



Bachelorarbeit

# UNTERSUCHUNG DES EINFLUSSES VON AUSREISSERN AUF DIE PROGNOSEGENAUIGKEIT VON FEINSTAUBKONZENTRATIONEN

Davin Ahn

Matr.-Nr.: 4797790

Betreut durch:

Prof. Dr.-Ing. habil. Dirk Habich

und:


Dr.-Ing. Claudio Hartmann

Eingereicht am 28 Februar 2023



## ERKLÄRUNG

Ich erkläre, dass ich die vorliegende Arbeit selbständig, unter Angabe aller Zitate und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

 Digital unterschrieben  
von Davin Ahn  
Datum: 2023.02.28  
17:10:37 +01'00'

Dresden, 28 Februar 2023



## ABSTRACT

Maschinelles Lernen ist in der heutigen Datenwelt ein wichtiges Werkzeug, um komplexe Probleme zu lösen und Entscheidungen zu treffen. Durch den Anstieg des Datenvolumens und Verbesserungen in der Datenqualität können Prognosemodelle präziser und fundierter werden, was bessere Entscheidungen ermöglicht. Derzeit sind bereits ausreichende Daten vorhanden, jedoch ist das Potenzial zur Verbesserung der Datenqualität noch nicht ausgeschöpft. Hier können Methoden wie Datenbereicherung, Datenreduktion und Ausreißererkennung helfen. Die vorliegende Arbeit konzentriert sich besonders auf die Ausreißererkennung in Zeitreihendaten, die durch Umweltdaten erfasst werden, bei denen Ausreißer häufig auftreten.

Die Arbeit untersucht die Auswirkungen der Ausreißererkennung auf die Vorhersagegenauigkeit von maschinellen Lernmodellen. Im Gegensatz zu früheren Arbeiten, die sich auf jeweils einen Algorithmus oder auf solche Algorithmen beschränkten, die auf nur einer bestimmten Eigenschaft basieren, untersucht diese Arbeit fünf Algorithmen, die auf unterschiedlichen Eigenschaften beruhen. Das hier verfolgte Ziel ist es, ein vertieftes Verständnis für die verschiedenen Algorithmen zur Ausreißererkennung zu entwickeln und deren Stärken und Schwächen zu identifizieren. Darüber hinaus kann eine gründliche Analyse der vorhandenen Algorithmen dazu beitragen, in einer konkreten Forschungssituation eine Empfehlung für den besten Algorithmus abzugeben.

Die Untersuchung hat zu dem Ergebnis geführt, dass sämtliche Algorithmen mit Ausnahme des LOF-Algorithmus die Vorhersagegenauigkeit in einem stark verzerrten Datensatz verbessern können.



# INHALTSVERZEICHNIS

<b>1</b>	<b>Einleitung</b>	<b>11</b>
<b>2</b>	<b>Vorbetrachtungen</b>	<b>13</b>
2.1	Zeitreihendaten . . . . .	13
2.2	Anomalien in Zeitreihendaten . . . . .	15
2.3	Modellierung der Feinstaubkonzentration . . . . .	16
<b>3</b>	<b>Verwandte Arbeiten</b>	<b>19</b>
<b>4</b>	<b>Algorithmen zur Ausreißererkennung</b>	<b>21</b>
4.1	Statistikbasierte Algorithmen zur Ausreißererkennung . . . . .	21
4.1.1	Interquartile Range (IQR) . . . . .	21
4.1.2	z-Score Filter (z-Score) . . . . .	22
4.2	Cluster-basierte Algorithmen zur Ausreißererkennung . . . . .	23
4.2.1	K-Means-Clustering (K-Means) . . . . .	23
4.3	Dichtebasierte Algorithmen zur Ausreißererkennung . . . . .	26
4.3.1	Local Outlier Factor (LOF) . . . . .	26
4.3.2	Isolation Forest (iForest) . . . . .	28

<b>5</b>	<b>Umsetzung</b>	<b>33</b>
5.1	Datensatz . . . . .	33
5.2	Implementierung des Algorithmus . . . . .	34
5.3	Feinstaubprognosemodell . . . . .	37
<b>6</b>	<b>Evaluation</b>	<b>39</b>
<b>7</b>	<b>Fazit &amp; Ausblick</b>	<b>43</b>



## ACKNOWLEDGMENTS

An dieser Stelle möchte ich mich von Herzen bei Prof. em. Prof. e.h. Dr. Dr. h.c. Dr. h.c. Werner Lehfeldt für seine wertvolle Unterstützung während meines Studiums und insbesondere beim Schreiben meiner Abschlussarbeit bedanken. Seine Hilfe bei der Verbesserung meiner Grammatik war von unschätzbarem Wert. Ohne ihn hätte ich meine Arbeit nicht auf diesem Niveau abschließen können. Daher bin ich ihm sehr dankbar für seine Unterstützung und Freundschaft.

Davin Ahn

Dresden, 28 Februar 2023



# 1 EINLEITUNG

Die Überwachung der Feinstaubkonzentration in der Luft ist ein wichtiger Faktor, wenn es darum geht, die Gesundheit und das Wohlbefinden der Menschen zu erhalten und zu verbessern. In den letzten Jahren haben Low-Cost-Sensoren es nicht nur entwickelten Ländern, sondern auch Entwicklungsländern ermöglicht, diese Überwachung zu automatisieren und zu verbessern. Dies hat jedoch gleichzeitig zur Erhöhung der Wahrscheinlichkeit von Ausreißern in den generierten Datensätzen geführt.

Die Analyse des Datensatzes spielt bei der Modellierung des maschinellen Lernens eine entscheidende Rolle. Ausreißer in einem Datensatz können das Modell beeinträchtigen und die Fehlerhaftigkeit von Vorhersagen bewirken. Daher ist es wichtig, Algorithmen zur Erkennung von Ausreißern zu entwickeln, um die Leistungsfähigkeit des Modells zu erhöhen. In der vorliegenden Arbeit wird eine Analyse mehrerer Algorithmen zur Ausreißererkennung durchgeführt, um auf diese Weise einen Beitrag zur Weiterentwicklung dieser Technologien zu leisten. Als Ausgangspunkt dafür behandelt die Einleitung die Zielsetzung und den Aufbau dieser Arbeit.

## ZIELSETZUNG

Das Ziel dieser Arbeit ist es, herauszufinden, wie sich das Vorhandensein von Ausreißern in Feinstaubmessdaten auf das Prognosemodell auswirkt. Dazu wird der mit fünf Algorithmen zur Ausreißererkennung (Interquartile Range, z-Score Filter, K-Means-Clustering, Local Outlier Factor und Isolation Forest) bereinigte Datensatz als Trainingsdaten für das Prognosemodell verwendet und anschließend ausgewertet.

## AUFBAU DER ARBEIT

Die Arbeit gliedert sich in folgende Kapitel: Kapitel 2 stellt Zeitreihendaten und Anomalien in Zeitreihendaten vor und beschreibt ein Konzept zur Modellierung der Feinstaubkonzentra-

tion. Kapitel 3 beschreibt verwandte Arbeiten und begründet die Notwendigkeit der vorliegenden Arbeit. Kapitel 4 befasst sich mit den fünf Algorithmen zur Ausreißererkennung, die statistik-, cluster- und dichte-basierte Algorithmen umfassen. Kapitel 5 beschreibt den hier verwendeten Datensatz, die Implementierung des Algorithmus und das Feinstaubprognosemodell. Kapitel 6 überprüft die Wirksamkeit der Algorithmen und führt eine vergleichende Analyse durch. Schließlich werden in Kapitel 7 die Ergebnisse zusammengefasst und wird ein Ausblick auf zukünftige Forschungsrichtungen gegeben.

## 2 VORBETRACHTUNGEN

Die Abschnitte 2.1 und 2.2 erläutern die Grundlagen der Zeitreihenprognose und führen eine Unterscheidung verschiedener Typen von Anomalien ein. Der Abschnitt 2.3 beschreibt den Arbeitsablauf zur Erstellung eines Feinstaubprognosemodells.

### 2.1 ZEITREIHENDATEN

Eine Zeitreihe ist das Ergebnis der Beobachtung von Messungen, die in bestimmten Zeitintervallen und mit einer bestimmten Abtastrate durchgeführt wurden. Somit kann eine Zeitreihe als eine Reihe aufeinander folgender Momente definiert werden. Eine Reihe kann univariat oder multivariat sein, letzteres dann, wenn mehrere Reihen gleichzeitig mehrere Dimensionen innerhalb derselben Zeitspanne umfassen. Die besonderen Merkmale von Zeitreihendaten bestehen darin, dass sie nach „Zeit“ sortiert und aufeinander folgende Werte miteinander korreliert sind. Mit anderen Worten, Elemente von Zeitreihendaten haben einen gemeinsamen Zeitkontext, so dass benachbarte Instanzen voneinander abhängen. [EA12]

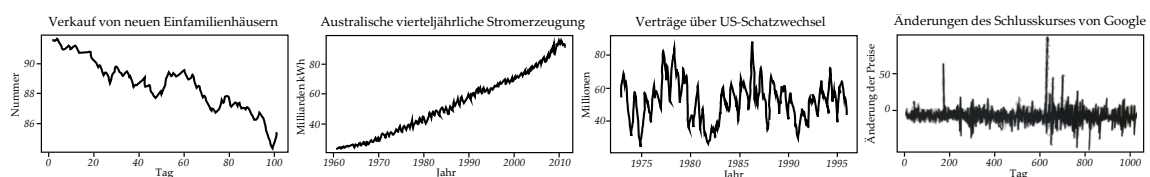


Abbildung 2.1: Muster der Zeitreihendaten [HA14]

Da Zeitreihendaten zeitabhängig sind, gibt es ein zeitliches Muster. Traditionell werden Zeitreihendaten in Trend, Saisonalität, Zyklus und zufällige Schwankung kategorisiert.

**Trend** bezieht sich auf langfristige Änderungen ohne sich wiederholendes Muster. Die erste und die zweite Grafik in Abbildung 2.1 zeigen fallende bzw. steigende Trends.

**Saisonalität** tritt auf, wenn eine Zeitreihe von saisonalen Faktoren wie der Jahreszeit oder dem Wochentag beeinflusst wird. Die Saisonalität stellt ein Muster dar, das sich gemäß einem

regelmäßigen Zyklus wiederholt, und weist immer eine feste und bekannte Frequenz auf. In der dritten Grafik von Abbildung 2.1 sind die zwischen 1975 und 1982 beobachteten Schwankungen ein Beispiel für ein solches Muster.

**Zyklus** ist ein weiteres klassifiziertes Muster, das sich in Wellenform ohne regelmäßigen Zyklus und ohne feste Frequenz wiederholt. Diese Muster weisen typischerweise Schwankungen von zwei oder mehr Jahren auf. Die dritte Grafik in Abbildung 2.1 zeigt ein saisonales Muster, bildet aber auch große Wellen über einen langen Zeitraum ab, z. B. so kurz wie sieben Jahre und so lang wie zehn Jahre, was ein Beispiel für einen Zyklus ist.

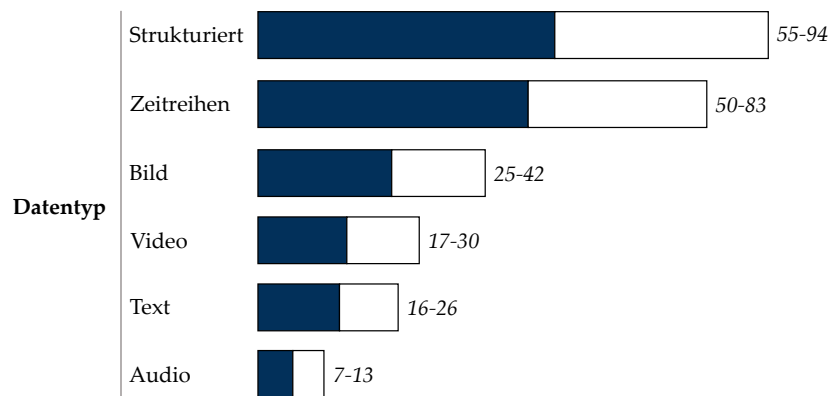
**Zufällige Schwankungen** sind, wie in der vierten Grafik in Abbildung 2.1 dargestellt, andere Unregelmäßigkeiten als Trend, Saisonalität und Zyklus.

Die saisonale und die zyklische Änderung sehen ähnlich aus, aber tatsächlich sind sie ganz verschieden. Wenn die Schwankungen eine feste Frequenz haben, dann sind sie saisonal. Wenn sich die Frequenz ändert, dann ist das Muster zyklisch. Im Allgemeinen ist die durchschnittliche Länge von Saisonalität kürzer als die Länge eines zyklischen Musters [HA14]. Diese Klassifizierung vereinfacht die Art der Daten.

In fast allen Bereichen werden Messungen über die Zeit durchgeführt. Diese Messungen führen zu einer Sammlung organisierter Daten, die als Zeitreihen bezeichnet werden [EA12]. Beispielsweise wird Forschung unter Verwendung von Zeitreihendaten in verschiedenen Bereichen durchgeführt, wie z.B. dem Studium biologischer Prozesse [BJGS12], dem Auffinden von Schadensquellen in mechanischen Systemen [SF01], der Bewertung der Pflanzenproduktivität und des Pflanzenmanagements [SYT<sup>+</sup>05] sowie der Untersuchung der Auswirkungen von Richtlinien [KCMC20].

% des gesamten Wertpotenzials

■ Bereich



**Abbildung 2.2:** Bereich der potenziellen Auswirkung des KI-Werts nach Datentyp [CMM<sup>+</sup>18]

Da es insbesondere Fälle gibt, in denen Geschäftsstrategien unter Bezugnahme auf solche Ergebnisse bestimmt werden, die aufgrund von Zeitreihendaten vorhergesagt werden, sind Zeitreihendaten in praktischer Hinsicht tatsächlich sehr wertvoll. So wird beispielsweise laut einem Paper des „McKinsey Global Institute (Abbildung 2.2)“ Zeitreihendaten ein höheres Wertpotenzial als Text- oder Bilddaten zugesprochen [CMM<sup>+</sup>18].

## 2.2 ANOMALIEN IN ZEITREIHENDATEN

Die intuitive Definition einer Anomalie wäre „eine Beobachtung, die so stark von anderen Beobachtungen abweicht, dass der Verdacht erweckt wird, dass sie durch einen anderen Mechanismus erzeugt worden ist“ [Haw80]. Anomalie wird mit verschiedenen Termini ausgedrückt, wie z. B. Neuheit, die als positiver Ausdruck im Wirtschaftsbereich verstanden wird, Anomalität, die als negativer Ausdruck bewertet wird, so etwa ein schlechtes Signal, und Ausreißer für etwas, was extrem weit außerhalb des Allgemeinen liegt. Daher können Anomalien für jeden Bereich und jedes Problem unterschiedlich definiert werden, was bedeutet, dass es keinen einzigen optimalen Algorithmus für die Anomalieerkennung gibt und dass je nach der definierten Anomalie unterschiedliche Algorithmen zur Problemlösung ausgewählt werden. Aus diesem Grund wird hier ein Ansatz zum Identifizieren von Anomalien beschrieben.

Anomalien können in drei Typen eingeteilt werden.

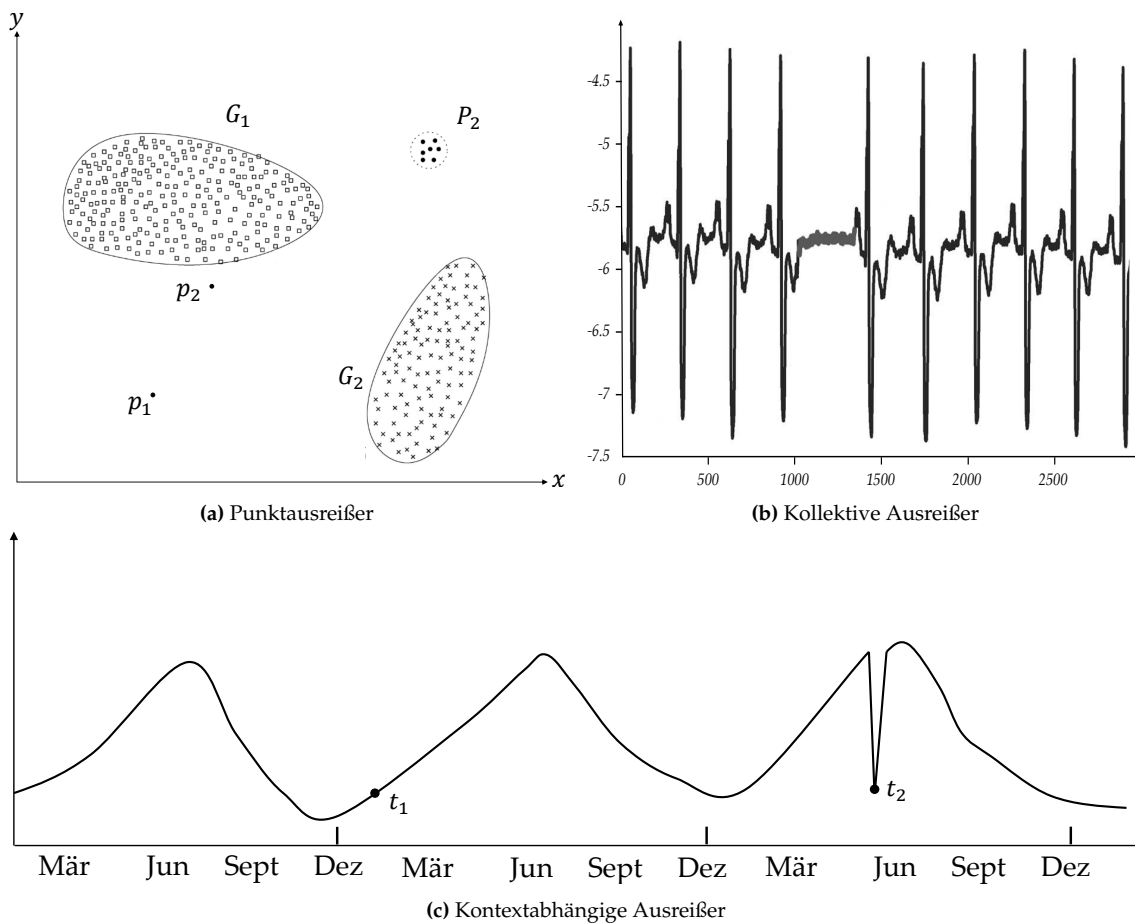


Abbildung 2.3: Typen der Anomalien [CBK09]

**Punktausreißer** Wenn ein beliebiger Datenpunkt in Bezug auf die übrigen Daten als anomal betrachtet wird, wird er als Punktausreißer bezeichnet. Dies ist die einfachste Art von Anomalie. Beispielsweise sind in Abbildung 2.3a die Punkte  $p_1$  und  $p_2$  sowie die Punkte im Bereich  $P_2$  Punktausreißer, da sie außerhalb der Grenzen des normalen Bereichs liegen [CBK09].

**Kollektive Ausreißer** Wenn eine Kollektion verwandter Datenpunkte in Bezug auf den gesamten Datensatz anomal ist, werden diese Punkte als kollektive Ausreißer bezeichnet. Ein einzelner Datenpunkt für sich genommen ist vielleicht keine Anomalie, aber was im Zusammenspiel mit anderen Daten geschieht, kann sehr wohl eine Anomalie sein. Kollektive Ausreißer werden meist in Sequenz- und in Diagramm Daten untersucht. Abbildung 2.3b ist ein Beispiel einer menschlichen EKG-Ausgabe [GAG<sup>+</sup>00]. Der Bereich zwischen 1000 und 1500 entlang der x-Achse weist auf eine Anomalie hin, da die gleichen niedrigen Werte ungewöhnlich lange vorhanden sind [CBK09].

**Kontextabhängige Ausreißer** Wenn bestimmte Datenpunkte in einem bestimmten Kontext anomal sind, werden sie als kontextabhängige Ausreißer bezeichnet. Diese Datenpunkte können in einem bestimmten Kontext kontextabhängige Ausreißer sein, in einem anderen Kontext jedoch als normal gelten. Hauptsächlich werden kontextabhängige Ausreißer in Zeitreihen und in räumlichen Daten untersucht. Abbildung 2.3c zeigt ein Beispiel für kontextabhängige Ausreißer in Zeitreihendaten, die die monatlichen Temperaturen in einer Region zeigen. Eine Temperatur für die Zeit  $t_1$  kann im Winter (Zeitpunkt  $t_1$ ) normal sein, aber dieselbe Temperatur wäre im Sommer (Zeitpunkt  $t_2$ ) eine Anomalie [CBK09].

Punktausreißer können innerhalb des gesamten Datensatzes auftreten, während kontextabhängige Ausreißer nur innerhalb einer Gruppe eines verwandten Datensatzes auftreten können. Kollektive Ausreißer können dagegen je nach Kontext der Daten insgesamt oder innerhalb einer Gruppe auftreten.

Da es in der vorliegenden Arbeit um Feinstaubdaten geht, konzentriert sie sich auf Zeitreihendaten, und die Ausreißererkennung in Zeitreihendaten entspricht dem kontextabhängigen Ausreißer. Wie der Graph in Abbildung 2.3c zeigt, ist die Zeitabhängigkeit der wichtigste Faktor für Zeitreihendaten. Dies liegt daran, dass sie zusätzliche Informationen enthalten kann, die sich für eine Verbesserung des Erkennens von Ausreißer verwenden lassen.

## 2.3 MODELLIERUNG DER FEINSTAUBKONZENTRATION

In diesem Abschnitt wird der Arbeitsablauf zur Erstellung eines Feinstaubprognosemodells beschrieben, insbesondere derjenige Teil, auf den sich die vorliegende Arbeit im Hinblick auf den gesamten Ablauf des maschinellen Lernens konzentriert.

Abbildung 2.4a zeigt den Ablauf des maschinellen Lernens. Das Feinstaubprognosemodell folgt dem gleichen Ablauf.

**Datenbeschaffung** Für maschinelles Lernen sind Daten notwendig. Der Datenerfassungsprozess im Umweltbereich sammelt Daten durch die Installation von Sensoren, die in der Lage sind, verschiedene in der Umwelt vorhandene Substanzen zu messen. Dabei können aufgrund technischer Probleme des Datenerfassungsgeräts oder menschlicher Fehler Ausreißerdaten in den Datensatz hineingeraten. Bei derartigen Ausreißerdaten kann es sich um fehlende Werte, um sehr große oder sehr kleine Werte oder um Werte in unorganisierten Einheiten handeln.



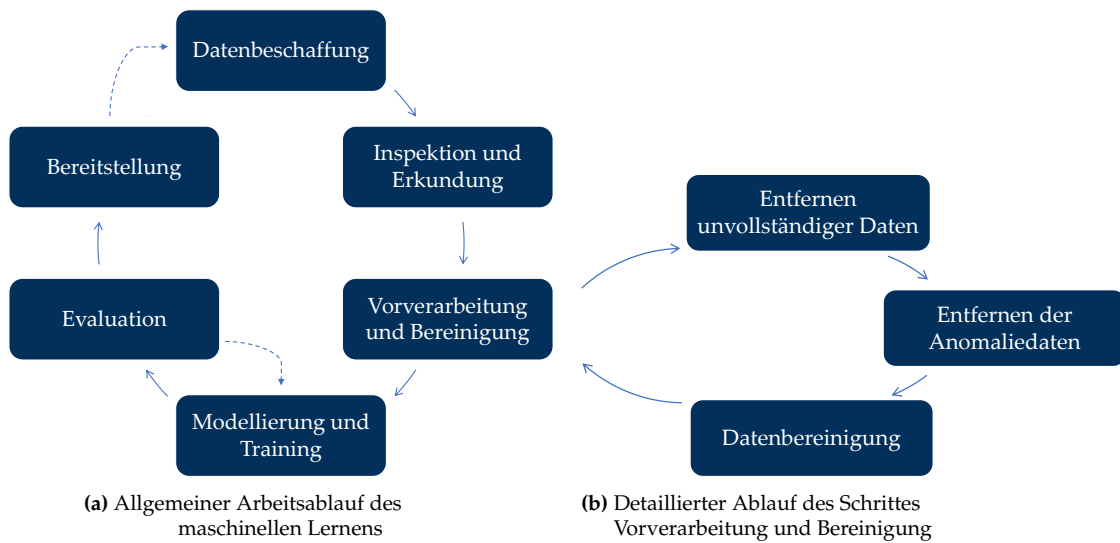


Abbildung 2.4: Arbeitsablauf des maschinellen Lernens

**Inspektion und Erkundung** Sobald die Daten gesammelt worden sind, besteht der nächste Schritt darin, diese zu untersuchen. Dieser Schritt wird auch als Exploratory Data Analysis (EDA) bezeichnet, bei der unabhängige Variablen, abhängige Variablen und Datentypen von Variablen untersucht werden, um die strukturellen Beziehungen aufzudecken, die durch verschiedene Features in den Daten hervorgerufen werden.

**Vorverarbeitung und Bereinigung** Auf diesen zweiten Schritt folgt der Datenvorverarbeitungsschritt, die im Ablauf des maschinellen Lernens anspruchsvollste Aufgabe ist. Dieser Schritt umfasst zahlreiche Unterschritte. Zum Beispiel umfasst er im Fall der Feinstaubprognose den Umgang mit fehlenden Daten, das Entfernen von anomalen Daten sowie die Datenbereinigung.

**Modellierung und Training** Wenn die Datenvorverarbeitung abgeschlossen ist, beginnt der Modellierungsschritt, d.h. der Schritt des Codierens für maschinelles Lernen. Wenn die Modellierung durch einen geeigneten maschinellen Lernalgorithmus abgeschlossen ist, dann wird das Modell mit Hilfe der vorverarbeiteten Daten trainiert. Wenn es damit zum Abschluss gelangt ist, ist es in der Lage, die gewünschte Feinstaubkonzentration vorherzusagen.

**Evaluation** In der Evaluationsphase wird die Leistung des Modells unter Zuhilfenahme von Testdaten bewertet. Wenn festgestellt wird, dass das maschinelle Lernmodell gemäß der gewählten Evaluationsmetrik unzureichende Leistung erbringt, kann dieser Prozess zum vorherigen Schritt zurückkehren und die Modellierung erneut durchführen. Die in dieser Arbeit als Evaluationsmetrik verwendete Metrik Symmetric Mean Absolute Percentage Error (SMAPE) ist eine von mehreren Metriken, die verwendet werden, um die Prognosegenauigkeit von Prognosemodellen zu messen [HA14]. Diese Metrik berechnet den durchschnittlichen absoluten Prozentfehler zwischen den vorhergesagten und den tatsächlichen Werten. Ein niedriger SMAPE-Wert deutet auf eine hohe Prognosegenauigkeit hin, während ein hoher Wert auf eine geringe Genauigkeit hinweist. Es ist wichtig, zu beachten, dass der SMAPE empfindlich auf Null- oder nahe bei Null liegende Werte reagieren kann, da der Fehler in Prozenten berechnet wird. Die Formel für SMAPE lautet wie folgt:

$$\text{SMAPE} = \frac{200}{n} \sum_{i=1}^n \frac{|\hat{y}_i - y_i|}{|\hat{y}_i| + |y_i|} \quad (2.1)$$

In Formel (2.1) ist  $y_i$  der tatsächliche Wert und  $\hat{y}_i$  der vorhergesagte Wert.

**Bereitstellung** Wenn das Modell auf dem Wege der Evaluation als erfolgreich trainiert befunden worden ist, geht es in die Bereitstellungsphase über. Wenn jedoch eine Situation vorliegt, in der das Modell aufgrund des allgemeinen Feedbacks erst noch weiter aktualisiert werden muss, kann es in die Erfassungsphase zurückkehren.

Die meisten Feinstaubprognosemodelle treffen Vorhersagen durch die Analyse meteorologischer Einflüsse und chemischer Komponenten, die in der Atmosphäre vorhanden sind. Das in dieser Arbeit verwendete Modell [Klingner et al.] analysiert hingegen nur meteorologische Einflüsse und trifft auf dieser Grundlage Vorhersagen. Wenn es also einen sehr hohen Messwert gibt, der nicht aus meteorologischen Daten erklärt werden kann, etwa weil jemand in der Nähe des Sensors ein Lagerfeuer oder etwas Ähnliches abgebrannt hat, dann sollten diese Daten im Schritt „Vorverarbeitung und Bereinigung“ entfernt werden, um einer Verminderung der Modellleistung vorzubeugen. Daher konzentriert sich die vorliegende Arbeit in dem Schritt „Vorverarbeitung und Bereinigung“ auf die Entfernung von anomalen Daten. Abbildung 2.4b zeigt diesen Schritt detaillierter. Wie hier zu sehen ist, wird die Qualität der Trainingsdaten verbessert, indem die Ausreißerentfernung beseitigt wird. Daher lernt das Feinstaubprognosemodell nur auf der Grundlage von Daten guter Qualität, so dass eine bessere Vorhersageleistung erzielt werden kann. In dem Paper [Bra] heißt es, dass die Leistung des Feinstaubprognosemodells durch das Entfernen der Ausreißer um 25.46% im Root Mean Square Error (RMSE) verbessert worden sei.

### 3 VERWANDTE ARBEITEN

Das Entfernen von Ausreißern kann die Leistung von Lernmodellen erheblich verbessern, wie auch bei einem Modell zur Vorhersage der Feinstaubkonzentration. In [SJLD15] wird untersucht, wie sich die Ausreißererkennung mit der Mahalanobis-Distanz Methode auf Feinstaubkonzentrationsdaten auswirkt, und wird gezeigt, dass das Entfernen von Ausreißern einen positiven Effekt ergibt. Ein anderes Paper [AO22] konzentriert sich auf das Auffinden von Ausreißern und Extremereignissen im Feinstaubdatensatz unter Verwendung des DBSCAN-Algorithmus. Die Ergebnisse zeigen die Wirksamkeit der Methode zur Identifizierung von Lärm und Extremereignissen, die durch lokale oder globale Faktoren verursacht werden können. Beide Arbeiten zeigen, dass das Entfernen von Ausreißern die Abweichungen der vorhergesagten Werte reduziert und somit die Prognoseleistung verbessert.

Meistens wird jedoch nur ein Algorithmus oder werden Algorithmen verwendet, der bzw. die auf nur einer Eigenschaft basieren, um Ausreißer im Feinstaubdatensatz zu erkennen, ohne Vergleiche mit auf unterschiedlichen Eigenschaften basierenden Algorithmen anzustellen. Zum Beispiel schlägt das Paper [JYK20] einen modifizierten IQR vor, so dass es kein Problem gibt, hohe Feinstaubkonzentrationen vorherzusagen. Denn der herkömmliche IQR kategorisiert Daten mit hohen Feinstaubkonzentrationen als Ausreißer und eliminiert diese, was dazu führt, dass das trainierte Modell nicht in der Lage ist, Vorhersagen für diese Art von Konzentrationen zu treffen. Ein anderes Paper [MKPT20] schlägt ein Framework vor, das mehrere statistische Algorithmen zur Ausreißererkennung verwendet, wie Z-Score, IQR, Grubbs's Test, Hampel's Test und Tietjen-Moore Test. Obwohl in diesem Paper mehrere Algorithmen miteinander verglichen wurden, sind alle rein statistikbasiert.

Im Bereich der Ausreißererkennung existieren bereits Algorithmen, die auf verschiedenen Eigenschaften basieren. Beispielsweise nutzt [WD16] beim „wireless sensor network“ das K-Means-Clustering zur Erkennung von Blackhole- und Misdirection-Knoten. Mit einer Erkennungsrate von 98,6% und einer falschen positiven Rate von 1,2% ist dies besser als bestehende Algorithmen. Ein adaptives Anomalie-Erkennungsschema für Cloud-Computing, das auf dem dichte-basierten LOF-Algorithmus basiert, wird in [HZZ<sup>+</sup>13] vorgestellt. Dieses Schema kann kontextabhängige Anomalien mit geringem Rechenaufwand effektiv erkennen. Mehrere andere Papers [LIPJ21] sowie [DF13] nutzen unterschiedliche Algorithmen zur Ausreißererkennung mit zufriedenstel-

lenden Ergebnissen. Bisher wurde jedoch kein Fall beschrieben, in dem Ausreißer mithilfe von Algorithmen mit unterschiedlichen Eigenschaften erkannt und die Unterschiede zwischen ihnen verglichen und analysiert wurden. Wenn jedoch solche Vergleiche und Analysen durchgeführt werden, können die Ergebnisse dazu beitragen, das Verständnis für die Stärken und die Schwächen eines jeden Algorithmus zu verbessern und in weiterführenden Arbeiten eine fundierte Entscheidung zu treffen. Tabelle 3.1 gibt an, welche eigenschaftsbasierten Algorithmen in den bisher erwähnten Papers und in dieser Arbeit verwendet wurden, um den Unterschied zwischen diesen Papers und der vorliegenden Arbeit zu verdeutlichen.

verwandte Algorithmen	Papers							
	[SjLD15]	[AO22]	[JYK20]	[MKPT20]	[WD16]	[HZZ <sup>+</sup> 13]	[LIP]21]	[DF13]
statistikbasiert	✗	✗	✓	✓	✗	✗	✗	✗
cluster(distanz)basiert	✓	✓	✗	✗	✓	✗	✓	✗
dichtebasiert	✗	✗	✗	✗	✗	✓	✗	✓

**Tabelle 3.1:** Vergleichstabelle der verwendeten Algorithmen

Daher werden in der vorliegenden Arbeit Ausreißer unter Zuhilfenahme von statistik-, cluster- und dichtebasierten Algorithmen entfernt, und im Ergebnis wird verglichen und analysiert, wie sich die Ausreißerentfernung mit jedem Algorithmus auf die Leistung des Lernmodells auswirkt.

## 4 ALGORITHMEN ZUR AUSREISSERERKENNUNG

In diesem Kapitel werden die fünf in dieser Arbeit verwendeten Algorithmen zur Ausreißerererkennung beschrieben. Jeder Algorithmus wird gemäß seiner Haupteigenschaft als statistik-, als cluster- oder als dichtebasierter Algorithmen klassifiziert.

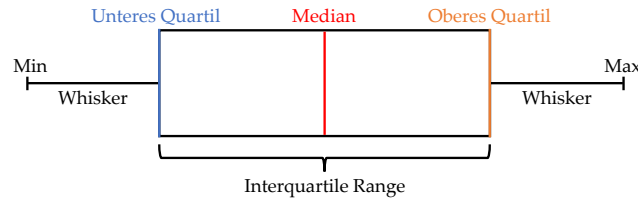
### 4.1 STATISTIKBASIERTE ALGORITHMEN ZUR AUSREISSERERKENNUNG

Die statistikbasierte Ausreißerererkennung ist eine frühe Methode der Ausreißerererkennung. Die Definition von anomalen Daten lautet hier: „ein Wert, der als teilweise oder als vollständig unterschiedlich von der Wahrscheinlichkeitsverteilung der meisten Werte angesehen wird“ [Ans60]. In diesem Kapitel werden zwei Algorithmen zur Erkennung von Ausreißern auf der Grundlage von Statistiken beschrieben.

#### 4.1.1 Interquartile Range (IQR)

Die Interquartile Range (Abbildung 4.1) ist die einfachste statistische Technik, die verwendet wird, um Ausreißer in univariaten und in multivariaten Daten zu erkennen. Sie verwendet Informationen wie unteres Quartil ( $Q_1$ ), Median (Median) und oberes Quartil ( $Q_3$ ), um solche Daten zu visualisieren.

Der für die Ausreißerererkennung definierte Interquartile Range (IQR) ist die Differenz zwischen dem oberen Quartil ( $Q_3$ ) und dem unteren Quartil ( $Q_1$ ). Datenpunkte außerhalb des Bereichs zwischen  $Q_1 - x * \text{IQR}$  und  $Q_3 + x * \text{IQR}$  werden als Ausreißer erkannt. In der obigen Gleichung steuert der Koeffizient  $x$ , auch bekannt als Whisker, des IQR den Bereich der Ausreißer. Bei einem kleinen Wert gelten viele Datenpunkte als Ausreißer, bei einem großen Wert hingegen nur



**Abbildung 4.1:** Ein Beispiel von IQR für univariate Daten

einige. Typischerweise setzt dieser Wert von  $x$  den IQR-Koeffizienten auf 1,5, was  $\pm 3\sigma$  aus einer Gaußschen Verteilung entspricht. Mit anderen Worten: Wenn der Wert von  $x$  auf 1,5 gesetzt wird, dann erkennt der Algorithmus 0,7% des ganzen Datensatzes als Ausreißer, wenn der Datensatz einer Gaußschen Verteilung folgt. Die Formel in (4.1) ist ein mathematischer Ausdruck von IQR, Obergrenze und Untergrenze.

$$\begin{aligned} \text{IQR} &= Q_3 - Q_1 \\ \text{Obergrenze} &= Q_3 + x * \text{IQR} \\ \text{Untergrenze} &= Q_1 - x * \text{IQR} \end{aligned} \tag{4.1}$$

#### 4.1.2 z-Score Filter (z-Score)

Der z-Score Filter ist ein in der Statistik häufig verwendeter Algorithmus, der misst, wie weit ein beobachteter Wert vom Mittelwert entfernt ist. Im allgemeinen Fall wird er benutzt, wenn ein verwendeter Datensatz einer Gaußschen Verteilung folgt. z-Score ist ein Wert, der misst, wie weit jeder Wert in einer Gaußschen Verteilung vom Durchschnitt abweicht. Diese statistische Technik wird unter Verwendung des beobachteten Werts, des Mittelwerts und der Standardabweichung wie folgt berechnet:  $X$  ist ein beobachteter Wert,  $\mu$  ist der Mittelwert, und  $\sigma$  ist die Standardabweichung.

$$\text{z-Score} = \frac{X - \mu}{\sigma} \tag{4.2}$$

Im Bereich der Ausreißererkennung wird ein Datenpunkt im Allgemeinen dann als Ausreißer definiert, wenn der z-Score größer oder kleiner als  $\pm 1,96$  ist [KM09]. Dies liegt daran, dass die Datenpunkte außerhalb dieses z-Scores ungefähr 5% des gesamten Datensatzes ausmachen.

Wie oben erwähnt, weisen allgemeine statistische Techniken eine optimale Leistung dann auf, wenn sie auf einen Datensatz angewendet werden, der einer Gaußschen Verteilung folgt. Wenn ein Datensatz nicht dieser Verteilung folgt, dann wird er in eine Verteilung geändert, die der Gaußschen Verteilung nahekommt, indem eine Log-Funktion auf den Datensatz angewendet wird, so dass allgemeine statistische Techniken angewendet werden können.

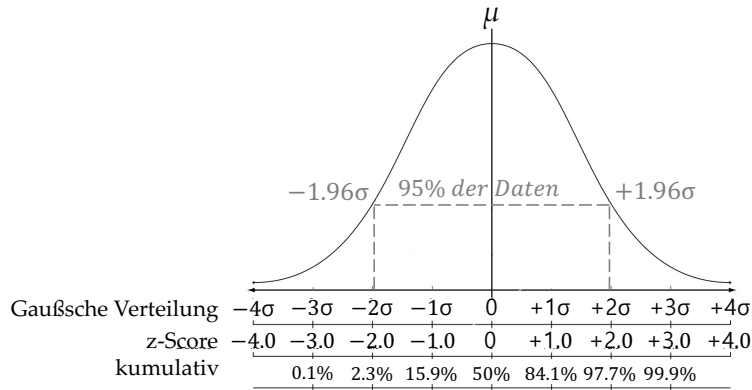


Abbildung 4.2: Konfidenzintervalle, die auf der Standardabweichung und der z-Score basieren

## 4.2 CLUSTER-BASIERTE ALGORITHMEN ZUR AUSREISSERERKENNUNG

Das Ziel des Daten-Clustering, auch bekannt als Cluster-Analyse, besteht darin, die natürliche(n) Gruppierung(en) einer Reihe von Mustern, Punkten oder Objekten aufzudecken [Jai10]. Der cluster-basierte Algorithmus wird je nach Fall in drei Typen unterteilt, und Ausreißer werden je nach Fall wie folgt definiert.

1. Normalwerte gehören zu einem Cluster oder zu mehreren Clustern, Ausreißer zu keinem Cluster. Nachdem Cluster in dem Datensatz gefunden und dazu gehörige Datenpunkte entfernt worden sind, werden die verbleibenden Datenpunkte als Ausreißer behandelt.
2. Bei geringem Abstand zum nächsten Schwerpunkt des Clusters handelt es sich um einen Normalwert, bei großem Abstand um einen Ausreißer. Nachdem das Clustering durchgeführt worden ist, wird der Abstand zwischen der Mitte eines Clusters und einem zu diesem Cluster gehörenden Datenpunkt als „Anomalie-Score“ ermittelt. Daten mit einem Anomalie-Score über einem vorhergestellten Threshold werden als Ausreißer klassifiziert.
3. Normale Datenpunkte gehören zu großen oder zu dichten Clustern und Ausreißer zu kleinen oder zu dünn besetzten Clustern. Die Größe oder die Dichte des Clusters ist ein Kriterium dafür, ob die dazu gehörenden Datenpunkte Ausreißer sind oder nicht.

In diesem Abschnitt wird der K-Means-Clustering für den zweiten Fall beschrieben.

### 4.2.1 K-Means-Clustering (K-Means)

K-Means-Clustering ist ein Clustering-Algorithmus, der jeden Datenpunkt dem diesem Datenpunkt am nächsten gelegenen Cluster zuweist [Llo82]. Als Hyperparameter sollten die maximale Anzahl der Iterationen  $L$ , die Toleranz  $\epsilon$ , die Anzahl der Cluster  $K$  und der anfängliche Mittelpunkt von einem Cluster  $\mu_j^{(0)}$ ,  $j = 1, \dots, K$  gesetzt werden. Das K-Means arbeitet gemäß folgendem Prozess, und die Abbildung 4.3 zeigt den Prozess visuell.

1. Die maximale Anzahl von Iterationen  $L$ , Toleranz  $\epsilon$  und die Anzahl von Clustern  $K$  werden eingestellt, und es werden beliebige Werte  $\mu_j^{(0)}$  zwischen den Maximal- und den Minimalwerten im Datensatz ausgewählt. Diese Werte werden als anfängliche Schwerpunkte des Clusters festgelegt.
2. Sei  $\mu_j^{(t)}$  der Schwerpunkt eines Clusters im  $t$ -ten Schritt. Zunächst wird für alle Datenpunkte  $x_n$  der Abstand zu jedem Schwerpunkt  $\mu_j$  berechnet. Jeder Datenpunkt gehört zu dem Cluster mit dem nächstgelegenen Schwerpunkt. Hier wird die euklidische Distanz eingesetzt. Das heißt, wenn

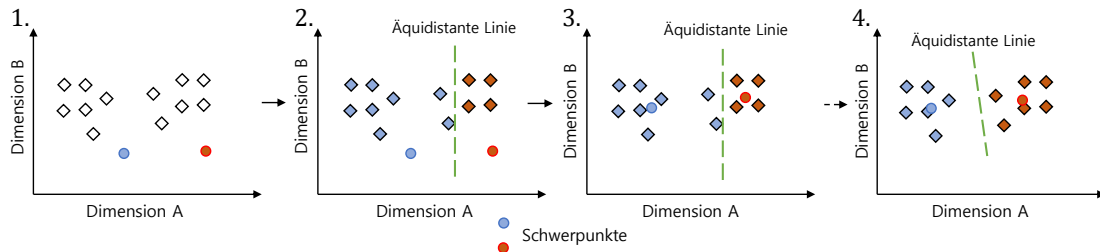
$$k = \operatorname{argmin}_j \|x_i - \mu_j^{(t)}\|^2, k \in K, \quad (4.3)$$

dann ist der Cluster von einem Datenpunkt  $x_i$  gleich  $k$ .

3. Um den Schwerpunkt eines jeden Clusters zu aktualisieren, wird der durchschnittliche Abstand der Datenpunkte innerhalb des Clusters berechnet. Für einen Cluster  $k$  wird der Schwerpunkt bei Iteration  $t + 1$  wie folgt aktualisiert:

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^n r_{ik} x_i}{\sum_{i=1}^n r_{ik}}, r_{ik} = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j \|x_i - \mu_j^{(t)}\|^2, \\ 0 & \text{sonst.} \end{cases} \quad (4.4)$$

4. Wenn  $t > L$  oder  $\|\mu_k^{(t+1)} - \mu_k^{(t)}\| < \epsilon$ , dann endet der Algorithmus, andernfalls geht er zurück zu Schritt 2 und wiederholt diesen.



**Abbildung 4.3:** Verfahren des K-Means-Clustering

Das K-Means-Clustering rekonstruiert kontinuierlich die Cluster des Datensatzes, bis jeder Schwerpunkt gegen einen bestimmten Wert konvergiert. Er konvergiert im Allgemeinen dann, wenn die Ähnlichkeit zwischen den Clustern in einem Datensatz maximal ist. Nach der Konvergierung eines jeden Schwerpunktes wird der Abstand zwischen dem Schwerpunkt von einem Cluster und einem zu diesem Cluster gehörenden Datenpunkt als „Anomalie-Score“ berechnet. Daten mit einem Anomalie-Score über einem vorhergestellten Threshold werden als Ausreißer klassifiziert.

Beim K-Means ist es wichtig, eine geeignete Anzahl von Clustern  $K$  auszuwählen, da man möglicherweise nicht weiß, welche Art von Clustering-Struktur der tatsächliche Datensatz aufweist. Das heuristische Verfahren kann dann verwendet werden, wenn der Wert  $K$  bei der Visualisierung des Datensatzes intuitiv erkannt werden kann. Im Allgemeinen wird jedoch ein Verfahren verwendet, das den Wert  $K$  im Datensatz abschätzen kann. Nach der Auswahl der Anzahl von



Cluster  $K$  ist es ebenfalls ein wichtiger Faktor beim Verbessern der Zeitkomplexität und der Leistung des Algorithmus, die anfänglichen Schwerpunkte des Clusters auszuwählen. In der vorliegenden Arbeit werden die Elbow-Methode, eine der am häufigsten verwendeten Techniken zum Schätzen der Anzahl von Clustern, sowie K-Means++ [AV06], eine der Methoden zum Ermitteln des anfänglichen Schwerpunkts von Clustern, erläutert.

**Elbow-Methode** ist eine Technik, die den  $SSE = \sum_{i=1}^n (y_i - \hat{y}_i)^2$  (Sum of Squared Error) für jede Anzahl von Clustern berechnet und visualisiert und sodann die Anzahl der Cluster auswählt, die dem Teil (Elbow) entspricht, der eine sanfte Neigung zeigt, nachdem er zunächst eine steile Neigung aufgewiesen hat. Diese Methode bestimmt beispielsweise im Fall von Abbildung 4.4, dass der Wert von  $K = 4$  am besten geeignet ist.

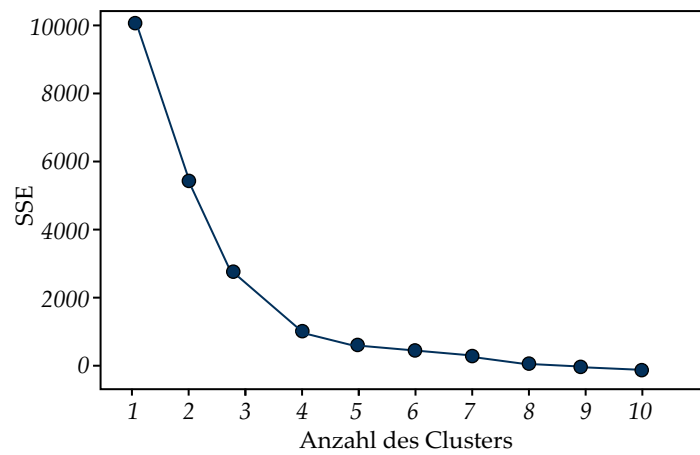


Abbildung 4.4: Elbow-Methode

**K-Means++** folgt diesem Verfahren:

1. Auswahl eines beliebigen Punkt aus den Datenpunkten. Dieser Punkt ist dann der erste Schwerpunkt.
2. Jeder nichtausgewählte Datenpunkt berechnet die Entfernung zum nächstgelegenen Schwerpunkt.
3. Der  $t$ -te Schwerpunkt wird gemäß der Wahrscheinlichkeit proportional zur Entfernung von jedem Punkt ausgewählt. Das heißt, ein Datenpunkt, der so weit wie möglich von einem bereits festgelegten Schwerpunkt entfernt platziert ist, wird als nächster Schwerpunkt festgelegt.
4. Die Schritte 2 und 3 werden solange wiederholt, bis  $K$  Schwerpunkte ausgewählt worden sind.

K-Means++ erfordert zusätzliche Zeitkosten, um den Anfangswert von K-Means festzulegen. Der gewählte Anfangswert ermöglicht es dem K-Means jedoch, die optimale Lösung in  $O(\log K)$ -Zeit zu finden.

Clustering-Algorithmen, einschließlich K-Means, verwenden distanzbasierte Maße, um die Ähnlichkeit zwischen Datenpunkten zu bestimmen. Daher werden für Datensätze mit unterschiedlichen Maßeinheiten für unterschiedliche Features empfohlen, die Daten auf einen Mittelwert von 0 und eine Standardabweichung von 1 zu standardisieren.

## 4.3 DICHTEBASIERTE ALGORITHMEN ZUR AUSREISSERERKENNUNG

Dichtebasierte Ausreißererkennungstechniken schätzen die Dichte der Nachbarschaft eines Datenpunkts. Ein Datenpunkt, der in einer Nachbarschaft mit geringer Dichte liegt, wird als anomal deklariert, während ein Datenpunkt, der in einer dichten Nachbarschaft liegt, als normal deklariert wird. In den beiden folgenden Unterabschnitten 4.3.1 und 4.3.2 werden zwei dichtebasierte Algorithmen, Local Outlier Factor und Isolation Forest, beschrieben.

### 4.3.1 Local Outlier Factor (LOF)

Local Outlier Factor gibt Informationen darüber an, wie weit jeder Datenpunkt von den anderen entfernt ist. Das wichtigste Merkmal von LOF besteht darin, Ausreißer nicht unter Berücksichtigung aller Daten als Ganzes, sondern aus lokaler Sicht anhand von Daten rund um den Datenpunkt zu beurteilen. Zum intuitiven Verständnis mag ein Beispiel wie das folgende dienen:

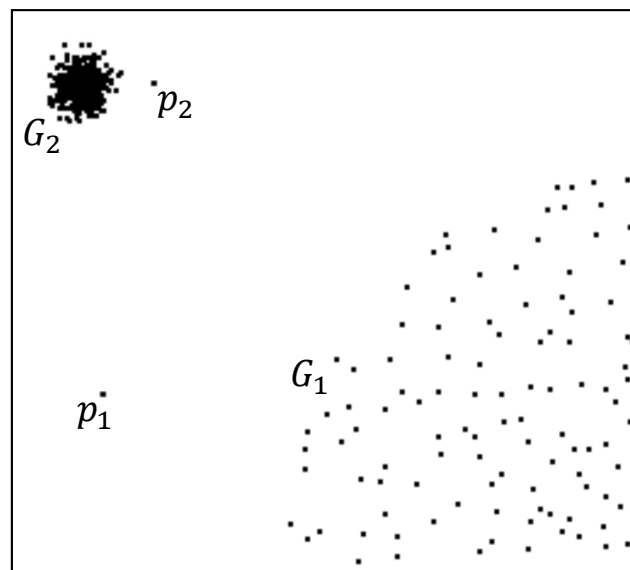


Abbildung 4.5: Visualisierung eines Datensatzes in 2D [BKNS00]

Hier existieren die Gruppe  $G_1$  mit niedriger Dichte, die Gruppe  $G_2$  mit hoher Dichte sowie die Ausreißer  $p_1$  und  $p_2$ . Die meisten Algorithmen zur Erkennung von Ausreißern vergleichen einen jeden Datenpunkt mit den gesamten Daten, um zu ermitteln, ob es sich tatsächlich um einen Ausreißer handelt oder nicht. Derartige Algorithmen betrachten  $p_2$  in Abbildung 4.5 nicht als Ausreißer. Da der Abstand zwischen  $G_2$  und  $p_2$  ähnlich dem Abstand zwischen Datenpunkten der Gruppe  $G_1$  ist, erweist es sich als schwierig,  $p_2$  insgesamt gesehen als Ausreißer einzustufen. Um diese Nachteile zu überwinden, zielt LOF darauf ab, unter Verwendung lokaler Informationen anzuzeigen, wie nah ein Datenpunkt bei dem Ausreißer liegt. Die Arbeitsweise dieses Algorithmus kann mit Hilfe von sechs Definitionen veranschaulicht werden [BKNS00]. In dieser Definition werden die Symbole  $p$ ,  $q$  und  $o$  verwendet, um Datenpunkte in einem Datensatz  $D$  zu bezeichnen.

**Definition 1:** (die Distanz zwischen den Objekten  $p$  und  $q$ )

Hier wird die Notation  $d(p, q)$  verwendet, um den Abstand zwischen Objekten  $p$  und  $q$  zu bezeichnen.

**Definition 2:** ( $k$ -distance eines Objekts  $p$ )

$k$ -distance( $p$ ) ist die Distanz von einem bestimmten Datenpunkt  $p$  zum  $k$ -ten nächsten Nachbarn.

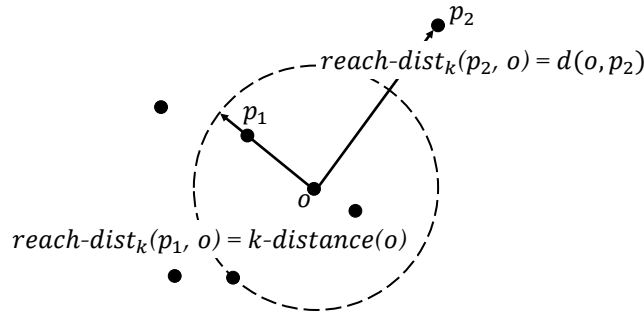
**Definition 3:** ( $k$ -distance Nachbarschaft eines Objekts  $p$ )

Die Anzahl der in der  $k$ -distance( $p$ ) enthaltenen Daten wird als

$$N_{k\text{-distance}(p)}(p) = \{q \in C \setminus \{p\} \mid d(p, q) \leq k\text{-distance}(p)\} \quad (4.5)$$

ausgedrückt, und die Notation wird durch die Abkürzung  $N_k(p)$  vereinfacht.

**Definition 4:** (Erreichbarkeitsdistanz eines Objekts  $p$  in Bezug auf Objekt  $o$ )



**Abbildung 4.6:**  $reach\text{-}dist_k(p_1, o)$  und  $reach\text{-}dist_k(p_2, o)$  für  $k = 3$  [BKNS00]

Sei  $k$  eine natürliche Zahl, dann wird die Erreichbarkeitsdistanz von einem Datenpunkt  $o$  zu einem anderen Datenpunkt  $p$  definiert als

$$reach\text{-}dist_k(p, o) = \max\{k\text{-distance}(o), d(o, p)\}. \quad (4.6)$$

Abbildung 4.6 zeigt das Konzept der Erreichbarkeitsdistanz für  $k = 3$ . Wenn der Datenpunkt  $p_2$  in Abbildung 4.6 weit von  $o$  entfernt ist, dann ist die Erreichbarkeitsdistanz zwischen ihnen intuitiv einfach die tatsächliche Distanz  $d(o, p_2)$ . Wenn er jedoch „nah genug“ ist, wie etwa  $p_1$  in Abbildung 4.6, dann wird die Distanz durch  $k\text{-distance}(o)$  ersetzt. Dieses Verfahren dient als Puffer, um sicherzustellen, dass der Wert von  $d(o, p)$  für alle  $p$  in der Nähe von  $o$  nicht zu klein wird, da später in Definition 6 die Dichte zwischen den Datenpunkten  $p$  und  $o$  durch diese  $reach\text{-}distance$  mit der Dichte zwischen anderen Datenpunkten verglichen wird. Die Stärke dieses Effekts kann durch den Parameter  $k$  gesteuert werden.

**Definition 5:** (lokale Erreichbarkeitsdichte eines Objekts  $p$ )

Die lokale Erreichbarkeitsdichte (local reachability density,  $lrd$ ) für den Datenpunkt  $p$  ist gegeben durch:

$$lrd_k(p) = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} reach\text{-}dist_k(p, o)} \quad (4.7)$$

Die Formel zeigt den Kehrwert des Mittelwerts von  $reach\text{-}dist_k(p, o)$  für den Datenpunkt  $p$ . Diese Formel gibt an, wie weit die  $k$  Nachbarn um den Datenpunkt  $p$  entfernt sind,

also wie dicht sie um den Datenpunkt  $p$  liegen. Mit anderen Worten: Wenn die  $k$  nächsten Nachbarn um den Datenpunkt  $p$  weit von  $p$  entfernt sind, dann ist die durchschnittliche  $reach-dist_k(p, o)$  groß und ergibt sodann die Formel  $lrd_k(p)$  einen kleinen Wert. Auch die umgekehrte Situation kann in Betracht gezogen werden.

**Definition 6:** ((lokaler) Ausreißerfaktor eines Objekts  $p$ )

$LOF_k(p)$  ist der Durchschnitt des Verhältnisses der Dichte  $lrd_k(o)$ ,  $o \in N_k(p)$  und der Dichte  $lrd_k(p)$ . Die Formel lautet wie folgt:

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|} \quad (4.8)$$

Der Wert von  $LOF_k(p)$  kann in drei Fälle unterteilt werden: (i)  $lrd_k(p) \approx lrd_k(o) \Rightarrow LOF_k(p) \approx 1$ ; (ii)  $lrd_k(p) < lrd_k(o) \Rightarrow LOF_k(p) > 1$ ; (iii)  $lrd_k(p) > lrd_k(o) \Rightarrow LOF_k(p) < 1$ . Da die Dichte zwischen den beiden Datenpunkten  $p$  und  $o$  mit der Dichte zwischen anderen Datenpunkten verglichen wird, wird auf diese Weise die  $reach-dist$ , die als ein Puffer fungiert, in Definition 4 definiert. Andernfalls kann das Verhältnis bei Datenpunkten, die sich an einem extrem dichten Ort befinden, aufgrund winziger Unterschiede sehr schwach werden.

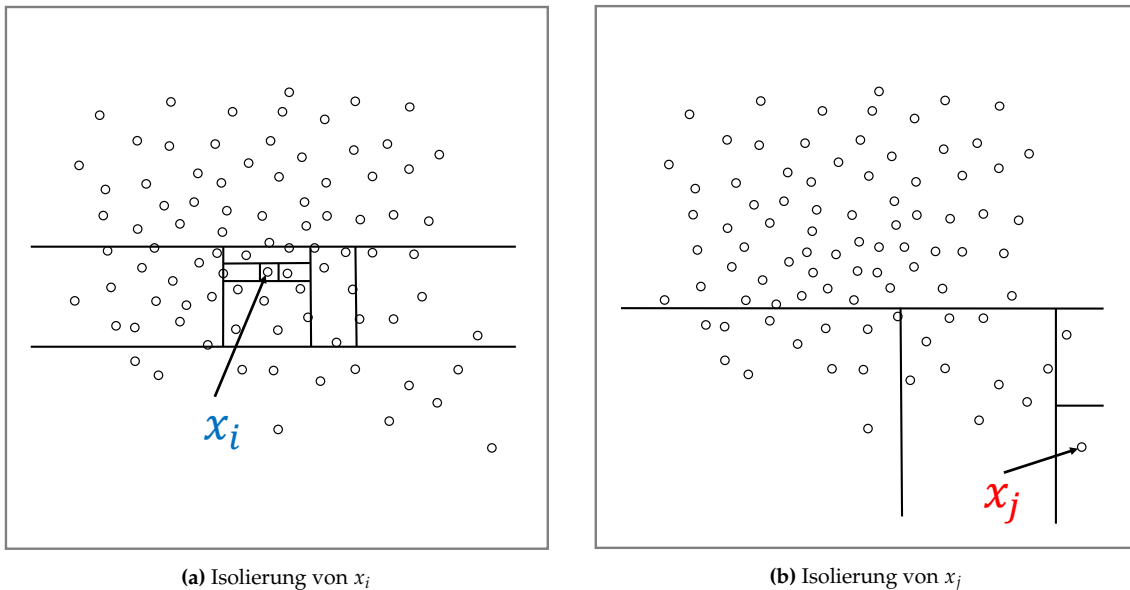
Datenpunkte mit einer geringeren lokalen Dichte als ihre Nachbarn werden als Ausreißer betrachtet, und daher kann ein Wert größer als 1 als Threshold angewendet werden. Es kann also derjenige Wert dieser  $LOF_k(p)$  als „Anomalie-Score“ angesehen werden, der angibt, wie nah  $p$  an einem Ausreißer liegt.

Bisher haben wir uns mit der Funktionsweise von LOF beschäftigt. Der Vorteil von LOF besteht darin, dass es einen Ausreißer dann erkennt, wenn ein Datenpunkt von einem sehr dichten Cluster nicht weit entfernt ist. Die Nachteile dieses Verfahrens bestehen darin, dass sich als chronisches Problem die Frage erweist, welcher Wert als  $k$  gesetzt werden soll. Empirisch wird jedoch empfohlen, ihn auf etwa  $10 \leq k \leq 20$  [BKNS00] einzustellen. Ferner ist es schwierig, herauszufinden, welcher Wert der  $LOF_k(p)$  als Threshold festgelegt werden soll. In einigen Datensätzen ist ein  $LOF_k(p)$ -Wert von 1,5 ein Ausreißer, aber in einem anderen Datensatz kann ein  $LOF_k(p)$ -Wert von 2,0 normal sein. Wie K-Means in Abschnitt 4.2.1 wird auch empfohlen, die Features von Daten zu standardisieren, um eine optimale Algorithmusleistung zu erzielen.

### 4.3.2 Isolation Forest (iForest)

Die Autoren des Papers [LTZ08], die den Isolation Forest-Algorithmus entwickelt haben, definieren Ausreißer als „wenige und unterschiedliche“ Dateninstanzen. Diese Eigenschaften zeigen, dass die Ausreißer anfällig gegenüber einem Mechanismus mit dem Namen Isolierung sind. Die Autoren schlagen einen Algorithmus namens Isolation Forest vor, der Anomalien ausschließlich mit Hilfe des Konzepts der Isolation erkennt, ohne Statistiken, Distanz oder Dichte zu messen [LTZ08]. In diesem Paper bedeutet Isolierung soviel wie „eine Instanz von den anderen zu trennen“. Da Ausreißer stark von den normalen Instanzen abweichen, werden sie beim Splitten schnell isoliert. Da umgekehrt normale Instanzen innerhalb des normalen Datenbereichs liegen, werden sie erst dann isoliert, nachdem sie den Splitprozess mehrmals durchlaufen haben. Abbildung 4.7a zeigt, dass eine normale Dateninstanz  $x_i$  durch vieles Splitten isoliert werden kann, da

sie innerhalb des normalen Datenbereichs liegt. Da andererseits  $x_j$  in Abbildung 4.7b weit von dem Bereich normaler Daten entfernt liegt, ist ersichtlich, dass diese Dateninstanz isoliert worden ist, nachdem sie nur wenige Splitprozesse durchlaufen hat.



**Abbildung 4.7:** Verfahren zur Ausreißererkennung mit Hilfe des Isolation Forest-Algorithmus [LTZ08]

Der Isolation Tree wurde auf der Grundlage der hier angeführten Beobachtungen entworfen. Diese kartesische Koordinatenebene wird durch beliebige Aufteilung ohne irgendwelche Regeln partitioniert, und die Aufteilung wird wiederholt, bis nur eine einzige Dateninstanz auf einer durch diese Partition erzeugten Unterebene liegt. Zu diesem Zeitpunkt wird die Situation, in der nur eine Dateninstanz in einer bestimmten kartesischen Koordinatenebene existiert, als „Isolation“ bezeichnet. Bevor dargelegt wird, wie dieser Algorithmus funktioniert, soll eine bei diesem Algorithmus verwendete Technik, das „Bootstrap-Aggregating“, beschrieben werden.

**Bootstrap Aggregation (Bagging)** ist ein Meta-Algorithmus für Ensemble-Lernverfahren, der entwickelt worden ist, um die Stabilität und die Genauigkeit von maschinellen Lernalgorithmen zu verbessern [Bre96]. Diese Technik erstellt zufällig und wiederholt aus einem Datensatz  $D$  der Größe  $n$  neue Datensätze  $D_i$ , ( $i = 1, 2, \dots, m$ ) der Größe  $n'$ . Solche durch Resampling erstellten neuen Datensätze  $D_i$  werden als „Bootstraps“ bezeichnet. Eine und dieselbe Dateninstanz kann in mehreren Bootstraps vorhanden sein. Das bedeutet, dass jeder Bootstrap eine Datenverteilung aufweist, die sich von der ursprünglichen Datenverteilung unterscheidet, also (positiv) verzerrt ist. Durch eine solche Verzerrung der Datenverteilung kann die Gefahr abgewendet werden, dass möglicherweise ein nur von einem bestimmten Rauschen abhängiges Modell erstellt wird. Daher kann mit einem Ensemble von  $m$ -Modellen ein robustes Modell entwickelt werden, wobei diese Modelle mit Hilfe des Bootstrap  $D_i$  trainiert werden. Wenn das Ergebnis des Modells ein diskreter Wert ist, dann wird es durch das Abstimmungsverfahren aggregiert. Wenn es sich hingegen um einen kontinuierlichen Wert handelt, dann wird es gemittelt. In dieser Arbeit wird das Abstimmungsverfahren deshalb verwendet, weil es sich bei ihm um ein Modell handelt, das Ausreißer einer Dateninstanz bestimmt.

Abbildung 4.8 zeigt den Prozess des Baggings visuell. Himmelblau, rot und blau gefärbte Elemente in Abbildung 4.8a bedeuten Datenobjekte, die in jedem Bootstrap enthalten sind,

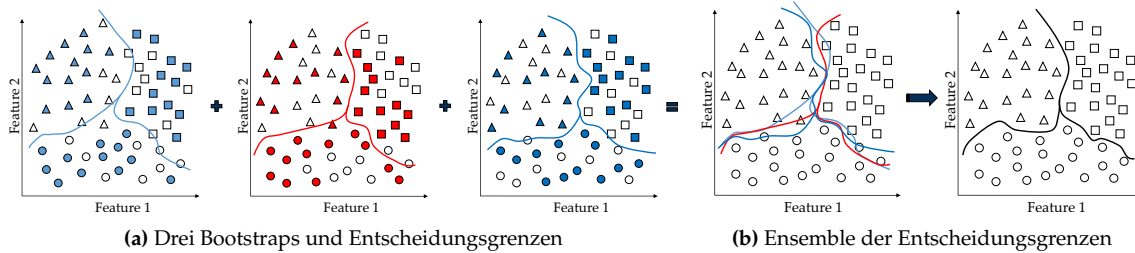


Abbildung 4.8: Verfahren der Bootstrap Aggregation [Pol06]

und weiße Elemente stellen Datenobjekte dar, die von jedem Bootstrap ausgeschlossen sind. Die durchgezogene Linie in jedem Diagramm repräsentiert die Klassifikationsgrenze des Lernmodells. Obwohl die Klassifikationsgrenze einzelner Modelle in Bezug auf die gesamten Daten nicht sehr genau ist, zeigt sich, dass das Aggregationsmodell (Abbildung 4.8b) für die Verteilung der gesamten Daten eine genauere Grenze ergibt.

Der Ansatz des Isolation Forest unter Verwendung der obigen „Bagging“-Technik ist wie folgt beschaffen:

1. Man konstruiert einen binären Tree, der den Bereich beliebig aufteilt, bis jede einzelne Instanz isoliert wird, die in der kartesischen Koordinatenebene vorhanden ist. Ein solcher Baum wird „Isolation Tree“ genannt.
2. Da der erste Schritt ein Ergebnis aus nur einem einzelnen iTree ergibt, ist dieses Ergebnis unzuverlässig, weshalb es notwendig ist, ein normalisiertes Ergebnis zu finden. Diese Idee wird als Isolation Forest umgesetzt, das heißt als ein Ensemble aus mehreren iTrees.
3. Man berechnet die durchschnittliche Pfadlänge einer jeder Dateninstanz und erhält die Anomalie-Scores unter Verwendung des später in Definition 3 definierten Anomalie-Scores. Dateninstanzen mit hohem Anomalie-Score werden als Ausreißer definiert.

Die drei in [LTZ08] angegebenen Definitionen sind eine wissenschaftliche Beschreibung der oben genannten Schritte.

#### Definition 1: (Isolation Tree (iTree))

Ein Isolation Tree  $T$  sei ein binärer Baum. Um einen iTree zu konstruieren, wählt dieser Algorithmus ein beliebiges Feature aus einem Bootstrap  $X = \{x_1, \dots, x_n\}$  aus, der durch das Bagging-Verfahren erstellt wird, sowie einen beliebigen Split-Wert. Danach splittet der Algorithmus den Bootstrap  $X$  rekursiv weiter, bis schließlich die folgende Bedingung erfüllt ist: (i)  $|X| = 1$ ; (ii)  $\forall x, y \in X, x = y$ ; (iii)  $\forall x \in X, h(x) > l$ , wobei  $h(x)$  die Pfadlänge und  $l$  die begrenzende Tiefe von  $T$  ist.

#### Definition 2: (Isolation Forest (iForest))

Abbildung 4.9 zeigt die Variation der Splitanzahl, die erforderlich ist, bis die beiden Instanzen  $x_i$  und  $x_j$  in sämtlichen iTrees isoliert sind. Da die durchschnittliche Pfadlänge mit zunehmender Anzahl von iTrees konvergiert, hat das Ensemble den Vorteil, dass mit ihm ein robusteres Modell erstellt werden kann. Somit wird der iForest als Ensemble aus mehreren iTrees aufgebaut.

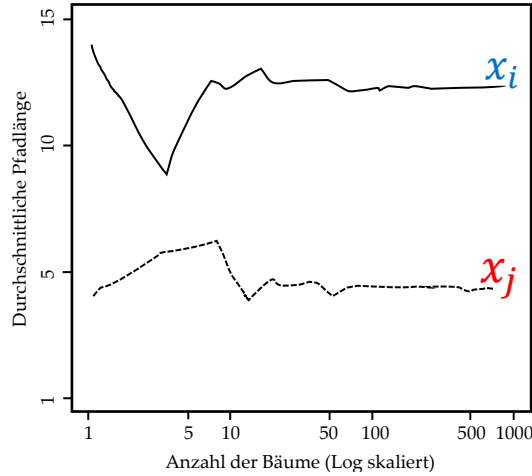


Abbildung 4.9: Konvergenz der durchschnittlichen Pfadlänge von  $x_i$  und  $x_j$  [LTZ08]

**Definition 3:** (Anomalie-Score)

Da ein iTree die gleiche Struktur wie das Binary Search Tree (BST) aufweist, ist die Situation beim Abbruch am externen Knoten die gleiche wie bei einer fehlgeschlagenen Suche im BST. Wir können also die Idee von BST übernehmen, um die durchschnittliche Pfadlänge von iTrees zu schätzen. Bei einem gegebenen Datensatz  $C = \{c_1, \dots, c_n\}$  gibt das Paper [Pre99] die durchschnittliche Pfadlänge fehlgeschlagener Suchen in BST wie folgt an:

$$c(n) = 2H(n-1) - \left(\frac{2(n-1)}{n}\right) \quad (4.9)$$

Hier ist  $H(i)$  eine harmonische Zahl und weist einen Wert von ungefähr  $\ln(i) + \gamma$  (Euler-Mascheroni-Konstante) auf. Der Anomalie-Score  $s$  der Instanz  $a$  ist definiert als:

$$s(a, n) = 2^{\frac{-E(h(a))}{c(n)}} \quad (4.10)$$

wobei  $E(h(a))$  das durchschnittliche  $h(a)$  des iTree ist. Daher gelten die folgenden Bedingungen:

Wenn  $E(h(a)) \rightarrow c(n)$ , dann  $s \rightarrow 0,5$

Wenn  $E(h(a)) \rightarrow 0$ , dann  $s \rightarrow 1$

Wenn  $E(h(a)) \rightarrow n-1$ , dann  $s \rightarrow 0$ .

$s(a, n)$  normalisiert den Wert von  $E(h(a))$  auf  $[0, 1]$ . Wenn  $E(h(a))$  zunimmt, dann nähert sich der Anomalie-Score  $s(a, n)$  dem Wert 0, wenn  $E(h(a))$  hingegen abnimmt, dann nähert sich der Anomalie-Score  $s(a, n)$  dem Wert 1. Dieses Verfahren berechnet den Anomalie-Score für alle Datenpunkte.

Unter der Annahme, dass alle Instanzen gesplittet sind, ist jede Instanz auf einem äußeren Knoten isoliert. In diesem Fall gibt es  $n$  äußere und  $n-1$  innere Knoten. Die Gesamtzahl der Knoten beträgt also  $2n-1$ , und das bedeutet, dass die räumliche Komplexität und die zeitliche Komplexität

linear mit  $n$  zunehmen. [AR04] berichtet, dass, wenn eine große Anzahl von Ausreißern in multivariaten Daten verteilt ist, das Masking-Phänomen sowie das Swamping-Phänomen auftreten, was die Leistungsfähigkeit bei der Suche nach Ausreißern vermindert. Das Masking-Phänomen bezeichnet hier ein solches Phänomen, bei dem Ausreißer fälschlicherweise als Normalwerte klassifiziert werden, wenn sie ein einzelnes Cluster bilden. Das Swamping-Phänomen liegt vor, wenn Normalwerte fälschlicherweise als Ausreißer klassifiziert werden, wenn sie in der Nähe von wirklichen Ausreißern liegen [Mur51]. Da der Algorithmus jedoch Bootstraps verwendet, können Masking- und Swamping-Probleme vermieden werden.



## 5 UMSETZUNG

Der erste Abschnitt dieses Kapitels beschreibt den Aufbau des für die vorliegende Arbeit ausgewählten Datensatzes sowie den Prozess der Vorverarbeitung der Rohdaten. Im folgenden Abschnitt werden konkrete Implementierungen der fünf in Kapitel 4 dargelegten Algorithmen beschrieben. Im letzten Abschnitt geht es schließlich um den allgemeinen Aufbau des Feinstaubprognosemodells.

### 5.1 DATENSATZ

Die vorliegende Arbeit verwendet zwei Feinstaubdatensätze,  $D_h$  und  $D_s$  aus dem DEUS-Projekt<sup>1</sup>. Diese Datensätze werden in Tabelle 5.1 beschrieben.

	Datensätze	
	$D_h$	$D_s$
Zeitraum der Sammlung	01. September. 2020 - 31. August. 2022	
Anzahl der Sensoren (Namen)	4 (215, 216, 218, DEUS)	
Features	<i>timestamp, sensor, temp, regen, windGe, pm</i>	<i>timestamp, sensor, pm</i>
Anzahl der Daten	70,956	6,123,678
Sammelintervall	eine Stunde	durchschnittlich 10,3 Sekunden

(a) Allgemeine Information zu den Datensätzen  $D_h$  und  $D_s$

	Name	Einheit	Datentyp
Datum	<i>timestamp</i>	Sekunde	datetime64
Sensor	<i>sensor</i>	-	String
Temperatur	<i>temp</i>	Grad Celsius ( $^{\circ}\text{C}$ )	
Niederschlag	<i>regen</i>	Millimeter pro Minute ( $\text{mm}/\text{min}$ )	float64
Windgeschwindigkeit	<i>windGe</i>	Meter pro Sekunde ( $\text{m}/\text{s}$ )	
Feinstaubkonzentration	<i>pm</i>	Mikrogramm pro Kubikmeter ( $\mu\text{g}/\text{m}^3$ )	

(b) Information zu den Variablen in den Datensätzen  $D_h$  und  $D_s$

**Tabelle 5.1:** Information zu den Datensätzen  $D_h$  und  $D_s$

<sup>1</sup><http://online.deus-smart-air.com/forschungsprojekt-bmvi/>

Von den beiden Datensätzen wird lediglich der Datensatz  $D_s$  zur Ausreißererkennung eingesetzt. Da das in dieser Arbeit verwendete Modell einen stündlich verfeinerten Datensatz erfordert, werden die Werte von  $pm$  in dem Datensatz  $D_h$  durch stündlich aggregierte Werte von  $pm$  in dem Datensatz  $D_s$  ersetzt, nachdem Ausreißer aus der Variable  $pm$  in dem Datensatz  $D_s$  herausgefiltert worden sind. Anschließend wird der neuerstellte Datensatz  $D_{gefiltert}$  unterteilt. Beispielsweise wird  $D_{IQR}$  in einen Trainingsdatensatz und einen Testdatensatz unterteilt und wird sodann zum Training und zur Evaluation des Modells verwendet. Der Datensatz  $D_s$  ist bereits in einen Trainingsdatensatz und einen Testdatensatz unterteilt, wird aber bei der Ausreißererkennung nicht separat aufgeteilt, sondern in einen großen Datensatz integriert. Dies liegt daran, dass beide Datensätze aus heuristischer Sicht zu viele Ausreißer aufweisen und die Leistung der Ausreißererkennung umso mehr verbessert werden kann, je mehr Daten dafür verwendet werden. Weiterhin ist von Bedeutung, ob die Ausreißererkennung für jeden Sensor separat oder für alle Sensoren auf einmal durchgeführt werden soll. In der vorliegenden Arbeit wird durch Teilung des Sensors der entsprechende Datensatz  $D_s$  als univariate Zeitreihendaten verwendet; denn univariate Zeitreihendaten können dabei helfen, die bei multivariaten Zeitreihendaten häufig auftretenden Masking- und Swamping-Probleme zu vermeiden. Erste Experimente haben gezeigt, dass die Aufteilung der Sensoren in einer um durchschnittlich 1,38% SMAPE besseren Genauigkeit resultieren.

Vor der Ausreißererkennung wird mit Hilfe des Pearson Correlation Coefficient (PCC) analysiert, wie jedes Feature in dem Datensatz  $D_h$  in Beziehung zur Feinstaubkonzentration  $pm$  steht. PCC ist ein Wert, der die Korrelation zwischen zwei Variablen quantifiziert und gemäß der Cauchy-Schwarz-Ungleichung einen Wert zwischen  $+1$  und  $-1$  annimmt. Das Ergebnis der Analyse der Features im Datensatz  $D_h$  außer *timestamp* und *sensor* mit PCC wird in Abbildung 5.1 dargestellt.

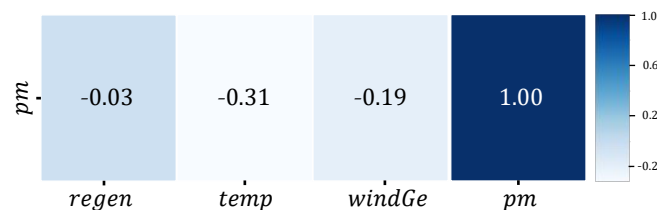


Abbildung 5.1: Die Korrelation zwischen  $pm$  und den Variablen

Da alle Features des Datensatzes außer dem  $pm$  als ausreißerfrei beurteilt werden und es gemäß dem PCC kein Feature gibt, das eine starke Korrelation mit  $pm$  aufwies, wird nur das Feature  $pm$  zur Ausreißererkennung verwendet. Außerdem wurden die duplikaten Daten der Anzahl 2536, die negativen Daten der Anzahl 4 von Datensatz  $D_h$  und die duplikaten Daten der Anzahl 43 sowie die negativen Daten der Anzahl 20 von Datensatz  $D_s$  aus dem Experiment ausgeschlossen.

## 5.2 IMPLEMENTIERUNG DES ALGORITHMUS

Das Forschungsfeld entwickelt sich aufgrund seiner Robustheit, Offenheit, Benutzerfreundlichkeit und des verfügbaren Open-Source-Materials um Python herum. Eben daher wurde Python als Programmiersprache für die Umsetzung der in Kapitel 4 beschriebenen Algorithmen ausgewählt. Wie in Tabelle 5.2 gezeigt, wurde Open-Source-Bibliotheken für z-Score, K-Means, LOF und iForest verwendet. IQR wurde in Python selbst implementiert. Die zusätzlich verwendeten Bibliotheken werden in Tabelle 5.3 aufgelistet.

Algorithmus	Ursprung des Codes
IQR	selbst implementiert in Python
z-Score	SciPy v.1.7.3 in Python
K-Means	selbst implementiert mit der scikit-learn v.1.0.2 in Python
LOF	scikit-learn v.1.0.2 in Python
iForest	scikit-learn v.1.0.2 in Python

**Tabelle 5.2:** Ursprung des Codes für angewandte Algorithmen

Bibliothek	Anwendungszweck
NumPy v.1.21.6	Datentyp der Variablen
Pandas v.1.3.5	Lesen der Daten
matplotlib v.3.2.2	Visualisierung der Anomalien
seaborn v.0.11.2	Rechnen der Dichte und der Wahrscheinlichkeit des Datensatzes

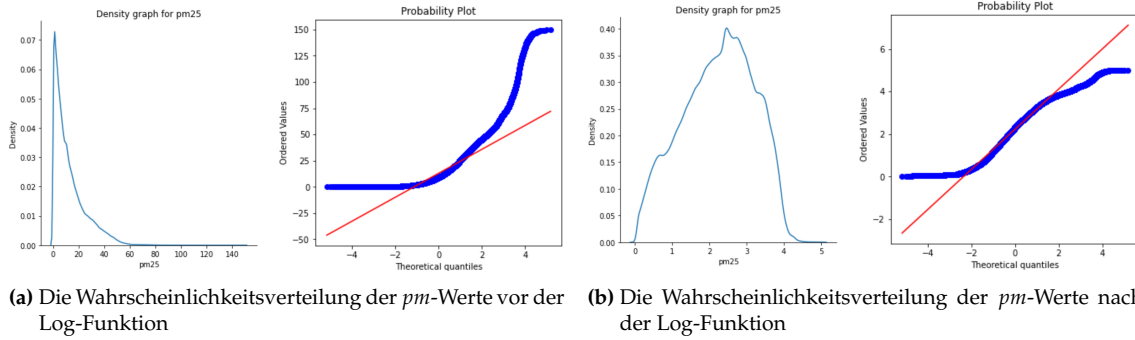
**Tabelle 5.3:** Zusätzlich verwendete Bibliotheken

Im Folgenden werden die Hyperparametereinstellungen der in Kapitel 4 beschriebenen fünf Algorithmen sowie die zur Verbesserung ihrer Leistung geleistete Arbeit beschrieben, bevor der Datensatz  $D_s$  durch diese Algorithmen gefiltert wird.

**IQR:** Als IQR-Koeffizient  $x$  wird der Wert 1,5 gesetzt, der  $\pm 3\sigma$  in der Gaußschen Verteilung entspricht.

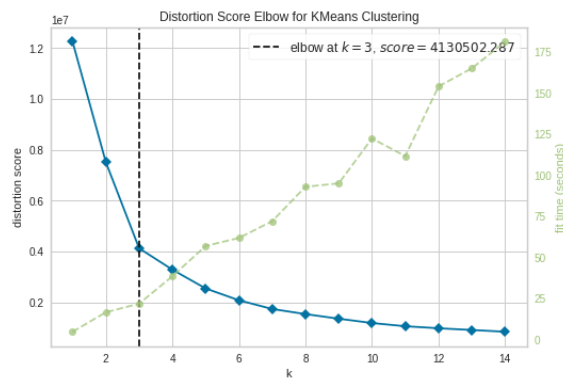
**z-Score:** Als Hyperparameter von z-Score wurde das Threshold von +1,64 verwendet. Laut dem Paper [KM09] ist es üblich, Datenpunkte mit den z-Score-Werten größer oder kleiner als  $\pm 1,96$  als Ausreißer zu definieren. Der in dieser Arbeit verwendete Datensatz scheint jedoch keinen negativen z-Score-Wert zu erfordern. Dies liegt daran, dass die negativen z-Score-Werte die  $pm$ -Werte nahe 0 bedeuten, und aufgrund der physikalischen Eigenschaften der Region, in der die Daten gesammelt wurden, ist es schwierig, die  $pm$ -Werte nahe 0 als Ausreißer zu klassifizieren. Daher wird +1,64, was 5% der Gesamtdaten als Ausreißer definiert, als z-Score-Wert festgelegt, da der im [KM09] vorgeschlagene z-Score-Wert  $\pm 1,96$  5% der Gesamtdaten entspricht. Außerdem wird für diesen Algorithmus, wie bereits in Abschnitt 4.1.2 erwähnt, dann die beste Leistung erzielt, wenn der Datensatz eine Gaußsche Verteilung aufweist. Daher wurde die Verteilung des Datensatzes mit Hilfe der Log-Funktion geändert. Abbildung 5.2 zeigt die Änderung der Wahrscheinlichkeitsverteilung der abhängigen Variablen  $pm$  mittels Log-Funktion. Je mehr Daten um die rote Linie in Abbildung 5.2 herum vorhanden sind, desto näher liegt der Datensatz an der Gaußschen Verteilung. Schließlich wurde eine Standardisierung durchgeführt, die am Ende des Abschnittes 4.1.2 beschrieben wird und mit der die Leistung des Algorithmus noch weiter verbessert werden soll.

**K-Means:** Abbildung 5.3 zeigt den am besten geeigneten Wert von  $K$  im Datensatz, der erzielt wird unter Verwendung der Elbow-Methode, die in Abschnitt 4.2.1 beschrieben worden ist. Gemäß dieser Methode wird  $K = 3$  als am geeignetsten angesehen, jedoch wird der eingestellte Wert des Hyperparameters  $K$  durch ein heuristisches Verfahren auf 4 gesetzt, wobei der Ergebniswert bei Einsatz dieser Methode als Hinweis darauf verwendet wird, welcher Wert von  $K$  am besten geeignet ist. Auch dieser Algorithmus erfordert eine Standardisierung, wie sie daher auch durchgeführt wurde. Da dieser Algorithmus in seinem Ursprung nicht für die Ausreißererkennung angelegt war, muss schließlich zusätzlich ein Threshold



**Abbildung 5.2:** Änderung der Wahrscheinlichkeitsverteilung der abhängigen Variablen  $pm$

gesetzt und müssen Instanzen, die diesen Threshold überschreiten, entfernt werden. In dieser Arbeit wird wie folgt verfahren: Eine Variable  $c$  (contamination) wird als Parameter der K-Means verwendet, und es werden die Instanzen des Datensatzes  $D$  in der Reihenfolge des höchsten Werts basierend auf Anomalie-Score sortiert. Danach wird die erste Instanz bis zur  $|D| \times c$ -ten Instanz aus der sortierten Liste entfernt.



**Abbildung 5.3:** Das mit Hilfe der Elbow-Methode erzielte Ergebnis

**LOF:** Als Wert  $k$  wurde der von [BKNS00] vorgeschlagene Wert 20 verwendet. Der Wert von  $LOF_k(p)$  wurde mit Hilfe eines heuristischen Verfahrens auf die Zahl 2.0 gesetzt. Dieser Algorithmus wurde ebenfalls standardisiert, da er bei einer Standardisierung der Daten eine bessere Leistung ergibt.

**iForest:** Der von scikit-learn bereitgestellte Isolation Forest-Algorithmus verwendet für Threshold dieselbe Methode in K-Means. Daher wurde der Wert der „contamination“ mit dem heuristischen Verfahren auf 0,04 gesetzt. Da dieser Algorithmus keine Standardisierung erfordert, verwendet er vorhandene Daten in unveränderter Form.

Am Ende der oben dargestellten fünf Algorithmen wird eine Visualisierung bereitgestellt. Schließlich wird der oben beschriebene Code auf Google Colab mit einem Intel(R) Xeon(R) und 12GB Arbeitsspeicher implementiert und getestet und wird in diesem System Python in der Version 3.8.16 verwendet.

## 5.3 FEINSTAUBPROGNOSEMODELL

In der vorliegenden Arbeit wird ein neuronales Netzwerk (NN) verwendet, um ein Modell zur Vorhersage der Feinstaubkonzentration zu trainieren und zu evaluieren. NN ist ein Modell, das in der künstlichen Intelligenz verwendet wird. Hierbei handelt es sich um einen Algorithmus, der durch Nachahmung eines biologischen neuronalen Netzwerks erstellt wird. Es besteht aus Schichten von Neuronen, die miteinander verbunden sind und die Dateneingabe in eine gewünschte Ausgabe umwandeln. NN ist in der Lage, komplexe und nichtlineare Zusammenhänge zu erlernen und Probleme zu lösen, die für herkömmliche Algorithmen schwierig oder unlösbar sind. Sie werden in zahlreichen Anwendungen wie etwa Bilderkennung [BWL20], Sprachverarbeitung [VSP<sup>+</sup>17] und Prognosemodellierung [SFGJ20] verwendet.

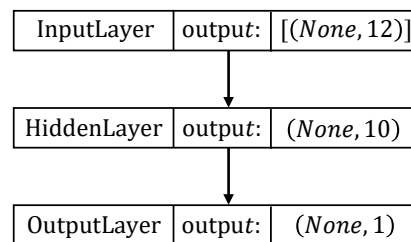


Abbildung 5.4: Zusammenfassung des Modells

Das in dieser Arbeit verwendete Modell [Klingner et al.] weist insgesamt drei Layer auf. Der erste Layer ist der Input-Layer. Er ist für den Empfang von Eingabedaten zuständig. Diese Ebene verwendet als Input zwölf Features, die etwas detaillierter sind als die sechs Features des ursprünglichen Datensatzes. Das Ableiten neuer Features liegt im Rahmen der Modellierungsphase und wird daher in dieser Arbeit nicht behandelt, sondern im nächsten Absatz der Verständlichkeit halber nur kurz erwähnt. Der zweite Layer fungiert als Hidden-Layer. Der dritte Layer ist der Output-Layer, der den Wert *pm* basierend auf den Eingabedaten vorhersagt. Die Struktur des Modells ist in Abbildung 5.4 zu sehen. Dieses Modell verwendet die Bibliothek „Keras“, eine in Python geschriebene Open-Source-Bibliothek für neuronale Netze, die den Prozess des Erstellens, Trainierens und Evaluierens von Deep-Learning-Modellen vereinfacht. Das Modell wird mit Hilfe des Trainingsdatensatzes trainiert, und seine Leistung wird unter Verwendung eines Testdatensatzes bewertet. Der gesamte Datensatz umfasst Trainingsdaten aus dem Zeitraum zwischen September 2020 und August 2021 sowie Testdaten aus dem Zeitraum zwischen September 2021 und August 2022.

Bevor das Modell trainiert wird, durchläuft es einige Vorverarbeitungsschritte, um die Leistung zu verbessern, wie z. B. das Ableiten neuer Features, das im vorigen Absatz erwähnt wurde. Ein Beispiel ist die Erstellung einer neuen Spalte „*pmshift*“, indem die Spalte *pm* um einen Index verschoben wird. Auf diese Weise kann das Modell die Feinstaubkonzentration des Vortages bei der Tagesprognose berücksichtigen, da sie von der Feinstaubbelastung an diesem Tag beeinflusst werden kann. Ein anderes Beispiel ist die Erstellung neuer Spalten „*tdiff1*“ und „*tdiff2*“ für Temperaturdifferenzen. Sie zeigen den Betrag der Temperaturänderung in einem bestimmten Zeitraum (05:00 bis 17:00, 17:30 bis 04:30). Diese Berechnung wird deshalb durchgeführt, weil es einen Zusammenhang zwischen der Feinstaubkonzentration und der Temperatur gibt. Solche Informationen unterstützen das Modell dabei, genauere Vorhersagen zu treffen. Eine detailliertere Erklärung dieser zwei Verfahren würde allerdings den Rahmen dieser Arbeit überschreiten. Schließlich wird die Leistung des Modells mit der SMAPE-Metrik analysiert.



## 6 EVALUATION

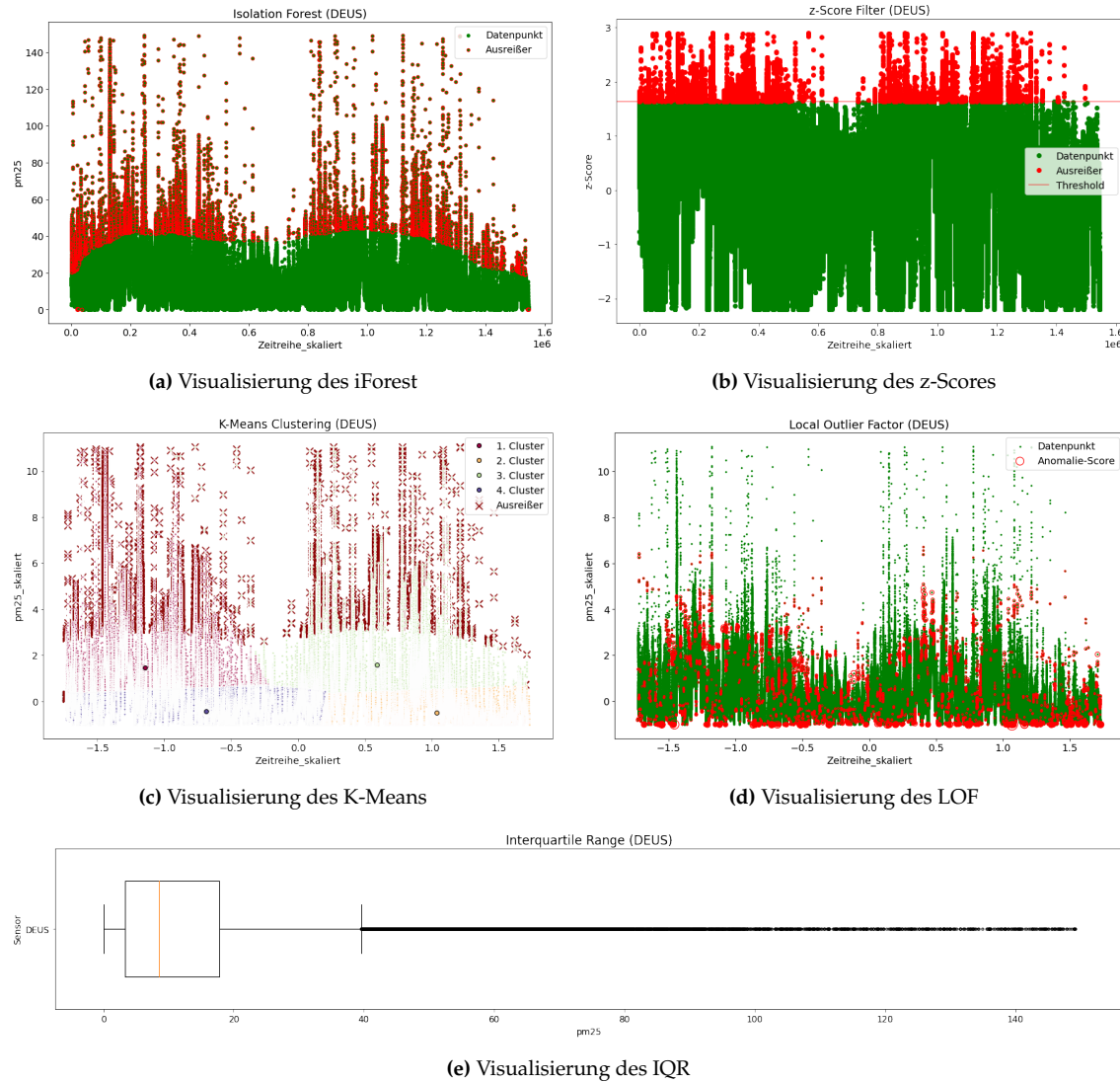
In diesem Kapitel wird anhand der SMAPE-Metrik in quantifizierter Form ermittelt, um wieviel die Leistung des Modells sich verbessert, indem die Ausreißer durch die vorgeschlagene Algorithmen entfernt werden. Schließlich werden die in der vorliegenden Arbeit erzielten Ergebnisse diskutiert.

### ERGEBNISSE

Wie oben erwähnt, wurden nach dem Verschmelzen des Trainings- und des Validierungsdatensatzes die Daten in vier verschiedene Datensätze für vier Sensoren aufgeteilt, und fünf Algorithmen wurden auf jeden Datensatz angewendet, um Ausreißer zu erkennen. Quantitative Informationen zu den Ergebnissen sind in Tabelle 6.1 und visuelle Informationen zu den Ergebnissen in Abbildung 6.1 zu finden. Abbildung 6.1 enthält der Einfachheit halber nur die Ergebnisse für den Sensor „DEUS“.

ungefiltert						
Algorithmus	SMAPE Validierung(%)					
	Sensoren				Statistik	
	215	216	218	DEUS	AVG	Diff.
-	55.15	56.76	57.02	54.98	55.98	0.00
gefiltert						
Algorithmus	SMAPE Validierung(%)					
	Sensoren				Statistik	
	215	216	218	DEUS	AVG	Diff.
iForest	53.25	53.81	54.60	52.58	53.56	-2.42
IQR	53.36	54.54	54.21	53.13	53.81	-2.17
z-Score	53.13	54.50	54.81	54.45	54.22	-1.76
K-Means	54.51	56.97	56.37	55.36	55.80	-0.18
LOF	55.15	56.84	57.05	55.12	56.04	+0.06

**Tabelle 6.1:** Die Ergebnisse in SMAPE


**Abbildung 6.1:** Ergebnisse in 2D

Gemäß Tabelle 6.1 zeigen die Ausreißererkennungen mit den iForest-, den IQR-, den z-Score- und den K-Means-Algorithmen einen positiven Effekt auf die Leistungsfähigkeit des Modells, wohingegen der LOF-Algorithmus eher einen negativen Einfluss ausübt. Unter den genannten vier Algorithmen haben wir das Modell mit der besten Verbesserung bei der Ausreißererkennung mit dem iForest konstruiert. Wie in Abbildung 6.1a dargestellt, die eine visuelle Metrik zur Ausreißererkennung verwendet, hat die Trennlinie zwischen normalen und Ausreißerdaten eine schöne Kurve erzeugt. Laut der numerischen Metrik SMAPE hat die Entfernung von Ausreißern die Leistung des Modells um 2,42% im Vergleich mit dem bestehenden Modell verbessert.

Dieses negative Ergebnis für den LOF scheint darauf zurückzuführen zu sein, dass zu viele signifikante Daten entfernt wurden, wie Abbildung 6.1d zeigt. Mit anderen Worten, es ist ersichtlich, dass dieses Ergebnis durch das Entfernen von Daten aus dichten Bereichen verursacht wurde. Dies ist eine Eigenschaft des LOF-Algorithmus, denn im Gegensatz zu anderen Algorithmen, bei denen nur Daten aus Bereichen mit geringer Dichte entfernt werden, entfernt der LOF-Algorithmus die Datenpunkte, die in einem dichten Bereich vorhanden sind, wenn sie niedriger als die durchschnittliche Dichte des Bereichs sind. Das K-Means-Clustering ist gleichfalls



weniger leistungsfähig als andere iForest-, IQR- und z-Score-Algorithmen, da einige Daten entfernt werden, die in dichten Bereichen vorhanden sind. Natürlich hat dieser Algorithmus auch eine kleine Menge sinnloser Daten entfernt, so dass er im Gegensatz zu LOF die Leistung des Modells verbessert hat.

Unter IQR und z-Score zeigt der IQR eine geringfügig bessere Leistung. Der Grund dafür kann durch folgenden Faktor erklärt werden: IQR ist ein robuster Algorithmus für die Datenstreuung, da es auf demjenigen Bereich basiert, in dem die Mehrheit der Daten liegt (1. Quartil bis 3. Quartil). z-Score ist dagegen auf den Mittelwert basiert und kann leicht von Ausreißern beeinflusst werden, da er unter Bezug auf den Abstand von jedem Datenpunkt zum Mittelwert berechnet wird. Dies bedeutet, dass der IQR robust gegenüber Ausreißern ist und sich gut für schiefe Verteilungen eignet, während der z-Score empfindlicher auf Ausreißer reagiert und besser für eine Gaußsche Verteilung geeignet ist. Daher scheint der IQR in diesem Fall besser als z-Score abzuschneiden, wie man daran erkennt, dass der in dieser Arbeit verwendete Datensatz eine starke Verzerrung aufweist. Da diese beiden auf Statistik basierten Algorithmen jedoch keine zusätzlichen Informationen aus Zeitreihendaten verwenden können, ist die Grenze dieser Algorithmen klar.

Der Grund für die beste Leistung von iForest, die 4% der Gesamtdaten als Ausreißer klassifiziert hat, lässt sich deduktiv daraus ableiten, dass dieser Algorithmus keine Daten aus dichten Bereichen entfernt, nicht von der Verteilung der Daten abhängt und die Zeitreihen berücksichtigt hat. Dies kann dahingehend interpretiert werden, dass die Aufmerksamkeit auf diese Teile eine optimale Leistung bei der Ausreißererkennung dann erzielen kann, wenn ein Datensatz aus intuitiver Sicht eine große Anzahl von Ausreißern oder eine stark verzerrte Verteilung aufweist.



## 7 FAZIT & AUSBLICK

Das Ziel dieser Arbeit besteht darin, zu untersuchen, wie die Entfernung von Ausreißern durch verschiedene Algorithmen die Leistung eines Prognosemodells beeinflusst, das auf dem Feinstaubdatensatz trainiert wurde. Zu diesem Zweck wurden fünf verschiedene Algorithmen zur Erkennung von Ausreißern untersucht, um den Einfluss eines jeden dieser Algorithmen auf die Leistung des Feinstaubprognosemodells zu analysieren. Die Ergebnisse zeigen, dass sämtliche Algorithmen außer LOF die Leistung des Modells verbessern. In Bezug auf LOF scheint es immerhin eine Möglichkeit zu geben, die Leistung zu verbessern, indem die optimalen Werte für die Hyperparameter bestimmt werden, wie in [XKC19] diskutiert. Laut dem Paper [SVWP<sup>+</sup>21] kann eine Leistungssteigerung von 13,0% in Form von AUC erzielt werden, wenn die optimalen Hyperparameter eingestellt werden. Darüber hinaus kann die Verwendung des in [HKB21] vorgestellten „Extended Isolation Forest“ eine Verbesserung der Leistung des iForest ergeben. Es ist jedoch zu beachten, dass die vorliegende Arbeit auch einige Einschränkungen aufweist. Zum Beispiel ist der verwendete Datensatz in Bezug auf Anzahl und Vielfalt der berücksichtigten Features begrenzt. In einer weiteren Arbeit könnten und sollten größere und vielfältigere Datensätze verwendet werden, um eine noch bessere Vorhersageleistung zu erzielen.

Es wird erwartet, dass die Ergebnisse dieser Arbeit dazu beitragen, Empfehlungen für den Einsatz von Algorithmen in bestimmten Forschungssituationen zu liefern und somit die Entscheidungsfindung in der Praxis zu verbessern. Zusammenfassend zeigt diese Arbeit, dass die Anwendung von Ausreißererkennungsalgorithmen bei der Vorhersage der Feinstaubkonzentration einen positiven Effekt hat. Es gibt jedoch noch weitere Möglichkeiten, die Leistung weiter zu verbessern. In der Zukunft könnten und sollten weitere Untersuchungen durchgeführt werden, um Schritt für Schritt die besten Algorithmen für diese Art von Problemen zu bestimmen und um die Leistungsfähigkeit von bereits vorhandenen Algorithmen weiter zu erhöhen.



# LITERATURVERZEICHNIS

- [Ans60] ANSCOMBE, Frank J.: Rejection of outliers. In: *Technometrics* 2 (1960), Nr. 2, S. 123–146
- [AO22] ASLAN, Meryem ; ONUT, Semih: Detection of Outliers and Extreme Events of Ground Level Particulate Matter Using DBSCAN Algorithm with Local Parameters. In: *Water, Air, Soil Pollution* 233 (2022), 06. <http://dx.doi.org/10.1007/s11270-022-05679-6>. – DOI 10.1007/s11270-022-05679-6
- [AR04] ACUÑA, Edgar ; RODRIGUEZ, Caroline: The Treatment of Missing Values and its Effect on Classifier Accuracy. In: BANKS, David (Hrsg.) ; MCMORRIS, Frederick R. (Hrsg.) ; ARABIE, Phipps (Hrsg.) ; GAUL, Wolfgang (Hrsg.): *Classification, Clustering, and Data Mining Applications*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2004. – ISBN 978-3-642-17103-1, S. 639–647
- [AV06] ARTHUR, David ; VASSILVITSKII, Sergei: k-means++: The advantages of careful seeding / Stanford. 2006. – Forschungsbericht
- [BJGS12] BAR-JOSEPH, Ziv ; GITTER, Anthony ; SIMON, Itamar: Studying and modelling dynamic biological processes using time-series gene expression data. In: *Nature Reviews Genetics* 13 (2012), Aug, Nr. 8, 552-564. <http://dx.doi.org/10.1038/nrg3244>. – DOI 10.1038/nrg3244. – ISSN 1471-0064
- [BKNS00] BREUNIG, Markus M. ; KRIEGLER, Hans-Peter ; NG, Raymond T. ; SANDER, Jörg: LOF: identifying density-based local outliers. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, S. 93–104
- [Bra] BRAATZ, Aaron: Evaluation von Ausreißer-Behandlung und Auswirkungen von Umwelteinflüssen auf Feinstaub-Analysen.
- [Bre96] BREIMAN, Leo: Bagging predictors. In: *Machine Learning* 24 (1996), August, Nr. 2, 123–140. <http://dx.doi.org/10.1007/BF00058655>. – DOI 10.1007/BF00058655. – ISSN 1573-0565
- [BWL20] BOCHKOVSKIY, Alexey ; WANG, Chien-Yao ; LIAO, Hong-Yuan M.: YOLOv4: Optimal Speed and Accuracy of Object Detection. In: *ArXiv abs/2004.10934* (2020)

- [CBK09] CHANDOLA, Varun ; BANERJEE, Arindam ; KUMAR, Vipin: Anomaly Detection: A Survey. In: *ACM Comput. Surv.* 41 (2009), jul, Nr. 3. <http://dx.doi.org/10.1145/1541880.1541882>. – DOI 10.1145/1541880.1541882. – ISSN 0360-0300
- [CMM<sup>+</sup>18] CHUI, Michael ; MANYIKA, James ; MIREMADI, Mehdi ; HENKE, Nicolaus ; CHUNG, Rita ; NEL, Pieter ; MALHOTRA, Sankalp: Notes from the AI frontier: Insights from hundreds of use cases. In: *McKinsey Global Institute* (2018), S. 28
- [DF13] DING, Zhiguo ; FEI, Minrui: An Anomaly Detection Approach Based on Isolation Forest Algorithm for Streaming Data using Sliding Window. In: *IFAC Proceedings Volumes* 46 (2013), Nr. 20, 12-17. <http://dx.doi.org/https://doi.org/10.3182/20130902-3-CN-3020.00044>. – DOI <https://doi.org/10.3182/20130902-3-CN-3020.00044>. – ISSN 1474-6670. – 3rd IFAC Conference on Intelligent Control and Automation Science ICONS 2013
- [EA12] ESLING, Philippe ; AGON, Carlos: Time-Series Data Mining. In: *ACM Comput. Surv.* 45 (2012), dec, Nr. 1. <http://dx.doi.org/10.1145/2379776.2379788>. – DOI 10.1145/2379776.2379788. – ISSN 0360-0300
- [GAG<sup>+</sup>00] GOLDBERGER, Ary L. ; AMARAL, Luis A. ; GLASS, Leon ; HAUSDORFF, Jeffrey M. ; IVANOV, Plamen C. ; MARK, Roger G. ; MIETUS, Joseph E. ; MOODY, George B. ; PENG, Chung-Kang ; STANLEY, H E.: PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. In: *circulation* 101 (2000), Nr. 23, S. e215–e220
- [HA14] HYNDMAN, R.J. ; ATHANASOPOULOS, G.: *Forecasting: Principles and Practice*. OTexts, 2014 <https://books.google.de/books?id=nmTQwAEACAAJ>. – ISBN 9780987507105
- [Haw80] In: HAWKINS, D. M.: *Introduction*. Dordrecht : Springer Netherlands, 1980. – ISBN 978-94-015-3994-4, 1-12
- [HKB21] HARIRI, Sahand ; KIND, Matias C. ; BRUNNER, Robert J.: Extended Isolation Forest. In: *IEEE Transactions on Knowledge and Data Engineering* 33 (2021), Nr. 4, S. 1479-1489. <http://dx.doi.org/10.1109/TKDE.2019.2947676>. – DOI 10.1109/TKDE.2019.2947676
- [HZZ<sup>+</sup>13] HUANG, Tian ; ZHU, Yan ; ZHANG, Qiannan ; ZHU, Yongxin ; WANG, Dongyang ; QIU, Meikang ; LIU, Lei: An lof-based adaptive anomaly detection scheme for cloud computing. In: *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops IEEE*, 2013, S. 206-211
- [Jai10] JAIN, Anil K.: Data clustering: 50 years beyond K-means. In: *Pattern Recognition Letters* 31 (2010), Nr. 8, 651-666. <http://dx.doi.org/https://doi.org/10.1016/j.patrec.2009.09.011>. – DOI <https://doi.org/10.1016/j.patrec.2009.09.011>. – ISSN 0167-8655. – Award winning papers from the 19th International Conference on Pattern Recognition (ICPR)
- [JYK20] JEON, Young T. ; YU, Suk H. ; KWON, Hee Y.: Improvement of PM Forecasting Performance by Outlier Data Removing. In: *Journal of Korea Multimedia Society* (2020)
- [KCMC20] KAN, Siyi ; CHEN, Bin ; MENG, Jing ; CHEN, Guoqian: An extended overview of natural gas use embodied in world economy and supply chains: Policy implications from a time series analysis. In: *Energy Policy* 137 (2020),

111068. <http://dx.doi.org/https://doi.org/10.1016/j.enpol.2019.111068>. – DOI <https://doi.org/10.1016/j.enpol.2019.111068>. – ISSN 0301-4215
- [KM09] KILLOURHY, Kevin S. ; MAXION, Roy A.: Comparing anomaly-detection algorithms for keystroke dynamics. In: *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, 2009, S. 125–134
- [LIPJ21] LI, Jinbo ; IZAKIAN, Hesam ; PEDRYCZ, Witold ; JAMAL, Iqbal: Clustering-based anomaly detection in multivariate time series data. In: *Applied Soft Computing* 100 (2021), 106919. <http://dx.doi.org/https://doi.org/10.1016/j.asoc.2020.106919>. – DOI <https://doi.org/10.1016/j.asoc.2020.106919>. – ISSN 1568-4946
- [Llo82] LLOYD, S.: Least squares quantization in PCM. In: *IEEE Transactions on Information Theory* 28 (1982), Nr. 2, S. 129–137. <http://dx.doi.org/10.1109/TIT.1982.1056489>. – DOI 10.1109/TIT.1982.1056489
- [LTZ08] LIU, Fei T. ; TING, Kai M. ; ZHOU, Zhi-Hua: Isolation forest. In: *2008 eighth ieee international conference on data mining IEEE*, 2008, S. 413–422
- [MKPT20] MAHAJAN, Manish ; KUMAR, Santosh ; PANT, Bhasker ; TIWARI, Umesh K.: Incremental Outlier Detection in Air Quality Data Using Statistical Methods. In: *2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI)*, 2020, S. 1–5
- [Mur51] MURPHY, R. B.: On Tests for Outlying Observations. In: *Princeton University* (1951)
- [Pol06] POLIKAR, R.: Ensemble based systems in decision making. In: *IEEE Circuits and Systems Magazine* 6 (2006), Nr. 3, S. 21–45. <http://dx.doi.org/10.1109/MCAS.2006.1688199>. – DOI 10.1109/MCAS.2006.1688199
- [Pre99] PREISS, Bruno R.: *Data Structure and Algorithms: With Object-oriented Design Patterns in Java*. John Wiley & Sons, 1999
- [SF01] SOHN, Hoon ; FARRAR, Charles R.: Damage diagnosis using time series analysis of vibration signals. In: *Smart Materials and Structures* 10 (2001), jun, Nr. 3, 446. <http://dx.doi.org/10.1088/0964-1726/10/3/304>. – DOI 10.1088/0964-1726/10/3/304
- [SFGJ20] SALINAS, David ; FLUNKERT, Valentin ; GASTHAUS, Jan ; JANUSCHOWSKI, Tim: DeepAR: Probabilistic forecasting with autoregressive recurrent networks. In: *International Journal of Forecasting* 36 (2020), Nr. 3, 1181-1191. <http://dx.doi.org/https://doi.org/10.1016/j.ijforecast.2019.07.001>. – DOI <https://doi.org/10.1016/j.ijforecast.2019.07.001>. – ISSN 0169-2070
- [SJLD15] SHAADAN, Norshahida ; JEMAIN, Abdul A. ; LATIF, Mohd T. ; DENI, Sayang M.: Anomaly detection and assessment of PM10 functional data at several locations in the Klang Valley, Malaysia. In: *Atmospheric Pollution Research* 6 (2015), Nr. 2, 365-375. <http://dx.doi.org/https://doi.org/10.5094/APR.2015.040>. – DOI <https://doi.org/10.5094/APR.2015.040>. – ISSN 1309-1042
- [SVWP<sup>+</sup>21] SOENEN, Jonas ; VAN WOLPUTTE, Elia ; PERINI, Lorenzo ; VERCRUYSEN, Vincent ; MEERT, Wannes ; DAVIS, Jesse ; BLOCKEEL, Hendrik: The effect of hyperparameter tuning on the comparative evaluation of unsupervised anomaly detection methods. In: *Proceedings of the KDD'21 Workshop on Outlier Detection and Description* Outlier Detection and Description Organising Committee, 2021, S. 1–9

- [SYT<sup>+</sup>05] SAKAMOTO, Toshihiro ; YOKOZAWA, Masayuki ; TORITANI, Hitoshi ; SHIBAYAMA, Michio ; ISHITSUKA, Naoki ; OHNO, Hiroyuki: A crop phenology detection method using time-series MODIS data. In: *Remote Sensing of Environment* 96 (2005), Nr. 3, 366-374. <http://dx.doi.org/https://doi.org/10.1016/j.rse.2005.03.008>. – DOI <https://doi.org/10.1016/j.rse.2005.03.008>. – ISSN 0034–4257
  
- [VSP<sup>+</sup>17] VASWANI, Ashish ; SHAZEER, Noam ; PARMAR, Niki ; USZKOREIT, Jakob ; JONES, Llion ; GOMEZ, Aidan N. ; KAISER, Ł u. ; POLOSUKHIN, Illia: Attention is All you Need. In: GUYON, I. (Hrsg.) ; LUXBURG, U. V. (Hrsg.) ; BENGIO, S. (Hrsg.) ; WALLACH, H. (Hrsg.) ; FERGUS, R. (Hrsg.) ; VISHWANATHAN, S. (Hrsg.) ; GARNETT, R. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 30, Curran Associates, Inc., 2017
  
- [WD16] WAZID, Mohammad ; DAS, Ashok K.: An efficient hybrid anomaly detection scheme using K-means clustering for wireless sensor networks. In: *Wireless Personal Communications* 90 (2016), S. 1971–2000
  
- [XKC19] XU, Zekun ; KAKDE, Deovrat ; CHAUDHURI, Arin: Automatic Hyperparameter Tuning Method for Local Outlier Factor, with Applications to Anomaly Detection. In: *2019 IEEE International Conference on Big Data (Big Data)*, 2019, S. 4201–4207