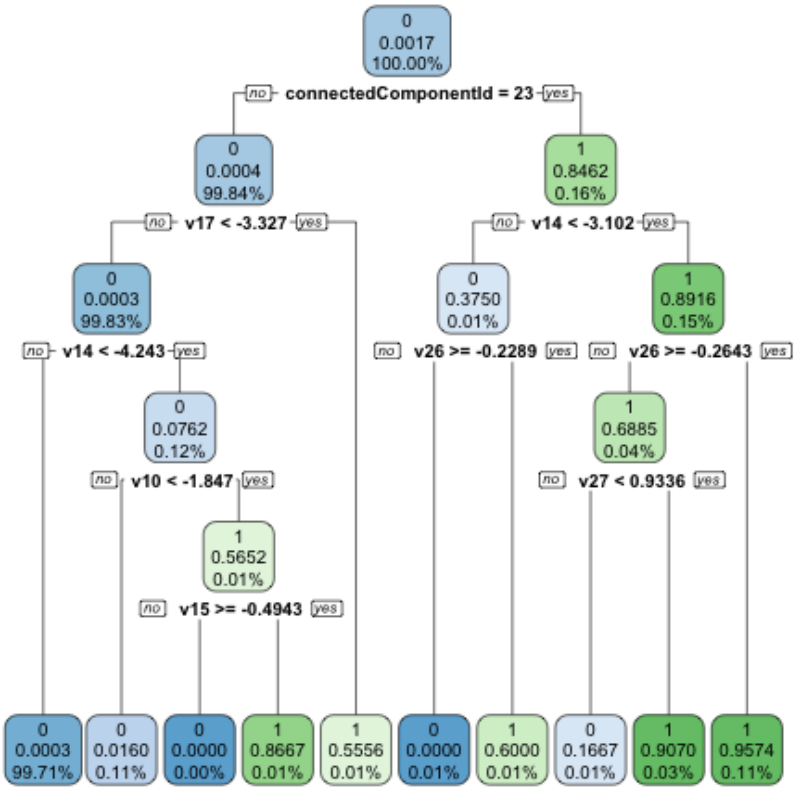
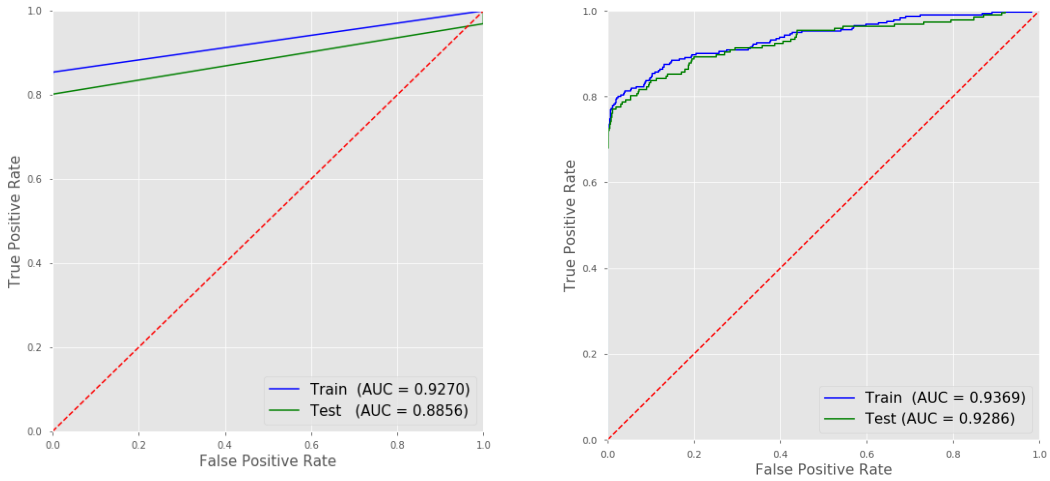


Abbildung 5.13.: Entscheidungsbaum auf dem Datensatz creditcard mit TSNE Linse mit Defaultparametern.

Zur Überdeckung der Linse wurde ein Gitter von jeweils 50 überlappenden Intervallen pro Dimension verwendet. Somit wird die Linse von insgesamt 2500 Intervallen überdeckt, die Stärke der Überlappung ist 20 %. Durch die in Kapitel 4 beschriebenen Schritte, ergibt sich der Entscheidungsbaum aus Abbildung 5.13. Das erste Splitattribut des Baumes in Abbildung 5.13 ist das topologische Feature *connectedComponentId*. Fällt eine Transaktion in die zusammenhängende Komponente 23, so handelt es sich hierbei mit hoher Wahrscheinlichkeit um Fraud, wie ein Vergleich der in der zweiten Zeile eines jeden Knotens des Entscheidungsbaumes angegebenen Wahrscheinlichkeiten für die positive Klasse in diesem Knoten, zeigt. Allerdings können Beobachtungen, bei denen die *connectedComponentId* 23 ist, dennoch als normale Transaktionen bewertet werden, wenn die im Baum ablesbaren Bedingungen zutreffen.

Liegt eine Transaktion in der zusammenhängenden Komponente mit der ID 23, so ist diese Beobachtung höchstwahrscheinlich ein Betrug.



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$

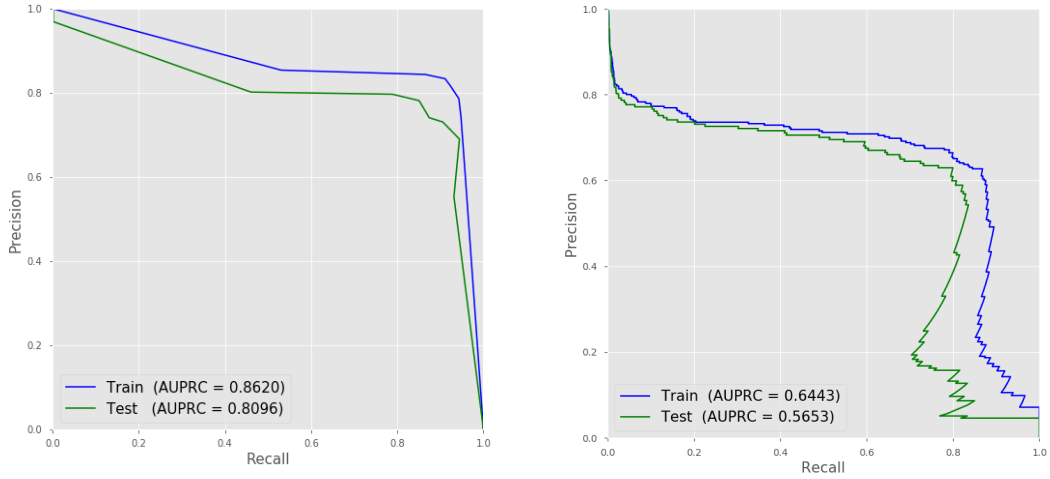
(b) Logistische Regression mit $\alpha = 50$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 5.14.: ROC Curve des Entscheidungsbaumes auf Trainings- und Testdatensatz bei Verwendung der TSNE Linse

Bei der Betrachtung des in Abbildung 5.14a dargestellten ROC Charts stellt man fest, dass sich im Vergleich zum Klassifikator, der keine topologischen Informationen nutzt, zwar der AUROC Wert auf dem Trainingsdatensatz verbessert hat, dieser Performancegewinn jedoch nicht auf dem Testdatensatz erkennbar ist. Hier liegt der AUROC Wert mit 88,56% gegenüber 90,35% beim Klassifikator ohne Durchführung einer topologischen Datenanalyse etwas niedriger.

Durch den Vergleich der ROC Charts des Entscheidungsbaumklassifikators aus Abbildung 5.14a und der logistischen Regression aus Abbildung 5.14b lässt sich erkennen, dass die Performance der logistischen Regression die Performance des Entscheidungsbaumes übertrifft. Zudem differiert die Fläche unterhalb dem ROC Chart der logistischen Regression aus 5.14b auf dem Trainings- und dem Testdatensatz nur gering.

Betrachtet man allerdings die Precision Recall Kurve in Abbildung 5.15a so stellt sich heraus, dass die Fläche unterhalb der Precision Recall Kurve sowohl auf dem Trainings- als auch auf dem Testdatensatz höher sind. Auffällig ist allerdings der deutliche Unterschied von über



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$

(b) Logistische Regression mit $\alpha = 50$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 5.15.: Precision-Recall Diagramm auf Trainings- und Testdatensatz bei Verwendung der TSNE Linse

5 Prozentpunkten zwischen in-sample und out-of-sample Performance, der auf Overfitting des Klassifikators hindeutet.

Für die logistische Regression ergibt sich, wie in der Abbildung 5.15b dargestellt, ebenfalls eine Zunahme des AUPRC Wertes um ca. 5 Prozentpunkte. Allerdings übertrifft der Entscheidungsbaumklassifikator die logistische Regression in puncto Performance auf dem Trainings- und dem Testdatensatz.

5.4. UMAP als Projektionsfunktion

Um zu untersuchen, ob eine weitere Verbesserung der Performance des Klassifikators, der topologische Information nutzt, möglich ist werden die Schritte des Gesamtprozesses aus Prozessdiagramm 4.1 auf Seite 53 wiederholt durchgeführt. In diesem Fall kommt als Linse eine UMAP Dimensionsreduktion zum Einsatz. Der Vorteil von UMAP gegenüber TSNE besteht in der geringeren Komplexität des Algorithmus. Während TSNE für die Reduktion des Trainingsdatensatzes auf zwei Dimensionen circa 20 Minuten braucht, erledigt UMAP diese Aufgabe - je nach Wahl der Parameter - in einer Zeit von ca. 3 bis 13 Minuten. Die Resultate

sind häufig vergleichbar zu denen von TSNE, in einigen Fällen besser. Ein weiterer Vorteil ist die Möglichkeit eine überwachte Dimensionsreduktion durchzuführen, indem das Target als zusätzlicher Parameter der *fit* Funktion übergeben wird. Zur Berechnung der Dimensionsreduktion nach UMAP wird aus dem Paket `umap-learn` die Klasse `UMAP` genutzt. In diesem Versuch wird die Dimensionsreduktion nach UMAP ohne Einbezug des Targets durchgeführt.

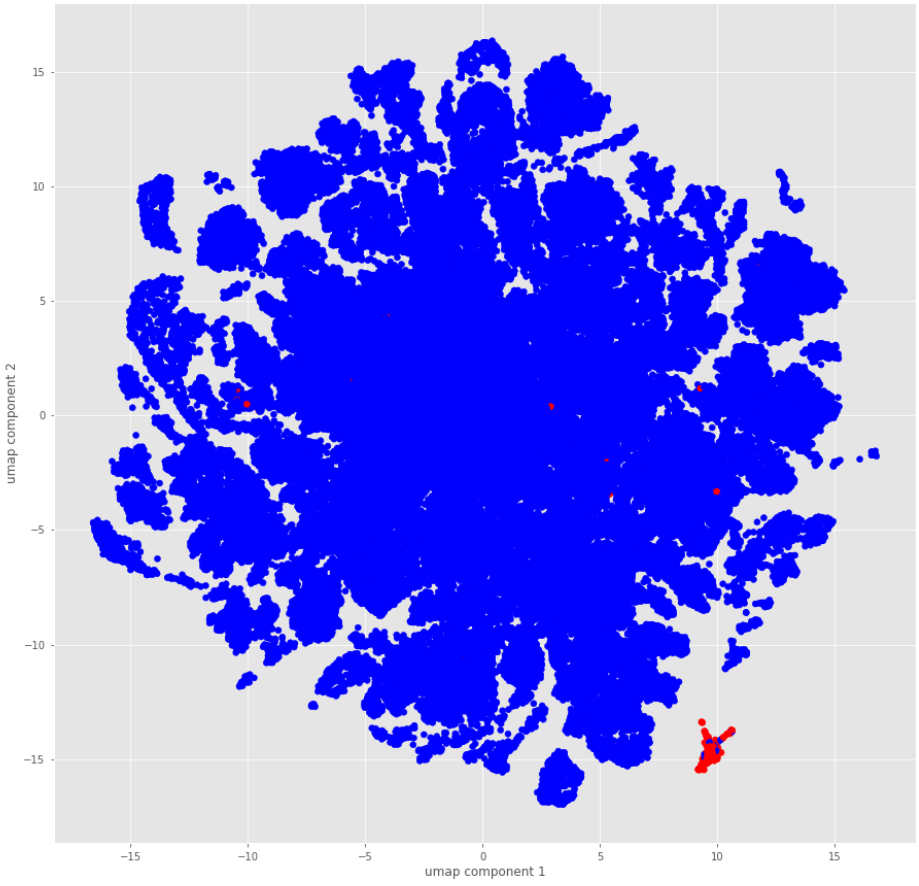


Abbildung 5.16.: Plot einer UMAP Dimensionsreduktion als Linse

Die im Scatterplot in der Abbildung 5.16 dargestellte Projektion des Trainingsdatensatzes verwendet 70 Nachbarn. Der Parameter *min_dist* wird auf 0,8 gesetzt und die *cosine* Metrik wird ausgewählt. Bei der Betrachtung der Abbildung 5.16 ist leicht zu erkennen, dass die

betrügerischen Transaktionen von den normalen weitestgehend getrennt sind. Die „Insel“ der auffälligen Transaktionen liegt in der Gegend um den Punkt (10,-15) und hat eine Form, die an den Buchstaben Y erinnert. Die Linse aus 5.16 wird durch 40 überlappende Intervalle je Dimension mit einer Stärke der Überlappung von 20% überdeckt.

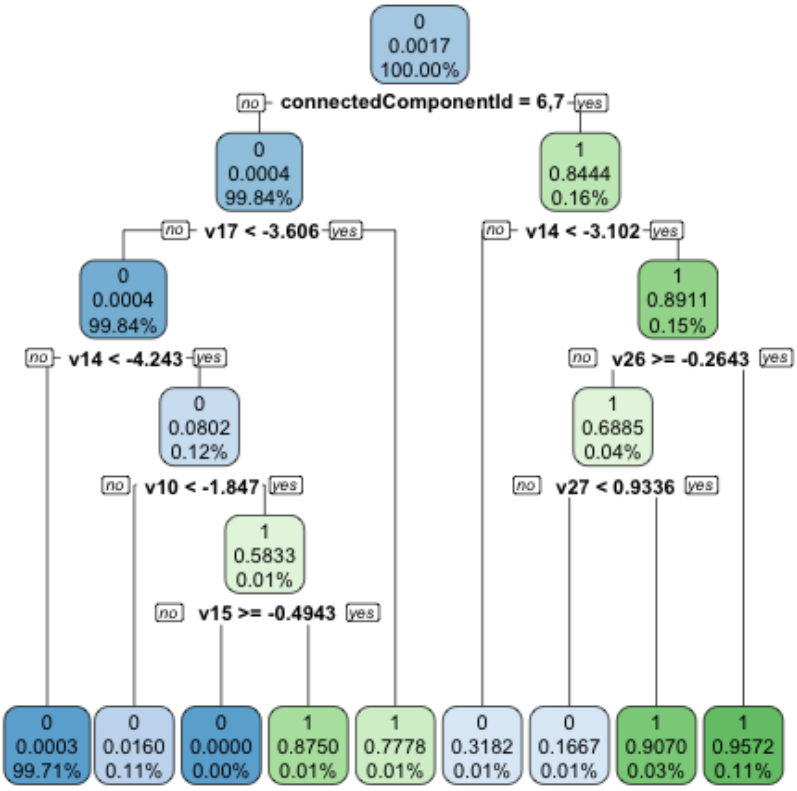
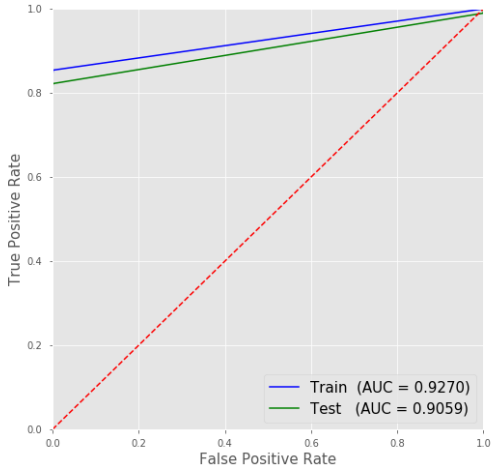
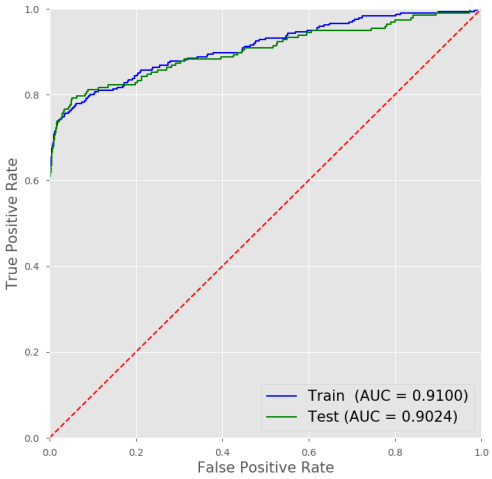


Abbildung 5.17.: Entscheidungsbaum auf dem Datensatz creditcard mit einer UMAP Linse mit Defaultparametern.

Abbildung 5.17 zeigt den Entscheidungsbaum, der auf dem Trainingsdatensatz trainiert wird, der durch die Durchführung einer topologischen Datenanalyse mit einer Dimensionsreduktion vom Typ UMAP um ein topologisches Feature angereichert ist. Der Wurzelknoten des Entscheidungsbaumes wird zunächst durch das topologische Feature *connectedComponentId* in zwei Knoten geteilt. Der rechte Knoten enthält überwiegend betrügerische Transaktionen, der linke Knoten enthält diese nur vereinzelt. Der Entscheidungsbaum in Abbildung 5.17 lässt den Schluss zu, dass sich in der zusammenhängenden Komponente 6 und 7 überwiegend Betrugsfälle befinden.



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$



(b) Logistische Regression mit $\alpha = 50$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 5.18.: ROC Curve des Entscheidungsbaumes auf Trainings- und Testdatensatz bei Verwendung einer UMAP Dimensionsreduktion als Linse

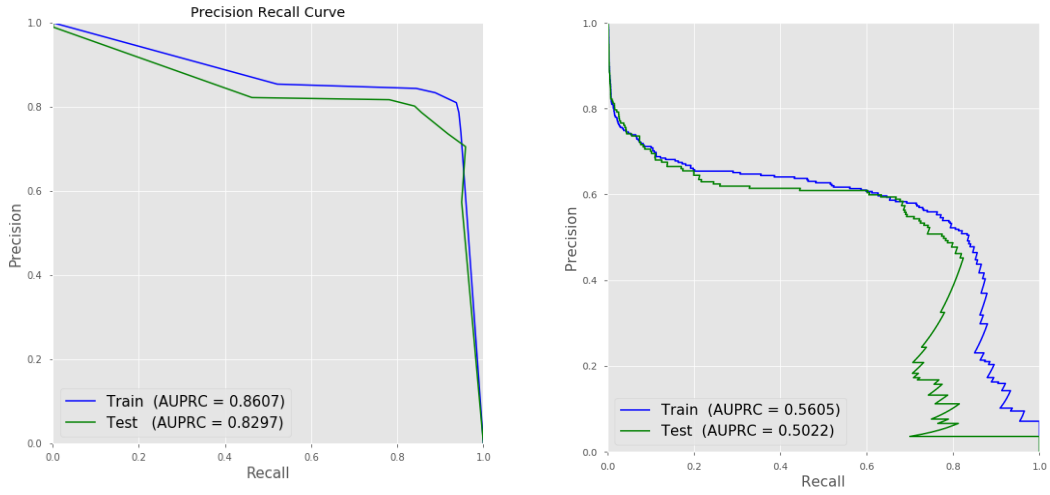
Bei der Betrachtung des in Abbildung 5.18a dargestellten Plots der ROC Kurve stellt sich heraus, dass hier im Vergleich zum ROC Chart aus Abbildung 5.2a des Klassifikators ohne topologische Informationen der AUROC Wert auf dem Trainingsdatensatz circa 1,3 Prozentpunkte höher liegt, dieser Gewinn an Performance jedoch auf dem Testdatensatz nicht gehalten werden kann. Allerdings ist der Verlust des AUROC im Gegensatz zum Entscheidungsbaum in Abbildung 5.1 eher gering. Auffällig ist, im Gegensatz zu den anderen Klassifikatoren, die auf einem Datensatz mit topologischem Feature trainiert wurden, die geringere Abweichung des AUROC zwischen Trainings- und Testset.

Bei der Betrachtung des ROC Charts der logistischen Regression aus Abbildung 5.18b fällt auf, dass der Performanceunterschied zwischen Trainings- und Testdatensatz gering ist. Vergleicht man die Performance auf dem Testdatensatz der beiden Klassifikatoren, so besitzen beide eine ähnlichen AUROC Wert. Der AUROC Wert des Entscheidungsbaumklassifikators ist jedoch minimal höher.

Dieser Eindruck bestätigt sich im Precision-Recall Diagramm in Abbildung 5.19a. Die blaue und die grüne Kurve liegen nah beieinander. Die blaue Kurve entsteht durch die Erzeugung der Precision Recall Kurve auf dem Trainingsdatensatz, die grüne auf dem Testdatensatz.

Das Precision Recall Diagramm 5.19a und die einbeschriebene Fläche unterhalb der Kurven zeigt, dass die Hinzunahme des topologischen Features, welches mithilfe der UMAP Linse erzeugt wurde, die Güte des Entscheidungsbaums verbessert. Die Zunahme des AUPRC Werts beträgt rund 3,7 Prozentpunkte auf dem Trainingsdatensatz und 4,91 Prozentpunkte auf dem Testdatensatz. Auch ist der Abstand zwischen AUPRC des Trainingsdatensatzes zum AUPRC auf dem Testdatensatzes geringer als bei den anderen Entscheidungsbäumen.

Während die topologische Datenanalyse in Verbindung mit der UMAP Linse wie oben beschreiben eine Verbesserung des AUPRC des Entscheidungsbaumklassifikators verursacht, so ergibt sich, wie in Abbildung 5.19b dargestellt, keine Zunahme der Qualität der Klassifikation bei der Nutzung der logistischen Regression mit LASSO. Darüber hinaus existiert in Abbildung 5.19b eine Abweichung von 6 Prozentpunkten zwischen dem AUPRC auf dem Trainings- und auf dem Testdatensatz.



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$

(b) Logistische Regression mit $\alpha = 50$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 5.19.: Precision-Recall Diagramm auf Trainings- und Testdatensatz bei Verwendung einer UMAP Dimensionsreduktion als Linse

Die Ergebnisse dieses Teils zeigen wiederum, dass durch die Hinzunahme eines Features, welches Informationen über die Topologie des Datensatzes enthält, die Performance eines Entscheidungsbaumklassifikators erhöht werden kann. Allerdings unterscheiden sich die Linsen, bei denen eine Verbesserung der Performance verzeichnet werden kann, je nach Art des verwendeten Klassifikators.

Zur Erleichterung des Verständnis der Beziehung zwischen Linse und durch die topologi-

sche Datenanalyse ermittelte Connected Component ID (CCID) ist in Abbildung 5.20 ein Scatterplot der UMAP Linse abgebildet. Die Einfärbung der Punkte geschieht durch die connectedComponentId auf dem Trainingsdatensatz. Durch den Vergleich des Scatterplots aus Abbildung 5.16 mit dem in Abbildung 5.20 lässt sich erkennen, dass die Fraud-Insel in Abbildung 5.16 eine Zusammenhangskomponente in der Darstellung in Abbildung 5.20 bildet.

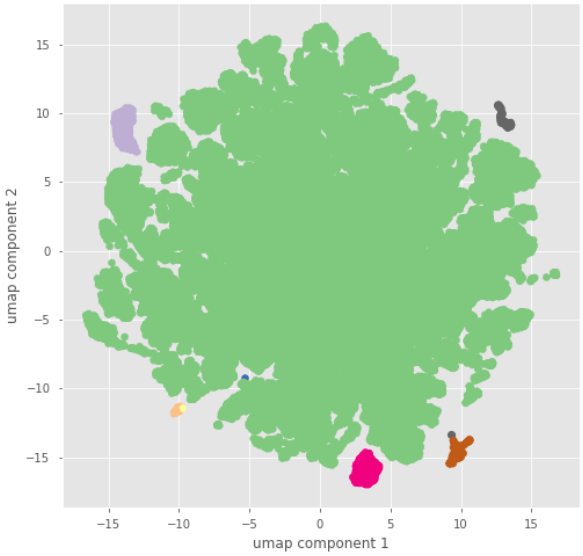


Abbildung 5.20.: Plot der UMAP Dimensionsreduktion mit Einfärbung nach connectedComponentId

5.5. Vergleich der Ergebnisse

In diesem Teil sollen die Ergebnisse dieses Kapitels zusammengefasst werden und ein Vergleich der Qualität der Klassifikatoren vorgenommen werden. Da, wie bereits erwähnt, der vorliegende Datensatz sehr unbalanciert ist, wird für die endgültige Bewertung die Fläche unterhalb der Precision Recall Kurve verwendet. Mit Ausnahme der Linse bestehend aus einer Kombination aus Anomalitätsscores eines Isolation Forests und der L_2-Norm tritt bei jeder Linse eine mehr oder weniger große Verbesserung dieses Performancemaßes beim Entscheidungsbaumklassifikator ein. Bei der Bestimmung der Klassifikatoren wurden die Parameter wie die Anzahl der Intervalle für die Überdeckung so gewählt, dass ein möglichst guter Entscheidungsbaumklassifikator entsteht. Durch den Vergleich der AUPRC Werte aus Tabelle 5.1 auf dem Testdatensatz, stellt sich heraus, dass der Gebrauch der UMAP Dimensionsreduktion

als Linse für den in Abbildung 4.2 beschriebenen Prozess zur Erzeugung eines topologischen Features, die Qualität des Klassifikators im Vergleich zum Klassifikator ohne Rückgriff auf topologische Informationen am meisten steigert.

Gütemaß	ohne topologische Features	LDA Linse	Isolation Forest und L_2-Norm	TSNE	UMAP
AUPRC	78,06%	81,83%	76,67%	80,96%	**82,97%**
AUROC	**91,35%**	87,05%	90,84%	88,56%	90,59%
f_1	80,54%	75,95%	**82,72%**	81,48%	82,01%
f_2	77,52%	76,07%	**81,19%**	79,46%	79,98%

Tabelle 5.1.: Tabelle ausgewählter Gütemaße des Entscheidungsbaumklassifikators auf dem Testdatensatz

In der Tabelle 5.2 finden sich die Werte der Gütemaße AUPRC, AUROC, f_1-Score sowie f_2-Score der logistischen Regression. Es fällt auf, dass die topologische Datenanalyse die Qualität der Vorhersage stark verringern kann. Wie im Verlaufe dieses Kapitels beschrieben, tritt bei der logistischen Regression bei den Linsen LDA und Isolation Forest und L_2-Norm besonders stark ausgeprägtes Overfitting auf, welches sich durch die beobachteten starken Differenzen der Performancemaße auf dem Trainings- und dem Testdatensatz bestätigt.

Gütemaß	ohne topologische Features	LDA Linse	Isolation Forest und L_2-Norm	TSNE	UMAP
AUPRC	50,22%	0,6%	5,0%	**56,53%**	50,22%
AUROC	90,24%	50,02%	58,21%	**92,86%**	90,24%
f_1	60,60%	0,0%	13,1%	**70,45%**	60,60%
f_2	60,79%	0,0 %	9,15%	**65,75%**	60,79%

Tabelle 5.2.: Tabelle ausgewählter Gütemaße der logistischen Regression auf dem Testdatensatz

Ein Blick auf die Tabelle 5.1 zeigt, dass der auf die UMAP Dimensionsreduktion basierende Entscheidungsbaum in Abbildung 5.17 nicht bezüglich allen Performancemaßen der Beste ist. Betrachtet man den ROC Index so ist durch die Durchführung der topologischen Datenanalyse keine Verbesserung des Klassifikators eingetreten. Bei der Linsenkombination bestehend aus Isolation Forest und L_2-Norm weist der resultierende Klassifikator den größten f_1 und f_2 Wert auf.

Bei der logistischen Regression ergeben sich nach allen in der Tabelle 5.2 aufgeführten Performancemaßen bei der Nutzung des Dimensionsreduktionsalgorithmus TSNE als Linse zur

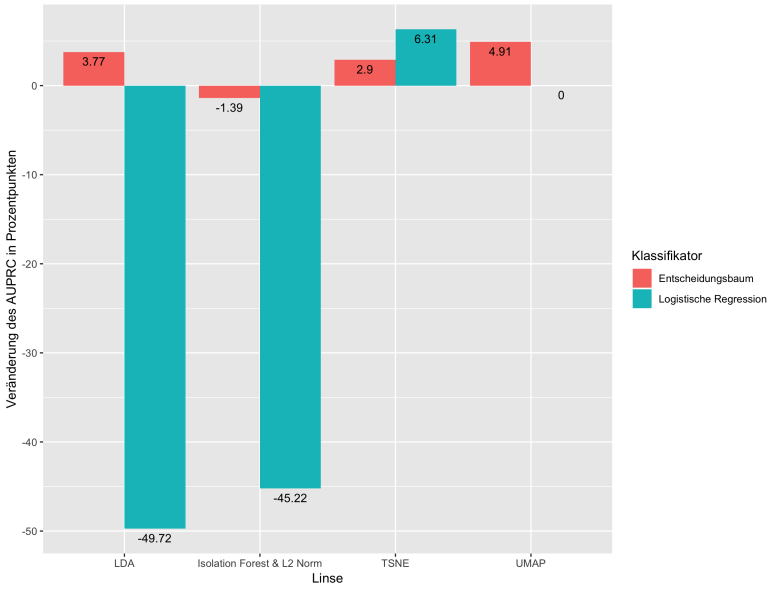


Abbildung 5.21.: Balkendiagramm der Veränderung des AUPRC auf dem Kreditkartentransaktionsdatensatz durch die topologische Datenanalyse. Die Veränderung des AUPRC wurde auf dem Testdatensatz gemessen.

Durchführung der topologischen Datenanalyse die beste Performance. Nichtsdestotrotz liefert der Entscheidungsbaum im Vergleich zur logistischen Regression auf dem Kreditkartentransaktionsdatensatz bessere Klassifikationsergebnisse.

Die Abbildung 5.21 stellt die Veränderung der Fläche unterhalb der Precision Recall Kurve auf dem Testdatensatz grafisch dar. Auffällig ist, dass die Performance der logistischen Regression bei der Nutzung der Linsen LDA und der Kombination aus Isolation Forest und L_2 Norm um mehr als 40 Prozentpunkte abnimmt. Andererseits verursacht die TSNE Linse den stärksten Anstieg des AUPRC der logistischen Regression. Der Entscheidungsbaumklassifikator kann durch die Linsen LDA, TSNE und UMAP um mehr als 2 Prozentpunkte verbessert werden. Einzig bei der Verwendung der Linsenkombination aus Isolation Forest und L_2 Norm tritt beim Entscheidungsbaum ein leichter Rückgang der Performance auf.

6. Ergebnisse auf German Credit Data Set des UCI ML Repository

Als zweiten Datensatz zur Überprüfung der Hypothese, dass die Hinzunahme eines durch die topologische Datenanalyse generierte Features zur Verbesserung der Performance eines Klassifikators führt, wird der German Credit Data Set aus dem UCI Machine Learning Repository verwendet. Einen kleinen Überblick über die Datengrundlage bietet Teilabschnitt 4.2.2. Aufgrund der kategorischen Merkmale wird zur Durchführung der topologischen Datenanalyse der Datensatz wie bereits in 4.2.2 beschrieben in eine numerische Darstellung überführt. Zur Durchführung der Klassifikation kommt auch hier das R Paket `rpart` zum Einsatz, welches den Gebrauch von kategorischen Features bei der Modellierung ermöglicht, wenn diese als Faktoren codiert sind. Die Durchführung der Experimente richtet sich, wie auch im Kapitel 5 nach den Prozessschritten aus Abbildung 4.1 und 4.2. Bei allen Experimenten mit dem German Credit Data Set wurde für den Komplexitätsparameter cp der Wert 0.015 verwendet, da der Defaultwert 0,01 zu einem Klassifikator führt, der deutliche Anzeichen von Overfitting aufweist.

Das Training der logistischen Regression erfolgt unter der Wahl der Werte $\alpha = 50$, *auto_trim_tol*=0.01 und *max_iter*=1000. Die Wahl dieser Werte erfolgte auf Basis der Klassifikation ohne Unterstützung durch die topologische Datenanalyse.

Wegen des kleineren Datensatzes wird der Datensatz stratifiziert in 80% Training und 20% Test unterteilt, um die Performance auf einem dem Klassifikator unbekannten Teil der Daten zu evaluieren.

Ohne die Unterstützung der topologischen Datenanalyse stellt sich der Entscheidungsbaum wie in Abbildung 6.1 dar. Die ROC Curve und das Precision Recall Diagramm des Entscheidungsbaumklassifikators sind in Abbildung 6.2a und 6.3a dargestellt. Auffällig ist hier, dass die Fläche unterhalb der Kurven für den Testdatensatz immer etwas größer sind als die des Trainingsdatensatzes.

Der AUROC Wert der logistischen Regression auf dem Testdatensatz ist, wie ein Vergleich

der Abbildungen 6.2a und 6.2b zeigt, etwa 3 Prozentpunkte höher als der des Entscheidungsbaumklassifikators. Auch bezogen auf die in den Precision Recall Kurven angegebene Fläche unterhalb dieser Kurven schneidet die logistische Regression leicht besser ab. Darüber hinaus ist die Abweichung der in der Abbildung 6.3b dargestellten AUPRC Werte für den Trainings- und den Testdatensatz bei der logistischen Regression sehr gering.

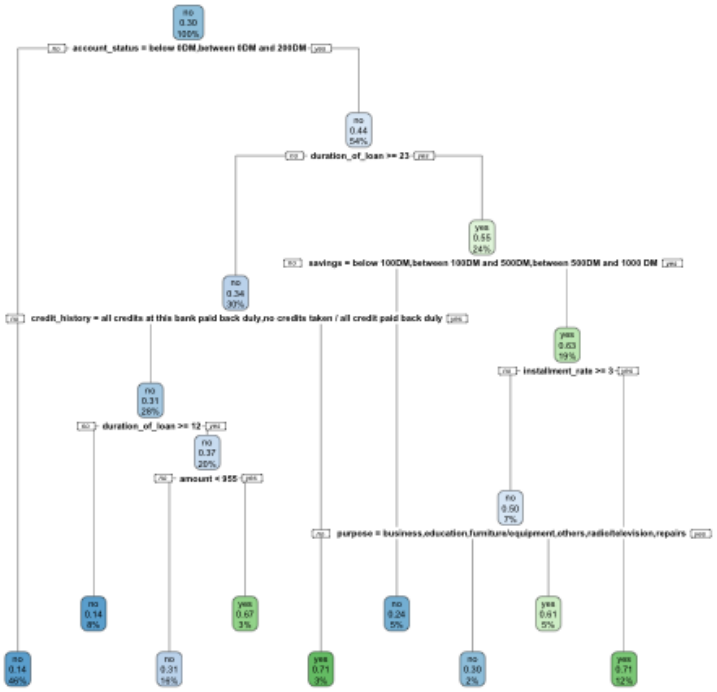
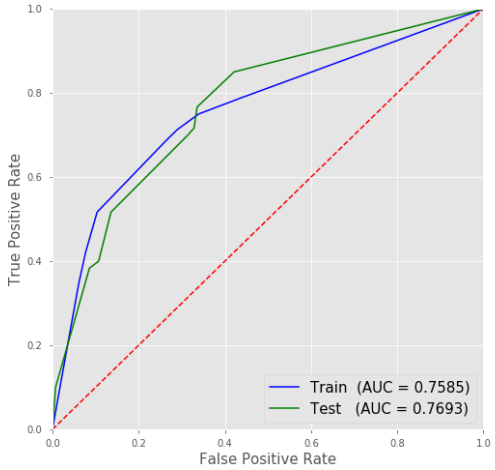
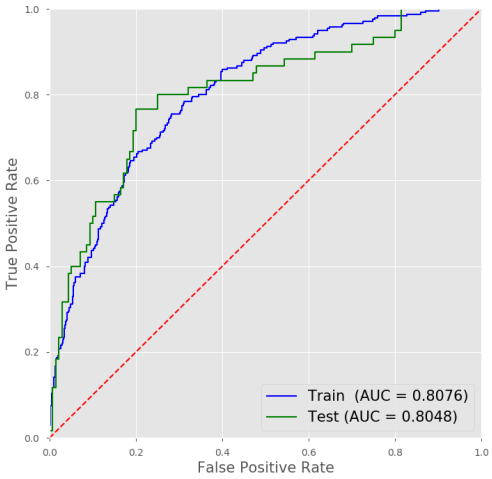


Abbildung 6.1.: Entscheidungsbaum des Ursprungsdatensatzes German Credit ohne topologische Informationen

Das wichtigste Splitattribut ist der Kontostand *account_status*. Fällt der Kontostand in den Bereich zwischen 0DM und 200DM oder ist das Konto bereits überzogen, so befindet sich der Datensatz in einem Knoten des rechten Teilbaumes und erhöht somit das Risiko, dass ebendieser Kredit ausfällt. Es ist der Abbildung 6.1 zu entnehmen, dass beispielsweise eine längere Kreditlaufzeit zu einer Erhöhung des Ausfallsrisikos führen kann. In diesem Kapitel soll wiederum gezeigt werden, ob auch auf einem anderen Datensatz eine Verbesserung des Klassifikators durch Hinzunahme eines topologischen Features erzielt werden kann.

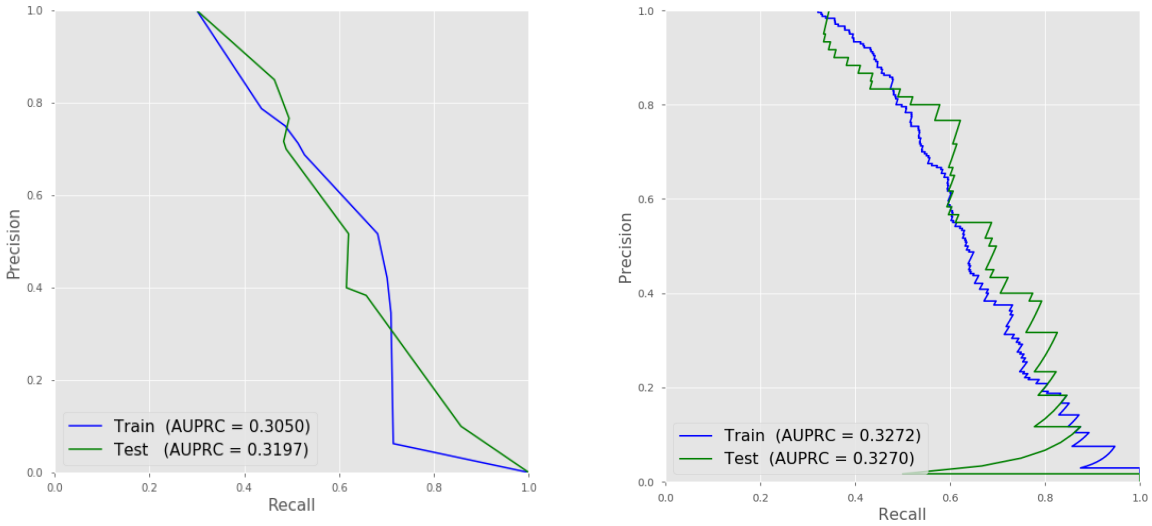


(a) Entscheidungsbaumklassifikator mit $cp = 0.015$



(b) logistische Regression mit Regularisierung $\alpha = 5.0$, auto_trim_tol=0.01, max_iter=1000

Abbildung 6.2.: ROC Curve ohne topologische Informationen auf dem German Credit Datensatz



(a) Entscheidungsbaumklassifikator mit $cp = 0.015$

(b) logistische Regression mit Regularisierung $\alpha = 5.0$, auto_trim_tol=0.01, max_iter=1000

Abbildung 6.3.: Precision-Recall Kurve des Entscheidungsbaumklassifikators ohne topologische Informationen auf dem German Credit Datensatz

6.1. Lineare Diskriminanzanalyse als Projektionsfunktion

Im Folgenden wird für die Durchführung der topologischen Datenanalyse die lineare Diskriminanzanalyse als Linse verwendet. Da das Target *class* nur die zwei Ausprägungen 1 und 0 besitzt, ist durch die lineare Diskriminanzanalyse nur die Reduktion der Dimension auf eine Dimension möglich. Das Ergebnis der linearen Diskriminanzanalyse des German Credit Datensatzes ist im Scatterplot der Abbildung 6.4 dargestellt. Betrachtet man den eindimensionalen Scatterplot, so gruppieren sich im negativen Bereich die pünktlich zurückgezahlten Kredite, wohingegen die Kredite bei denen Zahlungsverzögerungen auftreten eher im positiven Bereich liegen.

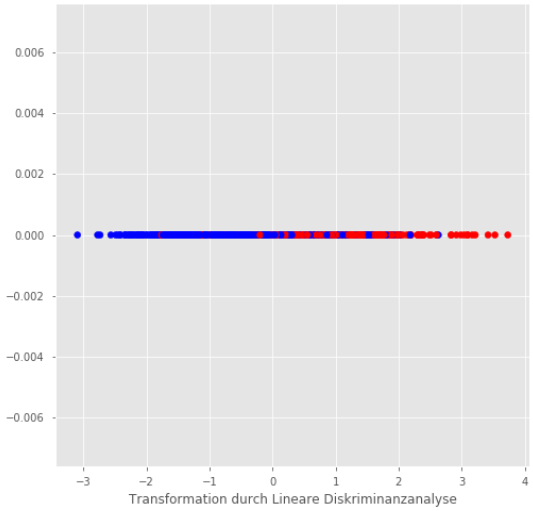
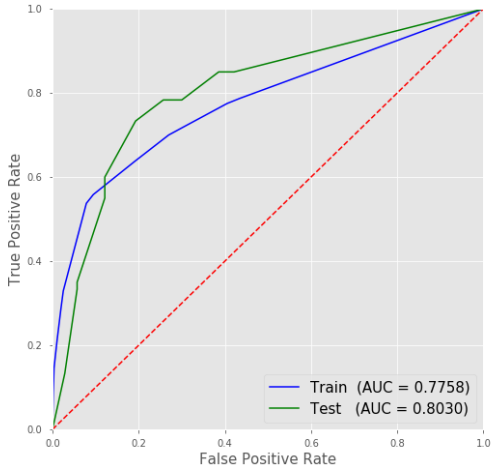
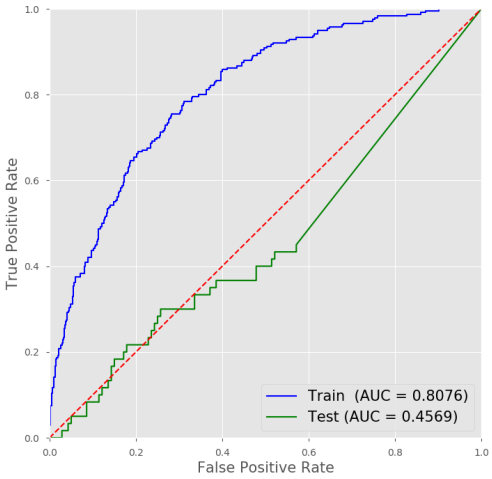


Abbildung 6.4.: Scatterplot der linearen Diskriminanzanalyse als Linse auf dem German Credit Datensatz

Für die Durchführung der topologischen Datenanalyse wurden 45 überlappende Intervalle mit einer Stärke der Überlappung von 30% verwendet. Die Vorhersage des topologischen Features auf dem Testdatensatzes geschieht wie beim Kreditkartentransaktionsdatensatz über die Zugehörigkeit der neuen Objekte zu den Nodes im Mapper Graphen. Durch das Ergebnis der Graphenanalyse auf dem Trainingsdatensatz werden die vorhergesagten Nodes in die *connectedComponentId* übersetzt.



(a) Entscheidungsbaumklassifikator mit $cp = 0.015$

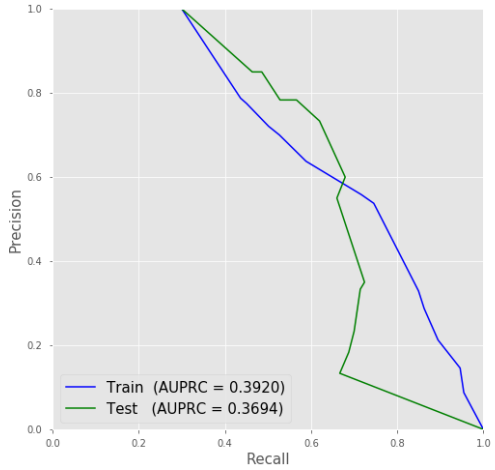


(b) logistische Regression mit Regularisierung $\alpha = 5,0$, auto_trim_tol=0,01, max_iter=1000

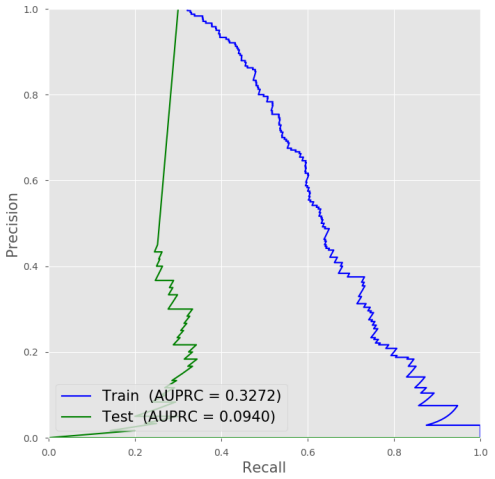
Abbildung 6.5.: ROC Kurve bei Nutzung der linearen Diskriminanzanalyse als Linse auf dem German Credit Datensatz

Anhand der ROC Curve und dem Precision Recall Diagramm in Abbildung 6.5a und 6.6a lässt sich erkennen, dass die Hinzunahme des topologischen Features *connectedComponentId* beide Gütemaße angehoben hat. Die AUROC steigt im Vergleich zum Entscheidungsbaum ohne topologische Informationen um ca. 3 Prozentpunkte, die AUPRC liegt bei der Verwendung der linearen Diskriminanzanalyse um ca. 5 Prozentpunkte höher.

Ein Anhub der Gütemaße bei der logistischen Regression ergibt sich, wie aus den Abbildungen 6.5b und 6.6b entnommen werden kann, nicht. Stattdessen nimmt die Fläche unterhalb der ROC Kurve und dem Precision Recall Diagramm deutlich ab. Auffällig ist ebenfalls die große Differenz der Performancemaße auf dem Trainings- und dem Testdatensatz, welches anzeigt, dass die logistische Regression übertrainiert ist.



(a) Entscheidungsbaumklassifikator mit $cp = 0.015$



(b) logistische Regression mit Regularisierung $\alpha = 5.0$, auto_trim_tol=0.01, max_iter=1000

Abbildung 6.6.: Precision-Recall Kurve bei Nutzung der linearen Diskriminanzanalyse als Linse auf dem German Credit Datensatz

6.2. Kombination aus Isolation Forest und L_2-Norm als Projektionsfunktion

Die nächste Linse, die zur Generierung des topologischen Features genutzt werden soll, ist die Kombination aus einem Isolation Forest und der L_2-Norm. Ein Scatterplot dieser Linsenkombination ist in Abbildung 6.7 abgebildet. Im Gegensatz zum Kreditkartentransaktionsdatensatz lässt sich hier keine deutliche Gruppierung der auffällig gewordenen Kredite erkennen. Dies liegt vermutlich daran, dass ein nicht bezahlter Kredit kein besonders seltenes Ereignis im Datensatz darstellt.

Als Parameter für den Isolation Forest wurde eine *contamination* von 0,3 genutzt. Der Parameter *n_estimators* gibt an, aus wie vielen Bäumen der Isolation Forest besteht. Hier wird ein Isolation Forest bestehend aus 500 Bäumen, die auf jeweils 500 zufällig ausgewählten Objekten des Trainingsdatensatzes erzeugt werden, verwendet.

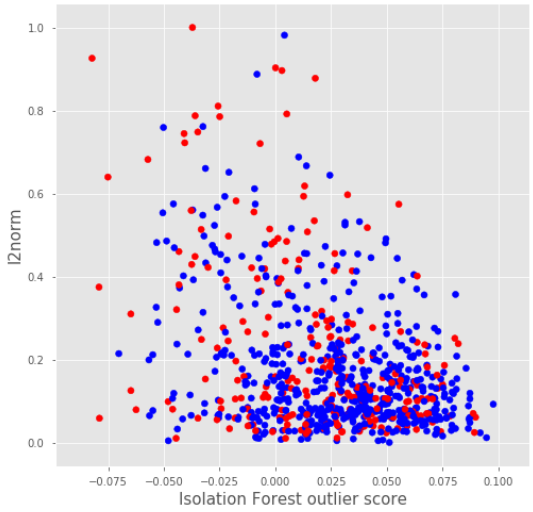
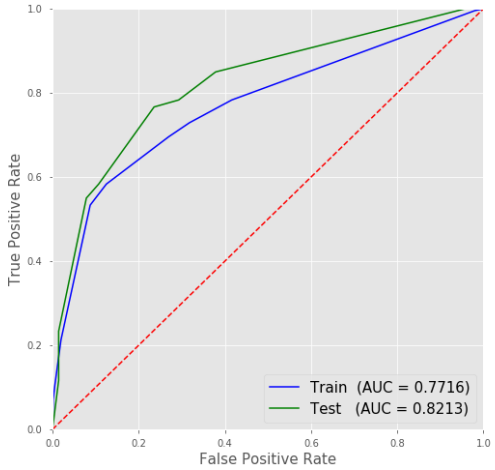
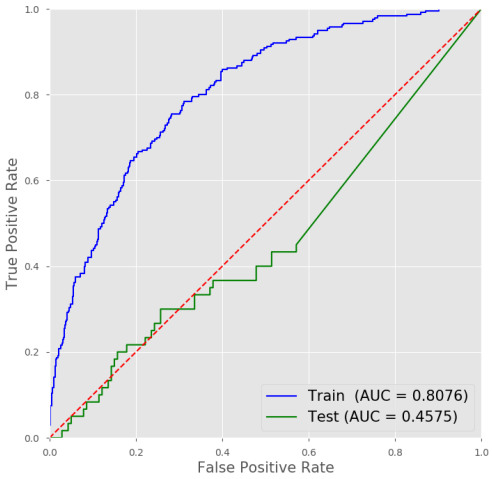


Abbildung 6.7.: Scatterplot der Linsenkombination aus Isolation Forest und L_2-Norm auf dem German Credit Datensatz. Parametrisierung: *contamination*=0.3, *n_estimators*=500, *max_samples*=500

Bei der Linsenkombination aus Isolation Forest und L_2-Norm werden zur Generierung der *connectedComponentId* 25 überlappende Intervalle je Dimension mit jeweils 70% Überlappung verwendet. Die Prädiktion der *connectedComponentId* auf dem Testdatensatz erfolgt über die *connectedComponentId* des nächstgelegenen Objektes des Trainingsdatensatzes. Als Maß für den Abstand der Beobachtungen im Trainingsdatensatz zur jeweiligen Beobachtung im Testdatensatz wurde der euklidische Abstand verwendet. Mit oben genannten Einstellungen wurde ein Entscheidungsbaum trainiert, der die in Abbildung 6.8a dargestellte ROC Curve und die Precision-Recall Kurve aus Abbildung 6.9b besitzen. Auch hier liefert ein Blick auf die in den Grafiken abgebildeten Werte eine Steigerung der Performance durch die Hinzunahme des neuen Features. Hier ist die Steigerung der Performancemaße mit über 5 Prozentpunkten respektive knapp 7 Prozentpunkten im Vergleich zum Klassifikator ohne topologische Information noch größer.

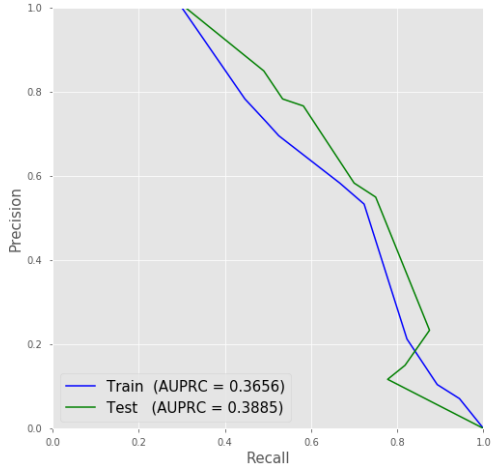


(a) Entscheidungsbaumklassifikator mit $cp = 0.015$

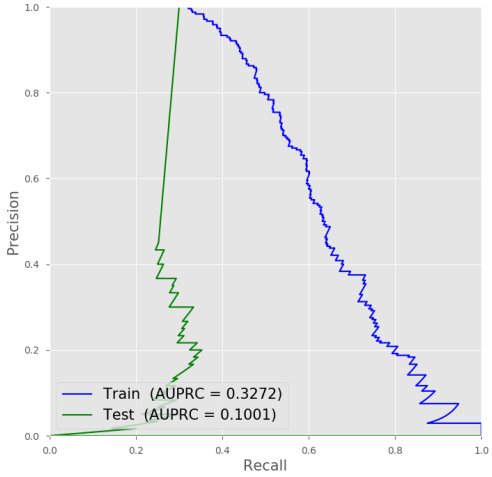


(b) logistische Regression mit Regularisierung $\alpha = 5.0$, auto_trim_tol=0.01, max_iter=1000

Abbildung 6.8.: ROC Kurve bei Nutzung der Linsenkombination Isolation Forest und L_2-Norm auf dem German Credit Datensatz



(a) Entscheidungsbaumklassifikator mit $cp = 0.015$



(b) logistische Regression mit Regularisierung $\alpha = 5.0$, auto_trim_tol=0.01, max_iter=1000

Abbildung 6.9.: Precision-Recall Kurve bei Nutzung der Linsenkombination Isolation Forest und L_2-Norm auf dem German Credit Datensatz

Die Abbildungen 5.10b und 5.11b der logistischen Regression zeigen, dass bei der Verwendung der logistischen Regression als Klassifikator durch die Hinzunahme des topologischen Features eine Verschlechterung der Performance auf dem Testdaten eintritt, wenn als Linse die Kombination aus Isolation Forest und L_2 Norm verwendet wird. Anhand der Diagramme lässt sich zusätzlich erkennen, dass durch die Hinzunahme des neuen Features die logistische Regression zu Overfitting neigt.

6.3. TSNE als Projektionsfunktion

Die TSNE Dimensionsreduktion der German Credit Daten, die als Linse für das Erzeugen des topologischen Features *connectedComponentId* genutzt wird, ist in der Abbildung 6.10 dargestellt. Für diese Darstellung wurde der Wert 1.000 für die Perplexität verwendet. Im Vergleich zur TSNE Dimensionsreduktion auf dem Kaggle Datensatz fällt hier auf, dass die Kredite, bei denen es Probleme mit der Tilgung gab, nicht an einer Stelle gehäuft auftreten, sondern im gesamten Wertebereich der dimensionsreduzierten Daten verstreut liegen. Dieser Effekt tritt auch bei anderen Werten für die Perplexität auf.

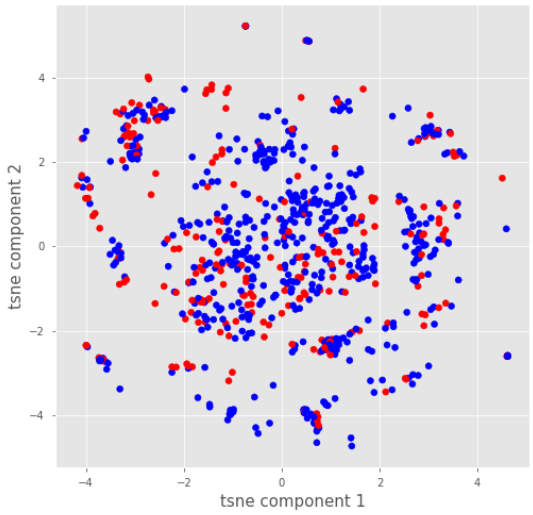
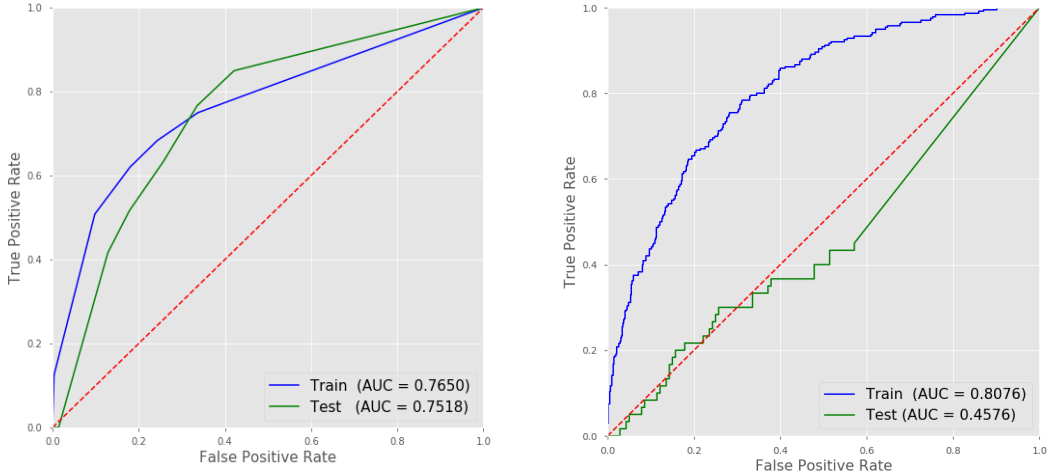


Abbildung 6.10.: Scatterplot der Linse TSNE dem German Credit Datensatz mit Perplexität 1000

Im Prozess zur Bestimmung der *connectedComponentId* jedes Objektes im Trainingsdatensatz wird die Linse analog zu Schritt 3 im Prozessdiagramm 4.2 mit einem überlappenden

Gitter überdeckt. Hierfür wurde die Anzahl der überlappenden Intervalle pro Dimension auf 25 gesetzt. Die Stärke der Überlappung beträgt 40%.

In diesem Fall wird das Feature *connectedComponentId* auf dem Testdatensatz durch die *connectedComponentId* des Nodes des Nodes vorhergesagt, dessen Clustermittelpunkt minimalen Abstand zum jeweiligen Datenpunkt des Testdatensatzes besitzt. Die Messung der Distanz der Datenpunkte des Trainingsdatensatzes bis zum Zentroid eines Nodes erfolgt durch die Metrik *mahalanobis*.



(a) Entscheidungsbaumklassifikator mit $cp = 0.015$

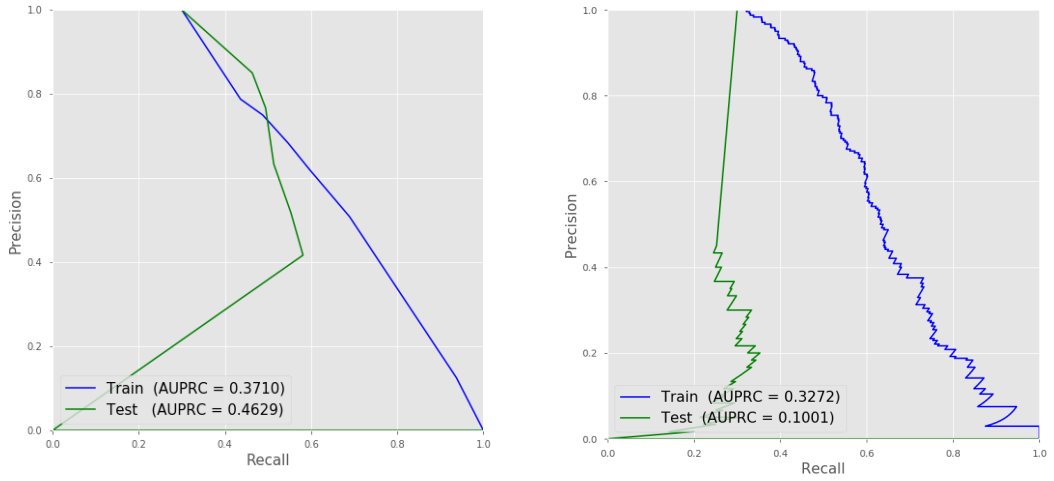
(b) logistische Regression mit Regularisierung $\alpha = 5.0$, auto_trim_tol=0.01, max_iter=1000

Abbildung 6.11.: ROC Kurve bei Nutzung der Linse TSNE auf dem German Credit Datensatz

In Abbildung 6.11a ist die ROC Kurve des Entscheidungsbaumes angegeben, der das topologische Feature *connectedComponentId* als zusätzlichen Input nutzt. Hierzu wurde die oben beschriebene Parametrisierung verwendet. Im Vergleich zum Entscheidungsbaum ohne topologisches Feature ergibt sich auf dem Testdatensatz eine leichte Verschlechterung des AUROC Performancemaßes um etwa einen Prozentpunkt. Zudem performt der Entscheidungsbaum hier auf dem Trainingsdatensatz etwas schlechter als auf dem Testdatensatz. Dies stellt einen Indikator für Overfitting dar.

Allerdings bestätigen sich die eben getroffenen Aussagen zur Verschlechterung der Performance durch die Durchführung der topologischen Datenanalyse nicht, wenn man zur Bewertung

der Performance des Klassifikators die Fläche unterhalb der Precision-Recall Kurve verwendet. Der Abbildung 6.12a ist zu entnehmen, dass die Fläche unterhalb der Precision-Recall Kurve des Testdatensatzes im Vergleich zum Klassifikator ohne topologisches Feature von rund 32 % auf 46,29 % ansteigt.



(a) Entscheidungsbaumklassifikator mit $cp = 0.015$

(b) logistische Regression mit Regularisierung $\alpha = 5.0$, auto_trim_tol=0.01, max_iter=1000

Abbildung 6.12.: Precision-Recall Kurve bei Nutzung der Linse TSNE auf dem German Credit Datensatz

Auffällig ist, dass mit der sehr starken Erhöhung des AUPRC keine Zunahme der weiteren Performancemetriken wie Accuracy, f_1 und f_2 Wert sowie AUROC einhergeht. Nichtsdestotrotz weist der Entscheidungsbaumklassifikator auf Basis des German Credit Datensatz bei einer Unterstützung durch die topologische Datenanalyse keine gute Qualität auf.

Durch die Betrachtung des ROC Charts aus Abbildung 6.11b und des Precision Recall Diagrammes aus Abbildung 6.12b der logistischen Regression ist ersichtlich, dass auch die Nutzung der TSNE Linse die Performance der logistischen Regression nicht verbessert. Ebenfalls tritt, wie auch bei den anderen Linsen, durch die Hinzunahme des topologischen Features deutliches Overfitting der logistischen Regression auf.

6.4. UMAP als Projektionsfunktion

Nun wird zur Generierung der *connectedComponentId* als Linse die UMAP Dimensionsreduktion verwendet. Die Darstellung in Abbildung 6.13 zeigt die UMAP Dimensionsreduktion bei

Verwendung der in Tabelle 6.1 dargestellten Parametrisierung. Die Reduktion der Dimension erfolgt hier auf die Dimension 2, da so eine zweidimensionale grafische Darstellung des Ergebnisses der Dimensionsreduktion möglich ist.

Parameter	Wert	Beschreibung
n_neighbors	10	Anzahl der betrachteten Nachbarn. Große Zahlen liefern eine globale Lösung, kleinere Anzahlen führen zu einer lokalen Lösung
min_dist	0,2	minimaler Abstand der dimensionsreduzierten Punkte
metric	chebyshev	Metrik zur Bestimmung der Abstände

Tabelle 6.1.: Verwendete Parametrisierung für UMAP Linse

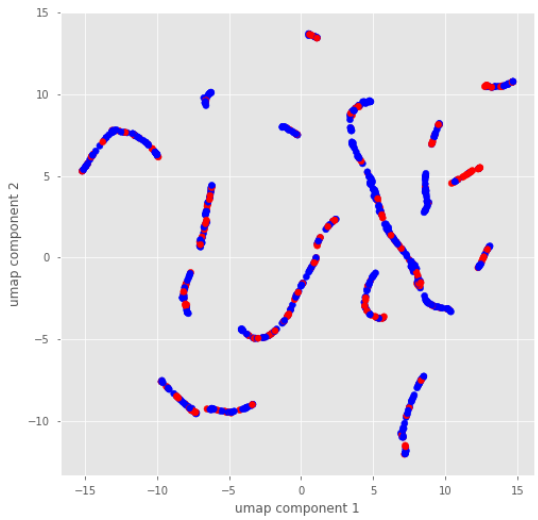
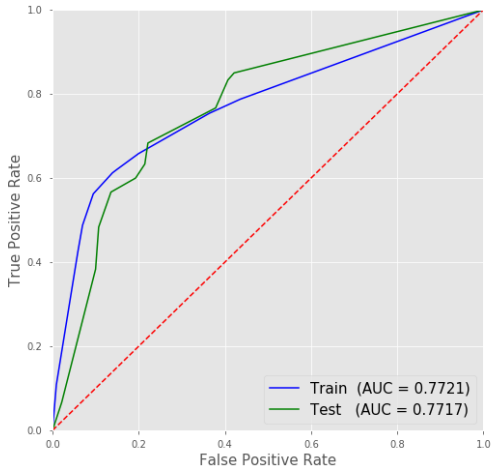


Abbildung 6.13.: Scatterplot der Linse UMAP dem German Credit Datensatz bei Parametrisierung aus Tabelle 6.1

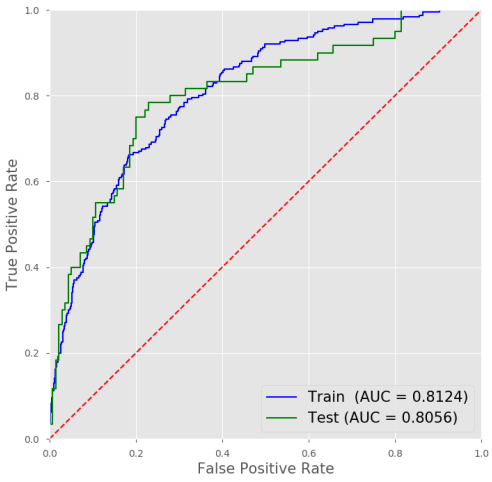
Die in diesem Teil dargestellten Ergebnisse entstehen durch die Überdeckung der Linse durch ein Gitter bestehend aus 20 Intervallen pro Dimension, die sich jeweils zu 20 % überlappen. Die für die Anwendung des Klassifikators auf die Daten des Testdatensatzes notwendi-

ge *connectedComponentId* muss anhand des Trainingsdatensatzes vorhergesagt werden. Dies geschieht hier, wie bereits bei der Kombination aus Isolation Forest und L_2-Norm, durch die *connectedComponentId* des nächsten Nachbarn im Trainingsdatensatz. Als Abstandsmaß findet hier die euklidische Metrik Verwendung.

Die Abbildungen 6.14a und 6.15a zeigen Grafiken zur Evaluation der Performance des Klassifikators, dessen Datenbestand durch ein topologisches Feature ergänzt wurde. Der Vergleich dieser Darstellungen mit denen des Klassifikators ohne topologische Information aus den Abbildungen 6.2a und 6.3a lässt den Schluss zu, dass die Hinzunahme des topologischen Features *connectedComponentId* einen positiven Einfluss auf die Qualität des Klassifikators besitzt. Allerdings ist die Zunahme des AUROC marginal. Im Vergleich zu den Ergebnissen bei Verwendung der Linsenkombination aus Isolation Forest und L_2-Norm liegen die Performancemaße auf dem Trainingsdatensatz oberhalb derer auf dem Testdatensatz. Jedoch ist der Unterschied der Performance zwischen dem Trainingsdatensatz und dem Testdatensatz sehr gering.



(a) Entscheidungsbaumklassifikator mit $cp = 0.015$



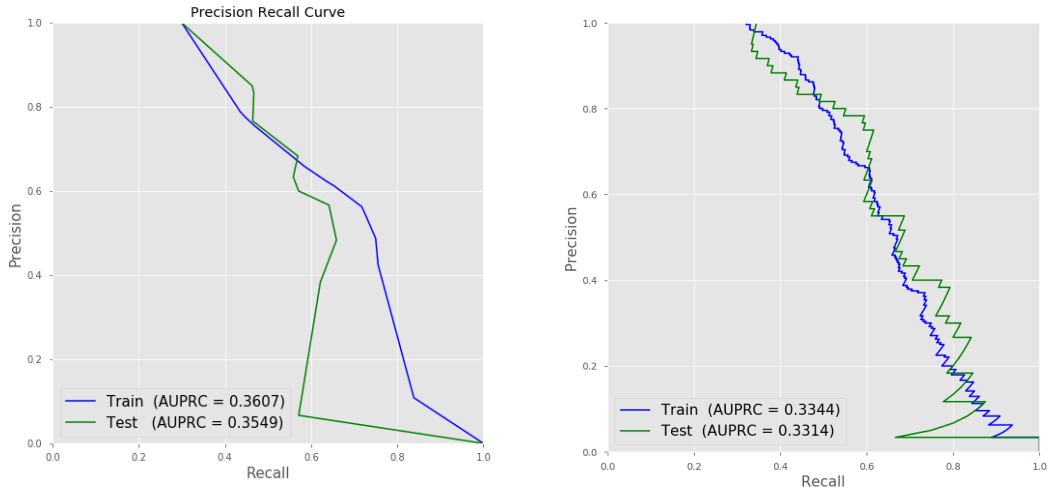
(b) logistische Regression mit Regularisierung $\alpha = 5.0$, auto_trim_tol=0.01, max_iter=1000

Abbildung 6.14.: ROC Kurve bei Nutzung der Linse UMAP auf dem German Credit Datensatz

Durch Hinzunahme des topologischen Features ergibt sich eine Zunahme des AUROC auf dem Testdatensatz um rund 0,2 Prozentpunkte. Die AUPRC hingegen verbessert sich von

rund 32% auf etwas über 35%.

Eine grafische Darstellung der Performance der logistischen Regression findet sich im ROC Chart in Abbildung 6.14b und im Precision Recall Diagramm aus Abbildung 6.15b. Anhand dieser Diagramme lässt sich ablesen, dass die Hinzunahme des topologischen Features, welches mithilfe der UMAP Dimensionsreduktion berechnet wurde, einen sehr leichten Anstieg des AUROC Wertes verursacht. Die Zunahme des AUPRC auf dem Testdatensatz liegt bei der logistischen Regression bei etwa einem halben Prozentpunkt.



(a) Entscheidungsbaumklassifikator mit $cp = 0.015$

(b) logistische Regression mit Regularisierung $\alpha = 5.0$, auto_trim_tol=0.01, max_iter=1000

Abbildung 6.15.: Precision-Recall Kurve bei Nutzung der Linse UMAP auf dem German Credit Datensatz

6.5. Vergleich der Ergebnisse

Gütemaß	ohne topologische Features	LDA Linse	Isolation Forest und L_2-Norm	TSNE	UMAP
Accuracy	76,00%	78,00%	**81,00%**	73,5%	77,50%
AUPRC	31,97%	36,94%	38,85%	**46,29%**	35,49%
AUROC	76,93%	80,30%	**82,13%**	75,18%	77,17%
f_1	56,36%	60,00%	**63,46%**	48,54%	60,17%
f_2	53,45%	56,90%	**58,10%**	44,17%	58,02%

Tabelle 6.2.: Tabelle ausgewählter Gütemaße des Entscheidungsbaumklassifikators auf dem Testdatensatz der German Credit Datensatzes

Gütemaß	ohne topologische Features	LDA Linse	Isolation Forest und L_2-Norm	TSNE	UMAP
Accuracy	**78,50%**	70,0%	70,00%	70,00%	**78,50%**
AUPRC	32,70%	9,40%	10,01%	10,00%	**33,14%**
AUROC	80,48%	45,69%	45,75%	45,76%	**80,56%**
f_1	**59,81%**	0,00%	0,00%	0,00%	**59,81%**
f_2	**55,75%**	0,00%	0,00%	0,00%	**55,75%**

Tabelle 6.3.: Tabelle ausgewählter Gütemaße der logistischen Regression auf dem Testdatensatz des German Credit Datensatzes

Wie in Tabelle 6.2 zu erkennen, liefert die Linsenkombination aus Abnormalitätsscore eines Isolation Forests mit der L_2-Norm der Features im Datensatz das beste Ergebnis bezogen auf die Mehrheit der in Tabelle 6.2 dargestellten Gütemaße . Durch die Verwendung der TSNE Linse bei der topologischen Datenanalyse zur Unterstützung eines Entscheidungsbaumklassifikators konnte die Fläche unterhalb der Precision Recall Kurve auf 46,29% angehoben werden. Somit stellt sich bei der Betrachtung des AUPRC Maßes heraus, dass die TSNE Linse die größte Verbesserung des Entscheidungsbaumklassifikators bringt.

Anhand der Performancemaße des Entscheidungsbaumklassifikators in Tabelle 6.2 ist zu erkennen, dass das topologische Feature *connectedComponentId* die Qualität des Entscheidungsbaums zwar verbessert, der Klassifikator jedoch als eher schlecht zu bewerten ist, da durch eine zufällige Klassifikation bereits ein AUPRC von 30% erzielt wird.

Die Tabelle 6.3 zeigt die Werte der in der Tabelle dargestellten Gütemaße der logistischen Regression auf dem Testdatensatz. Insgesamt lässt sich den Werten aus der Tabelle entnehmen, dass die Hinzunahme des topologischen Features zu keinem relevanten Performancegewinn führt, wenn die logistische Regression als Klassifikator gewählt wird. Ebenfalls

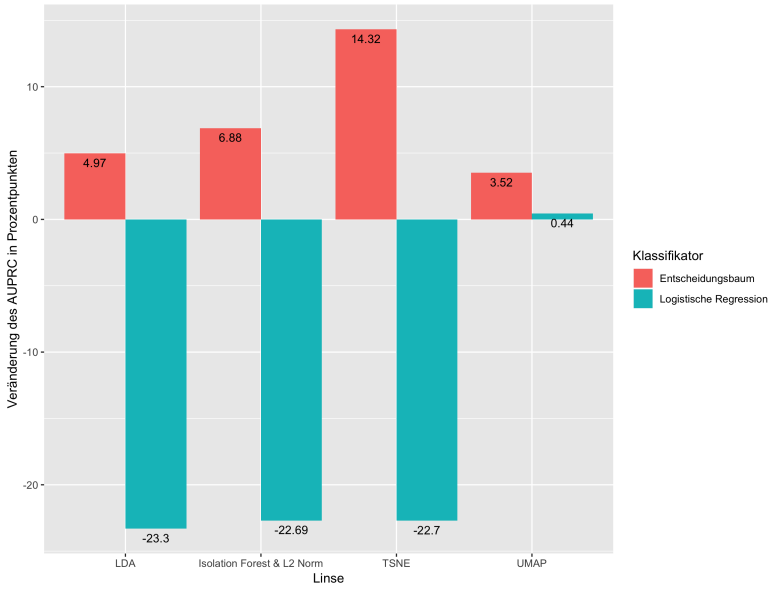


Abbildung 6.16.: Balkendiagramm der Veränderung des AUPRC auf dem German Credit Datensatz durch die topologische Datenanalyse. Die Veränderung des AUPRC wurde auf dem Testdatensatz gemessen.

führt die Hinzunahme der topologischen Information hier zu einem deutlichen Overfitting der logistischen Regression. Auch besitzt der durch die logistische Regression berechnete Klassifikator, wie auch der Entscheidungsbaumklassifikator absolut betrachtet keine gute Performance. Somit drängt sich der Eindruck auf, dass die Features des Datensatzes keine verlässliche Erklärung für den Ausfall der Kredite liefern.

Die sich auf dem German Credit Datensatz ergebenden Zu- und Abnahmen des AUPRC auf dem Testdatensatz sind im Balkendiagramm 6.16 dargestellt. Wie im Diagramm 6.16 ersichtlich, lässt sich der Entscheidungsbaumklassifikator durch die Anwendung der topologischen Datenanalyse durch jede der Linsen verbessern. Am stärksten ist der Anhub des AUPRC des Entscheidungsbaumes bei der TSNE Linse. Hier steigt der AUPRC Wert um etwa 14 Prozentpunkte. Die Anhübe der anderen Linsen bewegen sich zwischen 3 und 7 Prozentpunkten. Die Performance der logistischen Regression lässt sich nur durch die UMAP Linse etwas verbessern. Diese Verbesserung ist jedoch sehr gering ausgeprägt. Bei den anderen Linsen sinkt die Performance der logistischen Regression stark.

7. Ergebnisse auf dem Lending Club Datensatz

Dieses Kapitel zeigt die Ergebnisse auf dem Lending Club Datensatz bei der Durchführung des Gesamtprozesses der topologischen Datenanalyse aus dem Flussdiagramm in Abbildung 4.1 ab Seite 53. Eine Beschreibung des Datensatzes findet sich im Abschnitt 4.2.3 auf der Seite 64. Im Gegensatz zu den beiden bereits betrachteten Datensätzen, enthält der Lending Club Datensatz fehlende Werte, die während der Datenvorbereitung durch plausible Werte ersetzt werden müssen. Die hierzu genutzte Strategie zur Ersetzung dieser fehlenden Werte wird im Abschnitt 4.2.3 beschrieben.

Da der Lending Club Datensatz ebenfalls kategorische Features, wie z.B. eine durch Lending Club vorgenommene Bewertung der Kreditwürdigkeit durch Bonitätsstufen von A bis F, enthält, ist als zusätzlicher Schritt der Vorbereitung der Daten eine Umwandlung der kategorischen Features in eine numerische Darstellung notwendig, sodass die Anwendung der topologische Datenanalyse nach dem Prozessdiagramm 4.2 möglich ist. Als Komplexitätsparameter für den Entscheidungsbaum wird hier der Defaultwert $cp = 0.01$ verwendet.

Zum Training der logistischen Regression wurde als Regularisierungsparameter $\alpha = 1000$, eine Toleranz von 0.01 und als maximale Anzahl Iterationen 1000 genutzt.

Für die in diesem Kapitel vorgestellten Untersuchungen wird der Lending Club Datensatz zufällig unter Berücksichtigung der Klassenzugehörigkeit der Beobachtungen in 60% Trainingsmenge und 40% Testmenge geteilt. Das Target wird bei der Teilung der Daten in Training und Test in Betracht gezogen, damit sowohl in der Trainings- als auch in der Testmenge ein ähnliches Verhältnis zwischen Kreditausfällen und bezahlten Krediten wie im Gesamtdatensatz herrscht.

In der Abbildung 7.1 ist der Entscheidungsbaum, der auf dem Rohdatensatz bestimmt wurde, abgedruckt. Die erste Verzweigung des Baumes erfolgt durch das Feature *recoveries*. Liegt der Wert für das Feature oberhalb von 1.17, so klassifiziert der Entscheidungsbaum den jeweiligen

Kredit als Kredit, bei dem es zu Zahlungsausfällen kommt. Dies ist bei 2,76% der Kredite im Trainingsdatensatz der Fall. Der zweite Split erfolgt nach der Summe der insgesamt gezahlten Gebühren für eine verspätete Bezahlung des Kredites.

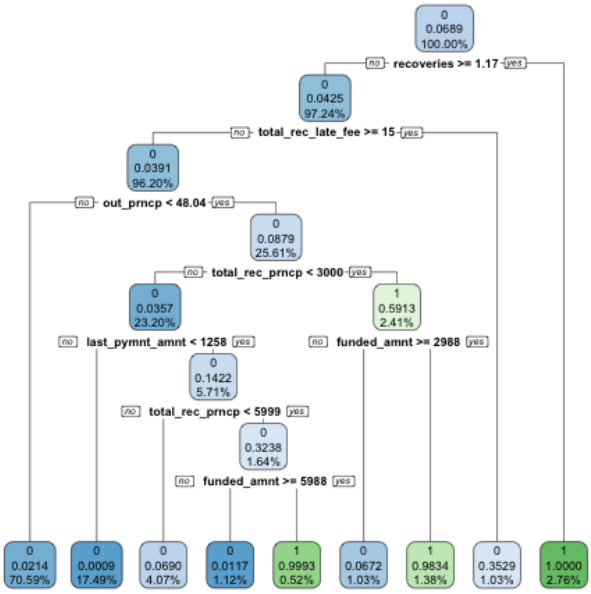
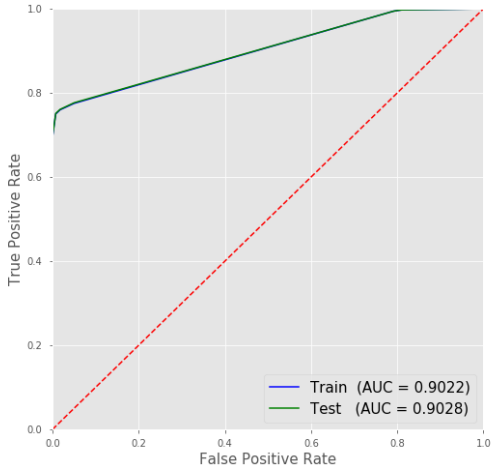


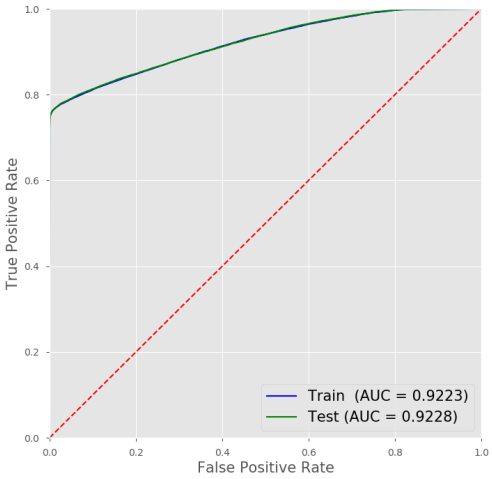
Abbildung 7.1.: Entscheidungsbaum ohne topologisches Feature auf dem Lending Club Datensatz bei Nutzung des Komplexitätsparameters $cp = 0.01$

Die ROC Kurve des auf dem Ursprungsdatensatz trainierten Entscheidungsbaumklassifikators findet sich in der Abbildung 7.2a. Wie beim German Credit Datensatz ist die Fläche unterhalb der ROC Kurve des Testdatensatz höher als die des Trainingsdatensatzes. Jedoch ist die Zunahme auf dem Testdatensatz gegenüber dem Trainingsdatensatz sehr gering.

Die ROC Kurve der logistischen Regression ohne die Nutzung des topologischen Features ist in Abbildung 7.2b dargestellt und weist für den AUROC Wert auf dem Testdatensatz einen um 2 Prozentpunkte höheren Wert auf.



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$

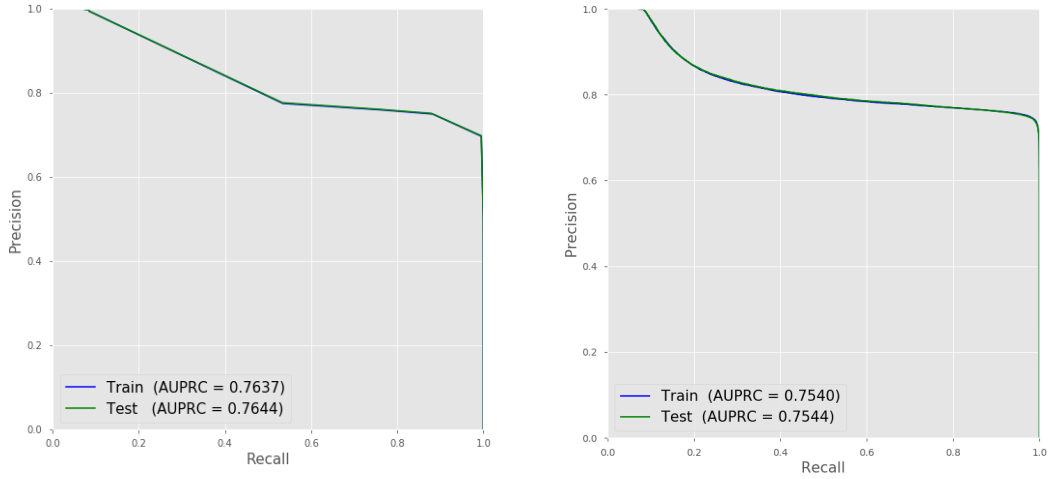


(b) [Logistische Regression mit $\alpha = 1000$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 7.2.: ROC Kurve der Klassifikatoren ohne topologisches Feature auf dem Lending Club Datensatz

Im Precision-Recall Diagramm in Abbildung 7.3a ist dieses Verhalten ebenfalls zu erkennen, dass der Klassifikator auf dem Testdatensatz leicht besser performt als auf dem Trainingsdatensatz. Mit einem AUPRC von rund 76% auf dem Testdatensatz handelt es sich bei dem Klassifikator, der auf den Ursprungsdaten trainiert wurde, bereits um einen guten Klassifikator. Bei dem naiven Klassifikator, der in jedem Fall die häufigste Klasse vorhersagt, entspricht die Fläche unterhalb der Precision-Recall Kurve genau der relativen Häufigkeit eines Ausfalles eines Kredites im Datensatz. Für den hier betrachteten Datenbestand ergibt sich daher beim naiven Klassifikator ein AUPRC von nur 6,8%.

Das Precision Recall Diagramm zur logistischen Regression ist in Abbildung 7.3b zu finden und besitzt ebenfalls die Auffällgkeit, dass der Klassifikator auf dem Testdatensatz leicht besser ist als auf dem Trainingsdatensatz. Vergleicht man die Precision Recall Diagramme der beiden Klassifikatoren aus der Abbildung 7.3a und 7.3b miteinander, so fällt auf, dass der Entscheidungsbaumklassifikator einen höheren AUPRC Wert aufweist als die logistische Regression.



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$

(b) Logistische Regression mit $\alpha = 1000$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 7.3.: Precision-Recall Kurve der Klassifikatoren ohne topologisches Feature auf dem Lending Club Datensatz

In den folgenden Abschnitten wird die topologische Datenanalyse wiederum nach dem Prozessdiagramm für den Gesamtprozess in Abbildung 4.1 auf Seite 53 durchgeführt. Die auf diese Weise erhaltenen Klassifikatoren werden mit dem Klassifikator, der keine topologische Informationen nutzt, verglichen und bewertet, ob eine Vergrößerung der Fläche unterhalb der Precision-Recall Kurve beobachtet werden kann, die gleichbedeutend mit einer Verbesserung der Performance des Klassifikators ist.

Bei der Bestimmung des neuen kategoriellen Features *connectedComponentId* durch die topologische Datenanalyse ist die Parametrisierung der Überdeckung der Linsen in Schritt 3 des Prozesses aus 4.2 essentiell. Parameter der Überdeckung der Linsen mit überlappenden Intervallen ist zum Einen die Anzahl der Intervalle *n_cubes* und zum Anderen die Stärke der Überlappung *overlap_perc*. Zur Bestimmung optimaler Werte dieser Parameter wurden einige Kombinationen dieser Werte getestet und der AUPRC auf dem Testdatensatz bestimmt. Für die Anzahl der Intervalle wurden Werte von 5 bis 80 in 5er Schritten verwendet. Die Stärke der Überlappung der Intervalle wird nacheinander auf die Werte $10^{-5}; 10^{-4}; 10^{-3}, 0,01; 0,05; 0,1; 0,2; 0,3; 0,4; 0,5; 0,6; 0,7$ und $0,8$ gesetzt. Die Kombination der Parameter, die den höchsten AUPRC Wert aufweist, wird als Optimum verwendet. Hierbei werden nur Kombinationen der Parameter *n_cubes* und *overlap_prec* verwendet, bei

denen die *connectedComponentId* im Entscheidungsbaum verwendet wird.

Die Anwendung der Klassifikatoren auf dem Testdatensatz erfordert die Vorhersage des topologischen Features. Hierfür wird der im Folgenden beschriebene Prozess durchgeführt. Zunächst werden die Beobachtungen des Testdatensatz einem Node des Graphen zugeordnet. Dies geschieht durch den Vergleich der Abstände der Beobachtungen im Testdatensatz zu den Clustermittelpunkten der Nodes im Graphen, die jeweils ein Cluster von Beobachtungen darstellen. Mit einem Python Dictionary in dem zu jedem Node die zugehörige *connectedComponentId* zugeordnet ist, können im Anschluss jeder Beobachtung des Testdatensatzes die passende *connectedComponentId* zugewiesen werden.

7.1. Lineare Diskriminanzanalyse als Projektionsfunktion

Wegen des binären Targets *loan_status* besteht der Output der Dimensionsreduktion durch die lineare Diskriminanzanalyse aus einem Vektor, dessen Komponentenanzahl der Anzahl der Beobachtungen im Trainingsdatensatz entspricht. In der Abbildung 7.4 ist die in diesem Teil verwendete Linse abgebildet. Rote Punkte in der Grafik kennzeichnen einen Ausfall eines Kredites, blaue Punkte stellen einen termingerecht bezahlten Kredit dar. Es fällt auf, dass unauffällige Kredite eher im negativen Bereich lokalisiert sind und die auffälligen Krediten ausschließlich im positiven Bereich der mithilfe der linearen Diskriminanzanalyse transformierten Werte liegen.

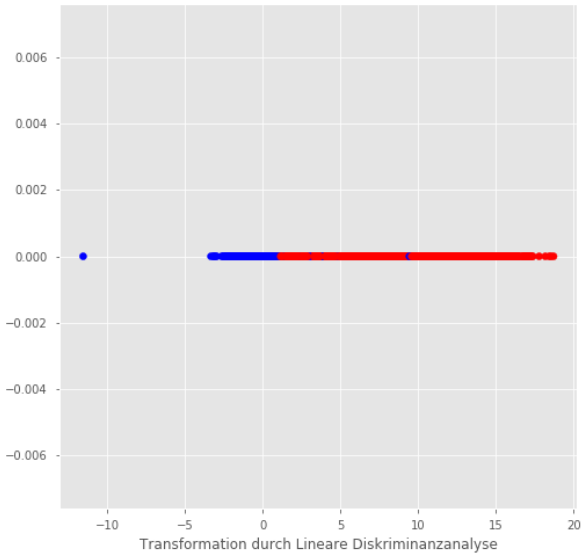
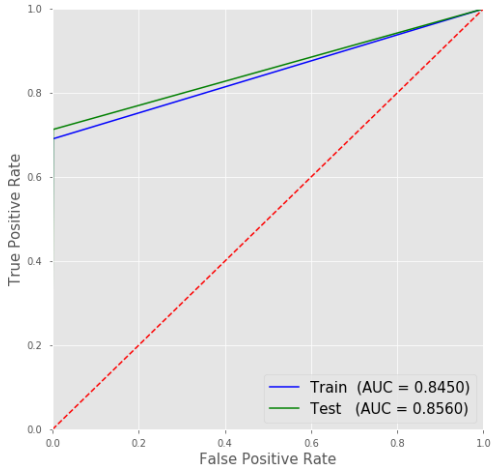
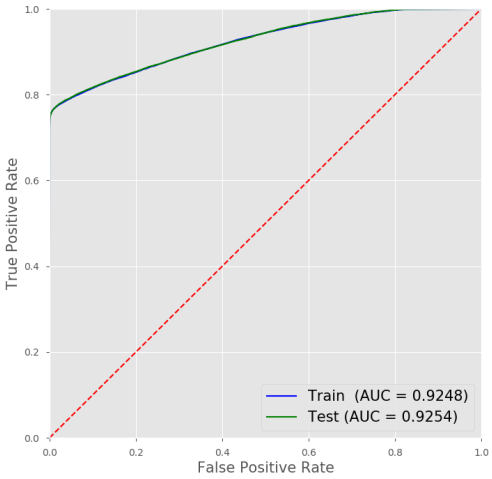


Abbildung 7.4.: Scatterplot der linearen Diskriminanzanalyse auf dem Lending Club Datensatz

Zur Parametrisierung der topologischen Datenanalyse zur Bestimmung des neuen Features *connectedComponentId* wurden 40 überlappende Intervalle mit einer Stärke der Überlappung von 0,0001 verwendet. Der ROC Chart des mithilfe dieser Parametrisierung berechnete Klassifikator findet sich in Abbildung 7.5a. Bei der Betrachtung des ROC Charts fällt auf, dass die Performance des Klassifikators hier auf dem Testdatensatz etwa ein Prozentpunkt besser ist als auf dem Trainingsdatensatz.



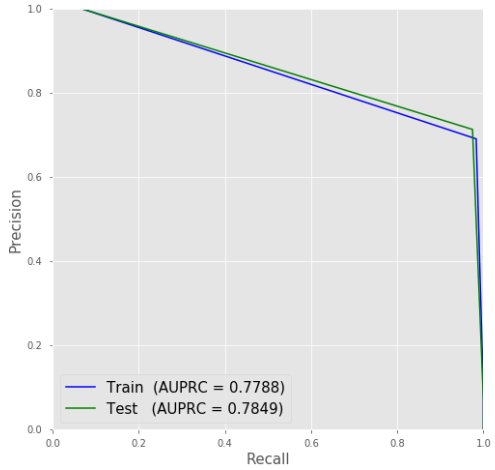
(a) Entscheidungsbaumklassifikator mit $cp = 0.01$



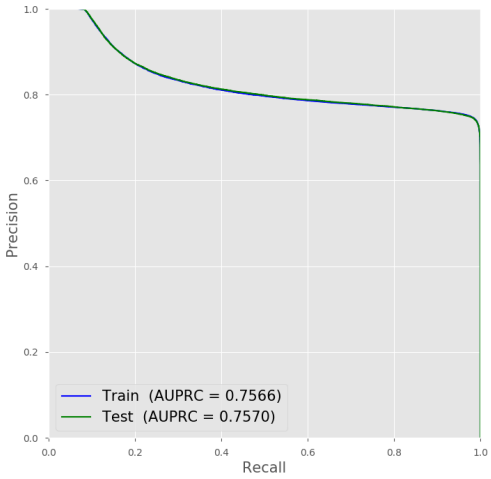
(b) Logistische Regression mit $\alpha = 1000$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 7.5.: ROC Chart der Klassifikatoren bei Verwendung der linearen Diskriminanzanalyse als Linse auf dem Lending Club Datensatz

Betrachtet man das Precision Recall Diagramm in Abbildung 7.6a, so bestätigt sich, dass auch bei einem Blick auf den Flächeninhalt unterhalb der Precision-Recall Kurve der Klassifikator auf dem Testdatensatz leicht bessere Ergebnisse zeigt als auf dem Teil des Datensatzes, der zum Training des Entscheidungsbaumes genutzt wurde.



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$



(b) Logistische Regression mit $\alpha = 1000$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 7.6.: Precision-Recall Kurve der Klassifikatoren bei Benutzung der linearen Diskriminanzanalyse als Linse auf dem Lending Club Datensatz

Bei der Verwendung der linearen Diskriminanzanalyse hebt das zusätzliche, topologische Feature die Performance des Klassifikators leicht an. Ein AUPRC auf dem Testdatensatz von 78,49% steht einem AUPRC von 76,44% auf dem Testdatensatz ohne topologische Informationen entgegen. Die Durchführung der topologischen Datenanalyse mithilfe der linearen Diskriminanzanalyse als Linse konnte die Qualität des Klassifikators um rund 2 Prozentpunkte verbessern.

Die Abbildungen 7.5b und 7.6b zeigen ROC Chart und Precision Recall Diagramm der logistischen Regression mit den angegebenen Einstellungen. Vergleicht man die ROC Charts der beiden Klassifikatoren miteinander, so erkennt man, dass der AUROC Wert der logistischen Regression etwa 7 Prozentpunkte höher liegt, als das Ergebnis des Entscheidungsbaumklassifikators. Bei der Betrachtung der Precision Recall Diagramm zeigt sich diese Beobachtung jedoch nicht. Durch die Hinzunahme des topologischen Features ist bei der logistischen Regression eine leichte Erhöhung des AUPRC zu verzeichnen. Die Verbesserung ist jedoch sehr geringen Ausmaßes und beträgt nur 0,3 Prozentpunkte.

7.2. Kombination aus Isolation Forest und L_2-Norm als Projektionsfunktion

Die in diesem Teil für die Generierung des topologischen Features verwendete Projektionsfunktion ist die Kombination aus einem Abnormalitätsscore eines Isolation Forest und der L_2-Norm des Trainingsdatensatzes. Die Ergebnisse dieser Linsenkombination sind in der Abbildung 7.7 in Form eines Scatterplots dargestellt. Eine Gruppierung von ordnungsgemäß bezahlten (blau) und überfälligen Krediten (rot) ist in Abbildung 7.7 nicht zu erkennen.

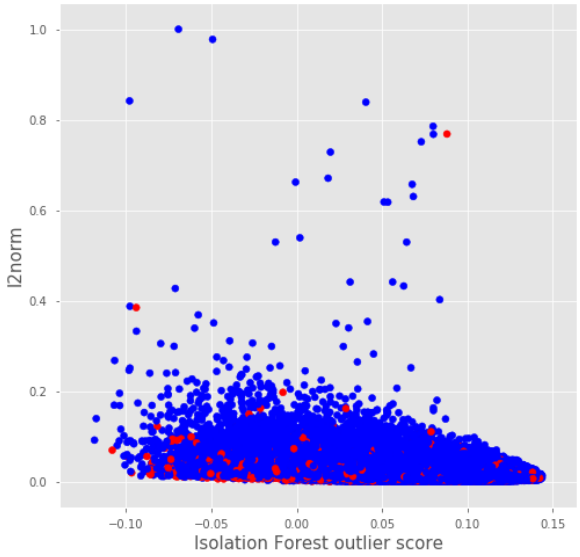
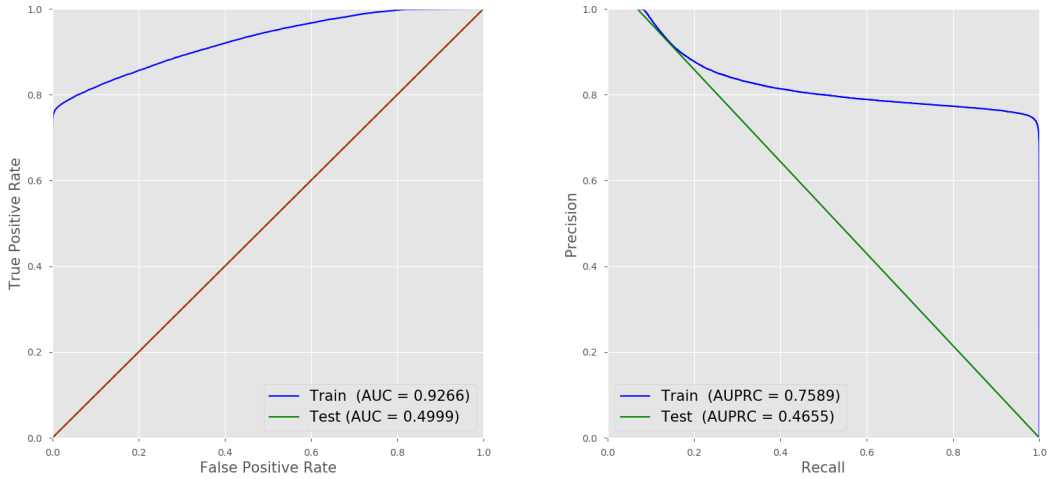


Abbildung 7.7.: Scatterplot der Linsenkombination aus Abnormalitätsscore eines Isolation Forest und L_2 Norm auf dem Lending Club Datensatz. Parametrisierung: *contamination*=0.06, *n_estimators*=500, *max_samples*=500

Bei der Verwendung der Linsenkombination aus Isolation Forest und L_2-Norm auf dem Lending Club Datensatz lässt sich beobachten, dass bei jeder Kombination der getesteten Parameterwerte das topologische Feature *connectedComponentId* nicht als Splitattribut des Entscheidungsbaumes verwendet wird. Die Nutzung von einer höheren Anzahl von Intervallen von über 100 für die Überdeckung der Linsenkombination aus Isolation Forest und L_2-Norm ändert an dem beobachteten Effekt nichts. Wegen der Nichtnutzung des topologischen Features *connectedComponentId* für die Klassifikation gleicht der bestimmte Entscheidungsbaum dem Entscheidungsbaum, der anhand des Ursprungsdatensatzes bestimmt wurde und in der



(a) ROC Curve

(b) Precision Recall Diagramm

Abbildung 7.8.: ROC Curve und Precision Recall Diagramm der logistischen Regression mit den Parametern $\alpha = 1000$, auto_trim_tol = 0.01, max_iter=1000 bei Verwendung der Linsenkombination aus Isolation Forest und L_2-Norm der Features

Abbildung 7.1 dargestellt ist. Somit stimmen die Maße zur Beurteilung der Performance der beiden Entscheidungsbäume überein. Es konnte daher keine Verbesserung der Klassifikationsergebnisse erzielt werden.

Bei der logistischen Regression hingegen, nimmt die Qualität des Klassifikators durch die Hinzunahme des topologischen Features stark ab, wie die Abbildung 7.8 zeigt. Darüber hinaus entsteht durch die Hinzunahme der CCID massives Overfitting der logistischen Regression, da die Performancewerte auf dem Testdatensatz stark von denen auf dem Trainingsdatensatz nach unten abweichen. Dieser Effekt liegt vermutlich daran, dass die hohe Anzahl von Intervallen von 150 gleichzeitig eine sehr hohe Zahl von zusammenhängenden Komponenten verursacht, die sich wiederum in einem komplexen Modell niederschlagen. Dieses wird zwar penalisiert geschätzt, dies hat offensichtlich nicht zu einem generalisierenden Modell geführt.

7.3. TSNE als Projektionsfunktion

In diesem Teil soll die Auswirkung auf die Güte der Klassifikation gezeigt werden, wenn zur Generierung des topologischen Features als Projektionsfunktion der Dimensionsreduktionsalgorithmus TSNE verwendet wird. Für die Perplexität wird der Wert 100 genutzt. Die bei der Durchführung der topologischen Schritte genutzten, dimensionsreduzierten Daten sind in der

Abbildung 7.9 in einem Scatterplot dargestellt. Es sind mehrere Inseln mit säumigen Krediten zu erkennen, allerdings sind diese nicht deutlich von den ordnungsgemäß bezahlten Krediten abgegrenzt.

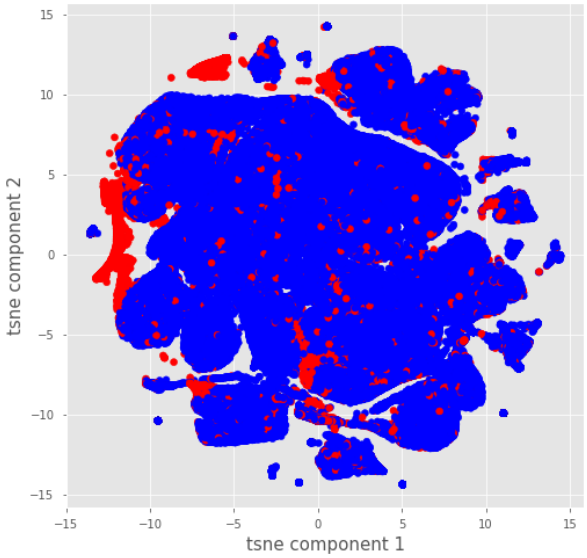
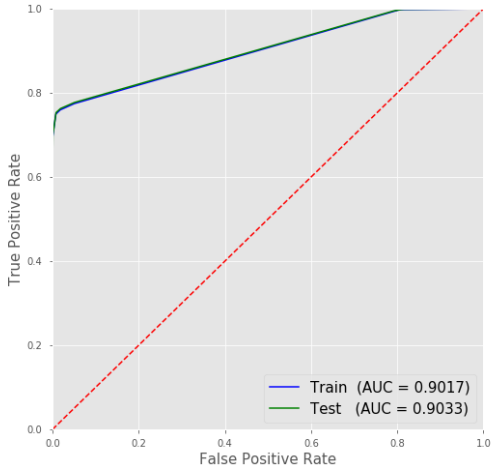


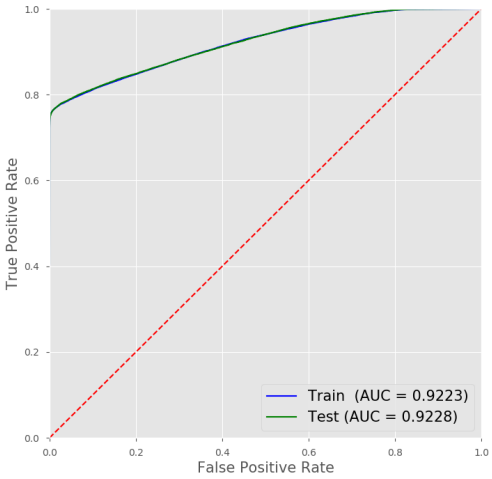
Abbildung 7.9.: Scatterplot der TSNE Dimensionsreduktion mit Perplexität 100 auf dem Lending Club Datensatz

Für die Überdeckung der TSNE Linse zur Bestimmung des neuen, topologischen Features wurden 15 überlappende Intervalle mit einer Stärke der Überlappung von 10^{-4} verwendet. Die Abbildung 7.10a zeigt die ROC Kurve des Klassifikators. Blickt man auf den Wert für die Fläche unterhalb der ROC Kurve, so ist ersichtlich, dass der Klassifikator auf dem Testdatensatz etwas besser performt als auf dem Trainingsdatensatz.

Diese Beobachtung, wenn auch in etwas geringerer Stärke lässt sich auch bei der logistischen Regression machen, deren ROC Kurve in Abbildung 7.10b dargestellt ist. Des Weiteren fällt auf, dass die logistische Regression bei der Verwendung der TSNE Linse einen etwas 2 Prozentpunkte höheren AUROC-Wert auf Trainings- und Testdatensatz besitzt als der Entscheidungsbaumklassifikator.



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$

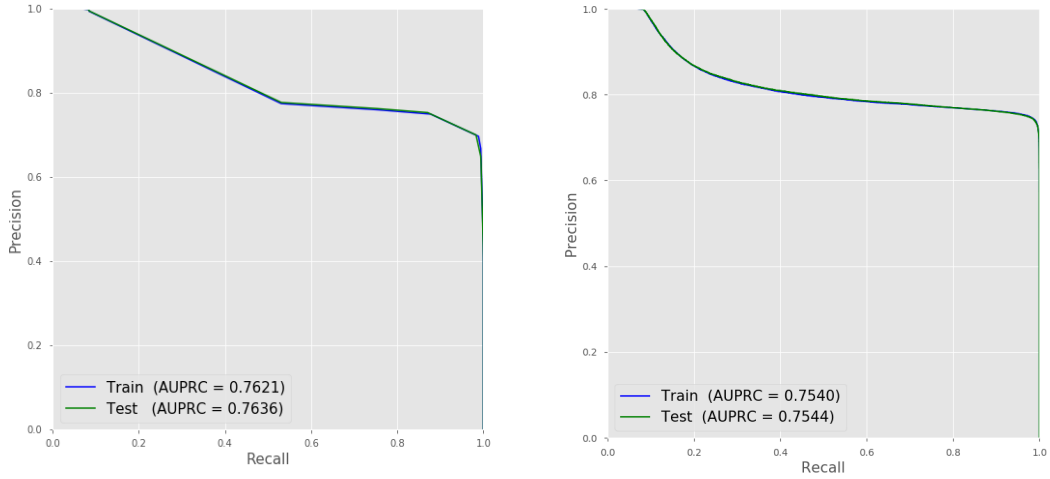


(b) Logistische Regression mit $\alpha = 1000$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 7.10.: ROC Kurve der Klassifikatoren bei TSNE als Projektionsfunktion auf dem Lending Club Datensatz

Auch bei der Betrachtung der Precision Recall Kurve in Abbildung 7.11a ist die Qualität des Klassifikators auf dem Testdatensatz im Vergleich zum Trainingsdatensatz leicht besser.

Das Precision Recall Diagramm aus Abbildung 7.11b stammt von der logistischen Regression. Durch den Vergleich der beiden Precision Recall Diagramme aus den Abbildungen 7.11a und 7.11b lässt sich erkennen, dass der AUPRC des Entscheidungsbaumklassifikators höher ist als der der logistischen Regression. Dagegen hat die Hinzunahme des topologischen Features bei der Klassifikation durch die logistische Regression nicht zu einem Anhub des Performancemaßes AUPRC geführt.



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$

(b) Logistische Regression mit $\alpha = 1000$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 7.11.: Precision Recall Kurve der Klassifikatoren bei TSNE als Projektionsfunktion auf dem Lending Club Datensatz

Im Vergleich zur Situation ohne topologische Informationen, performt der Entscheidungsbaumklassifikator, der das topologische Feature *connectedComponentId* nutzt, auf einem sehr ähnlichen Qualitätsniveau. Die Durchführung der topologischen Datenanalyse mit der Nutzung der TSNE Dimensionsreduktion führt daher nicht zu einer besseren Klassifikation durch Entscheidungsbäume oder die logistische Regression.

7.4. UMAP als Projektionsfunktion

Die Tabelle 7.1 zeigt die Werte für die Parameter der Dimensionsreduktion nach UMAP, die für die in diesem Teil durchgeführten Untersuchungen gewählt wurden. Zur Ermittelung des Ergebnisses von UMAP werden die zwanzig nächsten Nachbarn verwendet. Der minimale Abstand der dimensionsreduzierten Daten *min_dist* wird auf 0,2 gesetzt, als Metrik kommt die Metrik *canberra* zum Einsatz. Eine Darstellung des Ergebnisses der UMAP Dimensionsreduktion als Scatterplot, welche im weiteren Verlauf genutzt wird, befindet sich in Abbildung 7.12.

Parameter	Wert	Beschreibung
n_neighbors	20	Anzahl der betrachteten Nachbarn. Große Zahlen liefern eine globale Lösung, kleinere Anzahlen führen zu einer lokalen Lösung
min_dist	0,2	minimaler Abstand der dimensionsreduzierten Punkte
metric	canberra	Metrik zur Bestimmung der Abstände

Tabelle 7.1.: Verwendete Parametrisierung für UMAP Linse des Lending Club Datensatzes

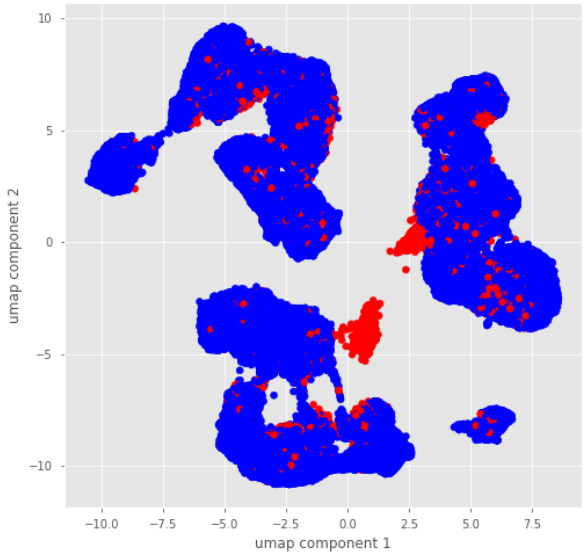
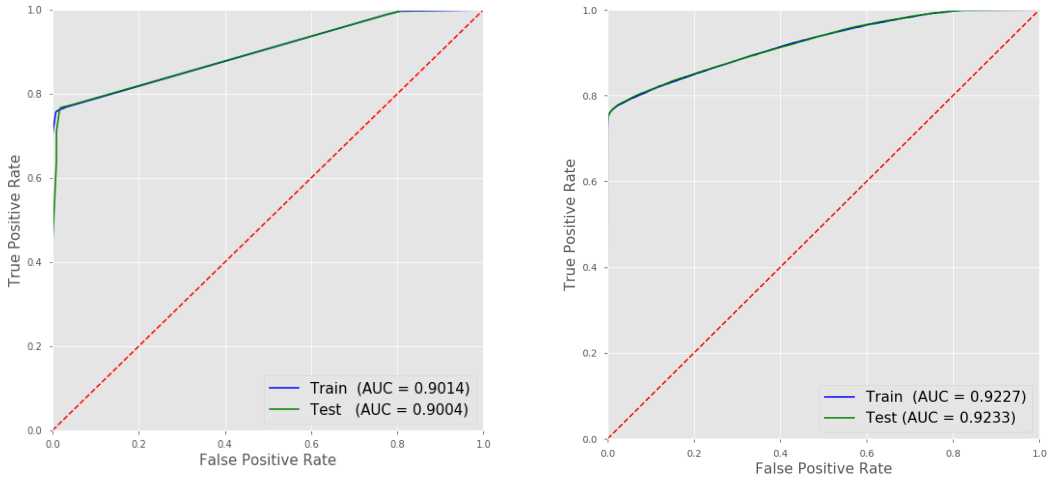


Abbildung 7.12.: Scatterplot der UMAP Dimensionsreduktion als Linse auf dem Lending Club Datensatz. Parametrisierung siehe Tabelle 7.1

Die hier abgedruckten Ergebnisse entstehen durch die Verwendung von 30 überlappenden Intervallen mit einer Überlappung von einem Promille für die Überdeckung der UMAP Linse. Die Abbildung 7.13a enthält einen Plot der ROC Curve, in den die Fläche unterhalb der ROC Curve für den Trainings- und den Testdatensatz als Teil der Legende eingetragen ist. Mit 90,14% auf dem Trainingsdatensatz und 90,04% auf dem Testdatensatz ist die Fläche

unterhalb der ROC Kurve auf dem Trainingsdatensatz leicht größer als auf dem Testdatensatz.

Die Analyse der Klassifikationsergebnisse der logistischen Regression erzeugt die Abbildung 7.13b, die einen ROC Chart der logistischen Regression darstellt. Die Fläche unterhalb der Kurve im ROC Chart auf dem Testdatensatz liegt mit 92,33% etwas mehr als 2 Prozentpunkte oberhalb dem AUROC des Entscheidungsbaumklassifikators bei der Verwendung der Linse UMAP.



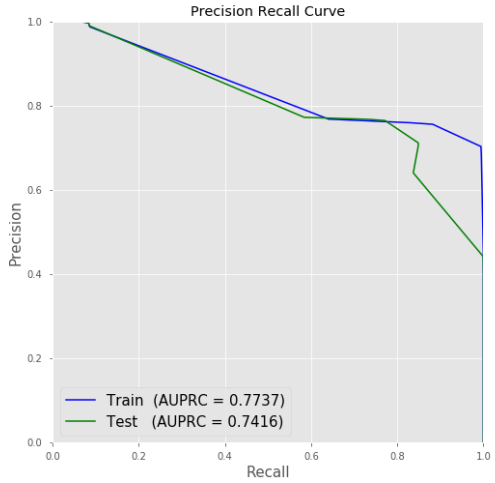
(a) Entscheidungsbaumklassifikator mit $cp = 0.01$

(b) Logistische Regression mit $\alpha = 1000$, auto_trim_tol = 0.01, max_iter=1000

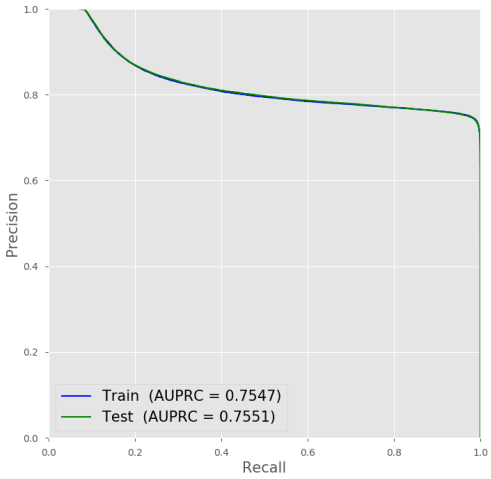
Abbildung 7.13.: ROC Kurve der Klassifikatoren bei Verwendung der UMAP Dimensionsreduktion als Linse auf dem Lending Club Datensatz

Die Precision-Recall Kurve des Entscheidungsbaumklassifikators findet sich in Abbildung 7.14a. Hier performt der Entscheidungsbaumklassifikator ebenfalls auf dem Trainingsdatensatz besser. Diese Auffälligkeit ist mit einem Unterschied von über 3 Prozentpunkten deutlicher ausgeprägt, als bei der ROC Kurve.

In Abbildung 7.14b ist das Precision Recall Diagramm der logistischen Regression zu finden. Im Gegensatz zum Entscheidungsbaumklassifikator, performt die logistische Regression auf dem Testdatensatz besser als auf dem Trainingsdatensatz. Allerdings ist dieser Unterschied sehr gering. Anders als beim Entscheidungsbaumklassifikator ergibt sich bei der logistischen Regression eine leichte Verbesserung des AUPRC Werts. Allerdings ist diese Zunahme marginal.



(a) Entscheidungsbaumklassifikator mit $cp = 0.01$



(b) Logistische Regression mit $\alpha = 1000$, auto_trim_tol = 0.01, max_iter=1000

Abbildung 7.14.: Precision-Recall Kurve der Klassifikatoren bei Verwendung der UMAP Dimensionsreduktion als Linse auf dem Lending Club Datensatz

Sieht man auf die Veränderung des AUPRC auf dem Trainingsdatensatz durch die Hinzunahme des topologischen Features des Entscheidungsbaumklassifikators, so gewinnt man den Eindruck, dass das neue Feature die Qualität der Klassifikation leicht anhebt. Allerdings schlägt sich dieser Effekt nicht auf den Testdatensatz nieder, da hier eine Abnahme des AUPRC um etwa 2 Prozentpunkte erfolgt. Daher ist auch bei der Verwendung der UMAP Dimensionsreduktion als Linse keine Verbesserung der Qualität der Klassifikation zu erkennen. Bei der logistischen Regression ist eine sehr leichte Zunahme des AUPRC zu verzeichnen.

7.5. Vergleich der Ergebnisse

Gütemaß	ohne topologische Features	LDA Linse	Isolation Forest und L_2-Norm	TSNE	UMAP
Accuracy	97,90%	97,90%	97,90%	97,85%	97,90%
AUPRC	76,44%	**78,49%**	76,44%	76,36%	74,16%
AUROC	90,28%	85,60%	90,28%	**90,33%**	90,04%
f_1	82,06%	**82,37%**	82,06%	81,78%	77,45%
f_2	74,25%	**75,34%**	74,25%	74,28%	73,58%

Tabelle 7.2.: Tabelle ausgewählter Gütemaße des Entscheidungsbaumklassifikators auf dem Testdatensatz des Lending Club Datensatzes

Gütemaß	ohne topologische Features	LDA Linse	Isolation Forest und L_2-Norm	TSNE	UMAP
Accuracy	**98,15%**	**98,15%**	6,90%	**98,15%**	**98,15%**
AUPRC	75,44%	**75,70%**	46,56%	75,44%	75,51%
AUROC	92,28%	**92,54%**	49,99%	92,28%	92,33%
f_1	84,66%	**84,67%**	12,90%	84,66%	**84,67%**
f_2	77,95%	77,94%	27,02%	77,95%	**77,97%**

Tabelle 7.3.: Tabelle ausgewählter Gütemaße der logistischen Regression auf dem Testdatensatz des Lending Club Datensatzes

Mit Ausnahme der linearen Diskriminanzanalyse führen die getesteten Linsen bei der Durchführung einer topologischen Datenanalyse nicht zu einer Verbesserung des Entscheidungsbaumklassifikators. Bei der Linsenkombination aus einem Abnormalitätsscore und der L_2-Norm ist das im Rahmen des Prozesses der topologischen Datenanalyse aus Abbildung 4.2 generierte topologische Feature nicht wichtig genug, um im Entscheidungsbaum genutzt zu werden. Die topologische Datenanalyse mithilfe der linearen Diskriminanzanalyse als Linse führt zwar zu einer Zunahme des AUPRC auf dem Testdatensatz, jedoch ist die Verbesserung mit etwas über 2 Prozentpunkten im Vergleich zu den anderen Datensätzen eher gering. Somit kann die Qualität der Vorhersage eines Entscheidungsbaumklassifikators auf dem Lending Club Datensatz durch topologische Datenanalyse nicht nachhaltig abgehoben werden.

In Abbildung 7.3 sind die wichtigsten Performancemaße der logistischen Regression in Abhängigkeit der verwendeten Linsen tabellarisch dargestellt. Der stärkste Anhub des AUPRC Wertes konnte durch die Verwendung der linearen Diskriminanzanalyse verursacht werden. Allerdings ist dieser Anhub nicht besonders stark. Bei Nutzung der Kombination von Isolation Forest und L_2-Norm als Linse nimmt die Qualität der Klassifikation durch die Hinzunah-

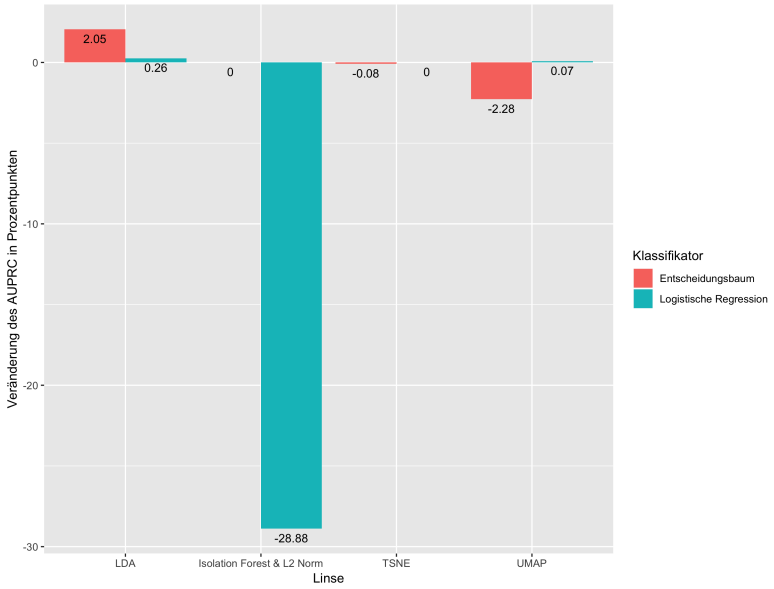


Abbildung 7.15.: Balkendiagramm der Veränderung des AUPRC auf dem Lending Club Datensatz durch die topologische Datenanalyse. Die Veränderung des AUPRC wurde auf dem Testdatensatz gemessen.

me des topologischen Features stark ab. Zusammengefasst lässt sich aufgrund der schwach ausgeprägten Zunahme des AUPRC durch die topologische Datenanalyse sagen, dass die topologische Datenanalyse auf dem Lending Datensatz nicht imstande ist, die Qualität der logistischen Regression nachhaltig zu verbessern.

Die Veränderung des AUPRC auf dem Testdatensatz der beiden Klassifikatoren ist grafisch in Abbildung 7.15 dargestellt. Auffällig ist der starke Rückgang der Performance der logistischen Regression bei der Nutzung der Linsenkombination aus Isolation Forest und L_2-Norm. Beide Klassifikatoren lassen sich durch die topologische Datenanalyse mithilfe der linearen Diskriminanzanalyse verbessern. Allerdings steigt der AUPRC des Entscheidungsbaumes bei der Verwendung der linearen Diskriminanzanalyse als Linse um etwa 2 Prozentpunkte, wohingegen bei der logistischen Regression durch die topologische Datenanalyse nur ein Anhub von einem viertel Prozentpunkt ausgelöst wird.

Anhand dieser Ergebnisse und den Vergleich der Ergebnisse deren anderen zur Untersuchung verwendeten Datensätzen ist der Schluss zulässig, dass die topologische Datenanalyse auf dem Lending Club Datensatz das geringste Potential besitzt, die Klassifikation zu verbessern.

Teil III.

Design und Implementierung einer Applikationen zur Unterstützung des Gesamtprozesses

8. Design und Implementierung der Applikation des Gesamtprozesses

Wie bereits in der Einleitung beschrieben, ist das Ziel dieser Masterthesis das Design und die Implementierung eines Prototyps zur Durchführung des in den beiden Flussdiagrammen 4.1 und 4.2 in Kapitel 4 gezeigten Prozessdiagrammen beschriebenen Prozess zur durch die topologische Datenanalyse unterstützte Klassifikation von Daten. Die in den vorangegangenen Kapiteln beschriebenen Experimente wurden in Jupyter Notebooks durchgeführt. In einem Jupyter Notebook kann interaktiv programmiert werden, der Output des produzierten Codes wird zusammen mit dem Code dargestellt. Die Interaktion mit dem Jupyter Notebook geschieht im Webbrowser.

Die bereits durchgeführten Experimente zeigen, dass der in dieser Masterthesis erarbeitete Prozess zur Vorhersage von Kreditausfällen beziehungsweise betrügerischen Transaktionen eine hohe Anzahl an Parametern aufweist, die einen starken Einfluss auf das Ergebnis besitzen und mit den anderen Parametern Wechselwirkungen aufweisen. Parameter des in dieser Masterthesis vorgestellten Prozesses zur Klassifikation von Daten unter Unterstützung der topologischen Datenanalyse sind zum Beispiel die gewählte Linse, deren Parameter, die Anzahl der Intervalle und die Breite der Überlappung für die Überdeckung der Linse sowie Methode und Metrik zur Messung der Abstände zur Vorhersage des topologischen Features auf dem Testdatensatz.

Eine Suche der optimalen Parameter durch mathematische Optimierungsverfahren ist wegen der hohen Anzahl von Parametern und der Wechselwirkungen nicht durchführbar. Dies führt zu der Notwendigkeit der experimentellen Bestimmung von Optimalwerten für die Parameter des Verfahrens. Zur Durchführung dieser Experimente ist bei der Nutzung von Jupyter Notebooks bei jeder Änderung der Datensätze oder eines Parameters ein höherer manueller Aufwand des Entwicklers nötig. Darüber hinaus ist es zur Speicherung der Ergebnisse notwendig, eine Kopie des Jupyter Notebooks anzulegen und diese nach den eigenen Vorstellungen abzuändern. Daher ist ein Prototyp einer Software hilfreich, welche die Durchführung dieser

Experimente erleichtert. Die Entwicklung und Implementierung dieses Prototypen wird in diesem Kapitel beschrieben.

Aufgrund konträrer Anwendungsszenarien wurden zwei interaktive Versionen und eine *batch* Version geplant und implementiert. Eine der interaktiven Versionen wurde mithilfe des Python Moduls `tkinter` als Desktopanwendung konzipiert. Auf Anforderung des Betreuers von Firamis wurde zusätzlich ein Webdashboard in Dash implementiert, um eine angenehmere Oberfläche zur Eingabe der notwendigen Parameter sowie zur Visualisierung der Zwischenergebnisse zu erhalten, die zur Vorführung vor Kunden geeignet ist.

Die interaktiven Versionen erlauben den dynamischen Eingriff des Benutzers in den Prozess der Klassifikation durch die Visualisierung von Zwischenergebnissen wie einem Scatterplot der berechneten Linse und dem erzeugten Mapper Graph und erlauben die Erforschung idealer Parameterwerte zum Erreichen guter Ergebnisse. Bei der *batch* Version hingegen müssen alle Einstellungen vor dem Beginn der Berechnung feststehen. Zur Untersuchung, ob der Klassifikator auch auf unbekannten Daten eine gute Performance zeigt, ist Crossvalidierung in die Funktionalität der *batch* Version integriert. Bei der interaktiven Version wurde auf Crossvalidierung verzichtet, um die Übersichtlichkeit und Performance des Prototypen zu gewährleisten. Durch die Crossvalidierung erfolgt die Berechnung der Linsen nicht wie bei den interaktiven Versionen einmal, sondern je nach Parametrisierung der Crossvalidierung mehrfach. Somit würden sich auch mehrere Zwischenergebnisse ergeben, die visualisiert werden. Dadurch nimmt die Übersichtlichkeit des Programmes ab. Ein weiterer Grund für den Verzicht auf die Crossvalidierung bei den interaktiven Versionen ist die längere Programmlaufzeit, da die Berechnung der Linse nicht nur einmal, sondern mehrfach erfolgt, welches mehr Zeit in Anspruch nimmt.

Eine wichtige Hilfsklasse, welche von anderen Klassen, die in der folgenden Beschreibung des Ablaufdiagrammes thematisiert werden, verwendet wird, ist die Klasse `ConfigParser`, welche zum Einlesen der Konfiguration des Prototyps dient. Die Konfigurationsdatei soll den implementierten Prototyp auf die vorhandene Konfiguration der Systeme des Benutzers einstellen. Zum Laden der Konfiguration des Prototyps verwenden einige der entworfenen Klassen die Hilfsklasse `ConfigParser`, welche eine Konfigurationdatei im YAML Format als Python Dictionary zur Verfügung stellt. Das Parsen der Konfigurationsdatei im YAML Format geschieht über das Modul `PyYAML` in der Version 3.12. Die Konfigurationsdatei enthält Informationen wie die URI und dem Port des Computers, welcher die Linsenberechnungen durchführt oder die

Größe des Plots des berechneten Entscheidungsbaumes. Weitere, in der Konfigurationsdatei zu spezifizierende Einstellungen sind die Speicherorte der Reports.

8.1. Datenvorbereitung

Der in diesem Kapitel beschriebene Prototyp deckt nicht die Kodierung von kategoriellen Features in numerische Features durch **Label Encoding** oder **One Hot Encoding** ab. Enthält ein Datensatz kategorielle Features, so ist eine zusätzliche numerische Version des Datensatzes bereitzustellen, bei der alle kategoriellen Features in numerische Features umkodiert wurden. Darüber hinaus ist der Prototyp darauf ausgerichtet, stets alle zur Verfügung stehenden Features zu nutzen. Die Auswahl von Features, die für die Klassifikation genutzt werden sollen, muss demnach als Teil der Datenvorbereitung vor der Nutzung des Prototyps durchgeführt werden.

Eine Darstellung des Ablaufes der Datenvorbereitung findet sich in Abbildung 8.1. Im weiteren Textverlauf werden die einzelnen Prozessschritte näher erläutert.

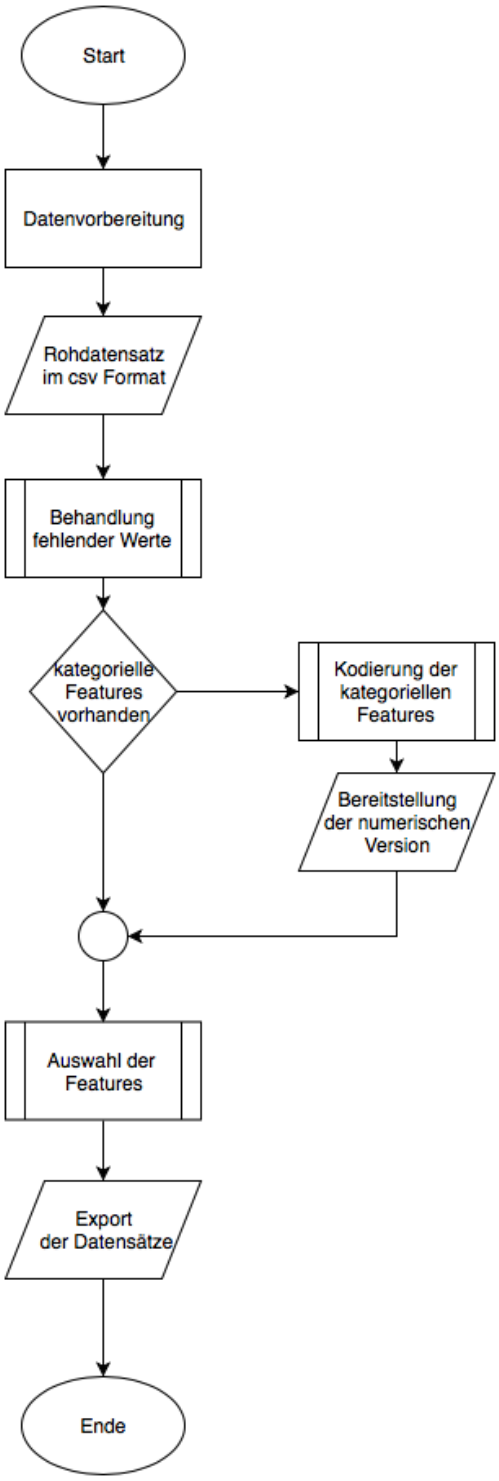


Abbildung 8.1.: Prozessablauf der Datenvorbereitung (offline)

Die Eingabe der Datenvorbereitung ist ein Datensatz im *csv* Format. Zunächst werden, falls vorhanden, die fehlenden Werte des Datensatzes ersetzt. Hierzu können verschiedene Strategien genutzt werden. Der einzige untersuchte Datensatz, der fehlende Werte enthält, ist der Lending Club Datensatz. Die explorative Datenanalyse zur Behandlung der fehlenden Werte erfolgt in einem Jupyter Notebook in Python durch Mittel der Bibliothek `pandas`. Die zur Behandlung dieser fehlenden Werte genutzten Strategien sind in Kapitel 4 beschrieben.

Der nächste in Abbildung 8.1 dargestellte Prozessschritt ist die Umwandlung der kategoriellen Features in eine numerische Darstellung. Hierzu wird für ordinale Features **Label Encoding** und für nominale Features **One Hot Encoding** verwendet. Zunächst muss ermittelt werden, welche kategoriellen Features ordinal oder nominal sind, damit die korrekte Kodierung gewählt werden kann. Dazu ist es notwendig, die Bedeutung der Features der Datensätze zu kennen. Eine Informationsquelle über die Bedeutung der Variablen in einem Datensatz sind sogenannte *Data Dictionaries*, in denen zu jedem Feature eine Beschreibung dieses Features zu finden ist. Auf dem Kreditkartentransaktionsdatensatz war der Schritt „Kodierung der kategoriellen Features“ nicht notwendig, da dieser Datensatz nur numerische Features enthält.

Die *csv* Datei des German Credit Dataset enthält im Ursprungszustand keine Namen der Features in der ersten Zeile. Allerdings konnten die Namen des Features mithilfe des *Data Dictionaries* zugeordnet werden. Im Anschluss wurden die wenig intuitiv benannten Ausprägungen, die aus dem Buchstaben A und einer fortlaufenden Zahl bestanden, mithilfe dem *Data Dictionary* in ihre Entsprechungen übersetzt. Dies wurde in R durch die Funktion *factor* realisiert. Als Argument wurden die neuen Ausprägungen angegeben. Das Ergebnis dieses Schrittes wurde im *csv* Format abgespeichert und bildet den Datensatz, der die kategoriellen Features enthält. Nun erfolgt die numerische Darstellung der kategoriellen Features. Die ordinalen Features des Datensatzes werden durch die Anwendung der *as.numeric* Funktion auf das jeweilige Feature in numerische Features umgewandelt. Dies funktioniert, da der Wert eines Faktor in R ein Integer ist. Für kategorielle Features, deren Ausprägungen keine Ordnung besitzen, wird durch die Funktion *dummyVars* des R Pakets `caret` ein **One Hot Encoding** durchgeführt. Hierbei wird ein Merkmal mit k Ausprägungen durch $k-1$ Merkmale mit den Ausprägungen 0 oder 1 ersetzt. Der daraus entstehende R Dataframe wird als csv exportiert und entspricht der numerischen Version des Datensatzes.

Auf dem Lending Club Datensatz erfolgt die Umwandlung von kategoriellen Features in numerische Features auf die gleiche Weise wie am Beispiel des German Credit Datensatz be-

schrieben.

Der nächste Schritt der Datenvorbereitung ist wie in Abbildung 8.1 dargestellt die Auswahl der Features, die zur Durchführung der Analyse verwendet werden sollen. Dieser Schritt wird auch als **Feature Selection** bezeichnet. Die Auswahl der Features muss als Bestandteil der Datenvorbereitung erfolgen, da der Prototyp diese Funktionalität nicht zur Verfügung stellt.

Der Output der Datenvorbereitung nach dem Prozessdiagramm aus Abbildung 8.1 ist der Export des ursprünglichen Datensatzes mit durchgeführter Selektion der Features und einer numerischen Version dieses Datensatzes, falls der Datensatz kategorielle Merkmale enthält.

8.2. Interaktive Versionen

Wie bereits einleitend erwähnt, wurden im Rahmen dieser Masterthesis zwei verschiedene interaktive Versionen zur Durchführung einer durch die topologische Datenanalyse unterstützte Klassifikation geplant und implementiert. Da die Unterschiede der beiden interaktiven Versionen nur die Interaktion mit dem Benutzer betreffen, ist das Design dieser beiden Versionen identisch. Bezüglich der Implementierung gibt es jedoch einige Unterschiede, weshalb die Beschreibung der Implementierung der interaktiven Versionen in zwei Teile untergliedert ist. Bevor die Klassifikation der Daten nach dem in Abbildung 8.2 dargestellten Prozess erfolgen kann, sind die in der Abbildung 8.1 beschriebenen Schritte zur Datenvorbereitung erforderlich.

8.2.1. Design der interaktiven Versionen

Eine Visualisierung des Designs des Prototyps zur Durchführung der Klassifikation unter Zuhilfenahme der topologischen Datenanalyse zeigt die Abbildung 8.2. Im Folgenden sollen die einzelnen Schritte des Prozesses näher erläutert werden. Eine Beschreibung der Implementierungsdetails findet sich im Abschnitt 8.2.2. Der Prototyp ist modular entwickelt, sodass diese Komponenten für alle vorgestellten Versionen genutzt werden können und nur die Schnittstellen gegebenenfalls angepasst werden müssen. Auf den folgenden Seiten wird das Design des Prototyps anhand der im Ablaufdiagramm aus Abbildung 8.2 dargestellten Schritte beschrieben.

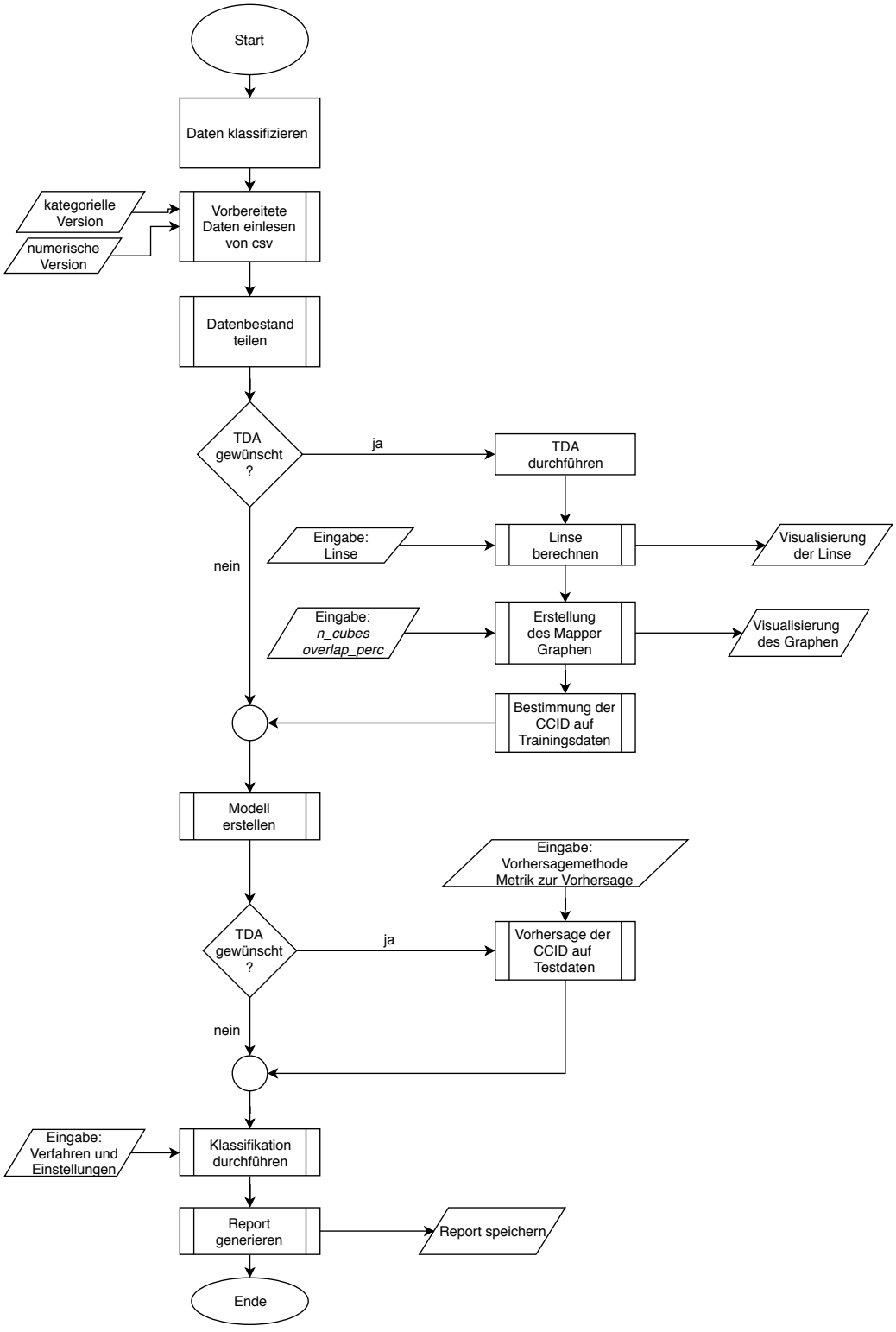


Abbildung 8.2.: Prozessablauf der Online-Analyse

Vorbereitete Daten einlesen von csv Datei

Der erste Prozessschritt aus dem Flussdiagramm in Abbildung 8.2 ist das Einlesen der zu klassifizierenden Daten. Enthält der Eingabedatensatz kategorielle Variablen, so ist die Bereitstellung einer numerischen Version dieses Datensatzes obligatorisch, wie bereits im Teil zur Datenvorbereitung beschrieben wurde. Die im Flussdiagramm dargestellte Eingabe der kategoriellen Daten kann entfallen, wenn als Klassifikator die logistische Regression mit Lasso Shrinkage gewählt wird. Bei Nutzung eines Entscheidungsbaumklassifikators liefert die Verwendung des Datensatzes mit kategoriellen Features eine besser lesbare Darstellung des Entscheidungsbaumes, da die Bedeutung des Vergleichswert bei der Durchführung der Splits direkt erkannt werden kann. Dieser Schritt entspricht dem Schritt 1 aus dem Prozessdiagramm 4.1 auf Seite 53. Das Laden der Daten geschieht in den jeweiligen Hauptprogrammen der unterschiedlichen Versionen.

Datenbestand teilen

Wie in der Abbildung 8.2 dargestellt, schließt sich eine zufällige Zuordnung der Daten zur Trainings- und Testdatenmenge an, um die Performance des Klassifikators auf einem unbekannten Datenbestand evaluieren zu können. Aufgrund der Unbalanciertheit des Targets der Datensätze unterschiedlichen Ausmaßes erfolgt die Teilung der Daten stratifiziert, das heißt, das Verhältnis der positiven Klasse zur negativen Klasse soll in der Trainings- als auch in der Testdatenmenge so groß sein, wie im Gesamtdatenbestand. Durch die Unbalanciertheit der Targets kann es sonst passieren, dass sich durch die zufällige Zuordnung beispielsweise keine Testobjekte der positiven Klasse im Testdatensatz befinden, was zu einem zu optimistischen Performanceindex führt. Das Teilen des Datenbestandes in Trainings- und Testmenge ist der 2. Schritt des Prozessdiagramms 4.1 auf Seite 53. Die Teilung der Datensätze erfolgt ebenfalls in den Hauptprogrammen der beiden interaktiven Versionen des Prototyps.

Durchführung der topologischen Datenanalyse

Um die Frage beantworten zu können, ob die topologische Datenanalyse die Qualität des Klassifikators verbessern kann, ist zum Vergleich das Training eines Klassifikators ohne topologische Informationen notwendig. Daher enthält das Ablaufdiagramm 8.2 zwei Stellen, an denen unterschieden wird, ob eine Durchführung der topologischen Datenanalyse gewünscht wird. Die Wahl, ob die topologische Datenanalyse eingesetzt werden soll, trifft der Benutzer über die Benutzeroberfläche. Entscheidet sich der Benutzer für die Durchführung der topologischen Datenanalyse, so beginnt der Prozess mit der Übergabe des Trainingsdatensatzes.

Dies ist im Flussdiagramm unter „TDA durchführen“ dargestellt und umfasst den Schritt 1 des Prozessdiagrammes 4.2 zur topologischen Datenanalyse.

Linse berechnen

Anschließend wird, wie in der Abbildung 8.2 dargestellt, die gewählte Linse berechnet. Dies geschieht für alle Linsen auf dem Cluster der Hochschule. Hierzu kommt die Klasse `LensCalculator` zum Einsatz, welche die benötigten Informationen, wie die Bezeichnung der gewählten Linse, die Parameter der gewünschten Linse sowie den Index der Objekte, die im Trainingsdatensatz enthalten sind, an das Cluster übermittelt und das Ergebnis der Berechnung für die weiteren Komponenten der implementierten Software bereitstellt. Die Klasse `LensCalculator` besitzt eine Methode zur Herstellung der Verbindung zum Cluster, eine Methode zur Durchführung der Berechnung, sowie eine Methode zur Beendigung der Verbindung. Zu diesem Schritt ordnet sich der Schritt 2 aus dem Prozessdiagramm der topologischen Datenanalyse aus Abbildung 4.2 auf Seite 54 zu. Die interaktiven Implementierungen bieten eine Visualisierung der Linse als Scatterplot, sodass der Benutzer beispielsweise passende Werte für die Überdeckung der Linse bestimmen kann.

Erstellung des Mapper Graphen

Nun erfolgt, wie in der Abbildung 8.2 dargestellt, die Erstellung des Mapper Graphens. Dies erfolgt durch die Klasse `KeplerMapperSplitted`. Die Klasse `KeplerMapperSplitted` wurde durch Vererbung von der Klasse KeplerMapper abgeleitet, damit eine Vorhersage des topologischen Features *connectedComponentId* auf dem Testdatensatz möglich ist, da hierzu Informationen benötigt werden, die die Klasse `KeplerMapper` bereitstellt. Um die gewünschte Funktionalität zu implementieren, war eine Anpassung der Methode *map* notwendig. Die Erstellung des Mapper Graphen besteht aus den Schritten (3), (4) und (5) des Prozessdiagrammes zur topologischen Datenanalyse in Abbildung 4.2. Eine Darstellung des Mapper Graphens bietet dem Benutzer als Zwischenergebnis die Möglichkeit, zu beurteilen, ob die Überdeckung der Linse geeignet gewählt wurde.

Bestimmung der CCID auf den Trainingsdaten

Aus dem im vorangegangen Schritt bestimmten Graph wird im nächsten Prozessschritt des Ablaufdiagrammes in Abbildung 8.2 mithilfe der Klasse `GraphAnalyser` die Zusammenhangskomponenten extrahiert, die das neue Feature CCID des Trainingsdatensatzes bilden. Dieser Schritt entspricht dem Schritt (6) des Prozessdiagrammes zur topologischen Datenanalyse auf Seite 4.2. Die Zuordnung der CCID erfolgt im Hauptprogramm.

Modell erstellen

Der eventuell durch die Ergebnisse der topologischen Datenanalyse angereicherte Trainingsdatensatz wird nun zum Training eines Klassifikators genutzt. Hierzu werden je nach gewähltem Klassifikationsverfahren unterschiedliche Klassen verwendet. Für Entscheidungsbäume kommt die Klasse `DecisionTreeBuilder` zum Einsatz, welche das Training des Entscheidungsbaums und die Vorhersage der Ergebnisse durchführt. Da kein in Python implementierter Entscheidungsbaumalgorithmus in der Lage ist, mit kategoriellen Merkmalen umgehen zu können, diese in den untersuchten Datensätzen jedoch enthalten sind, wird das Training des Entscheidungsbaumes und die Vorhersage der Klassen durch das Python Modul `rpy2` in R durchgeführt. Das Python Modul `rpy2` erlaubt die Verwendung von R Paketen direkt in der Programmiersprache Python. Die hierzu notwendigen Konvertierungen der Datentypen wie zum Beispiel von einem `pandas` Dataframe zum R Dataframe führt das `rpy2` Modul automatisch durch. Zur Bereitstellung der logistischen Regression mit LASSO zur Klassifikation von Daten wird die Klasse `LogisticRegressionBuilder` implementiert. Diese führt das Training des Modells und die Vorhersage der Klassen in Python durch und bedient sich hierzu der Funktionalität des Pythonbibliothek `statsmodels`. Der Grund für die Wahl dieser Bibliothek zur Implementierung der logistischen Regression mit Regularisierung besteht darin, dass wie aus R oder anderen Statistikprogrammen bekannt, eine Ausgabe einer Zusammenfassung des trainierten Modells erfolgt, welche die Werte der geschätzten Koeffizienten der Regressionsfunktion und deren Signifikanz angibt. Die Erstellung des Modells ist in dem Prozessdiagramm des Gesamtprozesses auf Seite 53 im Schritt (4) thematisiert. Zum Training der Modelle stellen beide Klassen die Methode *fit* zur Verfügung

Vorhersage der CCID auf den Testdaten

Für die Anwendung des trainierten Modells auf den Testdatensatz bei durchgeführter topologischer Datenanalyse ist es notwendig, das aus der topologischen Datenanalyse erhaltene neue Feature auf dem Testdatensatz vorherzusagen. Diese Funktionalität stellt die Klasse `ConnectedComponentPredictor` bereit. Zur Vorhersage der CCID wurden zwei Methoden verwendet, die beide in der Klasse `ConnectedComponentPredictor` implementiert sind. Die eine Methode ordnet jedes Objekt des Testdatensatz zu dem nächstgelegenen Clustermittelpunkt zu. Diese Cluster entsprechen den Nodes im Mapper Graphen, die jeweils zu einer zusammenhängenden Komponente gehören. Somit wird jedem Objekt des Testdatensatzes die CCID des Clusters zugewiesen, in dem das jeweilige Objekt verortet ist. Eine Alternative für die Vorhersage der CCID ist die Vorhersage auf Basis des Abstandes zwischen den Testobjekten und dem Trainingsobjekten. Die CCID eines Objektes aus dem Testdatensatz entspricht der CCID des nächstgelegenen Objekt im Trainingsdatensatz. Die in diesem Schritt durchgeführte Vorhersage der CCID auf dem Testdatensatz entspricht dem Schritt (5) im Prozessdiagramm des Gesamtprozesses auf Seite 53.

Klassifikation durchführen

Im nächsten Schritt folgt nun die Anwendung des Modells auf Trainings- und Testdatensatz. Dieser Schritt geschieht, wie das Training der Klassifikation auch, durch die Methode *predict* der Klassen `DecisionTreeBuilder` beziehungsweise `LogisticRegressionBuilder`.

Report generieren

Nach Berechnung der Klassifikationsergebnisse auf Trainings- und Testdatenmenge wird durch die Klasse `ModelEvaluator` ein Report erstellt, der Informationen zur Performance des Klassifikators beinhaltet und als PDF Dokument gespeichert wird. Die Evaluierung des Klassifikators und die Generierung des Reports ist der 6. Schritt des in Abbildung 4.1 auf Seite 53 dargestellten Gesamtprozess der durch die topologische Datenanalyse unterstützte Klassifikation.

8.2.2. Implementierung der interaktiven Versionen

Nachdem das Design der interaktiven Versionen beschrieben wurde, wird in diesem Teil genauer auf die Implementierung der interaktiven Prototypen eingegangen. Die Implementierungsdetails vieler Komponenten wie die Durchführung der topologischen Datenanalyse und

die weiteren Funktionen, die nicht zur Benutzeroberfläche zählen, unterscheiden sich zwischen den beiden interaktiven Versionen nicht. In diesem Teil wird zunächst die Implementierung der Komponenten beschrieben, die in beiden interaktiven Versionen zu finden sind. Die Reihenfolge der Vorstellung der Implementierung richtet sich nach dem in Abbildung 8.2 gezeigten Ablaufplan. Im Anschluss daran werden die Implementierungsdetails in separaten Teilen der beiden interaktiven Versionen vorgestellt. Hierbei wird besonders auf die Implementierung der Benutzeroberfläche, die Visualisierung der Linse und die Visualisierung des Mapper Graphs eingegangen.

Vorbereitete Daten einlesen und Datenbestand teilen

Die Prozessschritte „Vorbereitete Daten einlesen von csv“ und „Datenbestand teilen“ werden im Hauptprogramm durchgeführt, welches der Benutzer zum Start des Programmes verwendet. Das Einlesen der Datensätze in der Originalversion und der numerischen Version geschieht durch die Funktion *read_csv* der Bibliothek `pandas`, die eine Datei im *csv* Format in einen `pandas` Dataframe umwandelt [34]. Ebenfalls im Hauptprogramm durchgeführt wird die Teilung des Datenbestandes in Trainings- und Testdatenmenge. Hierfür kommt die Funktion *train_test_split* der Bibliothek `scikit-learn` zum Einsatz, welche durch das Setzen des Parameters *stratify* ein stratifiziertes Sampling des Datensatzes durchführt [42].

Durchführung der topologischen Datenanalyse

Der nächste Schritt ist die Durchführung der topologischen Datenanalyse. Hierzu wurde die Klasse `TDAPerformer` implementiert, welche die für die Durchführung der topologischen Datenanalyse notwendigen Funktionen bereitstellt und die Funktionen anderer Klassen aufruft, die diejenige Funktionalität bereitstellen. Das Kernstück der topologischen Datenanalyse ist in dieser Arbeit die Erstellung des Mapper Graphen, die durch die Klasse `KeplerMapperSplitted` bewerkstelligt wird.

Linse berechnen

Zur Durchführung der topologischen Datenanalyse ist als erster Schritt die Berechnung der verwendeten Linse notwendig. Aufgrund des hohen Aufwandes der Berechnung dieser Dimensionsreduktionsverfahren wird die Berechnung auf dem Cluster der Hochschule Darmstadt durchgeführt. Um eine bereits berechnete Linse nicht nochmals berechnen zu müssen, werden berechnete Linsenergebnisse auf dem Filesystem des Clusternodes im *csv* Format persistiert.

Für das Starten der Berechnung auf der externen Maschine und das Entgegennehmen des Ergebnisses der Linsenberechnung wurde die Klasse `LensCalculator` implementiert. Die Klasse `LensCalculator` verwendet das Python Modul `rpyc` um die Berechnung der Linsen auf einem externer Maschine durchzuführen. Für jede der vier in dieser Arbeit untersuchten Linsen wurde eine Funktion implementiert, die die Berechnung der Linse übernimmt und das Ergebnis zurückgibt. Diese Funktionen werden lokal implementiert und durch die Methode *teleport* des Connection-Objektes, welches bei der Herstellung der Verbindung erzeugt wird, zum Clusternode übermittelt. Die Parameter dieser Funktionen sind der Speicherort des verwendeten Datensatzes auf der entfernten Maschine und den Namen des Targets, die Indexe der Objekte im Trainingsdatensatz, die Bezeichnung der zu berechneten Linse und deren Parameter sowie eine Angabe, ob die Daten vor der Berechnung der Linse standardisiert werden sollen. Diese Standardisierung erfolgt durch die Klasse `StandardScaler` der Bibliothek `scikit-learn`. Der Name des Targets wird benötigt, um dieses bei der Berechnung der Linsen ausschließen zu können. Zur Implementierung der Linsen wurden verschiedene Module verwendet. Die Visualisierung der Linsenergebnisse erfolgt, je nach interaktiver Version, auf unterschiedliche Weisen. Die Implementierungsdetails hierzu werden in den Teilen 8.2.2 und 8.2.2 näher beleuchtet.

Zur Berechnung der **linearen Diskriminanzanalyse** wurde die Klasse `LinearDiskriminantAnalysis` der Bibliothek `scikit-learn` verwendet. Die durch den Aufruf der Funktion *fit_transform* wird die lineare Diskriminanzanalyse auf den Datensatz angewandt. Im Gegensatz zu den anderen Linsen muss bei der linearen Diskriminanzanalyse das Target zur Verfügung stehen, da das Ziel der linearen Diskriminanzanalyse eine Transformation der Daten ist, sodass der Abstand zwischen den Gruppen maximal ist.

Die nächste verwendete Linse ist eine Kombination eines Isolation Forests mit der L_2-Norm des Trainingsdatensatzes. Diese beiden Linsenbestandteile werden unabhängig voneinander berechnet und in einem `numpy` Array zusammengefasst. Die Berechnung des Isolation Forest erledigt die Klasse `IsolationForest` der Bibliothek `scikit-learn`. Die Bestimmung der Abnormalitätsscores, welche die erste Linsenkomponente der Isolation Forest und L_2-Norm Linse darstellt, geschieht durch die Funktion *decision_function* des Objektes der Klasse `IsolationForest`. Die Berechnung der L_2-Norm des Datensatzes erfolgt durch die *fit_transform* Funktion des KeplerMappers durch Setzen des Parameters *projection='l2norm'*. Die beiden Linsenbestandteile werden nun durch die in der Biblothek `numpy` implementierte Funktion *c_* spaltenweise zu einem Aray zusammengefügt, welches den Output dieser Lin-

senkombination enthält.

Die Dimensionsreduktion durch den TSNE Algorithmus erfolgt durch die Klasse `MulticoreTSNE`, welche in dem gleichnamigen Modul implementiert ist. Die Wahl dieser Klasse begründet sich darauf, da andere Implementierungen auf den gegebenen Daten zu schlecht skalierten, wie sich bei dem Vergleich der Implementierungen von TSNE in der Bibliothek `sklearn` mit denen der Module `MulticoreTSNE` und `bhtsne` herausstellt. Die Ergebnisse dieser Experimente sind im Anhang A wiedergegeben. Bei der verwendeten `MulticoreTSNE` Implementierung handelt es sich um eine Implementierung, welche zur Berechnung der Dimensionsreduktion mehrere Kerne verwendet. Die Berechnung der TSNE Dimensionsreduktion erfolgt durch die Funktion *fit_transform*. Für die Berechnung der UMAP Linse kommt das Modul `umap-learn` zum Einsatz.

Erstellung des Mapper Graphen

Aus der im letzten Schritt berechneten Linse erstellt die Klasse `KeplerMapperSplitted`, die aus der Klasse `KeplerMapper` durch Vererbung entsteht, einen Graphen. Die Vererbung wurde implementiert, da die Klasse KeplerMapper nicht in der Lage ist, die für die Vorhersage der CCID notwendigen Information bereitzustellen. Dies ist so, da die Klasse KeplerMapper nur zur Visualisierung der Graphen implementiert wurde, jedoch nicht zur Ableitung eines Features aus dem Graphen. Die Änderung der *map* Funktion der Klasse KeplerMapperSplitted besteht aus dem Hinzufügen der neuen Variable *cluster_centroid_dict*, welche die Namen und deren Zentroide aller Nodes im Graphen in Form eines Python Dictionaries enthält. Damit auf diese Information von außerhalb der Klasse zugegriffen werden kann, liefert die *map* Methode der Klasse `KeplerMapperSplitted` neben dem generierten Mapper Graph das beschriebene Dictionary als zweiten Rückgabewert zurück. Die Implementierung der Visualisierung der Mapper Graphen ist, wie bei den Linsenergebnissen bei der Version mit `tkinter` und der Version mit `Dash` unterschiedlich. Daher wird die Implementierung dieser Komponenten im Teil 8.2.2 beziehungsweise 8.2.2 beschrieben.

Bestimmung der CCID auf den Trainingsdaten

Das durch die topologische Datenanalyse generierte neue Feature CCID entspricht der ID der zusammenhängenden Komponente des Graphens in der der Node liegt, der das jeweilige Datenobjekt enthält. Dafür ist die Analyse der zusammenhängenden Komponenten des Mapper Graphs notwendig. Um diese Funktionalität bereitzustellen, wurde die Klasse `GraphAnalyser` erstellt, welche mithilfe des Moduls `networkx` die Analyse der Zusammenhangskomponen-

ten übernimmt. Hierzu ist es notwendig den Mapper Graphen in die Darstellung eines Graphen in dem Modul `networkx` zu übersetzen. Dieser Schritt erfolgt mithilfe der Funktionen *add_nodes_from* und *add_edges_from* eines Graphen in dem Modul `networkx`. Zur Überprüfung dieser Konvertierung wurde eine Möglichkeit der Visualisierung des importierten Graphs durch die Funktion *draw* des `networkx` Moduls geschaffen. Die Analyse der Zusammenhangskomponenten des Graphen erfolgt durch die Funktion *compute_connected_components*, deren Output für die weitere Verarbeitung in einem Python Dictionary bereitgestellt wird, welches die Namen der Nodes und die zugehörige ID der zusammenhängenden Komponente enthält. Mithilfe der zusammenhängenden Komponenten aus dem letzten Schritt wird nun durch die Methode *createConnectedComponentId* über Datensatzoperationen der Bibliothek `pandas` die CCID des Trainingsdatensatzes bestimmt. Hierzu wird der Datensatz nicht benötigt, da alle Informationen bereits im Graphen und der Analyse der zusammenhängenden Komponenten enthalten sind. Zur Einfärbung der Nodes wird allerdings in diesen Schritt das Target benötigt.

Modell erstellen

Die Erstellung des Klassifikationsmodells erfolgt, wie bereits erwähnt durch die Klassen `DecisionTreeBuilder` beziehungsweise `LogisticRegressionBuilder`, je nachdem welcher Klassifikationsalgorithmus vom Benutzer des Prototypen ausgewählt wurde. Zur Wahl steht ein Entscheidungsbaumklassifikator und eine logistische Regression mit LASSO Shrinkage. Da kein Python Modul zur Verfügung steht, welches Entscheidungsbäume erstellen kann und mit kategoriellen Daten umgehen kann, wurde der Umweg über R gewählt. Hierzu wurde das Modul `rpy2` verwendet, welche die Funktionalität von R in Python zur Verfügung stellt. Eine Voraussetzung des Moduls `rpy2` ist eine Installation der Software R. Als R Version wurde die Version 3.5.1 eingesetzt.

Zur Berechnung der Entscheidungsbäume wird das R Paket `rpart` verwendet. Zur Visualisierung des trainierten Entscheidungsbaumes werden noch die R Pakete `rpart.plot` und `grDevices` genutzt. Das Importieren der Pakete erfolgt in Python über die Funktion *importr*. Die Funktion rpart besitzt einen Parameter *control* zur Spezifizierung der Einstellungen des Entscheidungsbaumes wie etwa die maximale Tiefe des Baumes oder der Komplexitätsparameter *cp*. Diese Einstellungen werden in Python in einem Dictionary getroffen und durch eine Konvertierung in die Klasse ListVector für R nutzbar. Vor dem Training beziehungsweise der Vorhersage des Entscheidungsbaumes wird der Datentyp der kategoriellen Features CCID von einem String in den Datentyp *'category'* geändert, sodass die kategoriellen Features in R als Faktoren behandelt werden. Die CCID wird, falls die topologische Datenanalyse durchgeführt

wurde, ebenfalls in eine Kategorie umgewandelt.

Die Bereitstellung der logistischen Regression mit LASSO geschieht durch die Klasse `Logit` der Bibliothek `statmodels`. Da die verwendete Implementierung nicht direkt mit kategoriellen Variablen arbeiten kann, muss die CCID als kategorielles Feature einem **One Hot Encoding** unterzogen werden, sodass die Informationen der kategoriellen Variable CCID durch die logistische Regression genutzt werden können. Das **One Hot Encoding** erfolgt durch die Verwendung der Funktion *get_dummies* der Bibliothek `pandas`. Das Training des Modells geschieht mittels der Funktion *fit_regularized*, welche die Parameter der logistischen Regression schätzt. Bei der Optimierung der Parameter bezüglich einer Zielfunktion werden, wie bei LASSO üblich, große Werte der Parameter bestraft. Die Stärke dieser Penalisierung regelt der Parameter *alpha*, der durch den Benutzer vorgegeben werden kann. Der Defaultwert für *alpha* ist 50. Weitere Parameter, die der Benutzer eingeben kann, sind die Toleranz bei der Optimierung sowie die maximale Anzahl von Iterationen. Der Defaultwert für die Toleranz entspricht 0.001, die Anzahl der Iterationen beträgt im Default 500. Die Übergabe dieser Parameter an die Funktion fit der Klasse `LogisticRegressionBuilder` erfolgt in einem Python Dictionary.

Vorhersage der CCID auf den Testdaten

Für die Vorhersage des neuen, topologischen Features existiert für jede der beiden Varianten der Vorhersage ein Funktion der Klasse `ConnectedComponentPredictor`. Die Vorhersage auf Basis der CCID auf dem Trainingsdatensatz verwendet die in der Bibliothek `scikit-learn` implementierte Funktion *pairwise_distances_argmin*, um die ID des nächsten Nachbarn des Trainingsdatensatzes von jedem Objekt des Testdatensatzes herauszufinden. Dies erfordert die Angabe einer Metrik, die für die Berechnung des Abstandes verwendet werden soll. Das Resultat der Benutzung dieser Funktion ist ein Vektor, der die IDs der Nachbarn aus dem Trainingsdatensatz enthält. Die ermittelten Indexe werden nun dazu genutzt, die entsprechenden Werte des topologischen Features auf dem Trainingsdatensatz zu ermitteln, welche der Vorhersage auf dem Testdatensatz entsprechen. Die Rückgabe der vorhergesagten CCID erfolgt in Form eines numpy Arrays, welches die Werte des neuen topologischen Features auf dem Testdatensatz liefert. Diese Variante ist allerdings bei großen Datensätze sehr aufwendig, da sie eine Berechnung der Abstände von jedem Objekt im Testdatensatz zu jedem Objekt im Trainingsdatensatz erfordert. Aus diesem Grund wird die Verwendung dieser Vorhersagemethode für die CCID nicht empfohlen. Während eines Tests auf dem Kreditkartentransaktionsdatensatz mit etwa 280.000 Objekten ergibt sich eine Laufzeit von über 50 Minuten, während die

zentroidbasierte Methode in unter einer Minute fertig ist.

Eine weitere Methode, die eine bessere Performance liefert ist die zentroidbasierte Methode. Hierbei werden die Datensätze des Testdatensatzes zunächst den Nodes zugeordnet. Dies funktioniert, da ein Node ein Cluster von Datenpunkten repräsentiert, welche ein Zentroid besitzen. Diese Zentroide lassen sich innerhalb der *map* Funktion der Klasse `KeplerMapperSplitted` durch den Rückgabewert *cluster_centers_* der Instanz, die das Clustering der Daten innerhalb einer Zelle der Überdeckung der Linse durchführt, bestimmen und zusammen mit der Bezeichnung des Nodes in einem Dictionary speichern, welches als Parameter der Funktion zur Zuordnung der Testdatenobjekte zu den Nodes übergeben wird. Die Zuordnung der Objekte des Testdatensatzes zu den Nodes erfolgt durch die Berechnung der Abstände der Datenpunkte zu den jeweiligen Zentroiden durch die Funktion *cdist* der Bibliothek `scipy`. Ein Datenpunkt wird zu demjenigen Node zugeordnet, dessen Clusterzentroid ihm am nächsten ist, welches durch die Funktion *idxmin(axis=1)* realisiert wird. Die Anzahl der Spalten der Matrix entspricht der Anzahl der Nodes im Graphen, die Anzahl der Zeilen entspricht der Anzahl der Objekte im Testdatensatz. Die Funktion *idxmin(axis=1)* liefert den Namen der Spalte zurück, der in einer Zeile der Matrix den geringsten Wert aufweist. Aus der Zuordnung zu einem Node kann nun durch das Dictionary, welches zu jedem Node, die ID der zusammenhängenden Komponente enthält, die zugehörige CCID ermittelt werden. Das Dictionary, welches zu jedem Node die ID der zugehörigen zusammenhängenden Komponente enthält ist durch die Klasse `GraphAnalyser` bestimmt worden. Die Ausgabe der im `ConnectedComponentPredictor` implementierten Funktion zur Vorhersage der CCID durch den zuletzt beschriebenen Weg erfolgt als numpy Array, welches für jedes Objekt im Testdatensatz eine Vorhersage des neuen topologischen Features enthält. Diese Variante der Vorhersage ist effizienter, da die Distanz von jedem Objekt des Testdatensatzes nur zu einer geringeren Anzahl von Zentroiden berechnet werden muss, die der Anzahl von Knoten entspricht.

Klassifikation durchführen

Nachdem das topologische Feature auf dem Testdatensatz vorhergesagt wurde, steht nun die Anwendung des trainierten Modells an. Hierzu besitzen die Klassen `DecisionTreeBuilder` und `LogisticRegressionBuilder` jeweils die Methode *predict*, welche die Ergebnisse der Klassifikation berechnet. Die *predict* Methoden können sowohl die vorhergesagte Klasse, als auch einen Score ausgeben, der als Wahrscheinlichkeit interpretiert werden kann, dass das jeweilige Objekt positiv vorhergesagt wird. Die Wahl, welche Ausgabe gewünscht ist, geschieht

über den Parameter *type*. Sowohl beim Entscheidungsbaum als auch bei der logistischen Regression ist das neu hinzugekommene Feature analog zum Trainingsdatensatz in eine andere, auf den jeweiligen Klassifikationsalgorithmus abgestimmte Darstellung zu überführen. Beim Entscheidungsbaum erfolgt die Berechnung der Klassifikationsergebnisse durch die Funktion *predict_rpart* in R. Ist die Ausgabe der vorhergesagten Klasse erwünscht, so muss diese Ergebnis in ein numpy Array übersetzt und in jeder Komponente des Vektors 1.0 subtrahiert werden, da die Darstellung der Klassen in R durch die Werte 1 oder 2 erfolgt, jedoch die Werte 0 oder 1 gewünscht sind. Soll hingegen der Score ausgegeben werden, so liefert die Funktion *predict_rpart* für den Entscheidungsbaum jeweils die Wahrscheinlichkeiten für beide Klassen zurück. Die Wahrscheinlichkeit für eine positive Klassifikation findet sich daher in der zweite Spalte der retournierten Matrix. Bei der logistischen Regression erfolgt die Vorhersage der Klassifikationsergebnisse mithilfe der Funktion *predict* der Klasse `Logit`. Da die Ausgabe der verwendeten Implementierung nur einen Scorewert umfasst, muss dieser durch Setzen eines Schwellenwerts auf den Wert von 0,5 in eine binäre Klassifikation umgewandelt werden.

Report generieren

Um Informationen über die Qualität des trainierten Klassifikators zusammenzufassen, erstellt die Klasse `ModelEvaluator` einen Report im PDF Format, welcher die gewählten Einstellungen und die wichtigsten Performancemaße inklusive ROC Curve und Precision Recall Diagramm enthält. Darüber hinaus enthält der Report wahlweise eine Darstellung des Modells. Bei der Verwendung eines Entscheidungsbaumklassifikators ist dies ein Plot des Baumes, bei der logistischen Regression besteht diese Zusammenfasuung des Modells aus den Informationen zu den geschätzten Koeffizienten der Regressionsfunktion, wie sie durch die Bibliothek `statmodels` berechnet wurden.

Die Erstellung des PDF Reports geschieht durch LaTeX Templates, welche durch die Templating Engine `Jinja2` gerendert werden, sodass die gewünschte Ausgabe im Report erscheint. Diese Strategie erlaubt es, dass die Klasse zur Erstellung des zusammenfassenden Dokuments zu den Ergebnissen des Klassifikators nicht geändert werden muss, egal welche Version des Prototypen der Benutzer wählt. Durch die üblichen Kontrollstrukturen wie Verzweigungen und Schleifen kann der Inhalt des Reports dynamisch erstellt werden. Beispielsweise erfolgt die Ausgabe der Einstellungen durch ein Dictionary, dessen Inhalt durch eine for Schleife über alle Schlüssel und Werte im gerenderten Dokument ausgegeben wird. Die Klasse `ModelEvaluator` stellt Funktionen zur Berechnung der Performancemaße und zur Generierung der ROC Curves beziehungsweise Precision Recall Diagramme bereit und

stellt diese in einem Report Chunk zusammen. Ein Report Chunk enthält die Auswertung eines Modells auf Trainings- und Testdatenmenge. Wird Crossvalidierung verwendet, wie es die *batch* Implementierung des Prototyps erlaubt, so enthält der Report mehrere Chunks. Ein Chunk enthält jeweils eine Tabelle der Performancemaße Accuracy, f_1-Score, f_2-Score, AUROC und AUPRC jeweils auf Trainings- und Testdatensatz und, falls gewünscht, Plots der ROC Curve und der Precision Recall Curve. Die Erstellung eines solchen Chunks erfolgt durch die Funktion *createChunk* der Klasse `ModelEvaluator`. Die Funktion *createReport* fasst die Ergebnisse aller Chunks zusammen und erstellt den fertigen Report mit Darstellung von Informationen zum Namen des Datensatzes und der gewählten Einstellungen. Darüber hinaus bindet diese Funktion auch die Informationen zu den Modellen in den Report ein. Nachdem das Template des Report durch die Funktion *render* der Klasse `Template` gerendert wurde, wird dieser String in einer *.tex* Datei gespeichert und anschließend durch den Aufruf des Programmes `pdflatex` eine PDF Datei erzeugt. Der Aufruf des Programmes `pdflatex` erfolgt über die Klasse `Popen`, die im Modul `subprocess` implementiert ist und das Ausführen von Befehlen auf der Kommandozeile in Python erlaubt. Nach Kompilierung der *.tex* Datei wird diese gelöscht. Die fertige PDF Datei wird nun auf der Festplatte des Computers abgelegt.

Implementierung als GUI mit `tkinter`

Wie bereits beschrieben, unterscheidet sich die Implementierung der in Teil 8.2.2 beschriebenen Komponenten zwischen den beiden interaktiven Versionen nicht. Daher wird in diesem Teil nur eine Beschreibung der Implementierung der Benutzeroberfläche, der Visualisierung der Linsenergebnisse und des Mapper Graphen wiedergegeben. Zur Implementierung der grafischen Benutzeroberfläche wurde das Modul `tkinter` verwendet. Die Implementierung der Benutzeroberfläche geschieht direkt im Hauptprogramm durch Konstruktoraufrufe der Klassen, welche einen neuen Frame erzeugen oder ein Kontrollelement wie einen Button auf diesem Frame platzieren. Die Erzeugung dieser Frames und Kontrollelemente erfolgt dynamisch innerhalb von Funktionen, welche durch den Parameter *command* eines Buttons ausgeführt werden. Die Eingaben in jedem Schritt werden durch einen Klick auf Buttons als globale Variable bereitgestellt. Das Programm präsentiert zunächst einen Öffnen-Dialog in dem der Benutzer die Auswahl des Datensatzes im csv Format treffen kann, welcher zur Durchführung der Klassifikation genutzt werden soll.

Im Anschluss daran öffnen sich nacheinander zwei Dialogboxen, die die Eingabe des Namens des Targets und der Größe des Trainingsdatensatzes erfordern. Die Eingabe des Targets erfolgt durch ein Dropdownmenü, welches die Namen aller Features des ausgewählten Datensatzes

als Optionen enthält. Zur Implementierung der Abfrage der Größe des Trainingsdatensatzes wurde die Funktion *askstring* der Klasse `simpledialog` eingesetzt. In einer weiteren Dialogbox wählt der Benutzer aus, ob die Klassifikation durch eine Anwendung der topologischen Datenanalyse unterstützt werden soll. Ist dies nicht der Fall, werden in einem weiteren **Frame** Informationen zu dem zu verwendenden Klassifikationsalgorithmus abgefragt. Diese Informationen umfassen die Art des Klassifikators (logistische Regression mit LASSO oder Entscheidungsbaum) und die Parameter dieser Algorithmen. Mit einem Klick auf den Button **Run Classification** wird die Klassifikation durchgeführt und der Report erzeugt.

Hat der Benutzer jedoch die Durchführung der topologischen Datenanalyse, so erscheint, falls der bei Programmstart ausgewählte Datensatz kategorielle Variablen enthält, ein weiterer Öffnen Dialog, welche die Auswahl der numerischen Version des Datensatzes erlaubt. Diese numerische Version wurde, wie in der Abbildung 8.1 während der Datenvorbereitung, bereitgestellt. Im Anschluss daran erhält der Benutzer die Möglichkeit, die Linsen durch Anklicken der Checkboxen auszuwählen, die berechnet und im Anschluss als Scatterplot visualisiert werden sollen. Die Visualisierung der Linsen erfolgt hier durch die Klasse `LensVisualizer` im Browser. Die Klasse `LensVisualizer` nutzt zur Darstellung der Scatterplots der Linsen das Modul `bokeh`, welches interaktive Visualisierungen erstellen kann [8].

Auf Basis dieser Visualisierungen kann der Benutzer nun in einem weiteren Frame eine Linse auswählen, um einen Mapper Graphen zu erstellen. Im gleichen Frame erfolgt auch die Eingabe der Anzahl der Intervalle und die Stärke der Überlappung zur Überdeckung der verwendeten Linse. Möchte der Benutzer eine Überdeckung der Linse mit Rechtecken durchführen, so kann dies durch Angabe einer mit Semikolon separierten Folge von Zahlen erfolgen. Im gleichen Frame erfolgt darüber hinaus die Wahl der Methode und die hierfür genutzte Metrik zur Vorhersage der CCID auf dem Testdatensatz in Form von Dropdownmenüs. Sollte der mithilfe der Funktion *visualize* der Klasse `KeplerMapper` dargestellte Graph nicht zusagen, so kann durch Klick auf den Button **Back** ein Mapper Graph mit anderen Einstellungen generiert werden. Ist der Benutzer mit dem Graphen zufrieden, so wird durch einen Klick des Buttons **Proceed to Classification** der Frame geöffnet, welcher auch bei der Klassifikation ohne TDA zur Auswahl und Parametrisierung des Klassifikators dient.

Die Durchführung der Klassifikation und die Erstellung des Reports der Ergebnisse erfolgt, wie im Fall ohne die Durchführung der topologischen Datenanalyse, durch einen Klick auf die Schaltfläche **Run Classification**.

Implementierung als Weboberfläche mit Dash

Das Design der Variante als Webdashboard entspricht dem in Teil 8.2 vorgestellten Design, da sich nur die Implementierung der Benutzeroberfläche ändert. Für die einzelnen Komponenten des Prototyps gelten daher die in Teil 8.2.2 vorgestellten Hinweise zur Implementierung. In diesem Teil soll nun die Implementierung der Benutzeroberfläche thematisiert werden.

Aufgrund der Tatsache, dass der gesamte Prototyp in der Programmiersprache Python implementiert ist, erfolgt die Realisierung der Benutzeroberfläche in `Dash`. `Dash` dient zur Generierung analytischer Anwendungen im Webbrowser und wurde im Jahr 2016 von Chris Parmer vorgestellt [40]. Die Programmierung einer `Dash` Anwendung geschieht in zwei Schritten. Zunächst wird das Layout der Anwendung erstellt. Dies erfolgt unter anderem durch die in den Modulen `dash_core_components` und `dash_html_components` implementierten Komponenten, welche beispielsweise Buttons, Eingabefelder und Dropdownmenüs im Webdashboard darstellen. Die Erstellung des Layouts ist nah an der Erstellung einer Webseite mit HTML. Die HTML Tags wurden in Dash durch Klassen mit dem gleichen Namen ersetzt.

Interaktiv wird das Dashboard durch die Definition von Callbacks, welche die Eigenschaften der Komponenten, die im Layout definiert wurden, in Abhängigkeit der Werte anderer Komponenten verändern können. In Programm 8.1 ist ein Beispiel eines solchen Callbacks abgedruckt. Dieser Callback parst den Inhalt der *csv* Datei mit dem Namen *filename* des Datensatzes in einen Dataframe, um diesen im Anschluss als JSON serialisiert in der Komponente *input_data_buffer_tda* für die weiteren Aufgaben zur Verfügung zu stellen. Dies ist notwendig, da das Setzen von Variablen, auf die alle Callbacks Zugriff haben, in Dash aufgrund der parallelen Ausführung des Programms nicht möglich ist. Somit müssen alle Daten und Zwischenergebnisse, die von anderen Schritten benötigt werden, innerhalb der Webseite in einer sogenannten *hidden Div* Komponente gespeichert werden. Eine **hidden Div** Komponente enthält Daten als String, deren Inhalt jedoch im Browser nicht sichtbar ist. Für die Ablage der Daten in dieser Komponente sind Funktionen zur Serialisierung von Numpy Arrays implementiert worden, da diese beispielsweise die Linsenergebnisse enthalten.

```
1  @app.callback(Output('input_data_buffer_tda', 'children'),
2                [Input('upload-data-tda', 'contents'),
3                 Input('upload-data-tda', 'filename')])
4  def update_data(contents, filename):
5      if contents:
6          json_data = parse_contents(contents, filename)
7          return json_data
```

Programm 8.1: Beispiel eines Callbacks

Wie in Programm 8.1 dargestellt, erfolgt die Definition der Callbacks durch die Funktion *@app.callback*, welche als Argumente eine Instanz der Klasse Output und eine Liste von Input Instanzen besitzt. Zusätzlich kann auch eine Liste von Stateobjekten übergeben werden. Stateobjekte übergeben nur ihren Wert und triggern nicht die Ausführung der Callbackfunktion. Dies ist relevant, wenn die Callbackfunktion rechenintensiv ist und diese Berechnung nur erfolgen soll, wenn der Benutzer alle Parameter gesetzt hat. Als Argumente der Konstruktoren der Klassen `Output`, `Input` und `State` wird sowohl der Name der Komponente als auch die Eigenschaft übergeben, welche für das Callback relevant ist.

Nach dem Start des Programms wählt der Benutzer durch einen Klick auf einen Button aus, ob die Klassifikation mit oder ohne topologische Datenanalyse durchgeführt werden soll. Daraufhin wird das gewünschte Layout geladen und angezeigt. In jedem der beiden Layouts wählt der Benutzer über eine Uploadkomponente den Datensatz aus, welcher zur Klassifikation genutzt werden soll. Daraufhin werden unterhalb der Uploadkomponente die ersten 10 Datensätze in einer Tabellenansicht dargestellt und die Features des gewählten Datensatzes als Optionen im Dropdownmenü zur Auswahl des Targets dargestellt. Die Wahl der Größe des Trainingsdatensatz geschieht über einen Schieberegler, dessen Default auf 60% eingestellt ist. Hat der Benutzer die Durchführung der topologischen Datenanalyse gewählt, so wird, wenn die bereits hochgeladenen Daten nicht numerische Features enthalten, dem User eine Schaltfläche zum Hochladen der numerischen Version angeboten. Ist die numerische Version verfügbar, so folgt die Auswahl der Linse und die Eingabe der Linsenparameter. Die Visualisierung der Ergebnisse der Dimensionsreduktion erfolgt in einer `plot.ly` Graphkomponente, welche in das Layout der Webapplikation eingebunden ist.

Darunter erfolgt die Spezifizierung der Einstellungen zur Generierung des Mapper Graphens. Die Eingabe erfolgt über zwei Eingabefelder. Sollen in jeder Dimension unterschiedliche Intervallanzahlen für die Überdeckung verwendet werden, so sind die Anzahlen der Intervalle in jeder Dimension mit Semikolon getrennt in das Eingabefeld einzugeben. Die Darstellung des Graphen in der Webanwendung erfolgt über die Einbindung als `IFrame`.

Nach Wahl der Parameter für die Vorhersage des neuen Features wird der Klassifikationsalgorithmus bestimmt. Die Einstellungsmöglichkeiten entsprechen hier denen der anderen Implementierungen. Durch einen Klick auf die Schaltfläche *Classify* wird die Klassifikation durchgeführt und der Report mit dem Ergebnissen erzeugt.

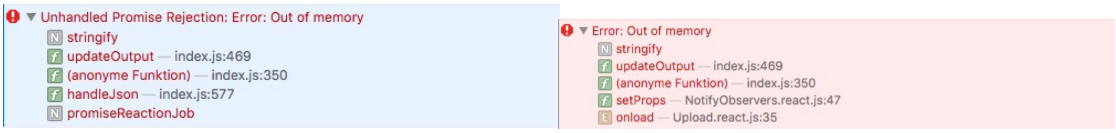


Abbildung 8.3.: Fehler im Zusammenhang mit Dash

Beim Testen der Implementierung in Dash fällt auf, dass diese auf dem kleinen German Credit Datensatz problemlos funktioniert, jedoch bei der Nutzung größerer Datensätze das Verhalten des Programmes von der Norm abweicht. Zunächst wurde für den Upload der Daten die Uploadkomponente des Moduls `dash_core_components` verwendet. Hierbei tritt beim Laden der Daten im Browser ein `OutOfMemoryError` auf. Dies hatte zur Folge, dass keine Übersicht über die 10 ersten Datensätze angezeigt wird, noch ein Update des Dropdownmenüs zur Wahl des Targets erfolgt. Darüber hinaus sind die Daten daher nicht ordnungsgemäß verfügbar, was eine Verwendung dieser Daten ausschließt. Durch eine Recherche im plotly Dash Forum unter `https://community.plot.ly/t/dash-upload-component-decoding-large-files/8033` wurde stattdessen die Upload Komponente des Moduls `dash_resumable_upload`, welche für größere Dateien optimiert ist, empfohlen. Hier erfolgt der Upload der Datei nicht als Ganzes, sondern zerlegt in kleinere Teile der Dateien, die im Anschluss zusammen gesetzt werden. Bei der Verwendung dieser Komponente zum Upload der Daten tritt auf den größeren Datensätzen erst nach einiger Zeit ein `Unhandled Promise Rejection Error: Out Of Memory` auf. Die genauen Meldungen zu den Fehlern können Abbildung 8.3 entnommen werden.

Die in diesem Forum von einem Forenmitglied namens Harold vorgebrachte Idee, durch die Nutzung des Moduls `Flask` über eine *html*-Uploadmaske den Upload des Datensatzes zu bewerkstelligen, löst das Problem nicht. Bei der Verwendung beider genannten Verfahren zum Upload der Daten sind weder eine Vorschau der Daten zu sehen noch ist eine Auswahl der Targetvariable möglich. Eine Vermutung ist, dass sich dieses Verhalten durch die Größe der hochgeladenen Datei von über 150 MB erklärt. Die Versuche mit einer kleineren Teilmenge der Daten von 50.000 Datensätzen bestätigten, dass die Gesamtdatenmenge zu groß ist, um diese serialisiert im Browser zu speichern und deshalb die Applikation nicht fehlerlos läuft, da in bei der Verwendung kleinerer Datensätze keine Probleme auftreten. Aus dieser Beobachtung heraus und der Notwendigkeit alle Daten und Zwischenergebnisse serialisiert im Browser zu speichern, scheint Dash nicht für das Arbeiten mit größeren Datenmengen geeignet zu sein. Ebenfalls die Nutzung eines Caches auf dem Filesystem zur Speicherung der Daten führt nicht zur fehlerlosen Funktion der Webapplikation. Eine weitere Methode, welche für die Speicherung der serialisierten Datensätze genutzt werden kann, ist die Komponente Store

des Moduls `dash_core_components`. Allerdings tritt bei der Nutzung dieser Komponente zur Speicherung der Datensätze bei den größeren Datensätzen ebenfalls der *Out of Memory Error* im Webbrowser auf. Da eine Lösung dieses Problems nicht möglich scheint, wurde ein Post für das Plotly Dash Forum verfasst, um bezüglich dieses Problems Lösungsansätze durch die Dash Community zu erhalten. Dieser Post blieb allerdings bis zum 23. Dezember 2018 unbeantwortet.

8.3. Batch Version

Neben den beiden interaktiven Versionen mit denen der Benutzer passende Einstellungen experimentell ermitteln kann, dient die ebenfalls implementierte Batch Version zur Validierung der anhand der ermittelten Parametereinstellungen berechneten Modelle. Hierzu werden die Daten zunächst in 80 % Trainings-und Validierungsdaten und 20 % Testdaten randomisiert eingeteilt. Aufgrund der Unbalanciertheit der verwendeten Datensätze erfolgt dies stratifiziert.

Durch die Crossvalidierung auf Basis des Trainings- und Validierungsdatensatzes wird in jedem Durchgang ein Trainingsdatensatz und ein Validierungsdatensatz erzeugt. In jedem Durchgang wird nun auf dem Trainingsdatensatz ein Modell erstellt, welches im Anschluss auf dem Validierungsdatensatz evaluiert wird. Das beste Modell wird am Schluss zur Klassifikation der Testdaten genutzt. Das Maß, welches zur Identifikation des besten Modells herangezogen wird, ist die Fläche unterhalb der Precision Recall Kurve (AUPRC). Zum Training der Modelle und zur Durchführung der topologischen Datenanalyse kommen die selben Komponenten wie bei den interaktiven Versionen zum Einsatz, welche bereits erläutert wurden. Daher werden in diesem Teil nur die Abweichungen des Designs und der Implementierung thematisiert.

8.3.1. Design und Implementierung der batch Version

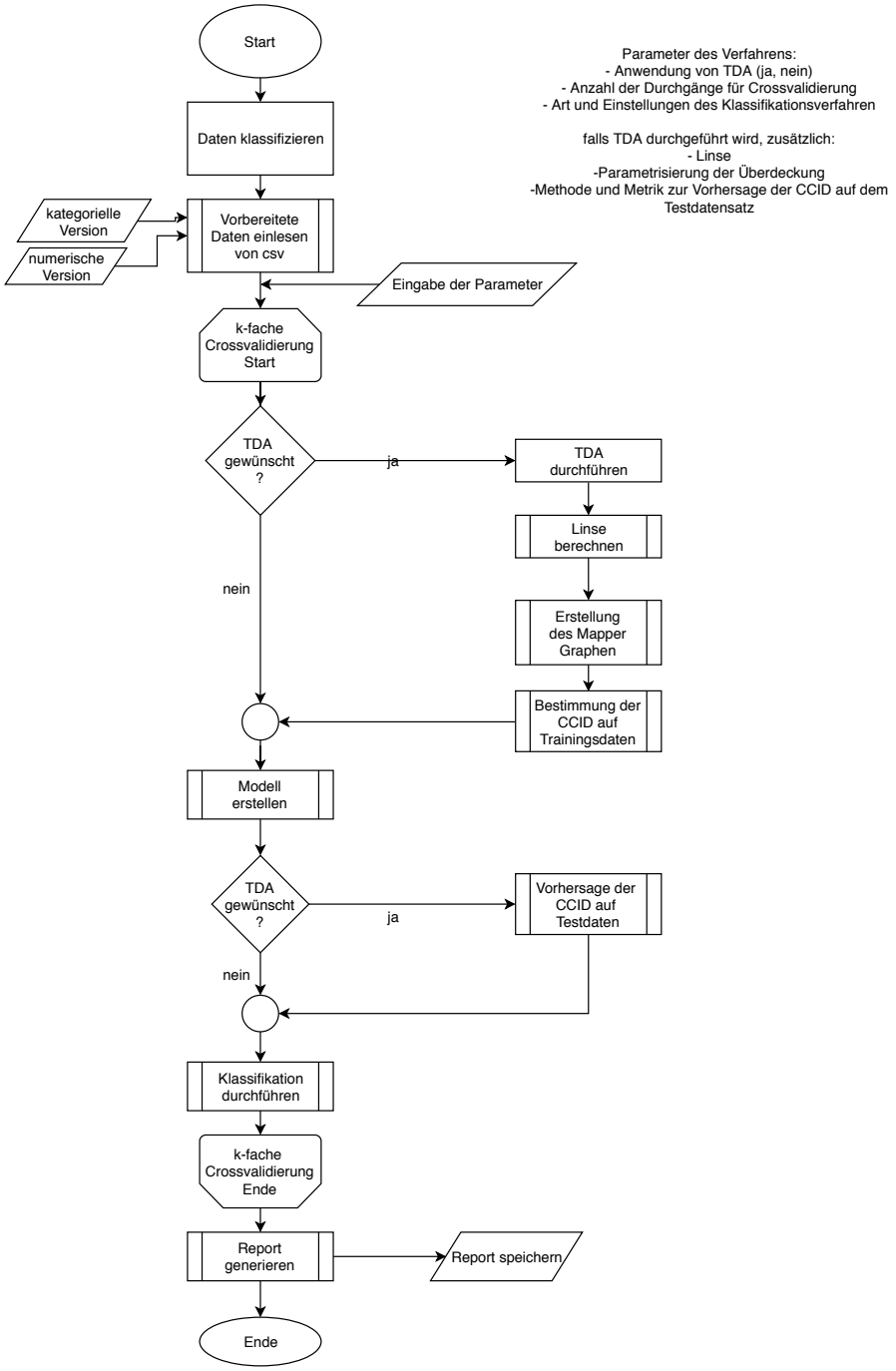


Abbildung 8.4.: Prozessablauf Batch-Analyse

Die Abbildung 8.4 stellt den Ablauf der *batch* Version grafisch dar. In der Grafik aus Gründen einer besseren Übersichtlichkeit nicht dargestellt ist die Teilung der Daten in Datensätze, die für die Crossvalidierung genutzt werden und in einen unabhängigen Testdatensatz und die Anwendung des besten Modells auf den Testdatensatz. Die Bestimmung des besten Modells erfolgt über das Performancemaß AUPRC auf dem Validierungsdatensatz. Die Durchführung des Klassifikationsprozesses erfolgt in der Klasse `TDAClassifier`. In der Methode *runTDAClassification* der Klasse`TDAClassifier` werden alle notwendigen Schritte der Klassifikation durchgeführt. Hierzu werden die bereits in Teil 8.2.2 beschriebenen Klassen verwendet.

Vorbereitete Daten einlesen

Der erste Schritt des in Abbildung 8.4 abgebildeten Ablaufdiagramm ist das Einlesen der Datensätze, die durch die Datenvorbereitungsschritte aus Abbildung 8.1 erzeugt wurden. Die Daten kategorieller Datensätze liegen in zwei Versionen vor, einmal in der ursprünglichen Form und einmal in rein numerischer Form. Das Einlesen der Daten erfolgt wie bei der interaktiven Version durch *read_csv* der Bibliothek `pandas`. Der zweite, nicht in der Abbildung 8.4 dargestellte, Schritt ist die Aufteilung der Daten, um einen Testdatensatz zu erhalten, um auf diesem die Performance des besten Modells auf Daten zu evaluieren, die nicht zum Training des Klassifikators genutzt wurden. Hierbei spricht man von einem Train-Validation-Test Ansatz. Die Implementierung der zunächst durchgeführten Teilung der Daten in Trainings- und Validierungsdatensatz und den Testdatensatz erfolgt, wie bei den anderen Versionen, mithilfe der Funktion *train_test_split* der `scikit-learn` Bibliothek.

Crossvalidierung

Im Anschluss an die Teilung der Daten auf die oben beschriebene Weise erfolgt nun als nächster Schritt des Ablaufdiagrammes aus Abbildung 8.4 die Durchführung der Crossvalidierung. Die Grundidee der k-fachen Crossvalidierung ist es, den Datensatz in k Teile aufzuteilen und zum Training des Modells jeweils $k-1$ Teile der Daten zu verwenden. Die übrige Datenmenge wird sodann zur Validierung des Modells verwendet. Dieser Prozess wird k mal wiederholt, bis jeder Teil der Daten zur Validierung verwendet wurde. Als Default wird hier eine Crossvalidierung mit $k = 5$ durchgeführt. Die Angabe des Parameters k erfolgt in der Konfigurationsdatei des implementierten Prototypen. Die zwischen den Schleifenbegrenzern abgebildeten Schritte sind Teil jedes Durchganges der Crossvalidierung und entsprechen den Schritten, die bereits bei den interaktiven Versionen beschrieben wurden. Ein Unterschied ist, dass hier auf eine Visualisierung von Zwischenergebnissen verzichtet wurde. Die Ausgabe der generierten Mapper Graphen ist jedoch durch Auswahl in der grafischen Benutzeroberfläche möglich. Die Imple-

mentierung der Crossvalidierung erfolgt durch die in `scikit-learn` implementierte Klasse `StratifiedKFold`. Die Methode *split* liefert eine Liste der Indexe der Objekte zurück, welche im jeweiligen Crossvalidierungsschritt den Trainings- beziehungsweise den Testdatensatz bilden. Durch Iteration über diese Liste können die Indexe der Datensätze ermittelt werden, welche im aktuellen Crossvalidierungsschritt den Trainings- und den Validierungsdatensatz bilden.

Vorhersage der CCID auf dem Testdatensatz

Nach Durchführung der Crossvalidierung folgt als nächster Schritt die Vorhersage der CCID auf dem Testdatensatz. Für diese Vorhersage wird entweder ein Trainingsdatensatz mit CCID für die lokale Vorhersage oder eine Zuordnung von Node und zugehörigem Zentroid für die zentroidbasierte Vorhersage benötigt. Diese Informationen werden jeweils in jedem Durchgang der Crossvalidierung in einer Liste gespeichert. Zur Prädiktion der CCID wird der Trainingsdatensatz beziehungsweise das Dictionary mit Zentroiden verwendet, welche zum besten Modell gehören. Ist das topologische Feature auf dem Testdatensatz vorhergesagt, so kann der Klassifikator auf diesen Datensatz angewandt werden und die Performancemaße der Klassifikation im Report festgehalten werden.

Report generieren

Da bei der Crossvalidierung k Modelle trainiert und evaluiert werden, besteht der Report zunächst aus k Chunks, die jeweils die Ergebnisse eines Durchlaufes der Crossvalidierung zusammenfassen. Diese bestehen, wie bei den interaktiven Versionen aus einer Tabelle der wichtigsten Performancemaße auf Trainings und Validierungsdatensatz, sowie einer Darstellung der ROC Curve und der Precision Recall Curve. Darüber hinaus enthält der Report eine statistische Zusammenfassung des AUPRC Wertes auf Trainings- und Validierungsdatensatz der einzelnen Durchgänge der Crossvalidierung. Dargestellt wird jeweils das arithmetische Mittel, die Standardabweichung und das Minimum und Maximum des Performancemaßes AUPRC auf allen mithilfe der Crossvalidierung trainierten Modellen. Der letzte Bestandteil des Reports der batch Version ist ein Chunk, der Informationen zur Güte der Klassifikation auf dem Testdatensatz enthält, wenn zur Klassifikation dieser Daten das beste Modell aus der Crossvalidierung genutzt wird. Die im Report ausgegebenen Informationen zur Performance, welche sich aus der Anwendung des besten Modells auf den Testdatensatz ergeben, entsprechen denen der anderen Chunks.

9. Fazit

Das Ziel dieser Masterthesis war es herauszufinden, ob durch die Anwendung der topologischen Datenanalyse eine Verbesserung eines Klassifikators möglich ist. Die aktuelle Forschung zur topologischen Datenanalyse in Verbindung mit Klassifikationsmodellen beschränkt sich fast ausschließlich auf die Generierung von Features aus der persistenten Homologie. Die Erstellung des Mapper Graphen erfolgt in der Literatur hauptsächlich zur Exploration der Daten, um aus den Erkenntnissen Informationen zur Modellgenerierung abzuleiten. Die direkte Ableitung von Features aus Ergebnissen des Mapper Graphen findet in der Forschung zu diesem Thema nicht statt. Um diese Lücke zu schließen, wurde ein Rahmen der Masterthesis der Prozess aus Abbildung 4.1 zur Durchführung der durch die topologische Datenanalyse unterstützte Klassifikation erarbeitet. Hierbei wird der Mapper Graph erzeugt und es werden mithilfe von Methoden aus der Graphanalyse die zusammenhängenden Komponenten bestimmt. Durch die Zusammenhangskomponenten des Mapper Graphs wird das neue topologische Feature *connectedComponentId* bestimmt. Jedem Trainingsobjekt wird als *connectedComponentId* die ID der Zusammenhangskomponente des Nodes zugewiesen, in dem das jeweilige Objekt liegt. Dies ist zielführend, da durch die Linsen eine Trennung der Objekte mit positiver Klasse von denen mit negativer Klasse verursacht werden soll. So befinden sich Objekte der negativen Klasse in einer anderen Komponente als die auffälligen Objekte des Datensatzes.

Zur Beantwortung der Frage, ob die topologische Datenanalyse durch die Hinzunahme eines topologischen Features Klassifikatoren verbessern kann, werden in den Kapiteln 5, 6 und 7 Experimente auf verschiedenen Datensätzen durchgeführt. Zur Klassifikation kommt ein Entscheidungsbaumklassifikator und eine L_1 regularisierte logistische Regression (LASSO) zum Einsatz. Das Training der Klassifikatoren erfolgt zunächst ohne topologisches Feature, um die Veränderung der Performance durch das neue topologische Feature bewerten zu können. Die Auswirkungen des topologischen Features auf die Güte der Klassifikation werden in Abhängigkeit der zur Erstellung des Mapper Graphen verwendeten Linse evaluiert. Das Maß zur Beurteilung der Vorhersagequalität ist die Fläche unterhalb der Precision Recall Kurve des Testdatensatzes, da diese Metrik deutlich unempfindlicher gegenüber der Unbalanciertheit der Klassen ist.

Während die topologische Datenanalyse auf dem Kreditkartendatensatz und dem German Credit Datensatz überwiegend zu Verbesserungen des Test-AUPRC des Entscheidungsbaumklassifikators führt, erfolgt ein Anhub des Test-AUPRC auf dem Lending Club Datensatz nur bei der Verwendung der linearen Diskriminanzanalyse als Linse. Der Anstieg des Test-AUPRC durch die topologische Datenanalyse bei der Nutzung des Entscheidungsbaumklassifikators unterscheidet sich je nach verwendeter Linse und verwendetem Datensatz. Die größte Verbesserung des Entscheidungsbaumklassifikators auf dem Kreditkartentransaktionsdatensatz verursacht die Linse UMAP mit einem Anstieg von knapp 5 Prozentpunkten. Nichtsdestotrotz sinkt der Test-AUPRC des Entscheidungsbaumklassifikators auf diesem Datensatz bei der Nutzung der Linsenkombination bestehend aus Isolation Forest und L_2 Norm um 1,4 Prozentpunkte. Bei dem German Credit Datensatz verursacht jede Linse einen Anhub des Test-AUPRC des Entscheidungsbaumklassifikators um mindestens 3,5 Prozentpunkte. Die größte Verbesserung der Performance des Entscheidungsbaumklassifikators auf dem German Credit Datensatz erfolgt durch die topologische Datenanalyse unter Verwendung der Linse TSNE mit einem Anstieg von 14 Prozentpunkten. Dennoch liegt der AUPRC des Testdatensatz nur bei rund 46 %, welches auf einen schlechten Klassifikator hindeutet. Diese Bewertung gilt auch für den Klassifikator ohne topologische Informationen mit einem Test-AUPRC von rund 32 %. Daher drängt sich die Vermutung auf, dass die Features des German Credit Datensatzes den Ausfall eines Kredits nicht gut erklären. Bei der Nutzung anderer Linsen auf dem Lending Club Datensatz bleibt der Test-AUPRC des Entscheidungsbaumklassifikators gleich oder nimmt leicht ab. Eine Begründung für die negativen Auswirkungen des topologischen Features auf den Entscheidungsbaum konnte nicht gefunden werden. Dies führt zu der Fragestellung, ob anhand der Eigenschaften des Datensatzes auf eine passende Linse geschlossen werden kann, welches eine offene Frage darstellt und daher im Ausblick noch einmal aufgegriffen wird.

Die durchgeführten Untersuchungen zeigen weiterhin, dass durch die Hinzunahme des topologischen Features oftmals nur eine geringe Verbesserung der Performance der logistischen Regression erreicht wird. Bei Verwendung der Linsenkombination Isolation Forest und L_2-Norm ist durch die Hinzunahme des topologischen Features auf allen Datensätzen eine sehr deutliche Verschlechterung von 20 bis 50 Prozentpunkten der Performance der logistischen Regression zu erkennen. Diese Klassifikatoren werden durch die topologische Datenanalyse unbrauchbar. Eine Begründung für diese starken Performanceverluste durch die topologische Datenanalyse könnte eine falsche Wahl der Parameter *n_cubes* und *overlap_perc* sein. Diese Parameter parametrisieren die Überdeckung der Linse, welche zur Bestimmung des Mapper

Graphens essentiell ist. Die Parameter *n_cubes* und *overlap_perc* wurden nicht auf das neue Klassifikationsverfahren angepasst, da ein Vergleich der Ergebnisse der beiden Klassifikationsverfahren möglich sein sollte. Eine deutliche Verbesserung der logistischen Regression konnte nur auf dem Kreditkartendatensatz mit der Linse TSNE erzielt werden. Hier vergrößert sich die Fläche unterhalb des Precision Recall Kurve des Testdatensatzes um 6,3 Prozentpunkte.

Die durchgeführten Experimente zeigen, dass die topologische Datenanalyse über die Generierung des Mapper Graphen ein sehr parameterreiches Verfahren der Datenanalyse ist. Dies ist so, da nahezu jeder Schritt des Prozess aus Abbildung 4.1 und 4.2 parametrisierbar ist und somit auf Eingaben vom Benutzer angewiesen ist. Die Experimente zeigen einen großen Einfluss der gewählten Linse auf die Qualität des Ergebnisses, da die Linse einen deutlichen Einflussfaktor auf die *connectedComponentId* darstellt. Darüber hinaus besitzt der überwiegende Teil der Linsen Parametrisierungsmöglichkeiten wie die Perplexität des Dimensionsreduktionsalgorithmus TSNE, welche einen Einfluss auf die Linse ausüben. Die Generierung des Mapper Graphs, welcher zur Bestimmung des topologischen Features notwendig ist, erfordert eine Parametrisierung der Überdeckung der Linse. Hieraus ergeben sich zwei weitere Parameter, für die ebenfalls optimale Werte gefunden werden müssen.

Da kein Verfahren zur Bestimmung optimaler Werte für die Anzahl der überlappenden Intervalle *n_cubes* mit einer relativen Überlappung von *overlap_perc* existiert, wurde im Kapitel 8 ein Prototyp konzeptioniert und implementiert, welcher die durch die topologische Datenanalyse unterstützte Klassifikation durchführt. Aufgrund unterschiedlicher Anforderungsprofile erfolgt das Design und die Implementierung des Prototyps als interaktive Versionen und als batch Version, welche dem Anwender erlaubt, die Qualität eines Klassifikationsmodell auf Basis von einer experimentell in der interaktiven Version des Prototyps herausgefundene Parametrisierung zu bewerten. Dies geschieht durch die Anwendung von Crossvalidierung. Eine der interaktiven Versionen ist ein Anwendungsprogramm mit grafischer Oberfläche, die andere interaktive Version ist als Webapplikation implementiert. Bei Datensätzen mit mehr als 50.000 Objekten treten bei Webapplikation allerdings Probleme mit der Bereitstellung der Daten auf. Diese sind im Kapitel 8 beschrieben und beruhen darauf, dass im Modul Dash zur Programmierung von Webapplikationen die Notwendigkeit besteht, hochgeladene Datensätze serialisiert im Browser zu speichern, sodass sie für andere Callbacks zur Verfügung stehen.

Wie im Kapitel 2 beschrieben, ist die einzige Anforderung an eine Linse die Stetigkeit. Aus diesem Grund kommen unzählige Funktionen und Algorithmen als Linse infrage. Daher ist es

zur Herstellung eines marktreifen Produktes notwendig, anhand der Eigenschaften eines Datensatz oder bestimmten Faustregeln eine optimale Linse inklusive optimaler Parametrisierung zu bestimmen. Die Tatsache, dass die Parametrisierung der Überdeckung der Linse eine großen Einfluss auf die Erstellung des Mapper Graphens und damit auf die *connectedComponentId* besitzt, zeigt die Notwendigkeit einer Methode zur Bestimmung optimaler Werte für die Anzahl überlappender Intervalle und die Stärke dieser Überlappung der Überdeckung der Linse.

Eine Alternative zur vorgestellten Weise der Nutzung von Informationen aus dem Mapper Graphen zur Erzielung eines besseren Klassifikationsergebnisses ist die Identifizierung der Features, welche die stärkste Trennung der Datenobjekte in verschiedenen Zusammenhangskomponenten verursachen. Hierzu wird zunächst der Mapper Graph generiert und die *connectedComponentId* wie beschrieben dem Datensatz zugewiesen. Nun wird eine multivariate Varianzanalyse durchgeführt, wobei die Features des Ursprungsdatensatz den unabhängigen Variablen entsprechen und die *connectedComponentId* die Rolle der abhängigen Variable der Varianzanalyse übernimmt. Die Features, welche die Zugehörigkeit zum Feature *connectedComponentId* am Besten erklären, werden zur Klassifikation verwendet. Dieses Verfahren wird **Feature Selection** genannt.

Insgesamt ist es möglich Klassifikatoren durch Hinzunahme eines topologischen Features als Zusammenfassung der Form des Datensatzes zu verbessern. Nichtsdestotrotz stellt die topologische Datenanalyse kein Allheilmittel dar, die schlechte Klassifikatoren zu Klassifikatoren mit einer guten Performance verhilft. Weiterhin haben die Experimente gezeigt, dass nicht bei jedem Datensatz eine Verbesserung der Qualität des Klassifikators zu erwarten ist. Daher ist es sinnvoll zu erforschen, ob anhand der Eigenschaften eines Datensatzes prognostiziert werden kann, ob die Anwendung der topologischen Datenanalyse beim jeweiligen Datensatz eine Verbesserung des Klassifikators verursachen wird. Eine weitere Herausforderung ist die Wahl der richtigen Parameter der topologischen Datenanalyse, da es hierfür keine etablierten Regeln gibt. Aus diesem Grund ist ein deutlicher Zeitaufwand zur Findung einer richtigen Parametrisierung einzuplanen. Es lohnt sich jedoch den Einsatz der topologischen Datenanalyse in Form des Mapper Graphs auf dem gegebenen Datenbestand zu evaluieren.

Literatur

[1] Henry Adams u. a. „Persistence Images: A Stable Vector Representation of Persistent Homology“. In: *J. Mach. Learn. Res.* 18.1 (Jan. 2017), S. 218–252. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=3122009.3122017.

[2] Khaled Almgren, Minkyu Kim und Jeongkyu Lee. „Extracting Knowledge from the Geometric Shape of Social Network Data Using Topological Data Analysis“. In: *Entropy* 19.7 (2017). ISSN: 1099-4300. DOI: 10.3390/e19070360. URL: http://www.mdpi.com/1099-4300/19/7/360.

[3] Ayasdi Inc. *Anti-Money Laundering Solution Deep Dive*. 2017. URL: https://s3.amazonaws.com/cdn.ayasdi.com/wp-content/uploads/2017/05/12132340/Machine_Intelligence_Apps_WP_051617v01.pdf (besucht am 29. 08. 2018).

[4] Ayasdi Inc. *Credit Card Fraud Detection and Modeling*. URL: https://s3.amazonaws.com/cdn.ayasdi.com/wp-content/uploads/2015/02/13112502/FinServ_Fraud_Detection_Solution_Brief.pdf (besucht am 08. 10. 2018).

[5] Ayasdi Inc. *Machine Intelligence Applications: Our Philosophy*. URL: https://s3.amazonaws.com/cdn.ayasdi.com/wp-content/uploads/2017/05/12132340/Machine_Intelligence_Apps_WP_051617v01.pdf (besucht am 08. 10. 2018).

[6] Ayasdi Inc. *TDA and Machine Learning: Better Together*. URL: https://s3.amazonaws.com/cdn.ayasdi.com/wp-content/uploads/2017/08/28134706/WP-TDAandML-BetterTogether-07172017.pdf (besucht am 08. 10. 2018).

[7] BaFin. *Big Data trifft auf künstliche Intelligenz*. URL: https://www.bafin.de/dok/10732556 (besucht am 19. 10. 2018).

[8] Bokeh Development Team. *Bokeh: Python library for interactive visualization*. 2018. URL: https://bokeh.pydata.org/en/latest/.

[9] Peter Bubenik. „Statistical Topological Data Analysis using Persistence Landscapes“. In: *Journal of Machine Learning Research* 16 (2015), S. 77–102.

[10] Gunnar Carlsson. „Topological pattern recognition for point cloud data“. In: *Acta Numerica* 23 (2014), S. 289–368. DOI: 10.1017/S0962492914000051.

[11] Frédéric Chazal und Bertrand Michel. *An introduction to Topological Data Analysis: fundamental and practical aspects for data scientists*. URL: https://arxiv.org/abs/1710.04019v1 (besucht am 14. 12. 2018).

[12] Jesse Davis und Mark Goadrich. „The Relationship Between Precision-Recall and ROC Curves“. In: *Proceedings of the 23. International Conference on Machine Learning*. 2006.

[13] Jia Deng, Wei Dong, Richard Socher u. a. „ImageNet: A large-scale hierarchical image database“. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Juni 2009, S. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[14] Tamal Dey, Sayan Mandal und William Varcho. „Improved Image Classification using Topological Persistence“. In: *Vision, Modeling, and Visualization*. Hrsg. von M. Hullin u. a. The Eurographics Association. 2017.

[15] Dua Dheeru und Efi Karra Taniskidou. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml (besucht am 14. 09. 2018).

[16] Alireza Dirafzoon, Namita Lokare und Edgar Lobaton. „Action classification from motion capture data using topological data analysis“. In: *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE. 2016, S. 1260–1264. DOI: 10.1109/GlobalSIP.2016.7906043.

[17] Pratik Doshi und Wlodek Zadrozny. „Movie Genre Detection Using Topological Data Analysis“. In: *Statistical Language and Speech Processing*. Hrsg. von Thierry Dutoit, Carlos Martín-Vide und Gueorgui Pironkov. Cham: Springer International Publishing, 2018, S. 117–128. ISBN: 978-3-030-00810-9.

[18] Rickard Brüel Gabrielsson und Gunnar Carlsson. *A look at the topology of convolutional neural networks*. URL: https://s3.amazonaws.com/cdn.ayasdi.com/wp-content/uploads/2018/11/07115708/1810.03234.pdf (besucht am 13. 12. 2018).

[19] Noah Giansiracusa, Robert Giansiracusa und Chul Moon. *Persistent homology machine learning for fingerprint classification*. URL: https://arxiv.org/abs/1711.09158v1 (besucht am 08. 10. 2018).

[20] Marian Gidea und Yuri Katz. „Topological Data Analysis of Financial Time Series: Landscapes of Crashes“. In: *Physica A: Statistical Mechanics and its Applications* 491 (2017), S. 820–834.

[21] Wei Guo und Ashis G. Banerjee. „Toward Automated Prediction of Manufacturing Productivity Based on Feature Selection Using Topological Data Analysis“. In: *IEEE International Symposium on Assembly and Manufacturing (ISAM)*. 2016.

[22] Wei Guo u. a. „Sparse-TDA: Sparse Realization of Topological Data Analysis for Multi-Way Classification“. In: *IEEE Transactions on Knowledge and Data Engineering*. 2017.

[23] Trevor Hastie, Robert Tibshirani und Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2. Auflage. Springer series in statistics. Springer, 2009.

[24] Christoph Hofer u. a. „Deep Learning with Topological Signatures“. In: *Neural Information Processing Systems (NIPS)*. 2018.

[25] Firas A. Khasawneh, Elizabeth Munch und Jose A. Perea. *Chatter Classification in Turning Using Machine Learning and Topological Data Analysis*. (Besucht am 08. 10. 2018).

[26] Genki Kusano, Kenji Fukumizu und Yasuaki Hiraoka. „Persistence Weighted Gaussian Kernel for Topological Data Analysis“. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, S. 2004–2013. URL: http://dl.acm.org/citation.cfm?id=3045390.3045602.

[27] Lending Club. *What do the different Note statuses mean?* URL: https://help.lendingclub.com/hc/en-us/articles/215488038-What-do-the-different-Note-statuses-mean- (besucht am 08. 10. 2018).

[28] Fei Tony Liu, Kai Ming Ting und Zhi-Hua Zhou. „Isolation-Based Anomaly Detection“. In: *ACM Trans. Knowl. Discov. Data* 6.1 (März 2012), 3:1–3:39. ISSN: 1556-4681. DOI: 10.1145/2133360.2133363. URL: http://doi.acm.org/10.1145/2133360.2133363.

[29] Laurens van der Maaten. *Barnes-Hut t-SNE*. 2015. URL: https://github.com/lvdmaaten/bhtsne (besucht am 26. 06. 2018).

[30] Laurens van der Maaten und Geoffrey Hinton. „Visualizing Data using t-SNE“. In: *Journal of Machine Learning Research* 1 (2008), S. 1–48.

[31] Herchel Thaddeus Machacon. „A Topological Data Analysis Approach to Visualizing Ebola Tweets“. In: *Japan Journal of Medical Informatics* 36.5 (2016), S. 253–269.

[32] Leland McInnes. *UMAP-Learn Documentation*. URL: https://umap-learn.readthedocs.io/en/latest/parameters.html (besucht am 14. 12. 2018).

[33] Leland McInnes, John Healy und James Melville. „UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction“. In: *ArXiv e-prints* (Feb. 2018). arXiv: 1802.03426 [stat.ML].

[34] Wes McKinney. „Data Structures for Statistical Computing in Python“. In: *Proceedings of the 9th Python in Science Conference*. Hrsg. von Stéfan van der Walt und Jarrod Millman. 2010, S. 51–56.

[35] Elizabeth Munch. „A User's Guide to Topological Data Analysis“. In: *Journal of Learning Analytics* 4.2 (2017), S. 47–61.

[36] Grzegorz Muszynski u. a. „Topological Data Analysis and Machine Learning for Recognizing Atmospheric River Patterns in Large Climate Datasets“. In: *Geoscientific Model Development Discussions* 2018 (2018), S. 1–24. DOI: 10.5194/gmd-2018-53. URL: https://www.geosci-model-dev-discuss.net/gmd-2018-53/.

[37] Monica Nicolau, Arnold J. Levine und Gunnar Carlsson. „Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival“. In: *PNAS* 108.17 (2011), S. 7265–7270.

[38] Nilson Report. „The Nilson Report“. In: *Nilson Report* 1118 (Okt. 2017). URL: https://nilsonreport.com/publication_newsletter_archive_issue.php?issue=1118.

[39] Nina Otter, Mason A Porter, Ulrike Tillmann u. a. „A roadmap for the computation of persistent homology“. In: *EPJ Data Science* 6.17 (2017), S. 1–38.

[40] Chris Parmer. *Dash Plotly Python Module*. URL: https://dash.plot.ly/introduction (besucht am 22. 11. 2018).

[41] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort u. a. *Scikit-learn: Isolation Forest Documentation*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html (besucht am 03. 12. 2018).

[42] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort u. a. „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830.

[43] Andrea Dal Pozzolo u. a. „Calibrating Probability with Undersampling for Unbalanced Classification“. In: *2015 IEEE Symposium Series on Computational Intelligence*. Dez. 2015, S. 159–166. DOI: 10.1109/SSCI.2015.33.

[44] Matteo Rucco, Emanuela Merelli, Damir Herman u. a. „Using topological data analysis for diagnosis pulmonary embolism“. In: *Journal of Theoretical and Applied Computer Science* 9.1 (2015), S. 41–55.

[45] Matthias Schubert. „Leonhard Euler und die 7 Brücken von Königsberg“. In: *Mathematik für Informatiker: Ausführlich erklärt mit vielen Programmbeispielen und Aufgaben*. Wiesbaden: Vieweg und Teubner, 2009, S. 317–358. ISBN: 978-3-8348-9585-1. DOI: 10.1007/978-3-8348-9585-1_13. URL: https://doi.org/10.1007/978-3-8348-9585-1_13.

[46] Gurjeet Singh, Facundo Mémoli und Gunnar Carlsson. „Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition“. In: *Eurographics Symposium on Point-Based Graphics*. Hrsg. von M. Botsch und R. Pajarola. The Eurographics Association, 2007.

[47] Justin Skycak. *Topological Data Analysis Theory, Practise, Software and Potential*. URL: https://de.slideshare.net/JustinSkycak/skycak-tda-expository-paper (besucht am 08. 10. 2018).

[48] Terry M. Therneau, Elizabeth J. Atkinson und Mayo Foundation. *An Introduction to Recursive Partitioning Using the RPART Routines*. URL: https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf (besucht am 08. 12. 2018).

[49] Dmitry Ulyanov. *Multicore-TSNE*. 2016. URL: https://github.com/DmitryUlyanov/Multicore-TSNE (besucht am 29. 08. 2018).

[50] Frank Wiebe. *Wie sich Kreditkartenbesitzer vor Betrügern schützen können*. URL: https://www.handelsblatt.com/finanzen/banken-versicherungen/kreditkartenbetrug-wie-sich-kreditkartenbesitzer-vor-betruegern-schuetzen-koennen/22892258.html (besucht am 16. 11. 2018).

[51] Wikipedia. *Königsberger Brückenproblem - Wikipedia, Die freie Enzyklopädie*. URL: https://de.wikipedia.org/w/index.php?title=K%C3%B6nigsberger_Br%C3%BCckenproblem&oldid=181654616 (besucht am 11. 12. 2018).

A. Gegenüberstellung zweier Implementierungen für die TSNE Dimensionsreduktion

Der TDA Mapper Algorithmus verwendet eine Projektionsfunktion, die die Dimension des Datensatzes auf eine oder zwei Dimensionen verringert. Hierzu wird unter anderem TSNE, ein Verfahren zur Visualisierung hochdimensionaler Datensätze in zwei oder drei Dimensionen, genutzt. Begonnen wurde mit der TSNE Implementierung aus dem scikit-learn Modul, wobei sich herausstellte, dass diese Implementierung nicht auf große Datenmengen skaliert. Auf alle Datensätze des bei Kaggle verfügbaren Datensatzes zur Erkennung von betrügerischen Transaktionen (ca. 284.000 Transaktionen) angewendet, füllt sich nach kürzester Zeit der RAM, was dazu führt, dass nach über 12 Stunden noch keine Ergebnisse berechnet werden. Aus diesem Grund wurden auf größere Datenmengen optimierte TSNE Implementierungen recherchiert. Zunächst wurde mit der Implementierung unter `https://github.com/lvdmaaten/bhtsne` experimentiert [29]. Mit dieser auf C++ basierenden Implementierung der TSNE Dimensionsreduktion war es möglich, die nötigen Berechnungen innerhalb von rund 2 Stunden auf einem Laptop mit 8GB RAM durchzuführen. Die Abbildung A.1 stellt die TSNE Komponenten in einem Scatterplot dar. Die Einstellungen sind in der Tabelle A.1 aufgelistet. Als Eingabe wurde der gesamte Datensatz inklusive des Targets verwendet. Als Einstellungsparameter wurden die Werte aus A.1 verwendet. Die Parameter *learning_rate* und *early_exaggeration* erhalten in der bhtsne Implementierung nach [29] als fest vorgegebene Werte die Werte der Tabelle A.2.

Parameter	Wert	Beschreibung
no_dims	2	Anzahl der Dimensionen nach der Dimensionsreduktion
perplexity	50	Tuningparameter, der vorgibt, ob das Modell eher lokal oder global sein soll. Typischerweise zwischen 5 und 50. Entspricht der Anzahl der berücksichtigten Nachbarn
theta	0.5	Trade-off Parameter zwischen Genauigkeit und Geschwindigkeit
randseed	4242	Zufallsseed zur Reproduzierbarkeit der Ergebnisse
verbose	True	Ausgabe von Zwischenergebnissen gewünscht?
initial_dims	31	Anzahl der Merkmale
use_pca	False	sollen die initialen Werte für die dimensionsreduzierten Daten durch eine Hauptkomponentenanalyse bestimmt werden
max_iter	1000	Anzahl der Iterationen

Tabelle A.1.: Verwendete Parameterwerte für die lokale TSNE Berechnung

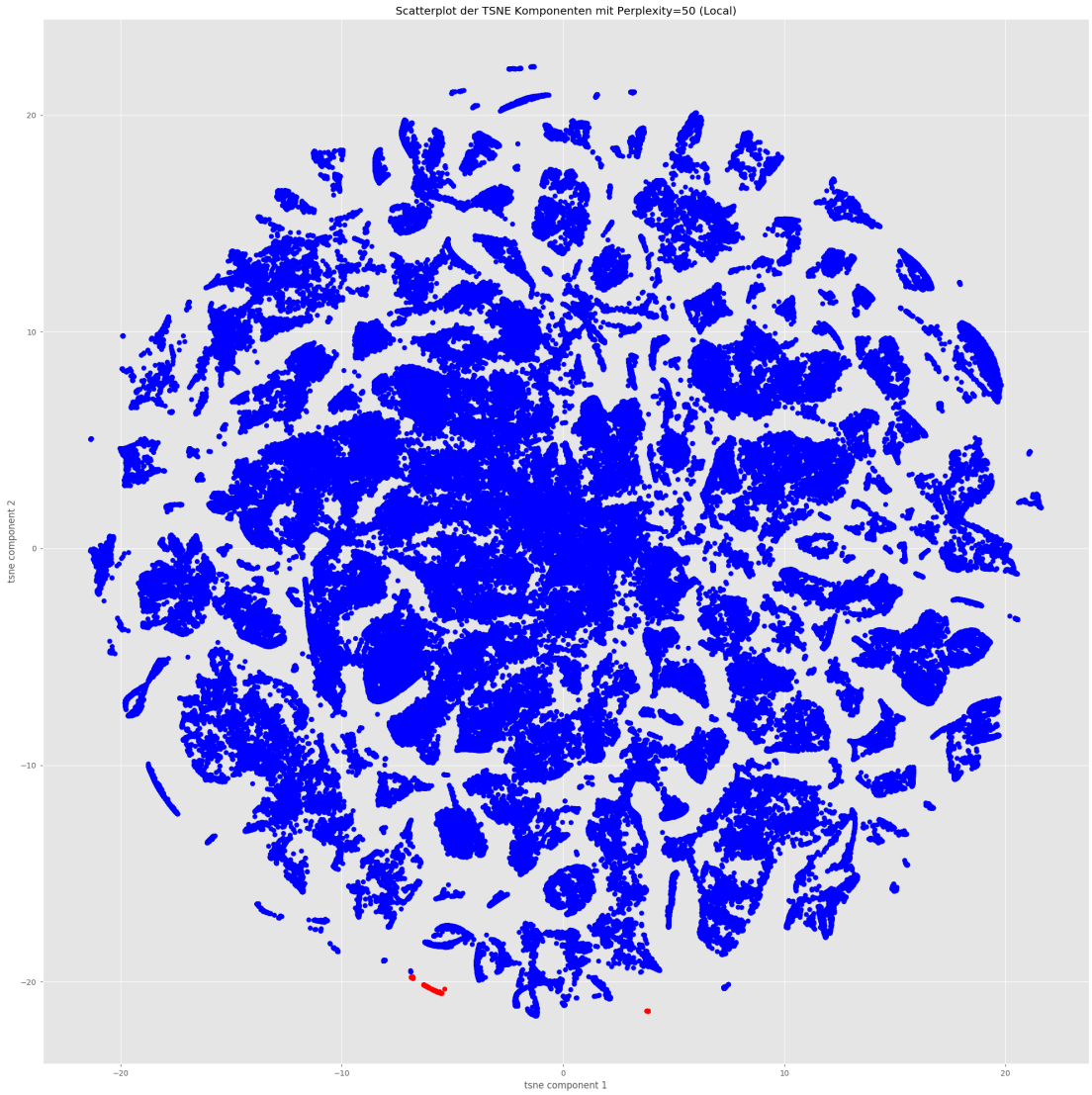


Abbildung A.1.: TSNE Visualisierung mit Perplexität 50 und auf dem gesamten Datensatz. Verwendet wurde die bhtsne Implementierung nach [29]

Als weitere Implementierung wurde eine Multicore TSNE Implementierung getestet. Diese findet sich unter `https://github.com/DmitryUlyanov/Multicore-TSNE` auf der Plattform GitHub [49]. Diese Implementierung ist an die vorgenannte angelehnt, ist jedoch für die parallele Verarbeitung durch mehrere Kerne optimiert. Die Implementierung wird auf einem Node des Hochschulcomputerclusters mit 128GB RAM getestet. Hier nimmt die Berechnung der TSNE Projektion des gesamten Kreditkartendatensatzes nur rund 33 Minuten in Anspruch. Als Visualisierung der Ergebnisse ergibt sich der Scatterplot aus A.2. Für die Berechnung der

in Abbildung A.2 dargestellten Komponenten der TSNE Dimensionsreduktion wurde die Information, ob es sich bei einer Transaktion um Betrug handelt in die TSNE Berechnung mit einbezogen. Mit Ausnahme der Perplexität, die auf 50 gesetzt und der Anzahl der für die Berechnung verwendeten Kerne (hier 8), wurden für die weiteren Parameter Defaultwerte verwendet. Diese lassen sich dem Quellcode entnehmen und sind in Tabelle A.2 aufgeführt.

Parameter	**Wert**	**Beschreibung**
n_components	2	gewünschte Anzahl der Dimensionen der Projektion
early_exaggeration	12	Skalierung der berechneten bedingten Wahrscheinlichkeiten zur Verbesserung der Optimierung der Zielfunktion (vgl [30, S. 10])
perplexity	50	Tuningparameter, der vorgibt, ob das Modell eher lokal oder global sein soll. Typischerweise zwischen 5 und 50. Entspricht der Anzahl der berücksichtigten Nachbarn
learning_rate	200	Learning-Rate für das Gradientenabstiegsverfahren
n_iter	1000	Anzahl der Iterationen
n_iter_without_progress	30	Stopkriterium, falls sich Zielfunktion der Optimierung nicht ändert
min_grad_norm	1e-07	Abbruchschwelle für den Gradienten
metric	'euclidean'	verwendete Metrik
init	'random'	Initialisierung für die dimensionsreduzierten Werte
verbose	0	Ausgabe von Zwischenergebnissen gewünscht?
random_state	4242	Random Seed für Reproduzierbarkeit
method	'barnes_hut'	Methode für die Durchführung von TSNE
angle	0.5	Trade-off Parameter zwischen Genauigkeit und Geschwindigkeit
cheat_metric	False	Auswahl, ob die quadrierte oder nicht quadrierte euklidische Distanz für die Berechnung der nächsten Nachbarn genutzt wird

Tabelle A.2.: Defaultwerte für Multicore Implementierung von TSNE

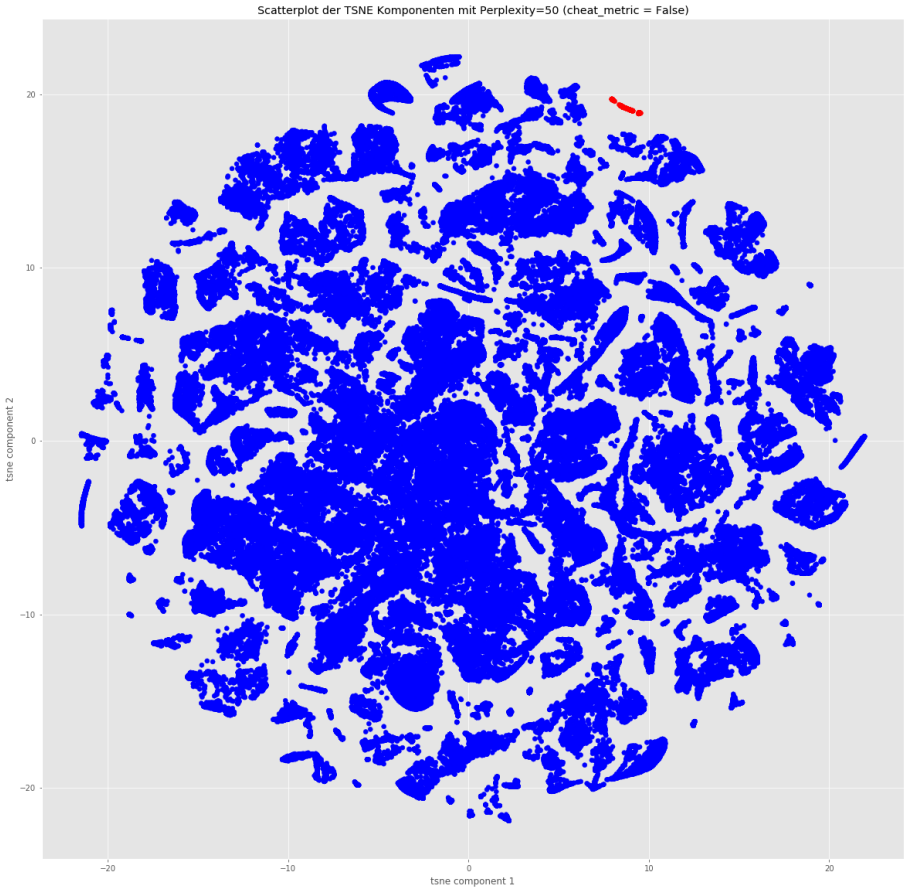


Abbildung A.2.: Scatterplot der TSNE Komponenten. Verwendete Perplexität 50. Verwenden der MulticoreTSNE Implementierung

Bei der Betrachtung der Scatterplots in Abbildung A.2 und A.1 fällt auf, dass sich die Lage der roten Punkte, die die betrügerischen Transaktionen charakterisieren, unterscheiden. Die beiden TSNE Implementierungen liefern somit nicht die selben Ergebnisse. Der Grund für die Abweichung der Ergebnisse konnte bisher noch nicht gefunden werden.

Anwendung des Kepler Mappers auf die TSNE Komponenten

Die durch TSNE mit den Einstellungen nach Tabelle A.2 auf zwei Dimensionen reduzierten Daten (dargestellt in Abbildung A.2) werden nun als csv auf dem Node gespeichert und auf den Laptop übertragen, wo durch die Verwendung des Kepler Mappers ein Graph der Daten erstellt wird. Der Parameter *n_cubes* für die Anzahl der Intervalle wird zunächst auf 25 gesetzt. Der Einfluss dieses Parameters wird im weiteren Verlauf der Arbeit noch evaluiert. Die Größe der Überlappung dieser Intervalle liegt im in A.3 dargestellten Graphen bei 0,2.

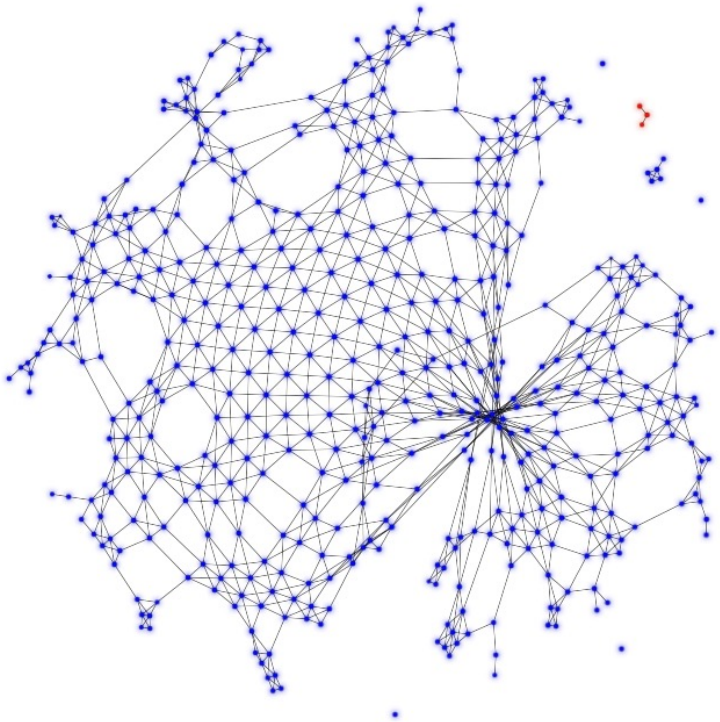


Abbildung A.3.: Visualisierung des durch den Kepler Mapper berechneten Graphen auf Basis der MulticoreTSNE Linse. Einfärbung der Nodes nach der häufigsten Klasse im Node (blau=normal, rot=Fraud). Verwendet 25 Intervalle pro Dimension mit 20 % Überlappung

Auf der Abbildung A.3 kann man gut die Trennung zwischen normalen und betrügerischen Transaktionen erkennen. Zur Einfärbung der Nodes wurde die häufigste Klasse im jeweiligen Node verwendet. Rot repräsentiert die betrügerischen Transaktionen, wohingegen normale Transaktionen blau gefärbt sind. Der in Abbildung A.3 dargestellte Graph wird dazu genutzt ein neues Feature connectedComponentId zu bilden. Hierzu wird zu allen Datensätzen aller Nodes in einer zusammenhängenden Komponente deren ID als Wert für das neue Feature *connectedComponentId* zugewiesen. Liegt ein Datenpunkt in unterschiedlichen Clustern, so

wird die ClusterId durch die ID des neuen Clusters ersetzt.

Zum Vergleich wurde der Graph mit gleichen Einstellungen des Kepler Mappers auch auf Basis der Dimensionsreduktion durch die bhtsne Implementierung, die lokal auf dem Laptop läuft erstellt, der in Abbildung A.4 abgedruckt ist. Hier ist keine Isolierung der Nodes mit überwiegend positiver Klasse festzustellen. Die betrügerischen Transaktionen liegen hier am Rand des Graphen mit den Nodes mit überwiegend normalen Transaktionen, sind jedoch durch Kanten mit diesen Nodes verbunden. In den in gelber Farbe dargestellen Nodes im Graph in Abbildung A.4 befinden sich ca. 80% betrügerische Transaktionen. Die Färbung der Nodes geschieht hier im Gegensatz zu den Ausführungen in Kapitel 4 nicht durch den Modus des Targets, sondern auf Basis des Anteils der betrügerischen Transaktionen in einem Node. Liegen überwiegend normale Transaktionen in einem Node, ist dieser im Graph blau eingefärbt. Sind mehr auffällige als normale Transaktionen in einem Node, so ist der Node im Graph rot gefärbt. Wegen der schnelleren Berechnung auf dem Cluster wird in der weiteren Forschung im Rahmen dieser Masterthesis zur Berechnung der Linse die Multicore Implementierung verwendet.

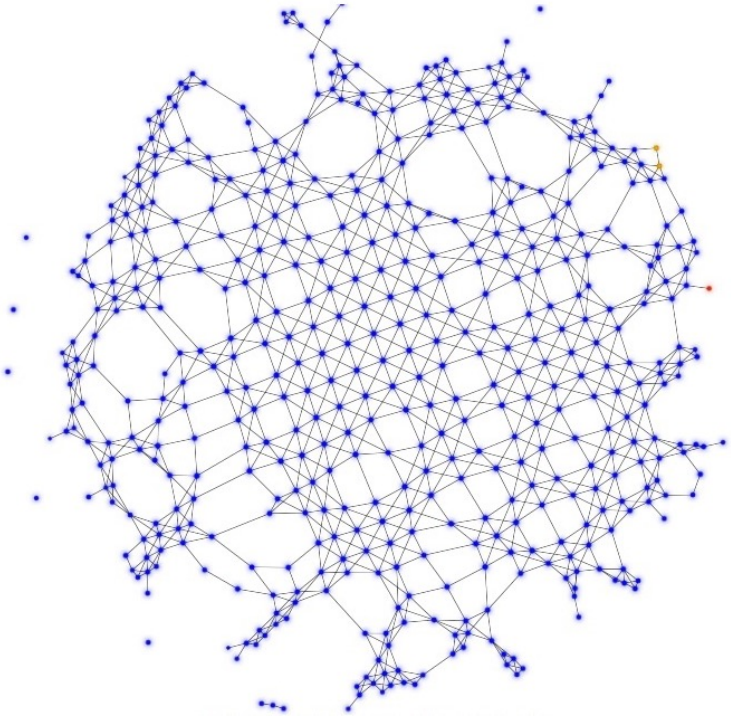


Abbildung A.4.: Visualisierung des durch den Kepler Mapper berechneten Graphen. Berechnung von TSNE durch bhtsne Implementierung. Einfärbung der Nodes nach der häufigsten Klasse im Node (blau=normal, rot=Fraud). Verwendet 25 Intervalle pro Dimension mit 20 % Überlappung