Ostfalia

Hochschule für angewandte Wissenschaften

Fakultät Maschinenbau

Institut für Mechatronik

Bachelorarbeit

Einsatzmöglichkeiten von Verfahren des unüberwachten Lernens für die Kategorisierung von Nutzertypen an Ladesäulen

---

Verfasser:	Alexander Kohn
Matrikelnummer:	70455099
Studiengang:	Maschinenbau
Eingereicht am:	13. September 2022

Erstprüfer:	Prof. Dr.-Ing. M. Strube
Zweitprüfer:	Prof. Dr.-Ing. C. Hartwig

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne unerlaubte Hilfe angefertigt, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Königslutter, 13.09.2022

Kurzfassung

Die rasante Zunahme von Elektrofahrzeugen erfordert einen effektiven Ausbau der Ladeinfrastruktur. Das Verhalten der Kunden spielt dabei eine wichtige Rolle. Deren Nutzerverhalten gibt Aufschluss für die Planung neuer Ladesäulen bzw. zur Optimierung der Auslastung bereits bestehender Ladesäulen. Im Rahmen dieser Arbeit wird das „Maschinelle Lernen“ für die Kategorisierung von Nutzertypen angewendet. Die Kategorisierung wird mit Hilfe der Clusteranalyse aus dem Bereich des unüberwachten Lernens durchgeführt. Für diese Untersuchung wurden die Daten der Ladesäulen von der Ostfalia genutzt und drei unterschiedliche Clusterverfahren verglichen: K-Means, DBSCAN und Gaussian Mixture Model (GMM). Das GMM konnte die besten Ergebnisse für die vorhandenen Daten erzielen. Dabei wurden sechs Cluster mit unterschiedlichen Tendenzen in ihren Charakteristiken mit Hilfe von Ankunftszeit, Abfahrzeit, maximaler Leistung, Ladeanteil und Gesamtverbrauch identifiziert. Das Ergebnis sind vier unterschiedliche Nutzertypen. Diese wurden als „Nachtlader“, „Zwischendurchlader“, „Arbeitslader“ und „Nachmittagslader“ bezeichnet.

Schlüsselwörter

Clusteranalyse, Clusterverfahren, K-Means, DBSCAN, GMM

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die Ladeinfrastruktur ist ein wichtiger Baustein für die Transformation des Verkehrssektors zur Elektromobilität. Ein flächendeckender Ausbau erhöht einen Anreiz zum Kauf eines Elektrofahrzeugs und verringert damit den Kohlendioxidausstoß durch die Reduzierung der Verbrennungsmotoren. Die Anzahl der Elektrofahrzeuge hat in den letzten Jahren deutlich zugenommen. Die meisten Besitzer laden ihre Elektrofahrzeugen zu Hause, was sich in Zukunft zwangsläufig stärker auf öffentliche Ladesäulen verschieben wird. Viele zukünftige Besitzer werden aus Ballungsgebieten kommen, wo sie kaum die Möglichkeit besitzen ihr Elektrofahrzeug an einem privaten Anschluss zu laden. Sie sind somit auf öffentliche Ladesäulen angewiesen. Dies belegt eine Studie der UNO, welche prognostiziert, dass 68% der Weltbevölkerung im Jahre 2050 in Großstädten leben wird [UNO18]. Das Laden von E-Fahrzeugen unterscheidet sich zeitlich vom Tanken von herkömmlichen Verbrennern und erfordert eine genaue Planung für den öffentlichen Bereich. Das Ziel beim Ausbau der Ladeinfrastruktur ist, dass alle Besitzer ihren Bedarf flexibel decken können. Um den steigenden Bedarf durch die Zunahme von Elektrofahrzeugen zu decken, ist es wichtig die Ladeinfrastruktur in kurzer Zeit effizient auszubauen und optimale Standorte zu wählen. Zudem müssen vorhandene Ladesäulen hinsichtlich ihrer Auslastung optimiert werden. Dafür müssen wichtige Größen, wie Art und Anzahl der Fahrzeuge, sowie die Ladeleistung und das Verhalten der Besitzer berücksichtigt werden.

1.2 Zielsetzung

Die heutige Informationsbasis für die Betreiber von Ladesäulen liefert meist keine hinreichenden Informationen zur potenziell möglichen Vermarktung dieser Ladesäulen. Aus den Ladedaten geht in der Regel nicht hervor, welcher Fahrzeugtyp bzw. Nutzertyp eine Ladesäule häufig aufsucht und wie sich die Auslastung der Ladesäulen optimieren ließe. Weiterhin ist nicht klar, welche Art der Kundengruppierung ein geeignetes Mittel für eine Potenzialanalyse an einem bestimmten Standort darstellt. Im Rahmen dieser Bachelorarbeit sollen Verfahren des maschinellen Lernens eingesetzt werden, um die heute nicht zur Verfügung stehenden Kategorisierungen von Nutzertypen experimentell zu generieren. Als Anwendungsbeispiel sollen die Ladesäulen der Ostfalia genutzt werden.

Ziel der Arbeit ist es, das Einsatzpotenzial des Maschinellen-Lernens für die Kategorisierung von Nutztypen an Ladesäulen zu untersuchen. Konkret wird dazu die Clusteranalyse aus dem Bereich des unüberwachten Lernens verwendet. Der Fokus liegt zum einen darauf, einen Überblick über verschiedene Clusterverfahren zu geben, zum anderen auf die Wahl eines potenziell geeigneten Clusterverfahrens für die Kategorisierung von Nutzertypen. Die ausgewählten Clusterverfahren werden daraufhin mit Hilfe der Daten von den Ladesäulen der Ostfalia experimentell Untersucht. Neben der Anwendung der Clusterverfahren und der Validierung der Ergebnisse, steht zudem die Vorgehensweise bei einer Clusteranalyse im Vordergrund. Dazu gehört die Analyse und Vorverarbeitung der Rohdaten von den Ladesäulen. Sowie die Auswahl von geeigneten Clusterverfahren anhand von verschiedenen Kriterien.

1.3 Anwendungsbeispiel Standort Ostfalia

Die Ostfalia betreibt seit August 2021 in Wolfenbüttel am Gebäude H zwei Ladesäulen der Firma Compleo und seit März 2022 eine weitere Ladesäule der Firma MENNEKES am Gebäude R. Außerdem sind zwei weitere Ladesäulen seit Dezember 2018 am Standort Salzgitter von der Firma MENNEKES im Betrieb.

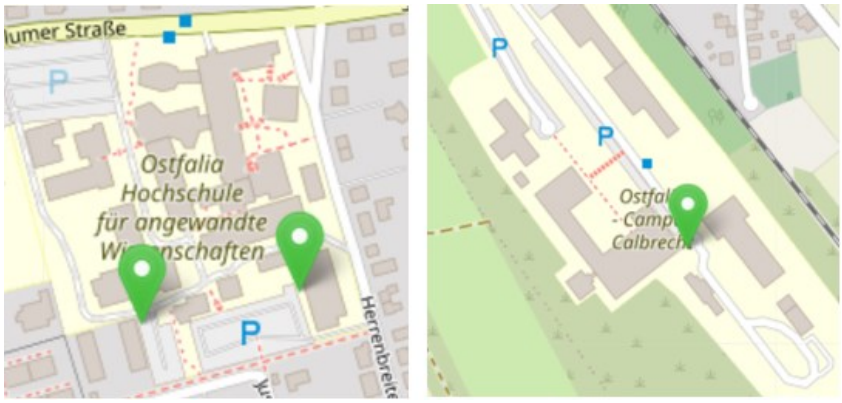


Abbildung 1.1: Standort Wolfenbüttel (links) und Salzgitter (rechts)

Die AC-Ladesäulen können eine maximale Ladeleistung von 22kW übertragen und besitzen jeweils zwei Typ-2-Stecker, sodass zwei Fahrzeuge gleichzeitig laden können. Die Kommunikation zwischen Ladesäule und Backend[1] wird über OCPP (Open Charge Point Protokoll) -1.6 JSON-Websocket von der Firma Open Charge Alliance durchgeführt. Das Backend wird durch die Firma Chargecloud bereitgestellt, welches das Monitoring über die Ladevorgänge und die Kostenabrechnungen ermöglicht. Der Zugriff und die Bezahlung an den Ladesäulen wird über RFID (Radio-Frequency-Identification) realisiert.

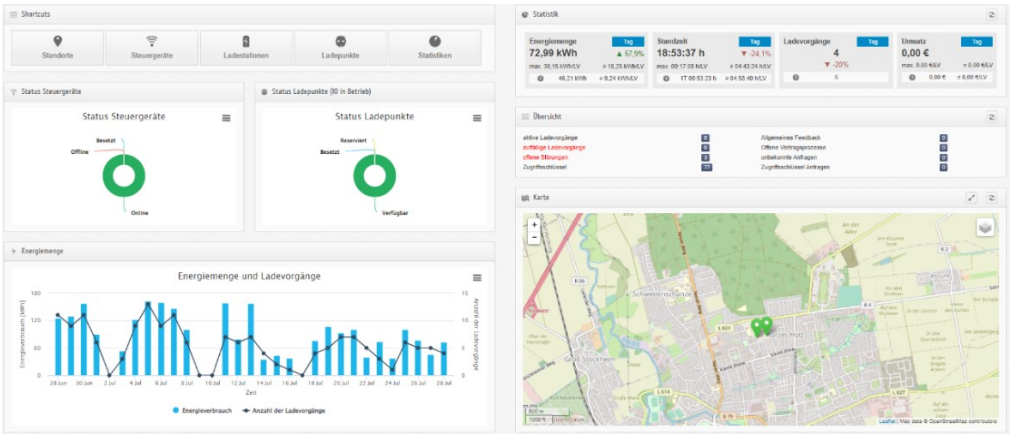


Abbildung 1.2: Dashboard von Chargecloud über das Monitoring der Ladevorgänge

[1] Backend: Softwareanbindung für die Fernsteuerung, Datenverarbeitung und -speicherung

2 Grundlagen

In diesem Kapitel wird ein Überblick über die Ladetechniken von Elektrofahrzeugen gegeben. Weiterhin werden Grundlagen des maschinellen Lernens vermittelt. Dabei liegt der Schwerpunkt auf dem sogenannten unüberwachten Lernen (eng.: Unsupervised-Learning) mit Methoden zur Clusteranalyse. Es wird ein umfangreicher Überblick über unterschiedliche Clusterverfahren gegeben, damit im Anschluss ein geeignetes Verfahren für die Kategorisierung von Nutztypen ausgewählt werden kann.

2.1 Laden an Ladesäulen

Das Laden von Elektrofahrzeugen kann mit Wechsel (AC)- oder Gleichstrom (DC) erfolgen. Die im Auto verbauten Akkumulatoren werden mit Gleichstrom geladen, daher muss der Wechselstrom aus dem Stromnetz gleichgerichtet werden. Wird die Umwandlung über ein Ladegerät im Fahrzeug vorgenommen, spricht man von AC-Laden. Geschieht die Umwandlung durch eine externe Ladevorrichtung, spricht man von DC-Laden. Das DC-Laden hat den Vorteil, dass kein Gleichrichter im Fahrzeug benötigt wird, wodurch Kosten und Gewicht eingespart werden [Sc22]. Dennoch besitzen derzeit alle E-Fahrzeuge ein Gleichrichter, damit die Besitzer flexibel laden können. Diese sind jedoch für eine geringere Stromaufnahme ausgelegt. Zusätzlich wird zwischen Normal- und Schnellladen unterschieden. Beim Normalladen werden maximal 22kW übertragen. Wohingegen alles über 22kW als Schnellladen bezeichnet wird [Sc22].

Die maximale Ladeleistung (Kilowatt (kW)) ergibt sich aus der verfügbaren Spannung und dem daraus resultierenden Strom. Eine normale Haushaltssteckdose (Schuko-Steckdose) kann z.B. 230 V mit 16A(einphasig) liefern, welches einer Leistung von 3.7 kW entspricht. Ein dreiphasiger Wechselstromanschluss kann bis zu 22 kW leisten. Die Ladeleistung und die speicherbare Kapazität von einem verbauten Akkumulator bestimmen im Wesentlichen die Ladezeit. Je höher die Ladeleistung, desto schneller ist der Ladevorgang beendet [Ka21].

Die folgende Tabelle 2.1 zeigt typische Kenngrößen beim Laden, die für unterschiedlichste E-Fahrzeuge zum Einsatz kommen.

Tabelle 2.1: Typische Kenngrößen beim Laden von E-Fahrzeugen in Anlehnung an [BD21]

	Ladetechnologie	**Ladeleistung (kW)**	**Ladestrom (A)**	**Netzanschluss der Ladeinfrastruktur**
Normalladen	AC, 1-phasig	bis 3,7	bis 16	AC, 1-phasig 230 V, 16 A
	AC, 3-phasig	bis 22	bis 32	AC, 3-phasig 400 V, 3x32 A
Schnellladen	DC	bis 350	bis 500	AC, 3-phasig 400 V, 3x125 A

Anzumerken ist, dass der Ladevorgang beim Laden von E-Fahrzeugen nicht durchgängig mit der maximalen Leistung durchgeführt wird. Die nächste Abbildung 2.1 zeigt einen Leistungsverlauf über die Zeit an einer Ladesäule mit einer maximal übertragbaren Ladeleistung von 22 kW:

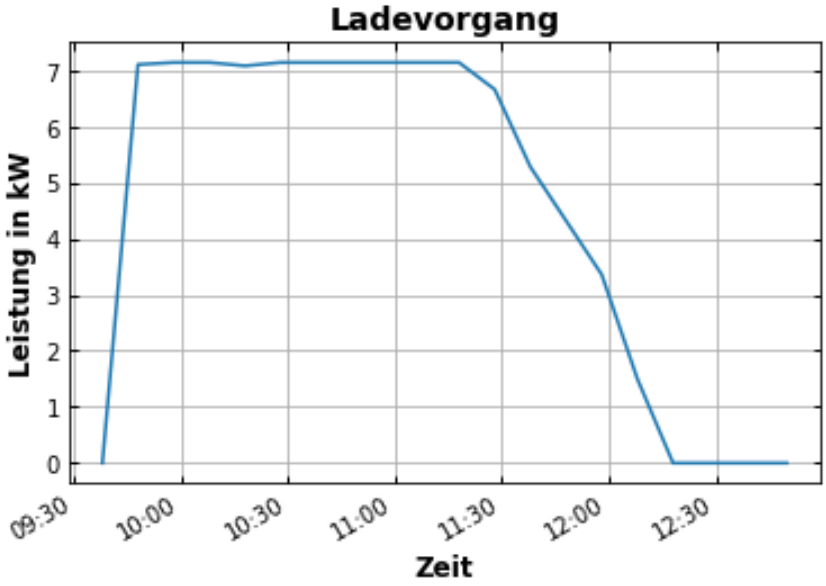


Abbildung 2.1: Leistungskurve eines Ladevorgangs

Zu Beginn steigt die Ladeleistung in kürzester Zeit auf die maximale Ladeleistung von etwa 7,2 kW und wird nahezu konstant beibehalten. Die Ladeleistung von 7.2 kW ist durch den Hersteller vorgegeben. Zum Ende hin fällt die Ladeleistung über einen längeren Zeitraum ab, bis der Akku vollgeladen ist. Lithium-Ionen-Batterien zeichnen sich im Bereich des Ladezustandes von 20 % bis 80 % durch ihre konstante Batteriespannung aus. Ab einem Ladezustand von ca. 80 % steigt die Spannung jedoch exponentiell an und die Gefahr besteht, dass die zulässige Spannung überschritten wird und zum Ausgasen der Batterie führt. Das kann im Extremfall zur Selbstentzündung führen. Aus diesem Grund wird bei einem hohen Ladezustand die Ladeleistung durch das interne Batteriemanagementsystem reduziert, und somit die zulässige Spannung nicht überschritten [Sa19].

Je nachdem welche Ladeleistung für das E-Fahrzeug verwendet wird, unterscheidet man zwischen vier unterschiedlichen Ladebetriebsarten (Moden), die in der Norm IEC 62196 beschrieben werden [Ka21]:

Mode 1:

- Das Laden wird mit Wechselstrom über eine Schuko- oder CEE-Steckdose durchgeführt.
- Der maximale Ladestrom beträgt einphasig 16 A (3,7 kW).
- Eine Fehlerstromschutzeinrichtung für die verwendete Steckdose muss vorhanden sein.
- Die Steckdose muss den Strom dauerhaft zulassen können. Bei einer Haushaltsteckdose sollte in der Praxis eine reduzierte Ladeleistung eingestellt werden (10 A).
- Es findet keine Kommunikation zwischen Steckdose und Fahrzeug statt.

Wird in der Praxis nur als Notlademöglichkeit mit reduzierter Ladeleistung genutzt.

Mode 2:

- Der maximale Ladestrom beträgt einphasig 16 A (3,7 kW) oder dreiphasig 3x32 A (22 kW).
- Die Ladeleistung verfügt über eine Steuer- und Schutzvorrichtung („In Cable Control and Protection Device“ (IC-CPD)).
- Das IC-CPD kann durch ein PWM-Signal mit dem Fahrzeug kommunizieren.

- Die Versorgung erfolg über eine Schuko- oder CEE-Steckdose, oder in Kombination mit einer Wallbox.

Mode 3:

- Das Laden wird durch eine Ladeeinrichtung (Wallbox, Ladesäule) bis maximal 63 A (43,5 kW) durchgeführt.
- Steuer- und Schutzfunktion, sowie Kommunikationsmodul sind fest in der Einrichtung installiert.
- Das Ladekabel ist entweder fest an der Ladeeinrichtung befestigt oder mit einer Steckervorrichtung verwendbar.

Mode 4:

- Beim Mode 4 wird anstelle von AC mit DC geladen. Dabei wird zwischen zwei Ladevarianten unterschieden:
  - das Laden mit maximal 38 kW über einen Typ-2-Stecker
  - das Laden bis maximal 350 kW über einen CCS (Combined Charging System) -Stecker oder CHAdeMo (Charge de Move) -Stecker (üblich bei japanischen Herstellern)
- Das Ladekabel ist fest mit der Ladestation verbunden.

In der Praxis sollen die Moden drei typische Ladeszenarien abdecken [Ka21]:

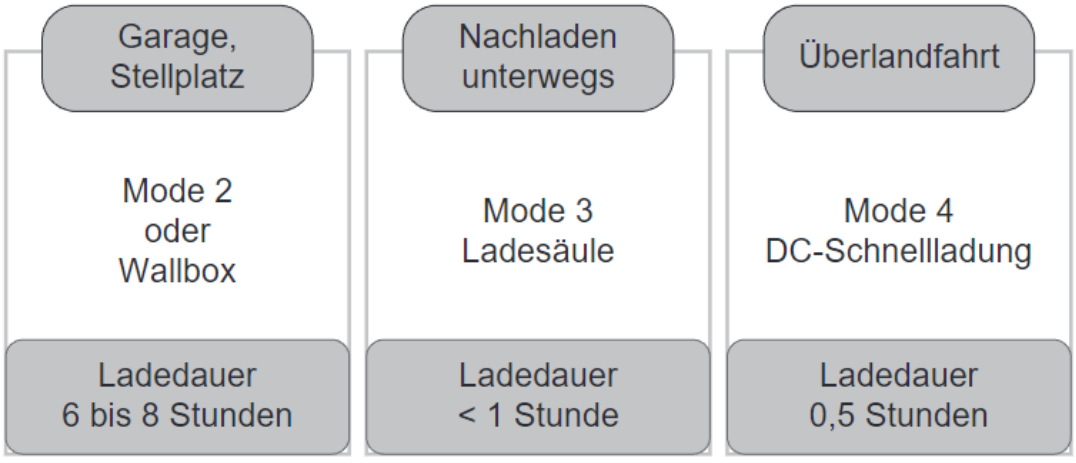


Abbildung 2.2: Typische Ladeszenarien [Ka21]

Mode zwei bezieht sich hierbei auf das Laden am Wohnort oder an der Arbeitsstelle. Mode drei ist für kürzere Aufenthalte z.B. beim Einkauf vorgesehen und Mode drei speziell zum Laden an Autobahnen mit kurzer Aufenthaltsdauer.

2.2 Maschinelles Lernen

Beim traditionellen Programmieren wird eine Kombination aus Regeln und Wissensträgern (Daten), manuell für die vorhandene Aufgabe erstellt und zum Lösen eines bestehenden Problem verwendet. Dieses Vorgehen kann bei komplexen Zusammenhängen einen hohen Aufwand bedeuten und stellt bei speziellen Problemen einen begrenzten Nutzen für verwandte Aufgaben dar. Das maschinelle Lernen setzt genau bei diesem Problem an und versucht selbständig in historischen Daten, also anhand von konkreten Beispielen, interne Muster zu erkennen. Die Muster werden dann für das Ableiten von allgemeingültigen Regeln genutzt. Die Regeln können für die Zuordnung oder Vorhersage von unbekannten Daten der gleichen Problemstellung verwendet werden. Die dafür eingesetzten Algorithmen können unabhängig vom Anwendungsfall genutzt werden und erfordern kein spezifisches Fachwissen über die Daten [Ma21].

Besonders hochdimensionale Daten, die viele Merkmale (engl.: Feature), z.B. Temperatur, Druck, Leistung etc. besitzen, sind für Menschen schwer zu interpretieren. Komplexe Zusammenhänge können nur mit viel Aufwand abgeleitet werden. Das maschinelle Lernen hingegen ist außergewöhnlich leistungsfähig im Umgang mit hohen Dimensionen und dabei im Wesentlichen nur durch die zur Verfügung gestellte Rechenleistung begrenzt [Ma21].

Lernverfahren

Das maschinelle Lernen kann laut [Ri19] in drei unterschiedliche Lernverfahren unterteilt werden:

- überwachtes Lernen
- unüberwachtes Lernen
- bestärktes Lernen

Im Rahmen dieser Arbeit wird das Lernverfahren des unüberwachten Lernens verwendet. Aus diesem Grund wird neben dem unüberwachten Lernen das überwachte Lernen für ein

besseres Verständnis und zur konkreten Abgrenzung kurz erläutert. Auf das bestärkte Lernen wird nicht weiter eingegangen.

Überwachtes Lernen

Ein überwachtes Lernverfahren liegt dann vor, wenn beim Lernprozess der Daten die Ausgabewerte bekannt sind und mit korrekten Beschriftungen versehen sind. Die beschrifteten Daten werden auch als sogenannte Labels bezeichnet. Bei den Ausgabewerten wird zusätzlich zwischen diskreten- und kontinuierlichen numerischen Merkmalen unterschieden. Nach [Ri19] gelten somit zwei verschieden Anwendungsfälle:

- **Klassifikation** bei diskreten Ausgabewerten
- **Regression** bei kontinuierlichen Ausgabewerten

Diese beziehen sich nur auf die Ausgabewerte. Die Art der Eingabewerte ist für beide Anwendungsfälle unbedeutend. Sie können kontinuierlich, diskret oder auch gemischt sein [Ma21]. In der folgenden Abbildung 2.3 wir das Konzept von Klassifikation und Regression anhand eines Beispiels dargestellt:

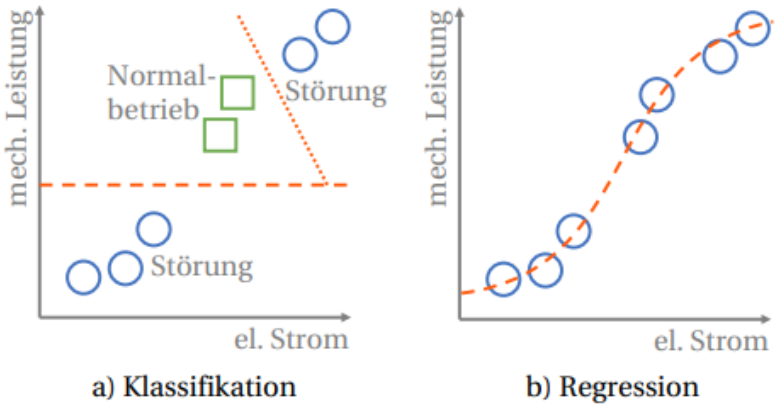


Abbildung 2.3: Anwendungsfall für a) Klassifikation und b) Regression [Ma21]

Das Beispiel zeigt den Zusammenhang zwischen mechanischer Leistung und elektrischem Strom einer Maschine. Im Fall der Klassifikation wird anhand dieser zwei Größen der Ausgabewert von Normalbetrieb oder Störung klassifiziert, welche als Klassen bezeichnet werden. Bei der Regression hingegen wird, in Abhängigkeit vom Eingabewert (mech. Leistung oder elektr. Strom), der jeweils nicht eingegebene Wert vorhergesagt. Beide Verfahren müssen im Vorfeld auf die konkreten Ausgabewerte trainiert werden, damit diese auf nicht zuvor gesehene Daten angewandt werden können.

Mathematisch gesehen wird also beim überwachten Lernen eine Funktion f^* gesucht, die den Zusammenhang zwischen $X \rightarrow Y$ am besten beschreiben kann. Bei der Klassifikation entspricht $Y \in \mathbb{N}$ und bei der Regression $Y \in \mathbb{R}$. Wichtig ist, dass für X immer eine Anzahl $n \in \mathbb{N}$ von Beobachtungen (Y) vorliegen muss [Ri19].

Unüberwachtes lernen

Im Fall des unüberwachten Lernens liegen nur unbeschriftete Daten vor oder die verfügbaren Beschriftungen werden explizit nicht im Lernprozess verwendet. Eingabewerte sind bekannt, aber keine Ausgabewerte. Wie beim überwachten Lernen werden Daten typischerweise von Menschen ausgewählt und bereitgestellt. Da jedoch keine Labels bereitgestellt werden müssen, ist der Aufwand geringer. Oft stehen große Datenmengen für das Training zur Verfügung. Das Ziel ist es, Strukturen bzw. Muster in den Daten zu entdecken. Die Suche erstreckt sich über die gesamten Eingabedaten und ist nicht, wie beim überwachten Lernen, auf Zusammenhänge mit bestimmten Zielvariablen beschränkt. Die größte Herausforderung beim unüberwachten Lernen besteht darin, dass es keine explizit richtigen oder falschen Ergebnisse gibt. Die Algorithmen besitzen keine konkreten Ziele, weshalb diese auch kein Feedback über die Qualität, der im Lernprozess entdeckten Muster, geben können. Aus diesem Grund ist eine umfangreiche Interpretation der Ergebnisse ein bedeutsamer Bestandteil des unüberwachten Lernens [SL19].

Das unüberwachte Lernen lässt sich nach [Pl21] in zwei wesentliche Typen unterteilen:

- **Clusteranalyse:** Verfahren der Clusteranalyse haben zum Ziel, den Datensatz in Gruppen (Cluster) einzuteilen, die sich durch ihre Charakteristiken unterscheiden.
- **Dimensionsreduktion**: Das Ziel einer Dimensionsreduktion ist, die Dimension von den Eingabedaten zu verringern, ohne bestimmte Charakteristiken, wie Lage und Abstände zwischen den Daten, bedeutsam zu verändern.

2.3 Clusteranalyse

In diesem Unterkapitel soll ein Überblick über die Clusteranalyse gegeben werden. Der Schwerpunkt liegt auf der Übersicht von unterschiedlichen Clusterverfahren. So kann ein geeignetes Verfahren für die Kategorisierung von Nutzertypen ausgewählt werden.

2.3.1 Distanz– und Ähnlichkeitsmaße

In der Praxis liegen Daten meistens als sogenannte Rohdaten vor, die aus Objekten (Datenpunkten) bestehen und mit Hilfe von Attributen (Merkmale) beschrieben werden. Diese Merkmale können nominal, binär, ordinal oder metrisch sein. Nominale Merkmale bestehen aus Kategorien ohne erklärbare Reihenfolge (z.B. Automarken). Binäre Merkmale bestehen aus Objekten mit zwei möglichen Zuständen (z.B. 0 und 1). Ordinale Merkmale bestehen aus Objekten mit einer sinnvollen Reihenfolge. Die Größenordnung zwischen den Objekten ist jedoch nicht bekannt (z.B. schlecht, gut, besser). Die zuvor aufgezählten Typen werden als quantitative Merkmale bezeichnet. Metrische Merkmale können ganzzahlig oder reell sein. Die Abstände der Objekte sind interpretierbar. Diese werden zusammen als qualitative Merkmale bezeichnet [HKP11]. In Abhängigkeit der Merkmalstypen müssen unterschiedliche Verfahren beim Clustern berücksichtigt werden. Alle Clusterverfahren basieren auf der Grundannahme, dass zwischen Datenpunkten eine Distanz bzw. Ähnlichkeit besteht. Diese werden auch als Proximitätsmaße bezeichnet und laut [Ba21] wie folgt definiert:

- **„Ähnlichkeitsmaße** spiegeln die **Ähnlichkeit** zwischen zwei Objekten wider: Je größer der Wert eines Ähnlichkeitsmaßes wird, desto ähnlicher sind sich zwei Objekte.“
- „**Distanzmaße** messen die **Unähnlichkeit** zwischen zwei Objekten: Je größer die Distanz wird, desto unähnlicher sind sich zwei Objekte. Sind zwei Objekte als vollkommen identisch anzusehen, so ergibt sich eine Distanz von Null“.

Wobei sich die Maße gegenseitig ergänzen, d.h. Ähnlichkeit = 1-Unähnilichkeit. In der Praxis wird bevorzugt die Unähnlichkeit für die Clusteranalyse gemessen. Bei der Verwendung von Distanzmaßen muss unbedingt darauf geachtet werden, dass die Merkmale miteinander vergleichbar sind, d.h. sie sollten die gleichen Einheiten besitzen [Ba21]. Angenommen der Datensatz besteht aus zwei Merkmalen, z.B. Verbrauch mit 1 kWh bis 2-kWh und Gewicht mit 1000 kg bis 2000 kg. Eine Änderung im Verbrauch wäre betragsmäßig kleiner gegenüber dem Gewicht. Das Gewicht würde bei der Verwendung eines Distanzmaßes eine höhere Gewichtung bekommen. Damit dies verhindert wird, müssen die Merkmale vor dem Clustern mit Hilfe einer Standardisierung einheitenlos gemacht bzw. auf eine einheitliche Skala gebracht werden. Bei einer Standardisierung wird

die Differenz zwischen dem Beobachtungswerten x_{ij} und dem Mittelwert $\bar{x}_i$ des Merkmales berechnet. Zusätzlich wird die Differenz durch die Standardabweichung s_j dividiert [Ba21]:

$$z_{ij} = \frac{x_{ij} - \bar{x}_i}{s_j} \tag{2.1}$$

Nach der Skalierung besitzt jedes Merkmal einen Mittelwert von Null mit einer Standardabweichung von Eins. Die Verteilung der Daten bleibt dabei erhalten. Nachdem die Standardisierung vorgenommen wurde, kann nun ein Proximitätsmaß berechnet werden. Die folgende Tabelle 2.2 soll einen Überblick für verschiedene Proximitätsmaße geben:

Tabelle 2.2: Übersicht von Proximitätsmaßen in Anlehnung an [Ba21]

	Metrische Merkmale	**Binäre Merkmale**	**Nominale- oder ordinale Merkmale**
Ähnlichkeitsmaße	• Kosinus • Pearson-Korrelation	• Einfache Übereinstimmung • Phi-4-Punkt Korrelation • Lambda (Goodman & Kruskal) • Würfel (Dice) Jaccard • Rogers & Animato • Russel & Rao	
Distanzmaße	• Euklidische Distanz • Quadrierte euklidische Distanz • Tscheyscheff • (City-) Block-Metrik • Minkowski-Metrik	• Euklidische Distanz • Quadrierte euklidische Distanz • Größendifferenz • Musterdifferenz • Lance & Williams	• Chi-Quadrat-Maß • Phi-Quadrat-Maß

Im Rahmen dieser Arbeit werden ausschließlich Clusterverfahren verwendet, die metrische Merkmale verarbeiten können. Für die Verfahren sind zwei wesentliche Distanzmaße vorgesehen. Die euklidische Distanz und die City-Block-Metrik (oder Manhattan-Distanz).

Der Ausgangspunkt für die zwei Distanzmaße ist die Minkowski-Metrik [Ba21]:

$$d_{k,l} = \left[\sum_{j=1}^{J} \left|x_{kj} - x_{lj}\right|^{r}\right]^{\frac{1}{r}} \tag{2.2}$$

Diese ermöglicht die Berechnung der Distanz zwischen zwei Punkten k und l über alle Dimensionen. Die Variable r ist die sogenannte Minkowski-Kontante. Für r = 1 erhält man die City-Block-Metrik, die auch als Manhattan-Distanz oder auch L1-Norm bezeichnet wird [Ba21]:

$$d_{k,l} = \sum_{j=1}^{J} \left|x_{kj} - x_{lj}\right|^{1} \tag{2.3}$$

Hierbei wird nicht die kürzeste Distanz zwischen zwei Punkten berechnet, sondern die Distanz durch vertikale und horizontale Schritte. Daher auch Manhattan-Distanz, weil die Distanz der Fortbewegung in einer Großstadt ähnelt.

Setzt man für r = 2 ein, erhält man die euklidische Distanz oder auch L2-Norm. Diese misst die Luftlinie, also die kürzeste Distanz zwischen zwei Punkten und ist eine der am weit verbreitetsten Distanzmaße für Clusterverfahren. Die Berechnung wird mit folgender Formel nach [Ba21] durchgeführt:

$$d_{k,l} = \sqrt{\sum_{j=1}^{J} \left|x_{kj} - x_{lj}\right|^{2}} = \sqrt{\left(x_{k1} - x_{j1}\right)^{2} + (x_{k2} - x_{l2})^{2} + \cdots + (x_{kn} - x_{ln})^{2}} \tag{2.4}$$

Im Grunde ist die euklidische Distanz eine Erweiterung vom Satz des Pythagoras für mehrdimensionale Daten. Anhand eines Beispiels soll die Berechnung verdeutlicht werden. Gegeben sind zwei Punkte in einem dreidimensionalen Raum mit den Koordinaten k(3,4,5) und l(5,3,2). Die Distanz der Punkte berechnet sich wie folgt:

$$d_{k,l} = \sqrt{(3-5)^2 + (4-3)^2 + (5-2)^2} = 3{,}74$$

Durch die Quadrierung werden größere Distanzen höher gewichtet und kleinere Distanzen geringer gewichtet. Wegen der Quadrierung sollte deshalb überprüft werden, ob eine Standardisierung der Merkmale erforderlich ist. Der wesentliche Vorteil von der Manhattan-

Distanz ist der Wegfall der Quadrierung, d.h. weniger Komplexität und damit weniger benötigte Rechenleistung.

Die Abbildung 2.4 soll den Unterschied zwischen euklidischer Distanz und Manhattan-Distanz durch ein graphisches Beispiel verdeutlichen:

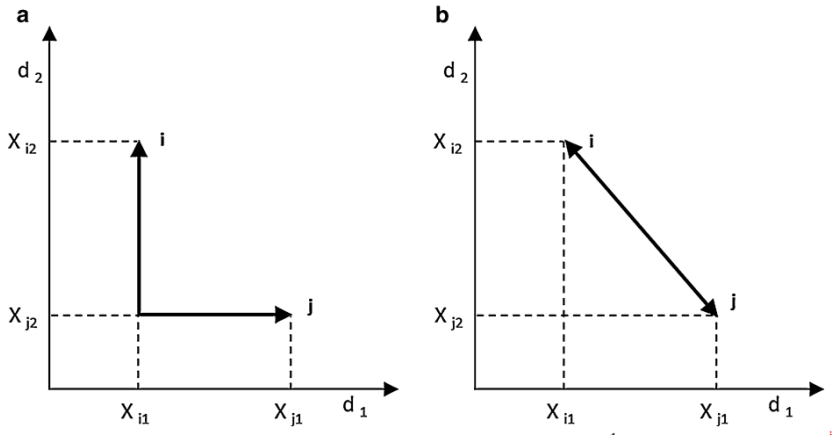


Abbildung 2.4: Unterschied zwischen Manhattan-Distanz(links) und euklidischer Distanz (rechts) anhand eines zweidimensionalen Beispiels [Sa19]

2.3.2 Übersicht von Clusterverfahren

In den vergangen Jahrzehnten hat das Gebiet der Clusteranalyse eine Vielzahl von unterschiedlichen Verfahren hervorgebracht. Diese haben teilweise sehr ähnliche Ansätze, was die Entscheidung für einen geeignetes Verfahren nicht leicht macht. Dennoch versuchen viele Literaturen eine Einordung der Verfahren zu finden, um die Entscheidung zu erleichtern. In [HKP11] teilen die Autoren die Verfahren in fünf übergeordnete Typen auf: „partitionierende-, hierarchische-, dichtbasierte-, gitterbasierte-, und modelbasierte Clusterverfahren“. Außerdem wird das graphenbasierte Clusterverfahren vorgestellt. Die Typen werden nachfolgend in konzentrierter Form erläutert und bedeutsame Vor- und Nachteile aufgezählt.

Partitionierendes Clusterverfahren

Das partitionierende Clustern hat zum Ziel, die Daten in disjunkte Gruppen (Cluster) einzuteilen (partitionieren). Das bedeutet, dass jeder Cluster aus mindestens einem Datenpunkt besteht und jeder Datenpunkt nur in einem Cluster vorkommt [CL20]. Die Anzahl der Cluster wird bei diesem Verfahren im Vorfeld festgelegt. Die festgelegte Anzahl von Clustern wird zu Beginn durch Schwerpunkte (Zentroiden) innerhalb der Daten

repräsentiert. Danach werden die einzelnen Datenpunkte den jeweiligen Zentroiden, in Abhängigkeit der Distanz, zugeordnet. Anschließend wird die anfänglich erzeugte Partitionierung durch einen iterativen Prozess verbessert. Dazu kommen unterschiedlichste Optimierungsverfahren zum Einsatz, welche zum Beispiel die Zentroiden verschieben oder die Datenpunkte zwischen den Clustern austauschen [HKP11]. Laut [HKP11] ist „das allgemeine Kriterium für eine gute Partitionierung, dass Datenpunkte im selben Cluster "nah" oder miteinander verwandt sind, während Datenpunkte verschiedenen Cluster "weit voneinander entfernt" oder sehr unterschiedlich sind".

Ein Vorteil der partitionierenden Clusterverfahren ist, dass diese in der Regel ein Hyperparameter[2] besitzen (Anzahl der Cluster) und daher sehr unkompliziert zu implementieren sind. In der Praxis werden häufig mehrere Versuche mit unterschiedlicher Anzahl von Clustern durchgeführt und das beste Ergebnis ausgewählt. Ein Nachteil des Verfahrens besteht darin, dass sphärische Cluster gebildet werden. Zudem können stark abweichende Datenpunkte die Lage der Zentroiden negativ beeinflussen und die Clusterbildung verzerren. In vielen Fällen kann dieses Problem durch eine Erhöhung der zu bildenden Cluster verhindert werden, sodass die Ausreißer in eigene Cluster zusammengefasst werden [Ma21]. Jedoch können dadurch viele einzelne Cluster entstehen, was ein Mehraufwand in der Interpretation bedeutet.

Zu den bekanntesten partitionierenden Clusterverfahren gehören unter anderem K-Means, PAM (K-Mediods), CLARA und CLARANS [PS16]. Laut [PS16] haben diese Algorithmen folgende Eigenschaften (siehe Tabelle 2.3):

Tabelle 2.3: Vergleich zwischen K-Means, K-Mediods, CLARA und CLARANS in Anlehnung an [PS16]

Eigenschaft	**K-Means**	**K-Mediods**	**CLARA**	**CLARANS**
Zeitkomplexität	O(i*k*n)	O(i*k(n-k)2)	O(ks2+k(n-k))	$O(n^2)$
Empfindlich gegenüber Ausreißer?	Ja	Nein	Nein	Nein
Vorgabe der Cluster erforderlich?	Ja	Ja	Ja	Ja
Effizienz	universell einsetzbar	kleine Datensätze	große Datensätze	große Datensätze

[2] Hyperparameter: Parameter die vor der Durchführung des Verfahrens festgelegt werden müssen.

Die Zeitkomplexität steht für die Berechnungsdauer, wobei i für die Anzahl der Iteration, k für die Anzahlt der Cluster und n für die Anzahl der Datenpunkte steht. Die Effizienz gibt an, für welche Datenmenge diese Verfahren optimiert sind. Diese Algorithmen sind für metrische Merkmale ausgelegt. Für kategorische Merkmale kann z.B. der K-Mode [Hu98] genutzt werden.

Hierarchische Clusterverfahren

Beim hierarchischen Clustern wird die Clusterhierarchie so eingestellt, dass immer die Cluster mit dem geringsten Abstand bzw. der größten Ähnlichkeit zusammengeführt werden. Beispielsweise kann ein übergeordneter Cluster von bereits zwei bestehenden Clustern erstellt werden [CL20]. Außerdem wird bei diesem Verfahren zwischen einer agglomerativen und divisiven Vorgehensweise unterschieden. Bei der agglomerativen Vorgehensweise wird der sogenannte „bottem-up“-Ansatz verwendet. Das bedeutet, dass zu Beginn jeder Datenpunkt ein Cluster repräsentiert. Die Cluster werden schrittweise zu neuen Clustern zusammengefasst, bis am Ende ein Cluster vorhanden ist. Das divisive Verfahren hingegen verfolgt den sogenannten „top down“-Ansatz. Dabei wird der gesamte Datensatz durch ein Cluster dargestellt. Dieser Cluster wird dann schrittweise in kleinere unterteilt [CL20].

Das Zusammenführen bzw. Aufteilen der Cluster wird in Abhängigkeit der Ähnlichkeiten bzw. der Distanz zueinander realisiert. Dafür können mehrere Vorgehensweisen in Betracht gezogen werden [HKP11]:

- minimale Distanz zweier Punkte zwischen den Clustern (Single-linkage)
- maximale Distanz zweier Punkte zwischen den Clustern (Complete-linkage)
- Distanz zwischen zwei Gruppenschwerpunkten (Centroid-linkage)
- mittlere Distanz aller Punkte zwischen den Clustern (Average-linkage)

Die erzielten Ergebnisse durch das hierarchische Clusterverfahren werden in der Regel mit einem sogenannten Dendrogramm dargestellt. Das Dendrogramm dient zur Interpretation und ist ein wesentlicher Bestandteil des Verfahrens. Die folgende Abbildung 2.5 zeigt ein beispielhaftes Dendrogramm, wobei zwölf Datenpunkte zusammengefasst sind.

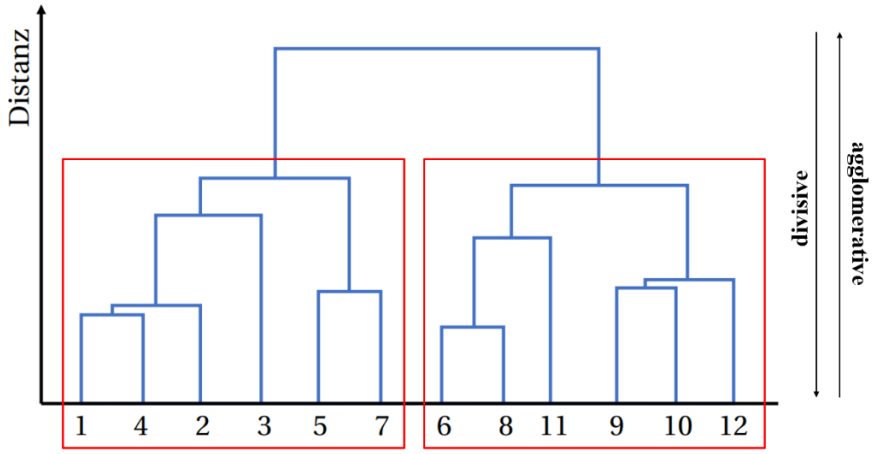


Abbildung 2.5: Beispiel eines Dendrogramm in Anlehnung an [Ma21]

Anhand des Dendrogramms können mehrere Rückschlüsse gezogen werden. Zum einen zeigen die vertikalen Linien den Abstand zwischen den Clustern, bis diese schlussendlich zusammengefasst werden. Außerdem ist eine deutliche Aufteilung der Punkte in zwei Hauptcluster a) und b) zu erkennen, welche wiederum mit einer deutlichen Distanz zueinander zusammengefasst werden. Mit dieser Erkenntnis kann die Annahme getroffen werden, dass sich innerhalb des Beispieldatensatzes zwei signifikante Cluster mit ausreichendem Abstand zueinander befinden. Zusätzlich fällt auf, dass zwischen Datenpunkt drei und dem Cluster (1 + 4 + 2) eine große Distanz bis zur Zusammenführung besteht. Der Datenpunkt drei könnte als Ausreißer interpretiert werden. [Ma21]

Der wesentliche Vorteil gegenüber anderen Verfahren ist, dass keine Hyperparameter im Vorfeld initialisiert werden müssen. Das macht die Implementierung unkompliziert. Nachteilig hingegen ist, dass häufig keine konkrete Zielfunktion minimiert wird. Dadurch werden alle Berechnungsschritte nur einmal durchgeführt. Fehlerhafte Zusammenführungen oder Aufteilungen können nicht rückgängig gemacht werden [KM14]. Die Interpretation eines Dendrogramms kann zudem, je nach Anwendungsfall, besonders herausfordernd sein. Das Beispiel in Abbildung 2.5 besitzt lediglich zwölf Datenpunkte. Bei einem Datensatz mit sehr vielen Datenpunkten nimmt die Komplexität enorm zu und das Erkennen der Cluster kann undeutlich werden. Außerdem kann das Erkennen von fehlerhaften Zuweisungen innerhalb des Dendrogramms bei vielen Datenpunkten nicht eindeutig interpretiert werden. Wie auch beim partitionierenden Verfahren werden vorzugsweise sphärische Cluster gebildet [HKP11].

Ein Übersicht von unterschiedlichen Algorithmen für die hierarchischen Clusterverfahren kann [DP16] entnommen werden.

Dichtebasierte Clusterverfahren

Dichtebasierte Clusterverfahren erzeugen Cluster anhand von Regionen mit hoher Datendichte und werden durch Regionen mit geringer Datendichte getrennt [HKP11]. Die Grundidee besteht darin, dass jeder Datenpunkt eine bestimmt Anzahl von anderen Datenpunkten in seiner Umgebung besitzen muss, damit ein Cluster gebildet wird[CL20].

Gegenüber partitionierenden und hierarchischen Verfahren, die sphärische Cluster bilden, besitzen dichtebasierte Verfahren die Fähigkeit, jegliche Formen innerhalb der Daten zu finden. Zudem können Datenpunkte in Regionen mit geringer Dichte als Ausreißer gekennzeichnet werden [HKP11]. Zusätzlich muss die Anzahl der Cluster nicht vorgegeben werden, da diese eigenständig in den Daten gesucht werden. Jedoch besitzen die Verfahren mehrere Hyperparameter, die im Vorfeld anhand der Datenverteilung bestimmt werden müssen und einen großen Einfluss auf die Clusterbildung haben [Ma21]. Des Weiteren können hohe Dimensionen die Clusterbildung verschlechtern [KM14]. Ein möglicher Grund dafür ist, dass die räumliche Trennung der Daten mit zunehmender Dimension größer wird und folglich die Dichte abnimmt [Ma21]. Die globalen Hyperparameter führen außerdem dazu, dass Daten mit variierender Dichte und potenziell unterschiedlichen Clustergrößen nicht ausreichend charakterisiert werden [HKP11].

Die mit Abstand meisten Algorithmen besitzen dichtebasierte Clusterverfahren, weshalb auf die Literatur [BM21] für einen umfangreichen Überblick verwiesen wird. Dennoch kann gesagt werden, dass die meisten Algorithmen auf Grundlage des sogenannten DBSCAN-Algorithmus [Es96] entstanden sind und für unterschiedlichste Anforderungen optimiert wurden.

Gitterbasierte Clusterverfahren

Gitterbasierte Clusterverfahren partitionieren den Eingaberaum in eine endliche Anzahl von Zellen, unabhängig von der Verteilung der Eingabedaten. Die einzelnen Zellen aus der erzeugten Gitterstruktur werden für die Clusteroperation verwendet [HKP11]. Anders ausgedrückt, es wird die Auflösung des Eingaberaums verringert, damit die Komplexität abnimmt. Der Vorteil der Gitterstruktur ist eine schnelle Berechnungszeit durch die lokalen Betrachtungen des Eingaberaums [HKP11].

Das effiziente Clustern von großen hochdimensionalen Daten zeichnet dieses Verfahren aus ([DP17], zitiert nach [CH13]). Eine Übersicht von verschiedenen Algorithmen kann [DP17] entnommen werden.

Modelbasierte Clusterverfahren

Diese Clusterverfahren basieren auf der Annahme, dass die Daten durch mathematische Modelle beschrieben werden können. Hierbei werden die Parameter der Modelle iterativ angepasst, bis diese die Daten bestmöglich beschreiben [PS16]. Zu den Verfahren gehören unter anderem die probabilistischen Modelle. Diese beschreiben die Wahrscheinlichkeitsverteilung in Daten. In der Praxis liegen die Daten häufig als sogenannte Mischverteilung vor, d.h. die Summe von unterschiedlichen Verteilungen. Ziel ist es, anhand der Mischverteilung die untergeordneten Verteilungen zu finden. Diese Verteilungen repräsentieren die Cluster [HKP11].

Probabilistische Clusterverfahren erzeugen sogenannte weiche Cluster, d.h. ein Datenpunkt kann mehreren Clustern unter Berücksichtigung einer Wahrscheinlichkeit zugeordnet werden. Die zuvor vorgestellten Verfahren erzeugen harte Cluster, wobei ein Datenpunkt genau einem Cluster zugeordnet wird [HKP11].

Ein typischer Vertreter dieses Verfahrens ist das Gaußsche-Mischverteilungsmodell (engl. Gaussian-Mixtur-Model (GMM)). Bei diesem Verfahren besteht die Annahme, dass die Daten aus eine Mischung von unterschiedlichen Gauß-Verteilungen mit unbekannten Parametern besteht [Gé20]. Der Vorteil bei diesem Verfahren besteht darin, dass die Cluster anhand einer Ellipse geformt werden, die sich durch Größe, Dichte und Ausrichtung unterscheiden können und sind somit den partitionierenden und hierarchischen Clusterverfahren überlegen. Diese bilden sphärische Cluster mit gleicher Größe [Gé20]. Bei überlappenden Cluster besteht die Möglichkeit, eine Aussage darüber zutreffen, mit welcher Wahrscheinlichkeit ein Datenpunkt zu den jeweiligen Clustern gehört (weiche Cluster).

Ein weiteres modelbasiertes Clusterverfahren ist die „Selbstorganisierende-Karte“ (engl. Self-Organizing-Map (SOM) [Ko90]. Diese Methode basiert auf einer speziellen Architektur eines künstlichen neuronalen Netzes. Die Grundidee der SOM ist, die Eingabedaten in einem niederdimensionalen Ausgaberaum darzustellen. Der Ausgaberaum wir als Merkmalskarte bezeichnet und beinhaltet die Haupteigenschaften der Eingabedaten [Mi17].

Graphenbasierte Clusterverfahren

Diese Clusterverfahren basieren auf der Annahme, dass die Zusammenhänge der Daten durch einen Graphen repräsentiert werden können. Die Knoten des Graphens repräsentieren die Datenpunkte und anhand der Kanten zwischen den Knoten wird die Beziehung zueinander beschrieben [XT15].

Ein typischer Vertreter dieses Verfahren ist das spektrale Clustern. Hierbei wird die Laplace-Matrix, welche die Zusammenhänge zwischen Knoten und Kanten beschreibt, für die Berechnung der Eigenwerte und Eigenvektoren verwendet. Die Eigenvektoren mit den niedrigsten Eigenwerten werden zur Bildung der Cluster genutzt. Bildlich gesprochen, der Graph wird durch die ausgewählten Eigenvektoren mit den dazugehörigen Datenpunkten in mehrere kleinere Graphen unterteilt. Dadurch, dass nicht alle Eigenvektoren aus der Matrix verwendet werden, führt das Verfahren eine Dimensionsreduktion durch. Die eigentliche Clusterbildung der Eigenvektoren kann im Nachhinein z.B. mit K-Means durchgeführt werden [HTF17]. Das Verfahren kann unterschiedliche Formen clustern und eignet sich besonders für hochdimensionale Datensätze. Außerdem ist es unempfindlich gegenüber Ausreißer. Nachteilig ist, dass das Verfahren eine hohe Zeitkomplexität besitzt und die Anzahl der Cluster festgelegt werden muss [XT15].

In den nächsten Unterkapiteln werden die ausgewählten Clusterverfahren näher erläutert, die im Rahmen der Arbeit untersucht werden. Der Grund für die Auswahl kann aus Abschnitt 5.2 entnommen werden.

2.3.3 K-Means

Der K-Means-Algorithmus oder auch Lloyd-Algorithmus [Ll82] ist ein partitionierendes Clusterverfahren [GCB05], [Oy19]. Der Algorithmus teilt die Daten anhand einer vordefinierten Anzahl von Clustern auf und jeder Cluster wird durch ein Zentroid repräsentiert. Die Datenpunkte, welchen die geringste Distanz zu einem der jeweiligen Zentroiden besitzen, werden dem jeweiligen Cluster zugeordnet [Ma21]. Der Algorithmus durchläuft dabei insgesamt fünf Schritte, die im Folgenden in Anlehnung an [Ma21] erläutert werden:

Schritt 1:

Der K-Means-Algorithmus besitzt einen Hyperparameter k, der die Anzahl der Zentroiden angibt, die im Vorfeld initialisiert werden müssen. Die Größe von k gibt also genau an, in wie viele Cluster der Datensatz aufgeteilt werden soll.

Schritt 2:

In Abhängigkeit von k werden zufällige Datenpunkte ausgewählt, die die Zentroiden repräsentieren. Die Abbildung 2.6 verdeutlicht die Initialisierung an einem zweidimensionalen Beispiel. In a) sind die zufälligen ausgewählten Zentroiden durch Kreis, Quadrat und Dreieck dargestellt:

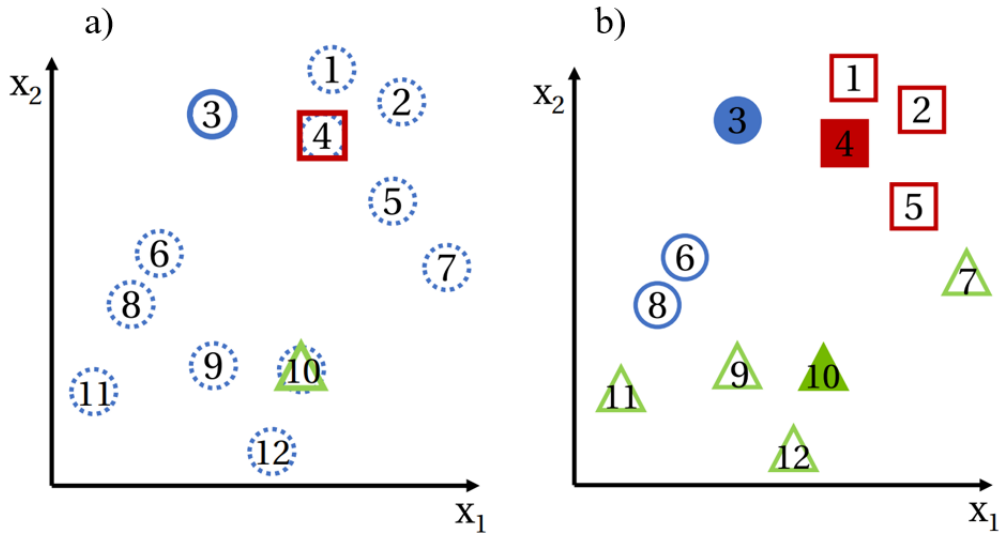


Abbildung 2.6: Initialisierung von K-Means am Beispiel eines zweidimensionalen Beispiels in Anlehnung an [Ma21]

Schritt 3:

In diesem Schritt werden den jeweiligen Zentroiden alle naheliegenden Datenpunkte zugeordnet. Die Zuordnung wir anhand der euklidischen Distanz berechnet. In Abbildung 2.7 b) wird die Zuweisung der Daten zu den jeweiligen Zentroiden gezeigt. Es ist deutlich zu erkennen, dass die zufällige Initialisierung der Zentroiden auf die Datenpunkte 3, 4 und 10 nicht optimal ist. Außerdem liegen die Zentroiden nicht in der Mitte der Cluster.

Schritt 4:

Der Zentroid der jeweiligen Cluster wird nun durch eine Mittelwertbildung der sich darin befindenden Datenpunkte neu berechnet (siehe Abbildung 2.7 c))

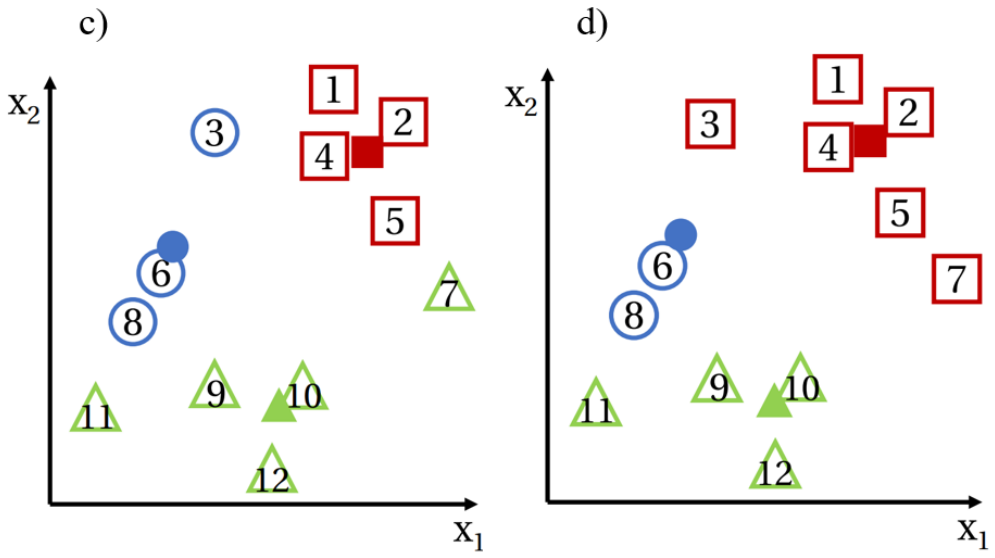


Abbildung 2.7: c) Aktualisierung der Zentroiden und d) neue Zuordnung der Datenpunkte (K-Means) in Anlehnung an [Ma21]

Schritt 5:

In diesem Schritt werden Schritt drei und vier wiederholt. Das ist notwendig, da sich Datenpunkt 3 und 7 näher am roten Zentroid befinden (siehe Abbildung 2.7 d)). Dadurch müssen alle Zentroiden aktualisiert werden. Dieser Vorgang wird so lange durchgeführt, bis keine neuen Zuordnungen der Datenpunkte erfolgen. Dabei versucht der Algorithmus die Summer der quadratischen Abweichungen zwischen den Datenpunkt x_i und den Zentroiden μ_k in den jeweiligen Clustern von allen Clustern mit folgender Zielfunktion zu minimieren [Ja10]:

$$J(k) = \sum_{k=1}^{k} \sum_{x_i \in c_i} \|x_i - \mu_k\|^2 \tag{2.5}$$

Der K-Means Algorithmus wurde von [AV16] weiterentwickelt. Beim standardmäßigen K-Means werden zufällige Datenpunkte ausgewählt, die im schlechtesten Fall nahe aneinander liegen (vgl. Abbildung 2.6 a) Datenpunkt 3 und 4). Der Algorithmus muss viele Aktualisierungen der Zentroiden für eine optimale Aufteilung der Daten durchlaufen. Das erfordert Zeit und kann zu suboptimalen Clustern führen [AV16]. Der sogenannte K-Means++ Algorithmus setzt an dieser Problematik der zufälligen Initialisierung an. Durch das Berechnen der Abstände zwischen dem ersten initialisierten Zentroid und allen vorhanden Datenpunkten, wird, in Abhängigkeit des am weitest entferntesten Datenpunkt, an diesem ein weiterer Zentroid initialisiert. Dieser Vorgang wird so lange wiederholt, bis

alle vorgegebenen Zentroiden (k) initialisiert sind und ermöglicht eine weite Streuung der Zentroiden. Der K-Means++ Algorithmus wird nach [AV16] wie folgt definiert:

1. Wähle ein zufälligen Zentroid μ_1 aus der Menge von X.
2. Wähle den nächste Zentroid μ_i , wobei $\mu_i = x' \in X$ mit der Wahrscheinlichkeit

$$\frac{D(x')^2}{\sum_{x \in X} D(x)} \tag{2.6}$$

   ausgewählt wird. Wobei $D(x')^2$ der am weit entfernteste Datenpunkten zum Zentroid und $\sum_{x \in X} D(x)$ die Summer der Abstände zwischen Zentroid und allen Datenpunkten ist.

3. Wiederhole Schritt eins, bis alle vorgegebenen Zentroiden ausgewählt sind.

Nach Schritt drei werden die zuvor erläuterten Schritte drei bis fünf aus dem K-Means-Algorithmus durchgeführt.

2.3.4 DBSCAN

DBSCAN (Density-Based Spatial Clustering ob Applications with Noise) ist das am häufigsten genutzte dichtebasierte Clusterverfahren (vgl. Kapitel 2.3.2) [Ma21]. Der Algorithmus teilt die Datenpunkte in drei Arten auf [Fr19]:

- Kernpunkt
- Randpunkt
- Rauschen bzw. Rauschpunkt

Die Art der Punkte werden durch zwei Hyperparameter bei der Durchführung von DBSCAN festgelegt. Der erste Hyperparamter ist der maximale Radius ε und beschreibt die Nachbarschaft eines Datenpunkts. Der zweite Hyperparameter ist die minimale Anzahl der Punkte (minPts) innerhalb von ε [Fr19]. Das folgende Beispiel soll den DBSCAN-Algorithmus erläutern.

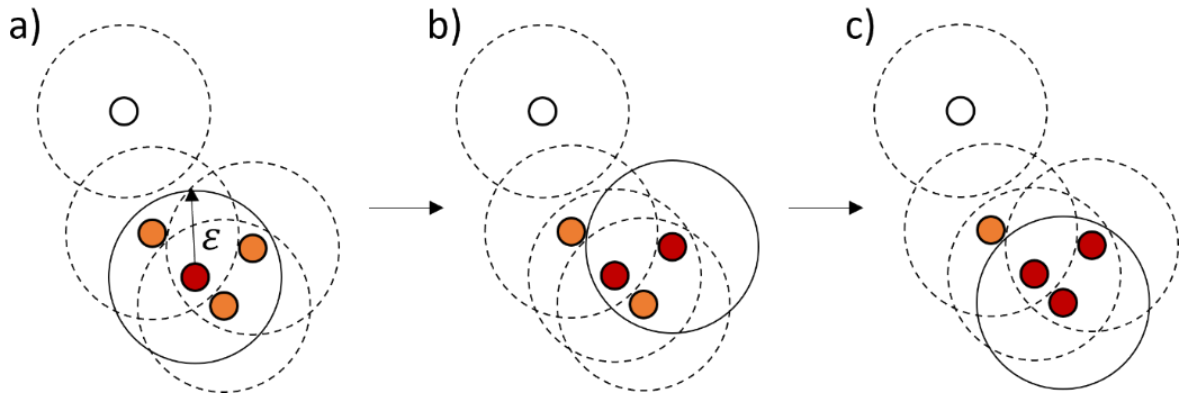


Abbildung 2.8: Vorgehensweise von DBSCAN

In Abbildung 2.8 ist ein Datensatz mit fünf Datenpunkten dargestellt. Der Hyperparameter von minPts wurde auf drei gesetzt. Der Algorithmus geht zu Beginn alle Punkte durch, bis ein Punkt im vorgegebenen ε-Radius mindestens zwei weitere Punkte besitzt. Der Ausgewählte Punkt zählt dabei mit. Die Anforderung an minPts gleich drei ist in a) erfüllt. Der Punkt wird als Kernpunkt (rot) beschriftet und alle weiteren im ε-Radius als Randpunkte (orange) (vgl. Abbildung 2.8 a)). Im Anschluss werden alle Randpunkte untersucht, ob diese mindesten zwei Punkte im vorgegebenen ε-Radius enthalten. Wenn das erfüllt ist, werden diese ebenfalls als Kernpunkte beschriftet (vgl. Abbildung 2.8 b)). Randpunkte, die einen Kernpunkt, aber nicht die Anforderung von mindesten zwei weiteren Punkten in seiner Nachbarschaft erfüllen, führen zum Abbruch der Clusterbildung. Diese Randpunkte bilden die Clustergrenze (vgl. Abbildung 2.8 c)). Besitzen ausgewählte Punkte keine weiteren Punkte im ε-Radius, werden diese als Rauschpunkte beschriftet und gehören nicht zum Cluster.

Eine wichtige Eigenschaft von DBSCAN ist, dass der Algorithmus beim Beschriften von Kernpunkten immer deterministisch und reihenfolgeunabhängig agiert. Das bedeutet, dass bei gleichen Beträgen der Hyperparameter immer die gleichen Cluster gebildet werden [Fr19]. Randpunkte hingegen sind reihenfolgenabhängig, d.h. sie können ihre Clusterzugehörigkeit beim Durchführen des Algorithmus ändern. Das wäre dann der Fall, wenn ein neuer Cluster gebildet wird und der Randpunkt in der Nachbarschaft von dessen Kernpunkten liegt. Zusätzlich ergibt sich daraus der Effekt, dass ein Cluster auch aus weniger als der minimalen Anzahl (minPts) bestehen kann [Fr19].

Wenn die vorhandeneren Daten mehr als zwei Dimensionen besitzen, kann die Wahl der Hyperparameter durch eine Visualisierung nicht abgeschätzt werden. Aus diesem Grund

wird häufig ein mathematischer Ansatz gewählt. Die Autoren in [RS16] schlagen einen Ansatz vor, wobei für jeden Datenpunkt die Distanz zu allen anderen Datenpunkten berechnet wird. Danach wird die minimale durchschnittliche Distanz bestimmt in der sich minPts befinden. Dieses Vorgehen wird mit aufsteigender Distanz so lange durchgeführt, bis alle Datenpunkte, in Abhängigkeit einer Distanz, immer minPts in ihrer Nachbarschaft besitzen. Sobald dieser Zustand erreicht ist, würde DBSCAN einen großen Cluster erzeugen. Im Anschluss werden die Distanzen in Abhängigkeit der Datenpunkte sortiert und graphisch dargestellt. Der Bereich mit der größten Änderung zwischen der Distanz und den Datenpunkten signalisiert einen optimalen Wertebereich für ε [RS16]. Das Beispiel in Abbildung 2.9 zeigt die Visualisierung der Methode.

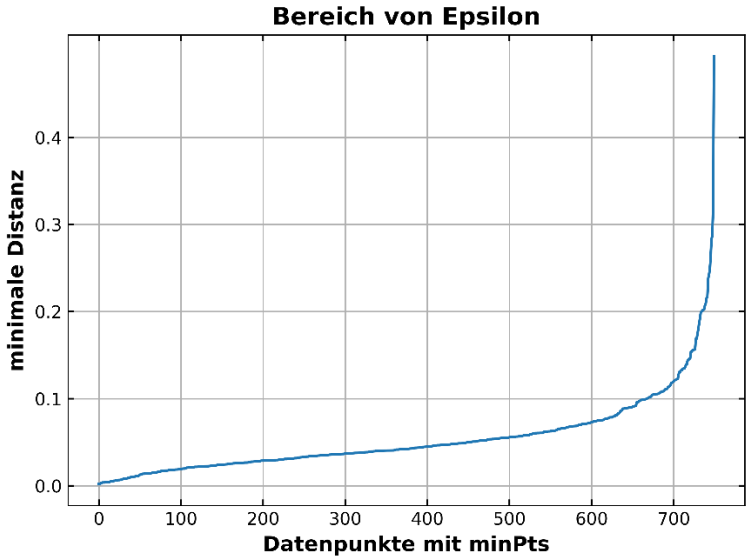


Abbildung 2.9: Analyse von ε mit festgelegter minPts

Zu erkennen ist, dass im Bereich zwischen etwa 0.09 bis 0.02 ein ausgeprägter Ellenbogen vorhanden ist, der als optimaler Wertebereich für ε gewertet werden kann. Die Grundidee bei dieser Methode ist, dass ein Kompromiss zwischen vielen kleinen Clustern und einem großen Cluster gemacht wird, was durch den Ellenbogen deutlich wird.

Dennoch muss bei dieser Methode der Hyperparamter minPts angegeben werden. Die Autoren in [Sa98] schlagen vor, die minimale Anzahl der Punkte auf das Zweifache der Dimensionen zu setzen, d.h. minPts = 2*D. Für zwei Dimensionen wird minPts <= 4 vorgeschlagen. Laut [Fr19] sollte bei der Wahl darauf geachtet werden, „Je größer MinPts, desto eher entstehen weniger kompakte Cluster, und je kleiner MinPts gewählt wird, desto eher entstehen ggf. Cluster aus Rauschpunkten“. Aus diesem Grund sollten bei der

Durchführung von DBSCAN immer mehrere Kombinationen der Hyperparameter untersucht werden.

2.3.5 Gaussian Mixture Modell

Wie bereits erwähnt, bezieht sich Gaussian-Mixture-Model (GMM) auf die Annahme, dass die Daten aus eine Mischung von Gauß-Verteilungen bestehen, deren Parameter unbekannt sind. Die Grundlage für ein GMM ist die Gauß-Verteilung, die durch die Dichtefunktion

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} * e^{-\frac{(x-\mu)}{2\sigma^2}} \tag{2.7}$$

mit dem Mittelwert μ, der Standardabweichung σ und der Datenpunkte x beschrieben wird. Beim GMM werden mehrere Dichtefunktionen an die Daten angepasst, weshalb sich folgende Gleichung ergibt [Bi06]:

$$p(x) = \sum_{k=1}^{K} \pi_k \, N(x_n|\mu_k, \Sigma_k) \quad mit \quad \sum_{k=1}^{K} \pi_k = 1 \tag{2.8}$$

Wobei K die Anzahl der Gauß-Verteilungen, π_k die Gewichtung der Gauß-Verteilungen und Σ_k die Kovarianzmatrix darstellt. Zusätzlich überführt N die Gauß-Verteilung in eine mehrdimensionalen Raum. Wichtig hierbei ist, dass die Kovarianzmatrix die Form der n-dimensionalen Gauß-Verteilungen steuert und es erlaubt, beliebige Ausrichtungen und Formen von Ellipsoiden zu erzeugen. Aus der Formel ergeben sich drei unbekannte Parameter (π_k, μ_k, Σ_k) die ermittelt werden. Um diese Parameter an die Daten anzupassen, wird der Erwartungs-Maximierungs-Algorithmus (engl. Expectation-Maximization-Algorithmus (EM-Algorithmus)) verwendet. Der EM-Algorithmus durchläuft zwei Schritte [Bi06]:

- Erwartungsschritt
- Maximierungsschritt

Zu Beginn werden reine Gauß-Verteilungen mit den Parametern (π_k, μ_k, Σ_k) in Abhängigkeit von k zufällig initialisiert. Anhand der Parameter wird die Wahrscheinlichkeit der Zugehörigkeit aller Datenpunkte zu den jeweiligen Gauß-Verteilungen mit folgender Gleichung berechnet (Erwartungsschritt) [Bi06]:

$$\gamma(z_{nk}) = \frac{\pi_k N(x_n|\mu_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k N(x_n|\mu_k, \Sigma_k)} \tag{2.9}$$

Befinden sich Datenpunkte nahe am Mittelwert einer Gauß-Verteilung, werden diese höher gewichtet und somit ergibt sich ein höheres $\gamma(z_{nk})$. Wobei z_{nk} die Datenpunkte mit der Zuweisung zu den jeweiligen Verteilungen berücksichtigt. Durch das Teilen der gesamten Datenpunkte aller Verteilungen wird der Wertebereich normalisiert. Ein Wert von $\gamma(z_{nk}) = 1$ bedeutet, dass der Datenpunkt genau im Mittelpunkt der Verteilung liegt [Bi06].

Anhand der Gewichtungen von $\gamma(z_{nk})$ werden die Parameter aktualisiert (Maximization-Schritt):

$$\mu_k^{neu} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma * x_n \tag{2.10}$$

$$\Sigma_k^{neu} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(x_n - \mu_k^{neu})(x_n - \mu_k^{neu})^T \tag{2.11}$$

$$\pi_k^{neu} = \frac{N_k}{N} \tag{2.12}$$

Wobei $N_k = \sum_{n=1}^{N} \gamma(z_{nk})$ ist. Wenn die Gauß-Verteilung die Datenpunkte optimal beschreibt, dann ist γ nahe eins und hat kaum Einfluss auf die Anpassung der Parameter. Sobald die Datenpunkte kaum von der Gauß-Verteilung erklärt werden, geht γ gegen Null und hat einen hohen Einfluss auf die Parameter [Bi06].

Nach der Durchführung der zwei Schritte, werden die angepassten Parameter über die sogenannte Log-Likelihood-Funktion überprüft:

$$\ln p(X|\mu, \Sigma, \pi) = \sum_{n=1}^{N} ln \left\{ \sum_{k=1}^{K} \pi_k N(x_n|\mu_k, \Sigma_k) \right\} \tag{2.13}$$

Die Log-Likelihood-Funktion ist die logarithmierte Mischverteilung der Datenpunkte aus der Summe aller Gauß-Verteilungen. Anhand der Log-Likelihood-Funktion wird beobachtet, ob die Änderung der Parameter eine Erhöhung dieser Funktion hervorruft. Wird eine Erhöhung erreicht, wird der EM-Algorithmus erneut zur Anpassung der Parameter durchgeführt, solange bis keine Anpassung erfolgt oder eine vorgegeben Anzahl von

Iterationen erreicht wird. Eine weitere Möglichkeit für die Begrenzung der Iterationen ist, die Festlegung von Schwellwerten für die einzelnen Parameter [Bi06]. Die Festlegung der Schwellwerte setzt jedoch voraus, dass eine umfangreiches Wissen über die Daten vorliegt. Ein ausführliche mathematische Beschreibung des GMM kann [Bi06] entnommen werden. Die folgenden Abbildung 2.10 verdeutlicht die Vorgehensweise vom GMM anhand eines zweidimensionalen Beispiels:

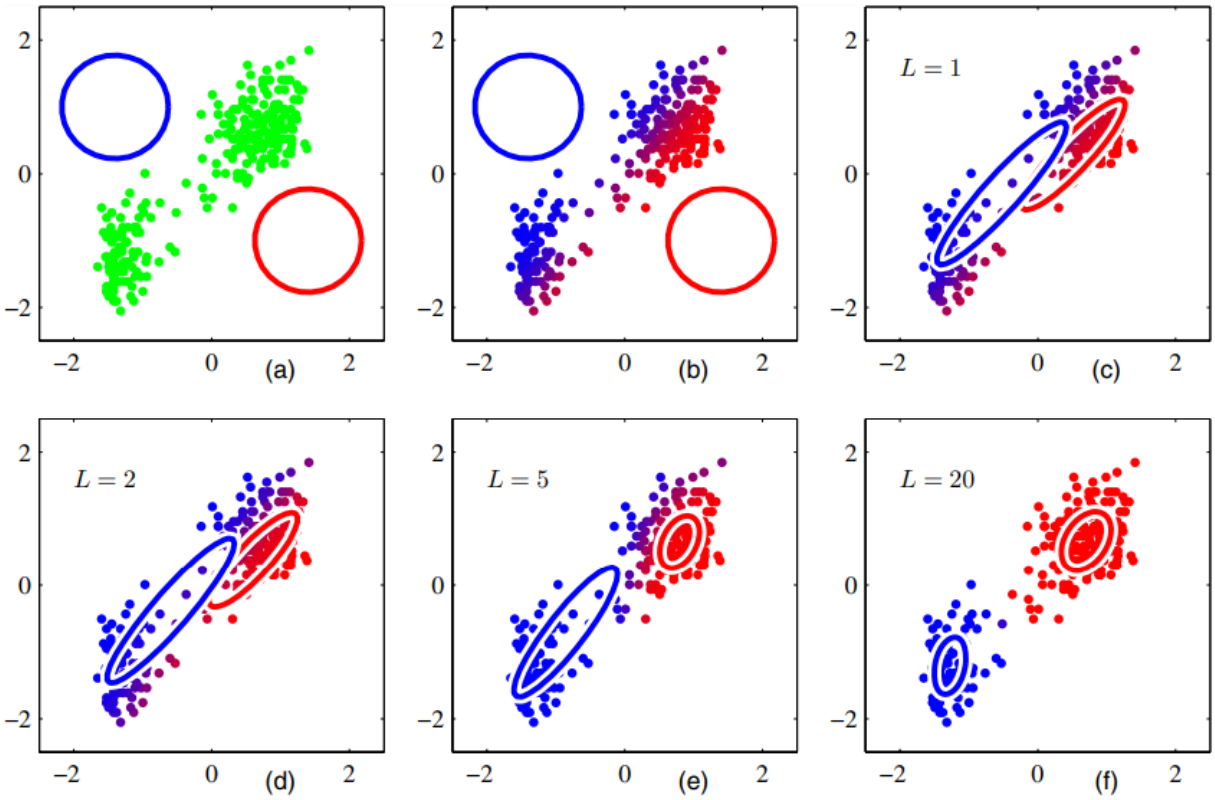


Abbildung 2.10: Visualisierung der Schritte eines EM-Algorithmus (zweidimensionales Beispiel) [Bi06]

In (a) findet die Initialisierung der reinen Gauß-Verteilungen statt (k=2). In (b) wird der Erwartungsschritt ausgeführt, wobei die Datenpunkte den Gauß-Verteilungen (Clustern) in Abhängigkeit der Gewichtung durch $\gamma(z_{nk})$ zugeordnet werden. Die lila markierten Punkte, demonstrieren Punkte, die zu beiden Clustern gehören. In (c) werden die Parameter durch den Maximierungsschritt anhand der Gewichtung von $\gamma(z_{nk})$ aktualisiert. Für (d), (e) und (f´) gelten die gleichen Schritte. Nach 20 Iterationen ist der EM-Algorithmus an seine Vorgaben konvergiert.

2.3.6 Validierung von Clusterverfahren

Nach der Durchführung einer Clusteranalyse müssen die Ergebnisse bewertet werden. Dazu gehört die Bestimmung der Anzahl von Clustern und die Beurteilung der Qualität von Clustern. Zusätzlich unterscheidet man dabei zwischen zwei Herangehensweisen [HKP11]:

- **Externe Bewertung:** Die wahren Ergebnisse (Labels) sind für die Bewertung der Clusterverfahren vorhanden.
- **Interne Bewertung:** Es sind keine Labels vorhanden, weshalb die Bewertung durch Eigenschaften innerhalb der Cluster bzw. zwischen den Clustern durchgeführt werden muss.

Im Rahmen dieser Arbeit muss auf die interne Bewertung zurückgegriffen werden, da keine gelabelten Daten vorliegen. Die interne Bewertung der Clusterbildung wird unter zwei wesentlichen Aspekten durchgeführt [HBV01]:

- **Kompaktheit**: Die Datenpunkte innerhalb der jeweiligen Cluster sollten so nahe wie möglich beieinander liegen.
- **Trennung**: Die Cluster sollten weit voneinander entfernt liegen.

Wichtig ist, dass die Bewertung der Kompaktheit bzw. Trennung mit der Bestimmung der Anzahl von Clustern einhergehen, d.h. anhand von Kompaktheit und Trennung können Rückschlüsse auf die Anzahl der Cluster gemacht werden.

Validierung von K-Means

In vielen Literaturen wurde die heuristische Ellenbogenmethode zum Ermitteln von k für den K-Means-Algorithmus genutzt [Ab21], [PT21], [Mo20]. Bei dieser Methode wird mit aufsteigendem k, die Summe der quadratischen Abweichung (siehe Gleichung 2.5) für alle Cluster zwischen Zentroid und den Datenpunkten berechnet. Sobald sich die Abweichung mit steigerndem k nur wesentlich reduziert, kann davon ausgegangen werden, dass neue Cluster nahe aneinander liegen und keinen großen Informationsgehalt bieten [HR20]. Die Abbildung 2.11 verdeutlicht die Ellenbogen-Methode anhand eines Beispiels:

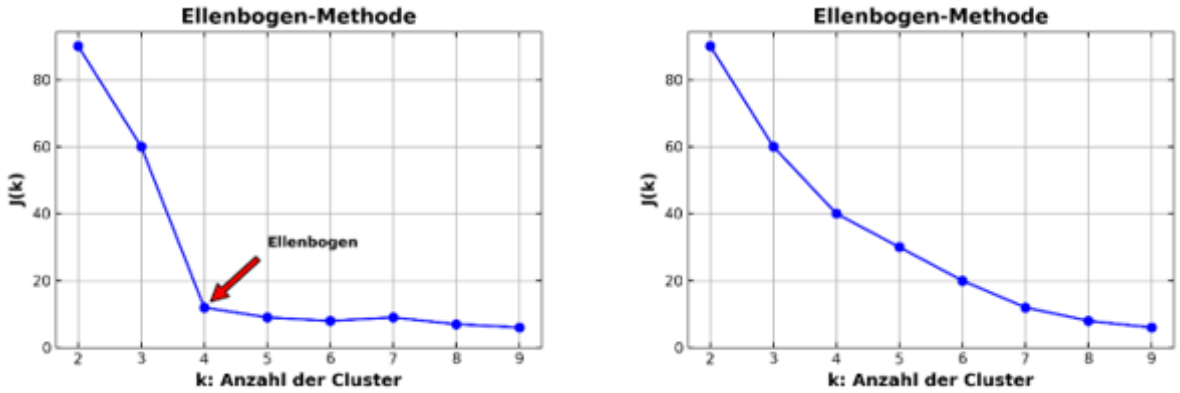


Abbildung 2.11: Visualisierung der Ellenbogen-Methode

Im linken Graph ist ein deutlicher Knick (Ellenbogen) bei k = 4 sichtbar. Nach dem Knick stagniert die Zielfunktion mit weiter steigendem k. Eine weitere Aufteilung der Daten ist nicht sinnvoll. In vielen Fällen besteht das Problem, dass der Ellenbogen nicht deutlich ausgeprägt ist [HR20]. Der rechte Graph soll diesen Fall verdeutlichen. Aus diesem Grund wird eine zweite Bewertungsmethode herangezogen.

In [YY19] wurden die Gap-, Silhouette- und Canopy-Methode verglichen. Die Autoren haben festgestellt, dass die Methoden die gestellten Anforderungen gleichermaßen erfüllen. Bei sehr großen Datensätzen unterscheiden sich die Methoden aber hinsichtlich ihrer Berechnungsdauer. Die Berechnungsdauer kann im Rahmen dieser Arbeit vernachlässigt werden. Der Datenumfang ist gering und es wird keine Echtzeitfähigkeit vorausgesetzt. Aufgrund dessen wurde die Silhouetten-Methode ausgewählt, da diese in der verwendeten Software-Bibliothek implementiert ist.

Anders als bei der Ellenbogen-Methode, die nur die Kompaktheit der Cluster berücksichtig, wird bei der Silhouette-Methode auch die Trennung berücksichtigt [Ro87]. Die Berechnung wird mit folgender Gleichung nach [Ro87] durchgeführt:

$$s(i) = \frac{b(i) - a(i)}{\max\{\mathrm{a(i)}, \mathrm{b(i)}\}} \tag{2.14}$$

Im ersten Schritt wird ein beliebter Punkt i in einem Cluster ausgewählt. Anschließend werden die Distanzen zwischen Datenpunkt i und allen weiteren Punkten im Cluster berechnet. Danach wird der Mittelwert a(i) von allen Distanzen gebildet. Ist der Wert von a(i) klein, deutet das auf eine hohe Kompaktheit der Clusters hin oder, bei einem hohen a(i), auf eine weite Streuung der Datenpunkte. Das ist damit zu begründen, dass bei einem kleinen

a(i) die Distanzen der Daten innerhalb der Cluster klein sind und damit die Unähnlichkeit gering ist. Die Daten deuten auf einen hohe Kompaktheit hin [Ro87].

Im nächsten Schritt wird ausgehend vom Datenpunkt i, die mittlere Distanz zwischen Datenpunkten in einem anderen Cluster berechnet. Das wird für alle weiteren Cluster wiederholt. Die kleinste mittlere Distanz zu einem Cluster wird für b(i) eingesetzt. Der Wert b(i) gibt also an, wie unähnlich der Datenpunkt zu dem am naheliegendsten Cluster ist [Ro87].

Im letzten Schritt wird s(i) durch die Differenz zwischen b(i) und a(i) berechnet. Zusätzlich wird durch die maximale mittele Distanz dividiert. Das ermöglicht eine Normalisierung auf einen Wertebereich von [-1, 1]. Es gelten folgende Zusammenhänge für a(i) und b(i) [Ro87]:

- 1 wenn a(i) << b(i) (Punkt ist sehr gut zugewiesen)
- 0 wenn a(i) = b(i) (Punkt befindet sich zwischen zwei Clustern)
- -1 wenn a(i) >> b(i) (Punkt ist sehr schlecht zugewiesen)

Diese Berechnung wird für alle Datenpunkte im Datensatz durchgeführt und anschließend der Mittelwert aller sogenannten Silhouetten ermittelt. Dieser Vorgang wird wiederum mit aufsteigendem k durchgeführt und der größte Mittelwert gibt das optimale k an [Ro87].

Validierung von GMM

Die zuvor erläuterten Methoden eigenen sich besonders für sphärische Cluster, die für K-Means gut geeignet sind. Besitzen die Cluster jedoch andere Formen oder sind unterschiedlich groß, neigen die Methoden zu schlechten Ergebnissen [Gé20].

Aus diesem Grund empfiehlt die Literatur [Gé20] für die Bestimmung der Anzahl von Clustern beim GMM die Verwendung vom Bayessche-Informationskriterium (BIC) und Akaike-Informaitonskriterium (AIC). Diese werden mit folgender Gleichungen berechnet[Gé20]:

$$BIC = Log(m)p - 2log(\hat{L}) \tag{2.15}$$

$$AIC = 2p - log(\hat{L}) \tag{2.16}$$

Mit den Variablen:

m = Anzahl an Datenpunkten

p = Anzahl der Parameter, die vom Modell erlernt wurden (Abhängig von der Anzahl der Gauß-Verteilungen, z.B. bei zwei Gauß-Verteilungen wären es vier Parameter (2μ, 2σ))

$\hat{L}$ = Maximaler Wert der Log-Likelihood-Funktion des Modells

Das BIC und AIC verhindern im Allgemeinen eine Überanpassung des GMM, in dem eine hohe Komplexität des Modells bestraft wird. Eine hohe Komplexität kann zu einer hohen Anzahl von Clustern führen, die den gegeben Datensatz genau beschreiben. Das Ziel ist es, einen Kompromiss zwischen $\hat{L}$ und p für die optimale Beschreibung der Daten zu finden. Dabei gibt die Log-Likelihood-Funktion an, wie gut das GMM die Daten beschreibt (vgl. Kapitel 2.3.5). Es wird immer ein maximales $\hat{L}$ angestrebt. Das $\hat{L}$ hängt vom EM-Algorithmus und der Anzahl von Cluster (k) ab. Die Erhöhung von k geht mit einer Erhöhung der Parameter einher (Cluster = Parameter) und steigert die Anpassungsfähigkeit vom GMM. Jedoch führen viele Cluster dazu, dass die Interpretierbarkeit abnimmt und keine allgemeingültige Aussage über den Datensatz gemacht werden kann. Aus diesem Grund wird als Gegengewicht das p eingeführt, das die Komplexität des GMM durch die Anzahl der Parameter berücksichtigt. Dadurch wird ein Gleichgewicht zwischen Anpassungsgüte und Modellkomplexität erreicht. In der Praxis wird die Anzahl der Cluster mit dem geringsten BIC oder AIC ausgewählt. Das BIC tendiert dazu, Modelle mit vielen Parametern stärker zu bestrafen und neigt zur Unteranpassung. Das AIC tendiert hingegen zu einer Überanpassung. Aus diesem Grund wird häufig ein Vergleich zwischen AIC und BIC durchgeführt, um das beste Ergebnis aus beiden Kriterien zu ermitteln [Gé20]. Wichtig ist, dass die Werte von AIC und BIC nicht interpretierbar sind und nur für den Vergleich von mehreren Modellen (hier GMM) verwendet werden.

Validierung von DBSCAN

Die Silhouetten-Methode kann für dichtebasierte Verfahren verwendet werden, unter der Voraussetzung, dass die Cluster sphärische Formen besitzen. Sobald das nicht der Fall ist, wovon man ausgehen sollte, sind distanzbasierte Validierungskriterien ungeeignet [Mo14]. Zudem werden die Cluster durch Bereiche mit geringer Dichte getrennt und können Rauschpunkte beinhalten. Die Silhouette-Methode kann die Rauschpunkte nicht von anderen Clustern unterscheiden und fließen mit in die Berechnung ein.

Aufgrund dessen wird die vorgeschlagene Methode aus [Mo14] verwendet. Die Autoren haben ein Validierung für dichtebasierte Clusterverfahren (DBCV) entwickelt. Die Methode

bewertet die Dichte zwischen und innerhalb der Cluster. Außerdem wird das Rauschen berücksichtigt. Der Validierungsindex wird nach folgender Formel berechnet [Mo14]:

$$DBCV(C) = \sum_{i=1}^{i=l} \frac{|C_i|}{|O|} V_C(C_i) \tag{2.17}$$

Wobei C_i die jeweiligen Datenpunkte im Cluster angibt und O die gesamten Datenpunkte aus dem Datensatz (inkl. Rauschpunkte). Der Parameter V_C beschreibt die Dichte innerhalb eines Clusters im Verhältnis zur Dichte, die ihn von anderen Clustern trennt. Demzufolge gilt für DBCV:

- V_C = 1 : Dichte ist innerhalb der Cluster hoch (gute Clusterbildung)
- V_C = 0 : Dichte ist gleich
- V_C = -1 : Dichte ist innerhalb der Cluster geringer (schlechte Clusterbildung)

Die genaue Herleitung von DBCV kann [Mo14] entnommen werden. Für den dazugehörigen Quellcode wird auf [Je22] verwiesen.

3 Verwandte Arbeiten

In diesem Kapitel werden relevante Literaturen im Bereich der Kategorisierung von Nutzertypen an Ladesäulen vorgestellt. Dadurch soll ein Überblick zum Stand der Technik verschafft werden, verschieden Ansätze analysiert und nützliche Informationen für das weitere Vorgehen gesammelt werden. Dabei wird der Fokus auf die Clusteranalyse gesetzt.

In [Xi18] wurde das Ladeverhalten von Nutzen mit dem K-Means-Clusterverfahren kategorisiert. Für die Clusteranalyse wurden Mittelwert und Standartabweichung von Ankunfts- und Abfahrzeit, sowie Pearson-Korrelationskoeffizienten zwischen Aufenthaltsdauer und Energieverbrauch genutzt. Das Ergebnis waren vier Nutzergruppen, die nicht weiter interpretiert wurden. Im Anschluss wurde das Ergebnis dazu genutzt, neue Nutzer mit Hilfe eines Neuronalen Netzes zu klassifizieren. Ein vergleichbarer Ansatz wurde auch in [Sh20] gewählt. Hier wurde der Mittelwert und die Standardabweichung der Ladezeit und die Verbindungszeit genutzt. Die Anzahl der Cluster wurde mit Hilfe der Ellenbogenmethode ausgewählt. Es wurden drei Cluster identifiziert, die für das Klassifizieren von neuen Nutzern mittels k-Nearest-Neighbors genutzt wurden. Das Thema des Ladeverhaltens von Nutzern wurde auch in [Xy16] untersucht. Es wurde k-Means für das Clustern von Ladeprofilen in drei unterschiedlichen Regionen von UK verwendet. Die Anzahl der Cluster wurde mit dem Davies-Boudin-Bewertungskriterium ermittelt. Die Zentren der Cluster wurden als typische Ladeprofile für die jeweilige Region festgelegt. In [De16] wurde das dichtebasierte Clusterverfahren DBSCAN verwendet. Anhand von Ankunftszeit und Abfahrtszeit wurden drei Cluster ermittelt. Der erste Cluster wurde „Laden in der Nähe von zu Hause“ bezeichnet, wobei die Ankunftszeit am Nachmittag/Abend und die Abfahrzeit morgens am nächsten Tag ist. Im zweiten Cluster „Laden nahe der Arbeit“ kommen die Nutzer morgens und fahren am Nachmittag. Im letzten Cluster „parken zum Laden“ kommen die Nutzer über den Tag verteilt, wobei kaum Leerlaufzeiten (parken, ohne zu laden) entstehen. Anzumerken ist, dass die Parameter für den Algorithmus empirisch anhand der Visualisierung durch das Streudiagramm ermittelt wurden. Der Datensatz enthielt 387.524 Ladesitzungen. Die Autoren in [JQ18] haben das Gaussian Mixture Modell für das Clustern von 221 Elektrofahrzeugen mit über 85000 Ladesitzungen (Nissan LEAF) anhand von Ladestart und Ladezustand der Batterie untersucht. Das Ergebnis wurde für das Erstellen von realistischen Profilen der Nutzer verwendet und im Nachgang mit gelabelten Daten validiert. Der Fehler betrug weniger als 1%.

Zusammenfassend kann man sagen, dass die Anzahl von Veröffentlichungen im Bereich des unüberwachten Lernens für die Analysen von Nutzerverhalten an Ladesäulen noch sehr begrenzt ist und viel Potenzial für weitere Untersuchungen bietet. Viele Literaturen haben in der Vergangenheit bevorzugt das überwachte Lernen für das Vorhersagen des Verbrauches an Ladesäulen oder das bestärkte Lernen für die Verwaltung von Ladeaktivitäten an Ladesäulen genutzt. Ein Überblick über verschiedene Literaturen kann aus [De21] entnommen werden. In Bezug auf die aufgeführten Veröffentlichungen wurden weiträumige Regionen mit umfangreichen Datensätzen zum Clustern verwendet. Es stellt sich die Frage, ob die verwendeten Algorithmen auch für kleinere Datensätze aussagefähige Ergebnisse liefern. Außerdem wurden keine Vergleiche zwischen unterschiedlichen Clusterverfahren durchgeführt, weshalb keine klare Aussage über die Wirksamkeit gemacht werden kann. Das Clustering diente mehr als Vorverarbeitungsschritt, um neue Nutzer zu klassifizieren. Zusätzlich fiel auf, dass die Autoren nur wenige Merkmale für das Clustern verwendet habe. In [Xi18] wären z.B. die Ladezeit und Leerlaufzeit als zusätzlich Merkmale denkbar. Auch in [De16] wurde nur Ankunft- und Abfahrzeit genutzt und im Nachhinein wurden weitere Merkmale den Clustern zugeordnet. Aus diesem Grund wäre eine Analyse mit weiteren Merkmalen durch ein Clusterverfahren denkbar.

4 Methodik

Nachdem die Grundlagen in Kapitel 2 näher beschrieben wurden, wird in diesem Kapitel die Methodik für einen möglichen Ablauf zur Clusteranalyse erläutert. Das ist insofern sinnvoll, dass beim Bearbeiten eines Projektes ein roter Faden entsteht. Dadurch werden bedeutsame Teilschritte im Vorfeld erkannt und ein systematisches Vorgehen gewährleistet. Die Autoren in [HBV01] schlugen einen Prozess in Anlehnung an den KDD-Prozess von Fayyed [Fa96] vor, der konkret für die Durchführung einer Clusteranalyse genutzt werden kann.

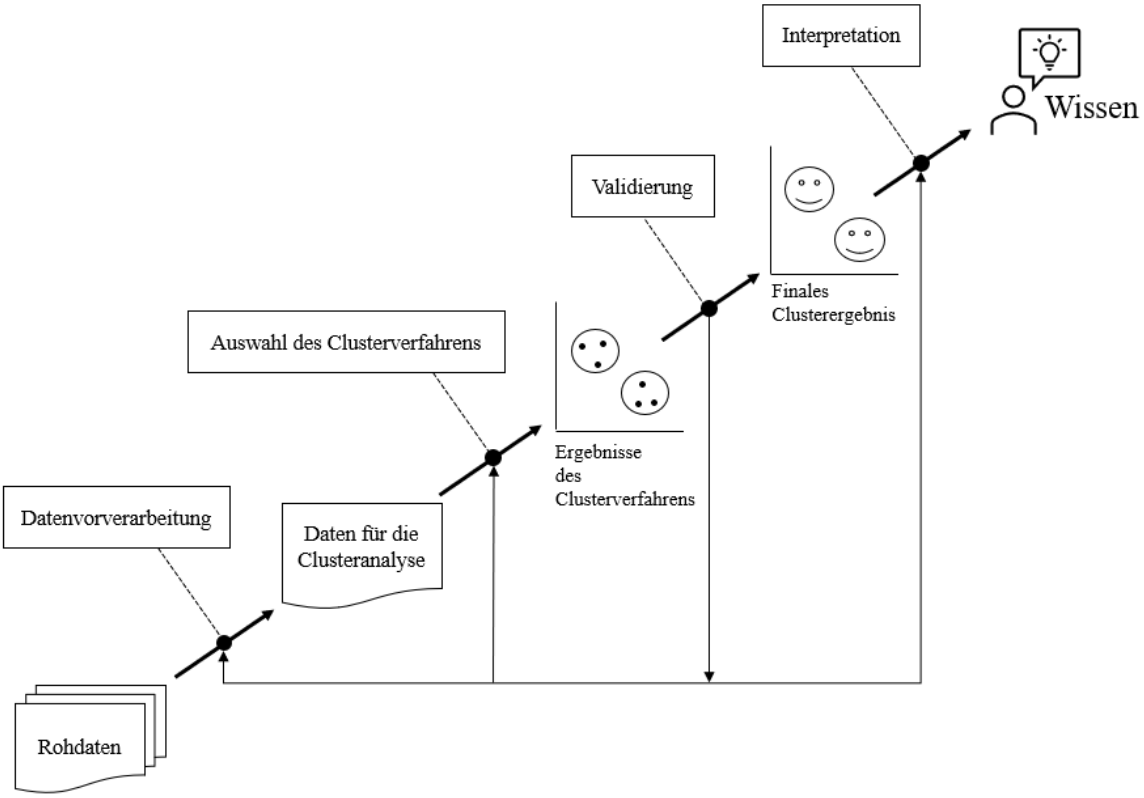


Abbildung 4.1: Prozessschritte bei eine Clusteranalyse in Anlehnung an [HBV01]

„Das Hauptanliegen des Clustering-Prozesses besteht darin, die Organisation von Mustern in "sinnvolle" Gruppen aufzuzeigen, die es uns ermöglichen, Ähnlichkeiten und Unterschiede zu entdecken und nützliche Schlussfolgerungen daraus abzuleiten“ [HBV01]. Der Prozess lässt sich dabei in vier Schritte untergliedern, die iterativ durchlaufen werden. Die einzelnen Schritte werden in den nächsten Abschnitten näher erläutert und mit wichtigen Ergänzungen erweitert.

4.1 Datenvorverarbeitung

Zu Beginn jedes Projektes wird ein Problem definiert, welches zu lösen gilt. Dazu werden Hypothesen in Abhängigkeit der Aufgabenstellung erstellt, die es zu belegen oder zu widerlegen gilt. Dazu werden die vorhandenen Datenquellen analysiert und beschrieben. Dafür können verschiedene Darstellungsarten genutzt werden, wie Tabellen, Diagramme oder Parameter. Das grundlegende Ziel ist es, einen Überblick über die Qualität der Daten zu bekommen und bedeutsame Eigenschaften und Muster zu erkennen [Ma21]. Aufgrund dessen können zum Beispiel fehlende Werte, Duplikate, ungültige Werte, Ausreißer, Datentypen und Muster festgestellt werden, die dann in Abhängigkeit der Anforderungen vorverarbeitet werden. Grundlegend sollte immer eine minimale Informationsredundanz angestrebt werden, d.h. Merkmale mit gleichem Informationsgehalt (hohe Korrelation) sollten reduziert werden [RB16]. Laut [Ba21] „führen hohe Korrelation zu einer impliziten Gewichtung bestimmter Aspekte beim Clusterprozess“. Damit besteht zum Beispiel bei partitionierenden Clusterverfahren die Gefahr, das sich der Schwerpunkt von Clustern an Bereichen mit hoher Korrelation orientiert und andere Merkmale weniger gewichtet [De18]. Die Auswahl von Merkmalen besitzt eine erhebliche Bedeutung, denn die Ergebnisse können häufig nicht mit Hilfe von vorhanden Labels validiert werden.

4.2 Auswahl des Clusterverfahrens

Nach der Auswahl von geeigneten Merkmalen und der Vorverarbeitung dieser, wird im zweiten Schritt ein geeigneters Clusterverfahren ausgewählt. Die Auswahl hängt von vielen Faktoren ab. Die Autoren in [HKP11] haben aus diesem Grund einige Kriterien an das Clusterverfahren aufgelistet, die dabei helfen einen geeignetes Verfahren auszuwählen:

- **Skalierbarkeit und Dimensionalität:** Einige Clusterverfahren kommen mit kleineren Datensätzen besser zurecht als mit großen Datensätzen. Außerdem kommen Clusterverfahren grundsätzlich mit niederdimensionalen Daten besser zu recht. Eine mögliche Ursache hierfür ist, die Verzerrung der Daten in hochdimensionalen Räumen.
- **Fähigkeit zum Umgang mit verschiedenen Attributen:** Viele Clusterverfahren können nur einen Datentyp verarbeiten. Der K-Means-Algorithmus kann zum Beispiel nur mit metrischen Merkmalen umgehen. Andere hingegen können binäre oder nominale Merkmale verarbeiten.

- **Entdecken von Clustern mit beliebiger Gestalt:** Beispielsweise neigen partitionierende und hierarchische Clusterverfahren zur Bildung von sphärischen Clustern. Dichtebasierte können hingegen jegliche Formen bilden.
- **Fachwissen über Eingabedaten:** Verschiedene Clusterverfahren besitzen Hyperparameter, die Fachwissen über z.B. die Anzahl der Cluster voraussetzen. Besonders bei hochdimensionalen Daten kann die Bestimmung eine Herausforderung darstellen.
- **Fähigkeit zum Umgang mit verrauschten Daten:** Bei verrauschten Daten eignen sich einige Clusterverfahren besser, wie z.B. dichtebasierte Verfahren, die diese kennzeichnen können.
- **Interpretierbarkeit und Nützlichkeit**: Um einen Nutzen aus Clusterverfahren zu generieren, müssen die Ergebnisse interpretierbar bzw. verständlich sein. Einige Verfahren können zudem mit nützlichen Anwendungen verknüpft werden. Wie zum Beispiel einer Dimensionsreduktion oder einer graphischen Darstellung der Ergebnisse.
- **Trennung der Cluster**: Einige Clusterverfahren teilen die Daten in feste Cluster auf, wobei alle Daten nur einem Cluster angehören (harte Cluster). Andere wiederum erzeugen überlappende Cluster, wodurch Daten zu mehreren Clustern gehören können (weiche Cluster).

Anhand dieser Kriterien können einige Vorüberlegungen für die Wahl von Clusterverfahren gemacht werden und zudem für die Vorverarbeitung der Daten einsetzt werden. Dennoch wird ersichtlich, dass nicht alle Kriterien vorab bekannt sein können. Zum Beispiel ist die Form oder Anzahl von Clustern bei mehrdimensionalen Daten in der Regel nicht bekannt. In den meisten Fällen möchte man genau das mit der Clusteranalyse feststellen.

4.3 Validierung der Ergebnisse

In diesem Schritt wird die Validierung der erzeugen Cluster durchgeführt. Hierzu müssen bedeutsame Eigenschaften, wie die Kompaktheit und Trennung der erzeugten Cluster überprüft werden. In Abhängigkeit der Resultate sind mögliche Anpassungen an den Hyperparameter vorzunehmen. Bei partitionierenden Clusterverfahren wäre z.B. die Anpassung von k denkbar. Außerdem könnte ein anderes Clusterverfahren besser geeignet sein.

4.4 Interpretation

Im Anschluss müssen die Ergebnisse der Clusteranalyse interpretiert werden. Dafür muss das Domänenwissen über die jeweiligen Anwendungsbereiche genutzt werden. Die Gefahr besteht, dass die Cluster bedeutungslose Muster besitzen und kein neues Wissen abgeleitet werden kann [Fa96]. Tritt dieser Fall ein, müssen gegebenenfalls Anpassungen an den vorigen Schritten vorgenommen werden.

5 Konzeption

Anhand der Vorgestelle Methodik in Kapitel 3 findet die Konzeptionierung für die Clusteranalyse der Ladesäulendaten statt. Das Ziel ist es, die vorhanden Datenquellen zu analysieren und geeignete Merkmale zu identifizieren. Im Anschluss werden geeignete Clusterverfahren für die experimentelle Umsetzung ausgewählt.

5.1 Analyse und Vorverarbeitung der Daten

Als Datenquelle dient das Dashboard der Firma Chargcloud für die Standorte der Ostfalia. Das Dashboard beinhaltet alle Ladevorgänge für Salzgitter und Wolfenbüttel. Zum Zeitpunkt des Exportierens wurden insgesamt 2863 Ladevorgänge aufgezeichnet, davon 1032 am Standort Wolfenbüttel und 1831 am Standort Salzgitter. Durch die relativ geringe Menge an aufgezeichneten Ladevorgängen wird die Annahme getroffen, dass sich das Verhalten der Nutzer von Wolfenbüttel und Salzgitter ähnelt. Beide Standorte besitzen ein ähnliche Infrastruktur, wobei sich der Standort in Salzgitter in einem weniger besiedelten Umfeld befindet. Dennoch kann davon ausgegangen werden, dass an beiden Standorten die Nutzung überwiegend durch Hochschulangehörige erfolgt.

Die Ladevorgänge können durch die folgenden Merkmale erklärt werden:

Abbildung 5.1:Verfügbare Merkmale für die Ladevorgänge

Die markierten Merkmale (✓) stellen die Vorauswahl für die weitere Analyse dar. Die nicht ausgewählten Merkmale enthalten entweder keine Einträge oder besitzen einzigartige Datenpunkte bzw. Hochzählungen, die keine Zusammenhänge zwischen den Ladevorgängen beschreiben. Die Ladevorgänge mit den ausgewählten Merkmalen können direkt über das Dashboard in eine CSV-Datei exportiert werden. Die ausgewählten Merkmale enthalten folgende Information:

- Gestartet = Uhrzeit und Datum des Anschlusses eines E-Fahrzeugs (Ladekabel angeschlossen).
- Beendet = Uhrzeit und Datum der Trennung eines E-Fahrzeugs (Ladekabel getrennt).
- Standzeit = Gesamte Zeitspanne in dem das E-Fahrzeug mit der Ladesäule verbunden ist.
- Verbrauch (kWh) = Gesamtverbrauch in Kilowattstunden des Ladevorgangs.

Durch das Datum im Merkmal Gestartet kann außerdem der Wochentag bestimmt werden. Der Wochentag gibt Auskunft darüber, an welchen Tagen die meisten Ladevorgänge stattfinden.

Zudem besteht die Möglichkeit, die Leistungskurven für alle Ladevorgänge einzeln zu exportieren. Das würde jedoch viel Aufwand bedeuten, weshalb das sogenannte „Web-Scraping“ verwendet wurde. Das Python-Packet Selenium [Se22] stellt die dafür erforderlichen Funktionen bereit. Das Web-Scraping ermöglicht die automatisierte Extraktion von Information von Webseiten. Im Grunde wird das manuelle Betätigen der Schaltsymbole durch das Eingabegerät über eine Algorithmus durchgeführt, der sich selbständig durch die Webseite bewegt und die vorgegeben Information, in diesem Fall die Leistungskurven, exportiert.

Nach dem Exportieren fiel auf, dass die Messung der Leistungskurve zu einem späteren Zeitpunkt durchgeführt wurde. Insgesamt ergeben sich 1287 Leistungsverläufe für die Clusteranalyse. Außerdem wurde zu Beginn die Messung mit einer Abtastzeit von 15 min durchgeführt, die im Laufe der Zeit auf 10 min und weiter auf 5 min runtergesetzt wurde. Wichtig ist, dass die Verläufe das reale Ladeverhalten widerspiegeln, d.h. einige Verläufe zeigen keine gesamte Vollladung (vgl. Abbildung 2.1). Die Verläufe können durch das Stoppen der Ladung (Stecker raus) unterbrochen werden.

Bedingt durch die geringe Anzahl von Leistungsverläufen und der starken Unregelmäßigkeiten, wird für die Clusteranalyse die maximale Ladeleistung extrahiert. Diese gibt Auskunft darüber, welcher Leistungstyp die Ladesäulen häufig aufsucht. Dies erlaubt eine grobe Einschätzung über den Fahrzeugtyp. Zusätzlich kann die Ladezeit entnommen werden, die sich durch die Zeitpanne ergibt, in der Leistung übertragen wird. Zusammen mit der Standzeit kann die Leerlaufzeit aus der Differenz zwischen Standzeit und Ladezeit ermittelt werden. Eine Leerlaufzeit entsteht, wenn das Fahrzeug vollgeladen ist und an der Ladesäule angeschlossen bleibt (keine Leistungsübertragung). Die Ladesäule wird blockiert und verhindert die Benutzung für andere E-Fahrzeuge.

Eine weitere Überlegung wäre die Temperatur mit in die Clusteranalyse einzubinden. Die Temperatur hat einen wesentlichen Einfluss auf den Verbrauch von E-Elektrofahrzeuge. Die optimale Betriebstemperatur von Lithium-Ionen-Akkus liegt zwischen 18 °C und 25 °C, weshalb bei niedrigen und hohen Temperaturen die Akkus klimatisiert werden müssen [Ka21]. Ein weiterer Faktor ist, dass die Besitzer die Heizung oder Klimaanlage verwenden, die einen hohen Mehrverbrauch verursachen. Um die Auswirkungen der Temperatur zu untersuchen, wurden die aufgezeichneten mittleren Temperaturen vom „Deutschen Wetterdienst“ für den Standort Braunschweig verwendet [DWD22]. Für die Standorte der Ostfalia wurden keine Temperaturen aufgezeichnet. Der Einfluss von der Temperatur auf den durchschnittlichen Verbrauch für die jeweiligen Monate zeigt die folgende Abbildung 5.2:

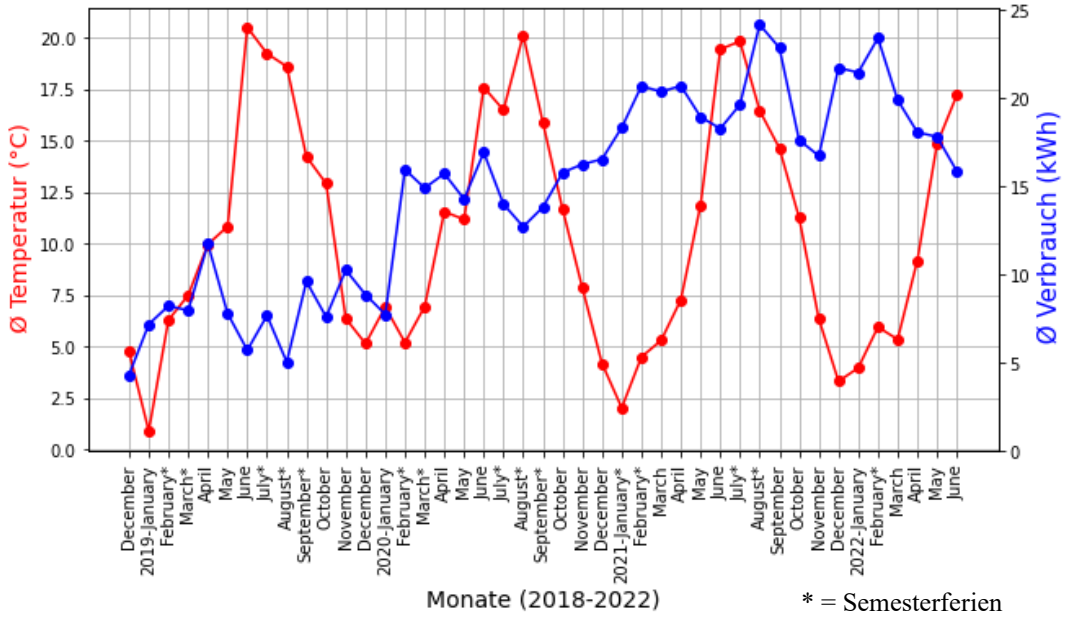


Abbildung 5.2: Einfluss der Temperatur auf den Verbrauch an den Ladesäulen der Ostfalia

Zu erkennen ist, dass der Verbrauch durch die Zunahme der E-Fahrzeuge und der zusätzlich installierten Ladesäulen über die Monate bzw. Jahre zunimmt. Durch die geringe Anzahl von Ladevorgängen und dem dynamischen Verhalten an der Hochschule ist der Verbrauch sehr dynamisch. Zusätzlich wurde das durch die Coronamaßnahmen verstärkt. Beobachtet man den Verbrauch im Januar 2021 genauer, steigt dieser bei geringer Temperatur. Der Anstieg könnte sich durch die vorigen Feiertage (Silvester, Weihnachten) erklären. Der Verbrauch erhöht sich aber auch im drauffolgenden Monat, obwohl die Temperatur zunimmt. Im Juli ist das gleiche Phänomen bei hohen Temperaturen zu beobachten. Der Verbrauch steigt weiter im darauffolgenden Monat, obwohl die Temperatur abnimmt. Anhand der Daten kann keine klare Aussage über den Einfluss von Temperatur auf den Verbrauch an den Ladesäulen gemacht werden. Aus diesem Grund wird die Temperatur nicht bei der Clusteranalyse berücksichtigt.

Die Uhrzeit von Gestartet und Beendet wird im weiteren Verlauf der Arbeit als Ankunftszeit und Abfahrzeit bezeichnet. Zusammengefasst ergeben sich die folgenden acht Merkmale für die weitere Analyse:

- Ankunftszeit
- Abfahrzeit
- Standzeit
- Ladezeit
- Leerlaufzeit
- Verbrauch (kWh)
- maximale Leistung (kW)
- Wochentag

Die Merkmale werden in einer CSV-Datei für die Analyse zusammengefast. Für die Analyse wird die CSV-Datei mit Hilfe von Pandas eingelesen. Zunächst werden alle Zeilen mit nicht vorhandenen Werten über die Leistungsverläufe entfernt. Daraus ergeben sich insgesamt 1287 Ladevorgänge. Ladevorgänge mit einer Standzeit unter 10min werden als fehlerhafte Anschlussversuche bewertet und entfernt. Zusätzlich werden alle fehlerhaften und leeren Messungen entfernt.

Im nächsten Schritt wird die Korrelation überprüft. Die folgende graphische Darstellung ist eine Veranschaulichung, wie stark die Variablen paarweise miteinander korreliert sind.

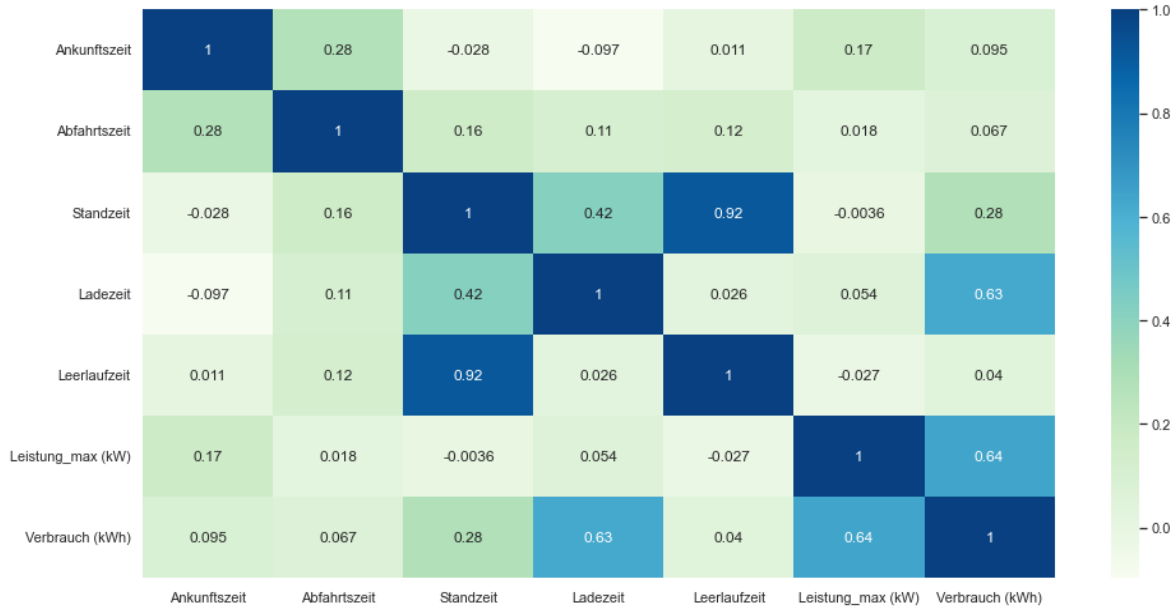


Abbildung 5.3: Korrelationsmatrix für alle Merkmale

Zwischen Leerlaufzeit und Standzeit besteht eine hohe positive Korrelation und kann beim Clustern mit Distanzmaßen einen wesentlichen Einfluss haben. Die Möglichkeit besteht, dass die zwei Merkmale bei der Clusteranalyse mehr gewichtet werden. Es kann nicht genau gesagt werden, wie sich die Korrelation im mehrdimensionalen Raum wirklich verhält. Um möglichen Einflüssen vorzubeugen, wird die Standzeit mit der Ladezeit zusammengefasst. Daraus wird ein neues Merkmal mit der Bezeichnung „Ladeanteil“ erzeugt. Der Ladeanteil ergibt sich aus der folgenden Definition:

$$Ladeanteil = \frac{Ladezeit}{Standzeit} * 100 \tag{5.1}$$

Durch den Ladeanteil lassen sich gleichzeitig die Ladezeit und Leerlaufzeit erklären. Ein Ladeanteil von 100 % bedeutet, dass keine Leerlaufzeit während des Aufenthalts an der Ladesäule entstand. Die Einführung vom Ladeanteil ermöglicht eine Reduzierung der Merkmale auf die Anzahl von Fünf und somit auch eine Reduzierung der Dimensionen. Die Einführung vom Ladeanteil auf die Clusterbildung wird bei der experimentellen Umsetzung näher Untersucht.

Im nächsten Schritt werden die Merkmale visualisiert. Dadurch soll ein Überblick über die Verteilung der Daten und über wichtige Eigenschaften, die für die spätere Interpretation

genutzt werden können, geschaffen werden. Hierbei ist wichtig, dass eine zweidimensionale Darstellung keine eindeutige Aussage über den Zusammenhang im mehrdimensionalen Raum gibt. In diesem Fall ist die Dimension jedoch gering und es können Annahmen getroffen werden, die für die Auswahl eines Clusterverfahrens helfen.

Die nächste Abbildung 5.4 zeigt die Ankunftszeit und Abfahrzeit der Ladevorgänge:

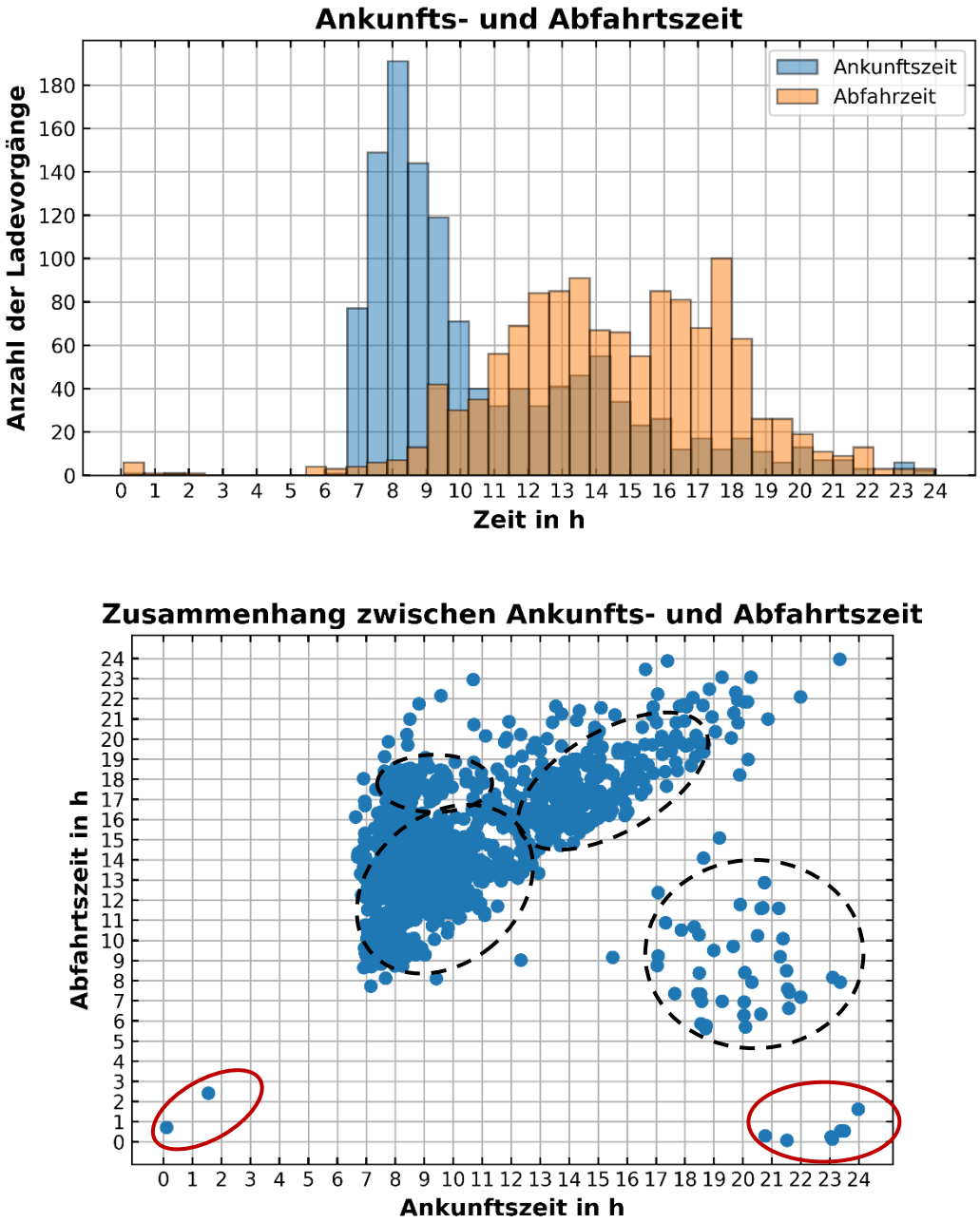


Abbildung 5.4: Ankunfts- und Abfahrtszeiten: Häufigkeitsverteilung (oben) und Streudiagramm (unten)

Die Häufigkeitsverteilung zeigt, dass in der Zeit zwischen ca. 7 Uhr und 10 Uhr die meisten Ladevorgänge beginnen, wobei der höchste Peak um 8 Uhr ist. Zwischen 11 Uhr und 16 Uhr reduzierten sich die Ankünfte auf ein vergleichsweise konstantes Niveau. Ab 16 Uhr weisen die Ankünfte ihr Minimum auf. Die Abfahrzeiten verteilen sich hingegen über einen weiten Zeitraum, wobei von ca. 12 - 14 Uhr und von ca. 16 - 18 Uhr die häufigsten Abfahrten standfinden. Das untere Streudiagramm zeigt vereinzelte Ladevorgänge (rot umkreist), die sich durch ihre Uhrzeiten von allen weiteren stark unterscheiden. Diese besitzen keinen repräsentativen Informationsgehalt für den gesamten Datensatz und werden für die weitere Clusteranalyse entfernt. Außerdem sind vier Gruppierungen anhand der Verteilung wahrnehmbar (gestrichelt umkreist), die sich durch Form oder Dichte unterscheiden.

In der Abbildung 5.5 ist als erstes die Häufigkeitsverteilung der maximalen Leistung für alle Ladevorgänge abgebildet. Die größten Ausprägungen befinden sich bei 3.7 kW und 7.4 kW. Weitaus geringer fallen Leistungen im Bereich von ca. 11 kW und 20 kW aus. Anzumerken ist, dass die Leistung in Abhängigkeit des Batteriezustandes variieren kann. Zusätzlich können auch Ladevorgänge vorkommen, die ausschließlich im Bereich des Leistungsabfalls (vgl. Abbildung 2.1) gemacht wurden. Dadurch kann die Leistung je nach Messzeitpunkt im gesamten Bereich unterhalb der maximalen Leistung liegen. Der Zusammenhang zwischen Gesamtverbrauch und max. Leistung anhand des Streudiagramms zeigt, dass Ladevorgänge mit geringer Leistung einen geringeren Wertebereich im Gesamtverbrauch besitzen. Mit zunehmender Leistung steigt der Wertebereich. Das Verhalten zeigt, dass Fahrzeuge mit geringerer Leistung im Durchschnitt eine geringere Batteriekapazität (Gesamtverbrauch) besitzen. Der Verbrauch hängt zudem stark von der Ladezeit ab. Die Datenverteilung weist auf vier Gruppierungen der Daten mit unterschiedlichen Formen und Dichten hin. Zusätzlich weisen mehrere Datenpunkte eine deutliche Streuung auf, die als Ausreißer gewertet werden können. Die Daten spiegeln dennoch reale Ladevorgänge wider, weshalb eine Entfernung von Daten kritisch betrachtet werden muss. Zudem ist unklar, wie sich die Daten im mehrdimensionalen Raum verhalten. Unter diesem Aspekt und der geringen Datenmenge, werden die Daten nicht entfernt. Die gleiche Begründung gilt auch für die Daten des Streudiagramms in Abbildung 5.5.

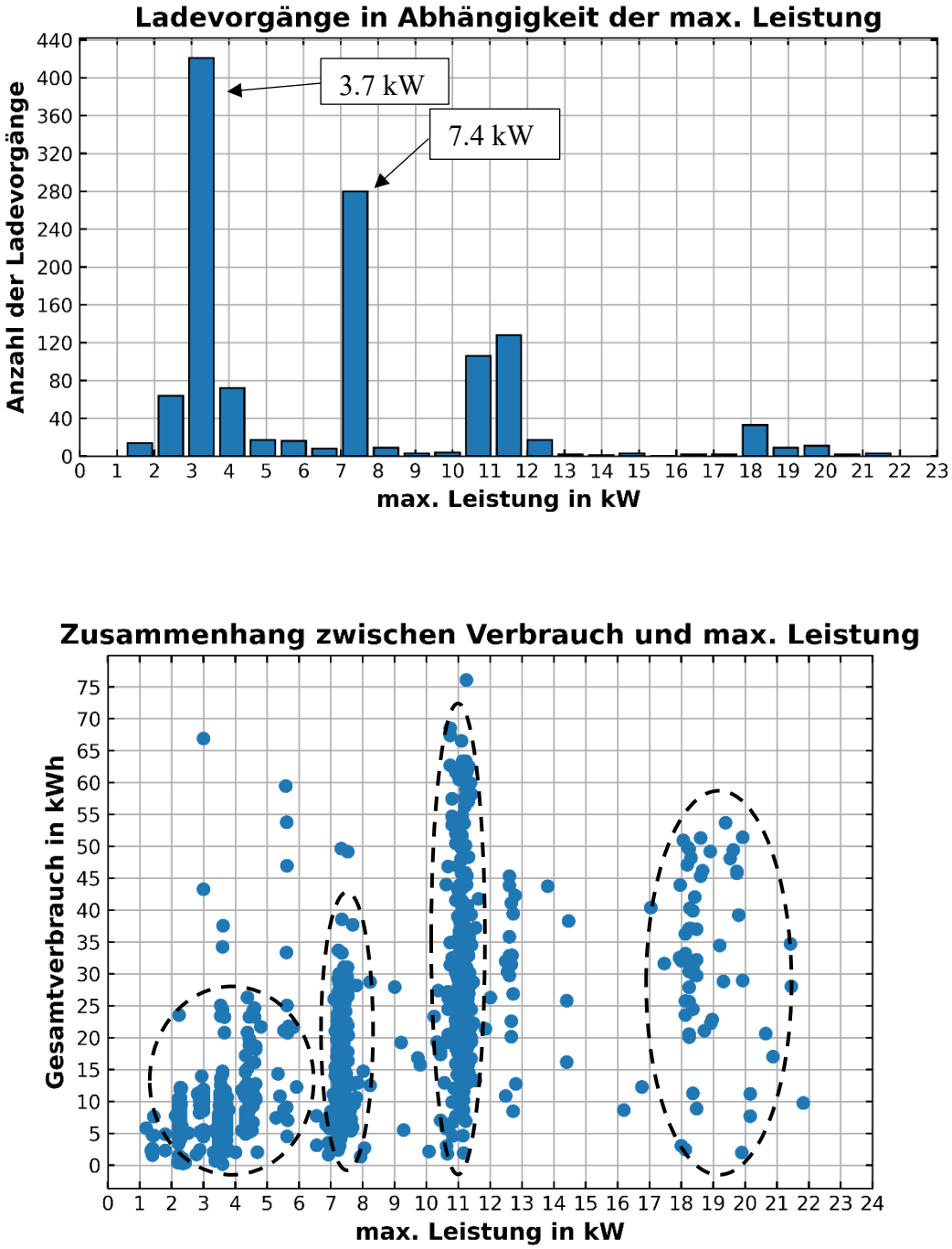


Abbildung 5.5: Häufigkeitsverteilung von max. Ladeleistung (links) und Zusammenhang zwischen Gesamtverbrauch und max. Leistung (rechts)

Die Häufigkeitsverteilung in Abbildung 5.6 zeigt, dass die 25 % der Ladevorgänge einen Ladeanteil von 100 % besitzen. Die übrigen Ladevorgänge besitzen eine gleichmäßige Verteilung über den gesamten Wertebereich unterhalb von 100 %.

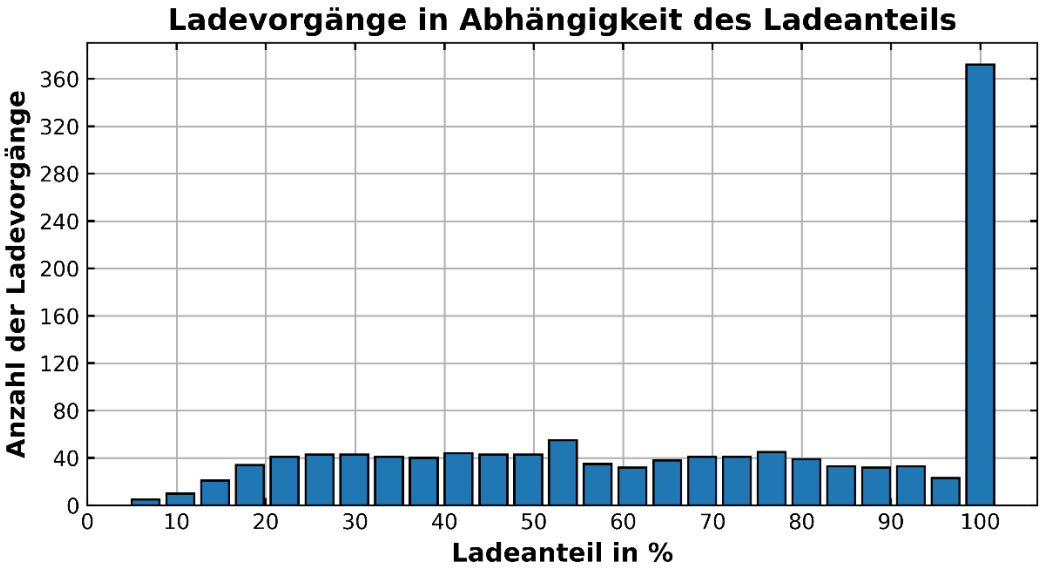


Abbildung 5.6: Häufigkeitsverteilung vom Ladeanteil

In Abbildung 5.7 ist die Häufigkeitsverteilung der Ladevorgänge in Abhängigkeit der Wochentage zu erkennen. Die Anzahl der Ladevorgänge ist am Wochenende deutlich geringer als unter der Woche. Das höchste Aufkommen ist am Mittwoch, dicht gefolgt vom Dienstag, Donnerstag und Montag. Der Freitag besitzt die wenigsten Ladevorgänge unter der Woche.

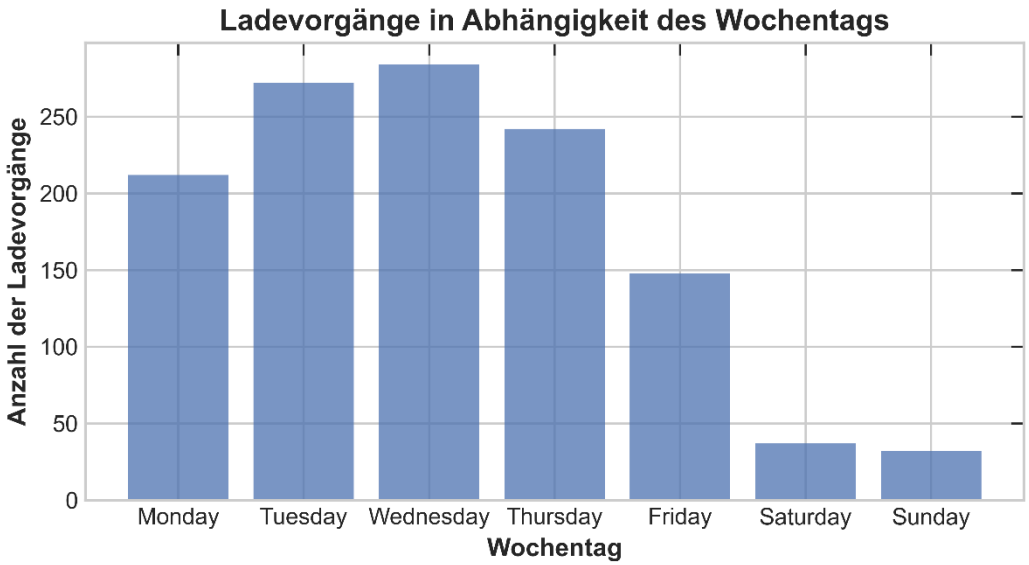


Abbildung 5.7: Häufigkeitsverteilung der Ladevorgänge in Abhängigkeit der Wochentage

Für die Clusteranalyse wird der Wochentag nicht weiter berücksichtig. Die meisten Clusterverfahren können keine Mischung aus metrischen- und kategorischen Merkmalen

verwenden. Außerdem besteht ein großes Ungleichgewicht zwischen Wochenende und den Tagen unter der Woche, die als Ausreißer gewertet werden könnten. Die Zuweisung der Clusterergebnisse auf die Wochentage wird im Nachhinein vorgenommen

Im letzten Schritt wird der gesamte Datensatz standardisiert. Dadurch wird eine unterschiedliche Gewichtung zwischen Merkmalen mit unterschiedlichen Einheiten verhindert. Dazu wird die im Kapitel 2.3.1 vorgeschlagen Formel 2.1 verwendet.

5.2 Auswahl des Clusterverfahrens

In diesem Kapitel wird die Entscheidung getroffen, welches Clusterverfahren verwendet wird. Die Analyse der Daten im vorigen Unterkapitel 5.1 hat gezeigt, dass die Datenverteilung auf mehrere Cluster hindeutet, die sich durch unterschiedliche Formen und Dichteverhältnisse unterscheiden. Die maximale Leistung deutet auf elliptische Clusterformen hin, wohingegen Ankunfts- und Abfahrzeit näherungsweise sphärische Clusterformen zeigen. Die Visualisierung im zweidimensionalen Raum kann jedoch nicht die Verteilung im mehrdimensionalen Raum erklären. Es könnten unbekannte Cluster vorhanden sein, die sich durch Form und Größe unterscheiden. Außerdem sind die Daten durch kleinere Gruppierungen mit geringer Datendichte gekennzeichnet, die wenig Informationsgehalt gegenüber anderen Dateigruppierungen besitzen. Diese könnten die Clusterbildung von Daten mit mehr Informationsgehalt negativ beeinflussen. Durch den Vergleich mit möglichen Kriterien für die Auswahl eines Clusterverfahrens aus Kapitel 4.2 und den Eigenschaften der aufgelisteten Clusterverfahren in Kapitel 2.3.2 wird ersichtlich, dass kein eindeutiges Clusterverfahren für diesen Anwendungsfall besteht. Aus diesem Grund werden mehrere Clusterverfahren ausgewählt, die sich grundlegend voneinander unterscheiden. Für die Implementierung wird die Bibliothek von Scikit-Learn mit der Programmiersprache Python genutzt, weshalb die Verfügbarkeit für ein Clusterverfahren berücksichtigt werden muss. Die Bibliothek verfügt zudem über die notwendigen Validierungsmethoden und weitere nützliche Hilfswerkzeuge für die Datenvorverarbeitung. Für die Clusteranalyse werden K-Means, DBSCAN und GMM verwendet, die zuvor im Kapitel 2.3.3, 2.3.4 und 2.3.5 näher erläutert wurden. Die wichtigsten Unterschiede der Clusteralgorithmen sind in Tabelle 5.1 zusammengefasst.

Tabelle 5.1: Gegenüberstellung von K-Means, DBSCAN und GMM

Eigenschaften	K-Means	DBSCAN	GMM
Clusterform	sphärisch, gleiche Größe	alle, variierende Größe	elliptisch, variierende Größe
Clusterart	hart	hart	weich/hart
Ausreißer	empfindlich	Unempfindlich, werden markiert	-
Parameter	k	ε, minPts	$k,\ (\pi_k, \mu_k, \Sigma_k)$
Validierung	Ellenbogen- und Silhouetten-Methode	DBCV	BIC, AIC

Der K-Means zeichnet sich durch die einfache Berechnung der Cluster aus, indem die Datenpunkte den jeweiligen Zentroiden zugeteilt werden. Die Zentroiden stellen den Mittelwert der Cluster dar und können für die Interpretation herangezogen werden. Der Nachteil ist, dass vorzugsweise sphärische Cluster mit gleicher Größe gebildet werden. DBSCAN kann hingegen Cluster mit unterschiedlichen Formen erzeugen und berücksichtigt Ausreißer. Die Anzahl der Cluster muss nicht vorgegeben werden. Dennoch müssen zwei Hyperparameter im Vorfeld initialisiert werden, die bei Daten mit unterschiedlichen Dichteverhältnissen schwer zu bestimmen sind. Das GMM bildet elliptische Cluster, die in Form, Größe und Ausrichtung variieren können. Die Cluster können zudem überlappen, wobei die Datenpunkte, unter Berücksichtig einer Wahrscheinlichkeit, den jeweiligen Clustern zugeordnet werden können. Die Berechnung ist jedoch wesentlich komplexer und der EM-Algorithmus kann bei jeder Durchführung von GMM unterschiedliche Anpassungen der Parameter hervorbringen. Das führt verstärkt zu unterschiedlichen Clusterergebnissen. Das Hauptproblem besteht darin, dass die Clusterformen im mehrdimensionalen Raum unbekannt sind. Die drei ausgewählten Algorithmen können zusammen alle möglichen Formen anhand der Datenverteilung erzeugen.

6 Experimentelle Umsetzung

In diesem Kapitel werden die zuvor ausgewählten Clusterverfahren für das Clustern der Ladesäulendaten verwendet und mit den in Kapitel 2.3.6 vorgestellten Validierungsmethoden validiert. Zusätzlich wird die Wirkungsweise der Verfahren anhand von zweidimensionalen Beispielen visualisiert und interpretiert. Die dabei gewonnen Erkenntnisse werden für die Interpretation der Clusteranalyse für die mehrdimensionalen Ladesäulendaten genutzt.

6.1 Clusteranalyse mit K-Means

Die Anzahl der Cluster wird im ersten Schritt mit der Ellenbogen-Methode (vgl. Kapitel 2.3.6) visualisiert und kann der Abbildung 6.1 entnommen werden.

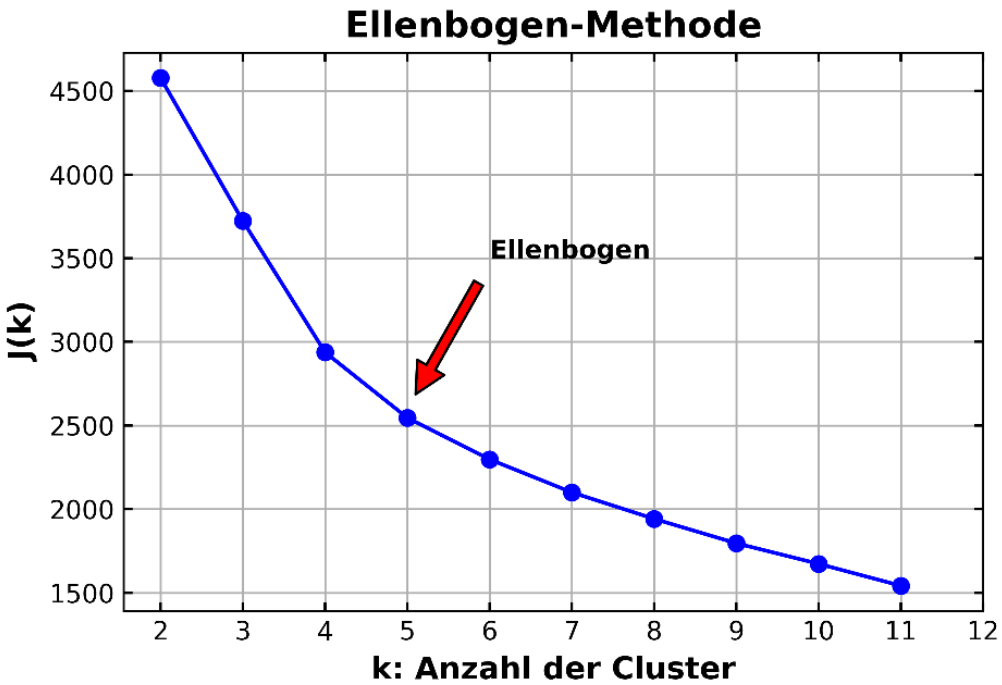


Abbildung 6.1: Ermittlung der Anzahl von Cluster mit Ellenbogen-Methode

Zu Beginn fällt die Summe der quadratischen Abweichung für alle Cluster stark ab. Ab der Anzahl von fünf Cluster stellt sich ein nahezu linearer Verlauf ein und die Abweichung minimiert sich um wesentlich geringere Beträge. Das bedeutet, dass die Zentroiden der Cluster bereits nahe aneinander liegen und eine Erhöhung der Clusteranzahl zu keiner bedeutsamen Aufteilung führt. Der Ellenbogen ist in diesem Fall nicht deutlich ausgeprägt, weshalb nur eine grobe Einschätzung über die Anzahl der Cluster gemacht werden kann.

Aus diesem Grund wird zusätzlich die Silhouetten-Methode verwendet. In Tabelle 6.1 sind die Ergebnisse aufgelistet. Zudem wird der Einfluss zwischen fünf und sieben Merkmalen untersucht.

Das Ergebnis zeigt, dass der höchste Mittelwert der Silhouetten (S) mit fünf Cluster erreicht wird (S = 0.3075 mit fünf Merkmalen). Bei sieben Merkmalen liegt der höchste Mittelwert bei zwei Cluster mit S = 0.2864. Das zeigt, dass die Reduzierung der Merkmale mit hoher Korrelation dafür gesorgt hat, dass die Anzahl der Cluster größer wird und die Trennung und Kompaktheit zugenommen hat. Dennoch ist der maximale Wert sehr gering und liegt betragsmäßig nahe an allen anderen Ergebnissen.

Tabelle 6.1: Ermittlung der Anzahl von Cluster mit Silhouetten-Methode

Anzahl der Cluster	**Silhouetten-Mittelwert**	
	5 Merkmale	**7 Merkmale**
2	0.2897	**0.2864**
3	0.2746	0.2537
4	0.2909	0.2692
5	**0.3075**	0.2861
6	0.3029	0.2584
7	0.2985	0.2640
8	0.2817	0.2635
9	0.2702	0.2660
10	0.2903	0.2741
11	0.2861	0.2760

Die Ergebnisse deuten darauf hin, dass innerhalb der Daten keine eindeutigen Cluster existieren bzw. das keine eindeutige Kompaktheit und Trennung der Cluster von K-Means erreicht wird. In Tabelle 6.2 sind die Datenpunkte der Merkmale aufgelistet, die die geringste Distanz zu den jeweiligen Zentroiden aufweisen.

Tabelle 6.2: Datenpunkte der Merkmale mit der geringsten Distanz zu den Zentroiden.

	Ankunftszeit	Abfahrtszeit	Ladeanteil	Leistung_max (kW)	Verbrauch (kWh)	Cluster
0	8.638889	12.081389	91.971274	3.66	10.42	Eins
1	8.618333	15.420278	36.741945	4.56	9.32	Zwei
2	15.991944	17.931111	77.882825	7.3965	10.56	Drei
3	10.979444	14.845556	86.79408	11.3418	35.761	Vier
4	17.870278	10.515	24.028304	10.74	34.91	Fünf

Die genaue Interpretation der Ergebnisse folgt in Kapitel 7.

Des Weiteren wird der K-Means mit Hilfe von zweidimensionalen Beispielen untersucht, um einen visuellen Vergleich zwischen allen Clusterverfahren zu erhalten und die Wirkungsweise besser zu verdeutlichen. Dazu wird das Streudiagramm von Ankunfts- und Abfahrtszeit (vgl. Abbildung 5.4), sowie von max. Leistung und Gesamtverbrauch (vgl. Abbildung 5.5) genutzt. Das Schaubild in Abbildung 6.2 zeigt die Aufteilung von Ankunfts- und Abfahrtszeit in drei Cluster.

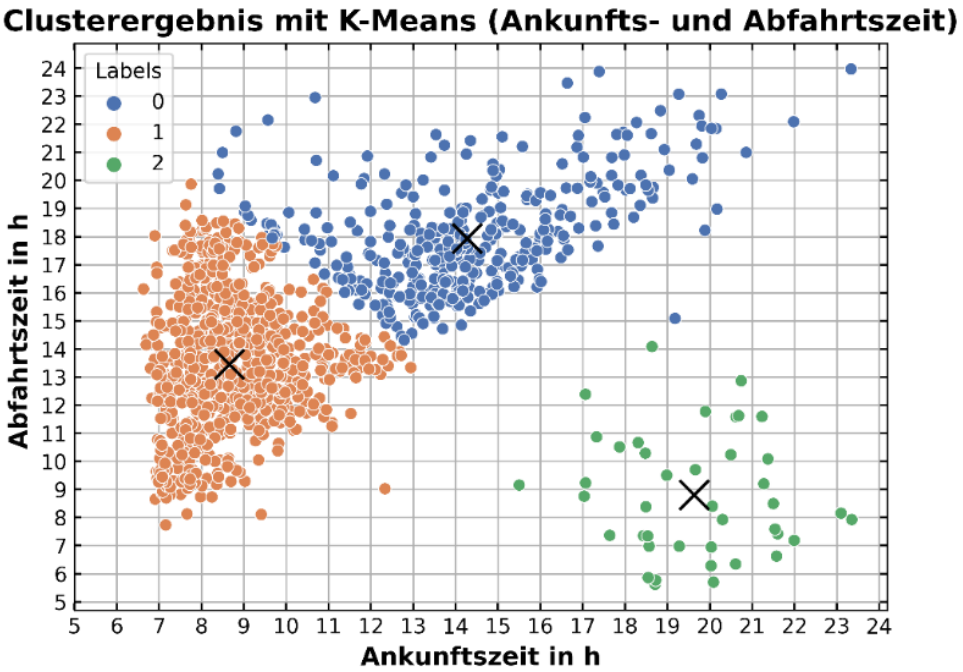


Abbildung 6.2: Clusterergebnis mit K-Means (Ankunfts- und Abfahrtszeit)

Die Kreuze symbolisieren die Zentroiden der Cluster. Wichtig hierbei ist, dass die Anzahl der Cluster anhand der Silhouette-Methode bestimmt wurden und nicht visuell. Dadurch werden gleiche Bedingungen, wie bei einer mehrdimensionalen Clusteranalyse, geschaffen.

Dieses Vorgehen wird auch bei DBSCAN und GMM durchgeführt. Zwischen dem blauen und orangenen Cluster befindet sich ein Bereich mit einer geringen Anzahl von Datenpunkten, welcher auch die Trennung der beiden Cluster zeigt. Betrachtet man die Datenpunkte an der Grenze zwischen beiden Clustern in diesem Bereich, besitzen alle eine nahezu gleiche Distanz zu den jeweiligen Zentroiden. Das lässt sich dadurch erklären, dass zwischen den Datenpunkten der jeweiligen Cluster und dessen Zentroiden die geringste euklidische Distanz besteht. Besitzt ein Datenpunkt eine identisch geringste Distanz zu mehreren Zentroiden, hängt die Zuweisung von der Reihenfolge der Berechnungen von K-Means ab.

Die Abbildung 6.3 zeigt das Clusterergebnis von max. Leistung und Gesamtverbrauch. Die Datenpunkte wurden in zwei Cluster aufgeteilt. Das Ergebnis verdeutlich sehr gut, dass K-Means bei Datengruppierungen mit unterschiedlicher Größe und Form eine beschränkte Leistungsfähigkeit besitzt.

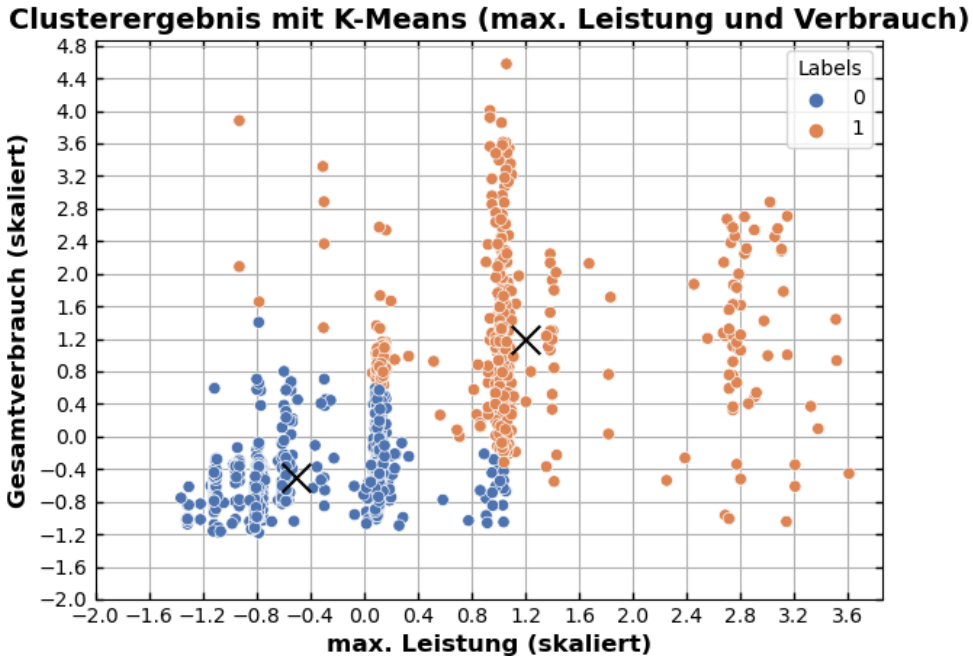


Abbildung 6.3: Clusterergebnis mit K-Means (max. Leistung und Gesamtverbrauch)

6.2 Clusteranalyse mit DBSCAN

Zu Beginn wird mit Hilfe der vorgestellten Methode in Kapitel 2.3.4 ein möglicher Bereich für ε ermittelt. Durch die fünf Merkmale (fünf Dimensionen) wird minPts auf zehn festgelegt (minPts = 2*D). Wie in der folgenden Abbildung 6.4 dargestellt, besteht ein ausgeprägter Ellenbogen im Wertebereich der minimalen Distanz zwischen etwa 0.4 und 1.4, wobei die Änderung exponentiell zunimmt. Dieser Wertebereich wird für die weiterführende Clusterbildung verwendet.

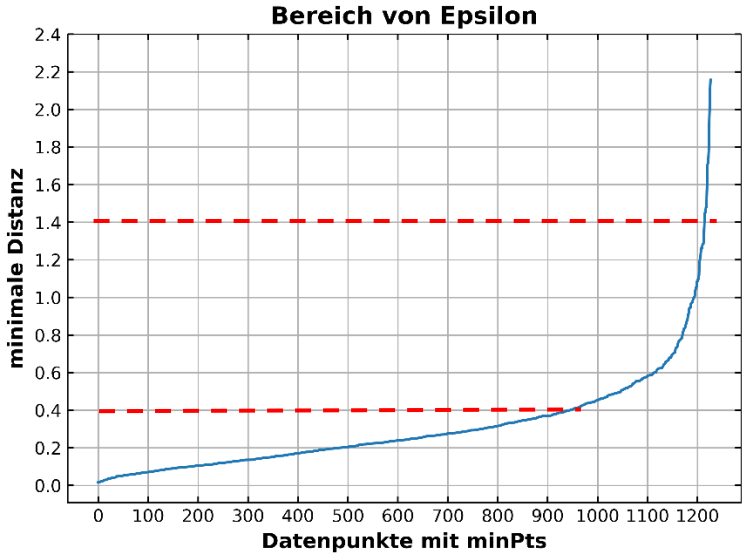


Abbildung 6.4: Ermittlung eines möglichen Wertebereichs für Epsilon

Die Clusteranalyse mit DBSCAN wird somit im Intervall von $0{,}4 \leq \varepsilon \leq 1{,}4$ mit einer ausgewählten Schrittweite von 0.05 durchgeführt und mit DBCV validiert. Die Schrittweite wurde anhand von Vorversuchen ausgewählt. Diese bietet einen Kompromiss zwischen einer genauen Abtastung der Datenpunkte und der intensiven Berechnungsdauer von DBCV.

Durch die Kombination konnten keine sinnvollen Cluster anhand der Daten gebildet werden. Alle Kombinationen von ε mit minPts führten zu einem Cluster mit Rauschpunkten oder zu extrem vielen Clustern. Eine mögliche Ursache dafür ist, dass die Dichte zu stark innerhalb der Daten variiert und die globalen Hyperparamter keine allgemeingültige Aussage über eine unterschiedliche Dichteverteilung machen können. Um die Wirkungsweise von DBSCAN dennoch zu untersuchen, werden ebenfalls die zweidimensionalen Beispiele verwendet, wie bei K-Means. Die folgende Abbildung 6.5 zeigt das Clusterergebnis von Ankunft- und Abfahrtszeit.

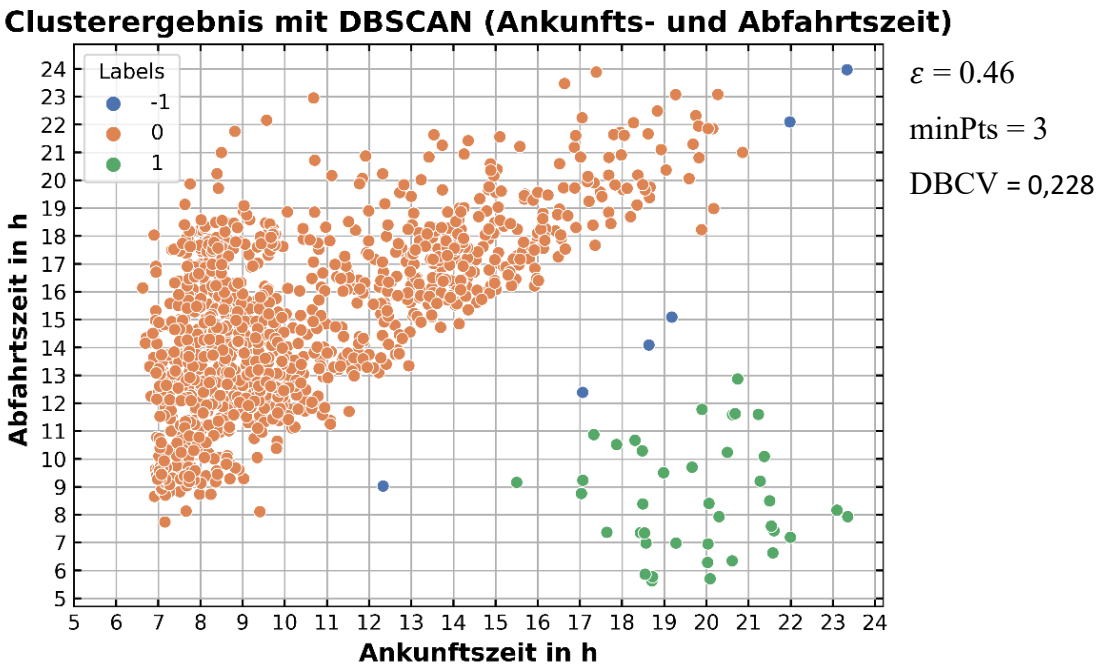


Abbildung 6.5: Clusterergebnis mit DBSCAN (Ankunfts- und Abfahrtszeit)

DBSCAN hat zwei Cluster erzeugt, die sich durch ihre Form und Größe unterscheiden. Außerdem wurde eine geringe Anzahl von Rauschpunkten erkannt, die eine deutliche Abweichung zu den Clustern aufweisen. Im Vergleich zu K-Means wird deutlich, dass die schwankende Dichte im orangenen Cluster dazu geführt hat, dass keine zwei Cluster gefunden wurden.

Das Ergebnis von max. Leistung und Gesamtverbrauch zeigt sechs gefundene Cluster (siehe Abbildung 6.6). Die zwei eindeutig getrennten Datengruppierungen im Bereich zwischen 1kW und 8kW wurden in einem Cluster (orange) zusammengefasst. Das demonstriert die limitierte Fähigkeit von DBSCAN bei unterschiedlichen Dichteverhältnissen. Dennoch zeigen die Cluster, dass DBSCAN bei unterschiedlichen Formen und Größen der Datengruppierungen gute Ergebnisse liefert. Zudem sind deutlich abweichende Datenpunkte zuverlässig als Rauschpunkte gelabelt wurden. Neben den großen Clustern wurden drei kleine Cluster (rot, pink und braun) erkannt, die eine minimale Anzahl von Datenpunkten aufweisen und genau den minimalen Anforderungen der Hyperparameter entsprechen. Das zeigt, dass bei einer hohen Anzahl von Clustern, dennoch Cluster mit hohem Informationsgehalt existieren können. Besonders bei mehrdimensionalen Daten, die nicht visualisierbar sind, sollte bei der Beurteilung auf die Anzahl der Datenpunkte je Cluster geachtet werden.

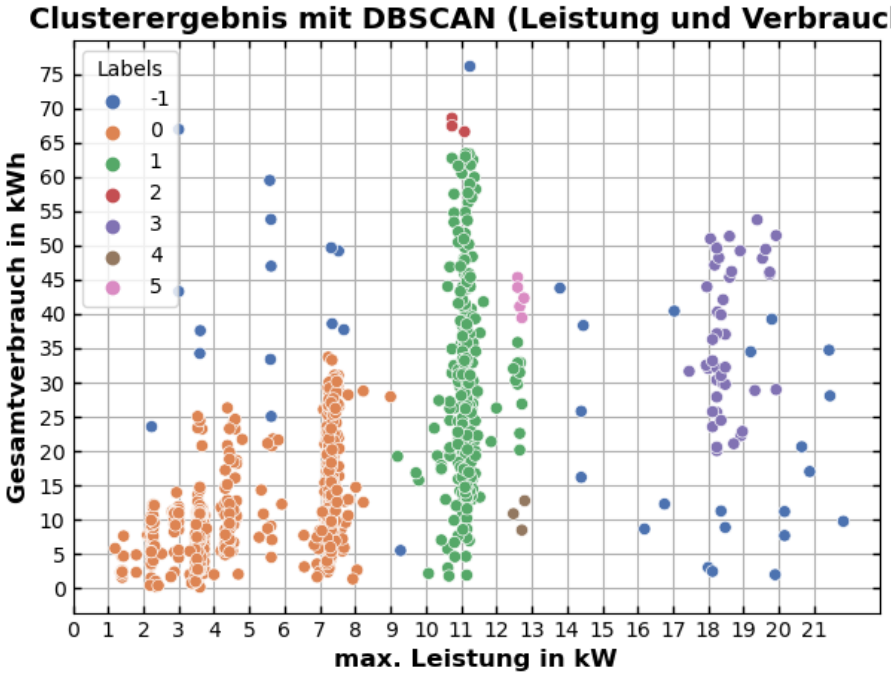


Abbildung 6.6: Clusterergebnis mit DBSCAN (max. Leistung und Gesamtverbrauch)

Die Clusterergebnisse von den zweidimensionalen Beispielen geben ein mögliche Ursache für die minderwertigen Clusterergebnisse im mehrdimensionalen Raum mit fünf Merkmalen. Durch die variierende Datendichte können keine Cluster mit unterschiedlichen Charakteristiken von DBSCAN gebildet werden und somit keine Rückschlüsse auf eindeutige Nutzertypen, die sich durch die Ladevorgänge unterscheiden. Abschließend kann gesagt werden, dass DBSCAN für diesen Datensatz ungeeignet scheint. Das bedeutet jedoch nicht, dass keine eindeutigen Cluster im Datensatz vorhanden sind. Andere Clustermethoden sind für diesen Anwendungsfall möglicherweise besser geeignet. Im weiteren Verlauf wird nur noch ein Vergleich zwischen K-Means und GMM durchgeführt.

6.3 Clusteranalyse mit GMM

Zu Beginn wird die Anzahl der Cluster vom GMM bestimmt. Dazu wird mit aufsteigendem k der BIC und AIC bestimmt. Durch die zufällige Initialisierung und der variierenden Anpassung durch den EM-Algorithmus sind mehrere Initialisierungen notwendig, damit ein eindeutiges Minimum für das optimale k abgelesen werden kann. In Abbildung 6.7 ist der Verlauf für AIC und BIC dargestellt. Bis zur Anzahl von sechs Clustern fallen die Werte von BIC und AIC signifikant ab und erreichen ein Plateau, wobei keine deutliche Minimierung von BIC und AIC stattfindet. Das Vorgehen ähnelt der Ellenbogen-Methode von K-Means. Aus diesem Grund wird die Anzahl der Cluster auf sechs festgelegt. Somit wird ein Kompromiss zwischen Clusteranzahl und Interpretierbarkeit gemacht. Im Gegensatz zu K-Means wurde ein zusätzlicher Cluster gefunden.

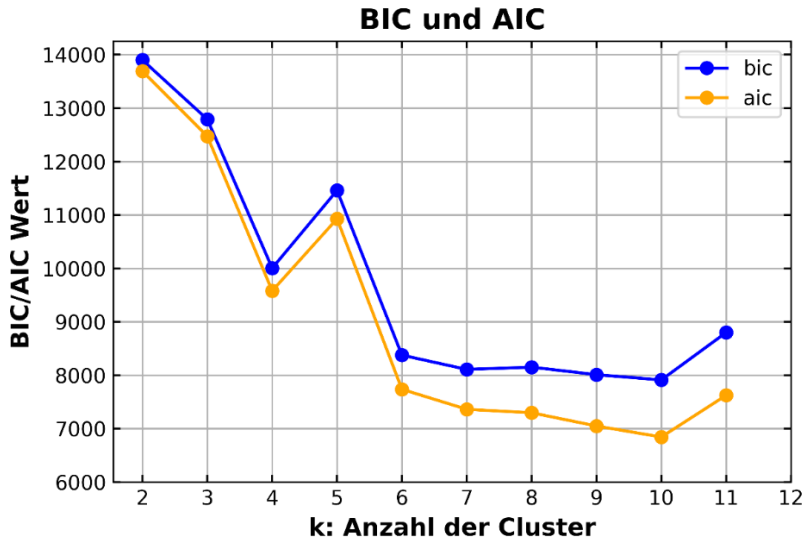


Abbildung 6.7: Ermittlung von Anzahl der Cluster mit AIC und BIC

Nach der Ermittlung der Clusteranzahl wird das GMM für die Clusterbildung der Daten genutzt. Dabei wurde festgestellt, dass die Clustergrößen nach jeder Initialisierung hinsichtlich ihrer Datenmenge deutlich variieren. Dadurch kann nicht sichergestellt werden, welches der Ergebnisse eine optimale Anpassung an die Daten darstellt. Aus diesem Grund werden mehrere Initialsierungen durchgeführt, wobei jedes Mal das AIC berechnet wird. Im Anschluss wird das Ergebnis mit dem geringsten AIC-Wert ausgewählt. Der Grund für die Wahl von AIC ist, dass die Werte sensibler auf die Ergebnisse reagieren und eine höhere Tendenz zur Überanpassung besteht. Vergleicht man die Verläufe von AIC und BIC in Abbildung 6.7, regiert das AIC im Bereich von k = 6 bis k = 11 betragsmäßig höher. Das BIC zeigt hingegen nur eine minimale Änderung. Der Unterschied zur Bestimmung von k

ist, dass die Anzahl der Parameter durch die festgelegte Clusteranzahl für jede Initialisierung gleich ist. Das bedeutet, dass nur der Einfluss der Log-Likelihood-Funktion in die Berechnung von AIC einfließt. Dementsprechend wird nur die Anpassungsgüte des GMM vom AIC berücksichtigt.

Die genau Interpretation der Ergebnisse erfolgt in Kapitel 7.

In Abbildung 6.8 ist das Clusterergebnis von GMM von Ankunfts- und Abfahrzeit dargestellt. Im Gegensatz zu K-Mean wurde ein weiterer Cluster gebildet. Das Ergebnis zeigt drei näherungsweise elliptische Clusterformen (rot, blau und orange), die für den GMM typisch sind.

Die Bildung von elliptischen Formen bietet somit den Vorteil, dass im Vergleich zu K-Mean die Aufteilung in kompaktere Cluster ermöglicht wird. Der grüne Cluster zeigt außerdem die Fähigkeit zur Bildung von kreisförmigen Clustern.

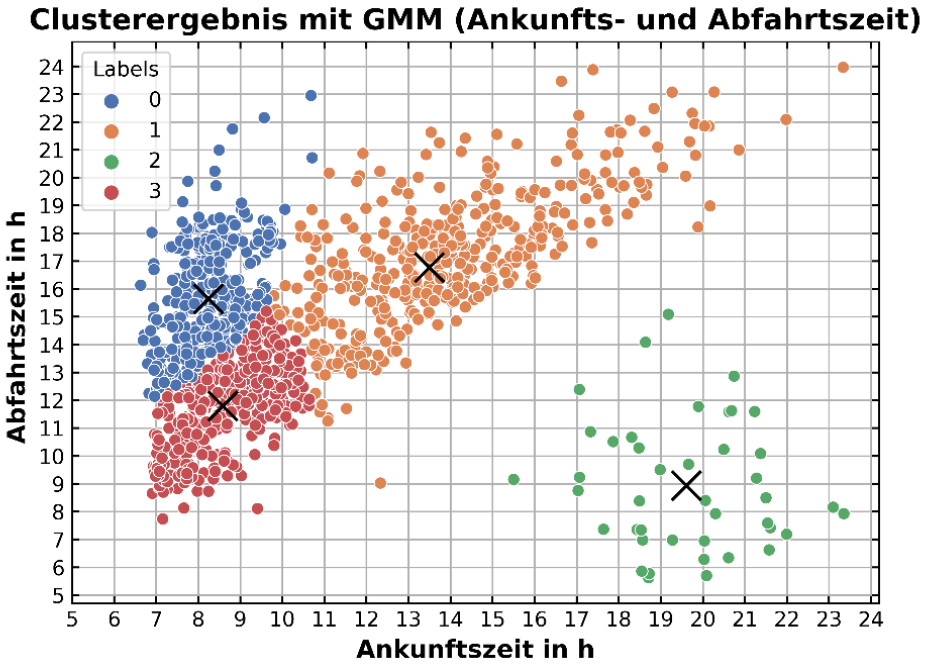


Abbildung 6.8: Clusterergebnis mit GMM (Ankunfts- und Abfahrtszeit)

Das GMM hat sechs Cluster anhand von max. Leistung und Gesamtverbrauch gebildet (siehe Abbildung 6.9).

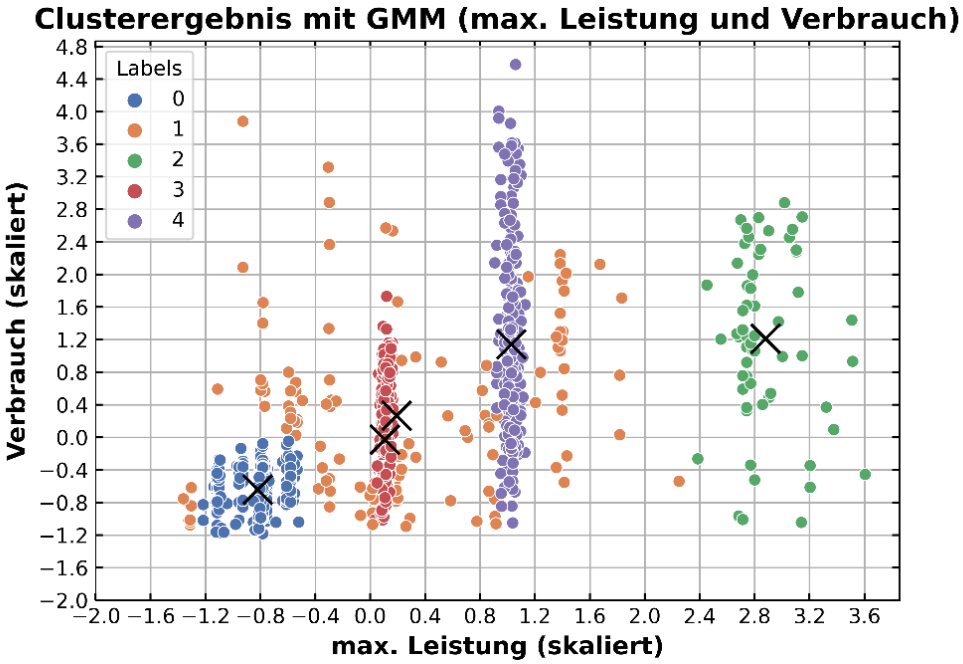


Abbildung 6.9: Clusterergebnis von GMM (max. Leistung und Gesamtverbrauch)

Im Vergleich zu K-Means und DBSCAN konnte hier das beste Ergebnis erzielt werden. Der GMM hat die ersichtlichen Gruppierungen der Datenpunkte in kompakte und eindeutig getrennte Cluster unterteilt. Zudem wurde ein globaler Cluster (orange) erzeugt. Dieser Cluster beinhaltet zum einen die ersichtlichen Ausreißer und zum anderen die Datenpunkte, die zwischen den anderen Clustern liegen. Damit wird ersichtlich, dass GMM die Fähigkeit besitzt mit Ausreißern umzugehen und Datenpunkte zu identifizieren, die keine eindeutige Zuordnung besitzen. Der Vergleicht zwischen dem roten bzw. lila Cluster und dem blauen zeigt, dass die Cluster verschiedene Ausrichtungen besitzen können.

Die zweidimensionalen Clusterergebnisse bestätigen alle aufgeführten Vorteile, die in Kapitel 2.3.2 für modellbasierte Clusterverfahren beschrieben wurden. Außerdem bestätigen die gefundenen Cluster, dass die Anzahl von sechs Clustern für die eigentliche mehrdimensionale Clusterbildung ein plausibler Wert ist, der zumindest die gefundenen Cluster im Zweidimensionalen abdeckt.

7 Interpretation und Diskussion

Durch die Ergebnisse der zweidimensionalen Beispiele im vorigen Unterkapitel 6.3, hat das GMM die besten Ergebnisse erzielt. Besonders die Clusterbildung von max. Leistung und Gesamtverbrauch hat gezeigt, dass GMM für diesen Datensatz besser geeignet scheint. Aus diesem Grund und zur Eingrenzung des Aufwands für die folgende Interpretation, wird K-Means nicht weiter behandelt. Das Ziel der folgenden Interpretation besteht darin, die Cluster hinsichtlich ihrer Charakteristiken zu untersuchen und mögliche Nutzertypen zu identifizieren. Durch die Ermittlung von BIC und AIC wurden sechs Cluster für die Initialisierung festlegt. Zusätzlich wurden mehrere Initialisierungen mit Hilfe des AIC verglichen. Das Clusterergebnis mit dem geringsten AIC wird für die folgende Interpretation genutzt.

Zu Beginn werden die Mittelwerte der Merkmale und die Anzahl der gelabelten Datenpunkte von den jeweiligen Clustern untersucht (siehe Tabelle 7.1).

Tabelle 7.1: Anzahl gelabelter Datenpunkte und Mittelwerte der Merkmale für alle Cluster (GMM)

	Cluster	Labels	Ankunftszeit	Abfahrtszeit	Standzeit	Ladeanteil	Leistung_max (kW)	Verbrauch (kWh)
0	Eins	364	9.434802	14.673462	5.238660	57.584236	3.463098	7.876343
1	Zwei	241	8.059995	14.544799	6.484804	44.905412	7.783878	17.701216
2	Drei	187	12.502858	14.905628	2.402770	99.999343	9.734598	21.510765
3	Vier	264	11.906985	16.004753	4.188676	65.515252	9.854762	22.058019
4	Fünf	121	11.050333	13.230450	2.180117	99.997359	3.355408	6.665273
5	Sechs	50	18.945683	10.100272	13.714589	26.574107	8.809100	31.316660

Die daraus ersichtlichen Informationen werden für alle Cluster aufgelistet:

Cluster „Eins“ :

- besitzt 28 % aller Datenpunkte
- durchschnittliche Ankunftszeit um 9:30 Uhr
- durchschnittliche Abfahrtszeit um 14:45 Uhr
- durchschnittliche Standzeit von 5 h mit einer Leerlaufzeit von 40 %
- durchschnittliche max. Ladeleistung von 3.5 kW
- durchschnittlicher Gesamtverbrauch von 8 kWh

Cluster „Zwei“ :

- besitzt 19.6 % der Daten aus allen Ladevorgängen
- durchschnittliche Ankunftszeit um 8:00 Uhr
- durchschnittliche Abfahrtszeit um 14:30 Uhr
- durchschnittliche Standzeit von 6,5 h mit einer Leerlaufzeit von 55 %
- durchschnittliche max. Ladeleistung von 7,8 kW
- durchschnittlicher Gesamtverbrauch von 18 kWh

Cluster „Drei“ :

- besitzt 15,2 % der Daten aus allen Ladevorgängen
- durchschnittliche Ankunftszeit um 12:30 Uhr
- durchschnittliche Abfahrtszeit um 15:00 Uhr
- durchschnittliche Standzeit von 2.4 h mit keiner Leerlaufzeit
- durchschnittliche max. Ladeleistung von 9.7 kW
- durchschnittlicher Gesamtverbrauch von 22 kW

Cluster „Vier“ :

- besitzt 21,5 % der Daten aus allen Ladevorgängen
- durchschnittliche Ankunftszeit um 12:00 Uhr
- durchschnittliche Abfahrtszeit um 16:00 Uhr
- durchschnittliche Standzeit von 4.2 h mit einer Leerlaufzeit von 35 %
- durchschnittliche max. Ladeleistung von 9.9 kW
- durchschnittlicher Gesamtverbrauch von 22 kWh

Cluster „Fünf“ :

- besitzt 10 % der Daten aus allen Ladevorgängen
- durchschnittliche Ankunftszeit um 11:00 Uhr
- durchschnittliche Abfahrtszeit um 13:15 Uhr
- durchschnittliche Standzeit von 2.2 h mit keiner Leerlaufzeit
- durchschnittliche max. Ladeleistung von 3.5 kW
- durchschnittlicher Gesamtverbrauch von 8 kWh

Cluster „Sechs“ :

- besitzt 4 % der Daten aus allen Ladevorgängen
- durchschnittliche Ankunftszeit um 19:00 Uhr
- durchschnittliche Abfahrzeit um 10:00 Uhr
- durchschnittliche Standzeit von 14 h mit einer Leerlaufzeit von 75 %
- durchschnittliche Ladeleistung von 8.8 kW
- durchschnittlicher Gesamtverbrauch von 31 kWh

Im nächsten Schritt wird die Verteilung der Daten im jeweiligen Cluster untersucht. Die Verteilung wird mit Hilfe eines Boxplot dargestellt. Ein Boxplot besteht aus einer Box, die 50% der Daten repräsentiert. Die schwarze horizontale Linie stellt den Median dar, die rote Line den Mittelwert. Der Bereich zwischen dem kleinsten und dem größten Wert, wird durch die anliegenden vertikalen Linien an der Box dargestellt. Datenpunkte, die sich außerhalb der Linien befinden, werden als Ausreißer bezeichnet. Unterhalb und oberhalb des Medians befinden sich 50 % der Daten. Die 50 % werden außerdem durch den oberen bzw. unteren Rand der Box in jeweils 25% aufgeteilt [Ba21].

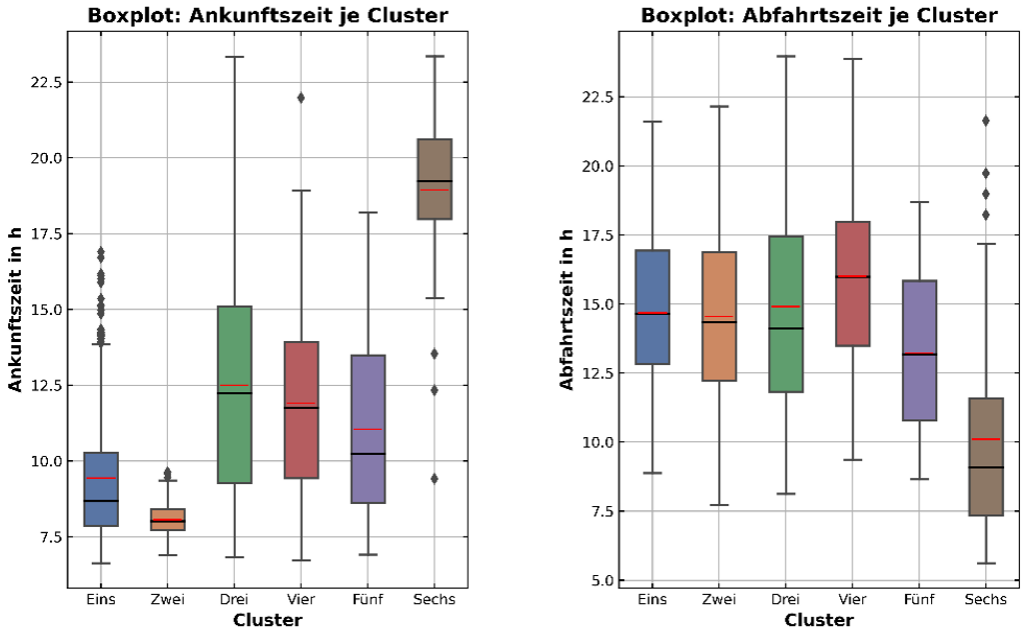


Abbildung 7.1: Boxplot: Ankunftszeit (links) Abfahrtszeit (rechts)

Die Boxplots für Ankunftszeit und Abfahrtszeit in Abbildung 7.1 zeigen die Verteilung der Daten innerhalb der Cluster. Für die Ankunftszeit (linkes Schaubild) weist der zweite Cluster die niedrigste Streuung im Bereich des Morgens auf. Der erste Cluster zeigt eine höhere Streuung auf und tendiert mehr zum Vormittag. Die Cluster Drei und Vier besitzen einen

Median bzw. Mittelwert im Bereich der Mittagszeit. Der Cluster Fünf tendiert zum Vormittag. Der Cluster Sechs zeichnet sich durch die Ankunftszeiten am Abend aus.

Die Cluster Eins, Zwei, Drei für die Abfahrtszeiten (rechtes Schaubild) zeigen eine nahezu gleiche Streuung und besitzen ihren Mittelwert im Bereich des Nachmittags. Der Cluster Vier tendiert hingegen mehr zum Abend und der Cluster Fünf zum Mittag. Der Cluster Sechs deckt größtenteils einen Bereich vom Morgen bis hin zum Vormittag ab.

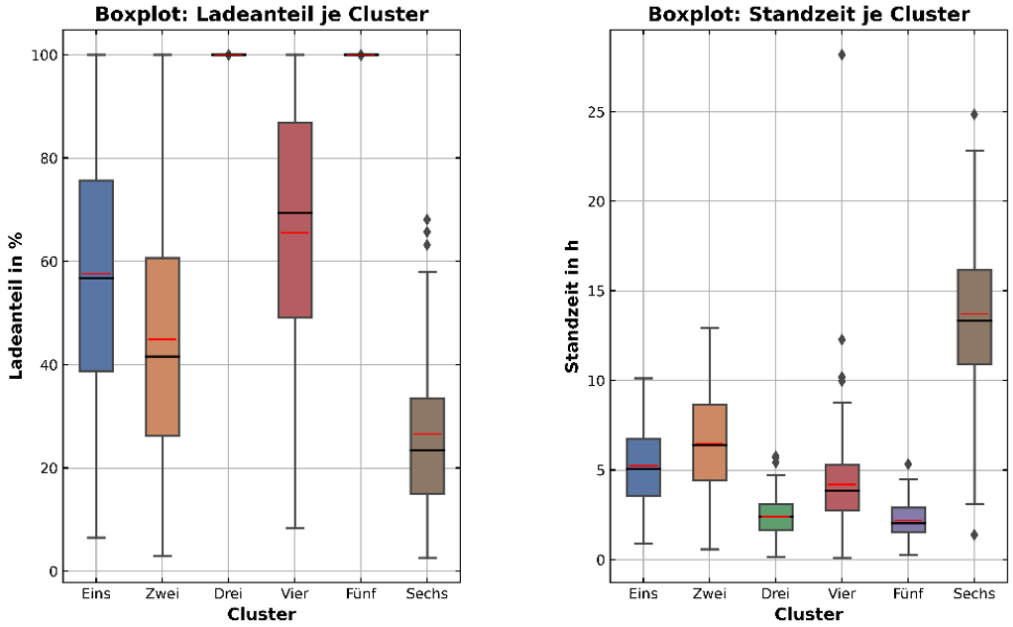


Abbildung 7.2: Boxplot: Ladeanteil (links) Standzeit (rechts)

Die Cluster Drei und Fünf besitzen einen Ladeanteil von 100 % (linkes Schaubild), d.h. die Ladevorgänge besitzen keine Leerlaufzeit. Der Cluster Sechs besitzt den niedrigsten Ladeanteil mit einem Mittelwert von etwa 25 %. Die Cluster Eins und Zwei besitzen eine weite Streuung und tendieren zu ihrem Mittelwert. Der Cluster Vier weist auch eine weite Streuung auf, tendiert jedoch zu einem hohen Ladeanteil.

Der Boxplot von Cluster Sechs für die Standzeit (rechtes Schaubild) zeigt einen deutlichen Unterschied zu den anderen Clustern auf. Der Mittelwert liegt bei ca. 13 h mit einem Maximum bei 24 h.

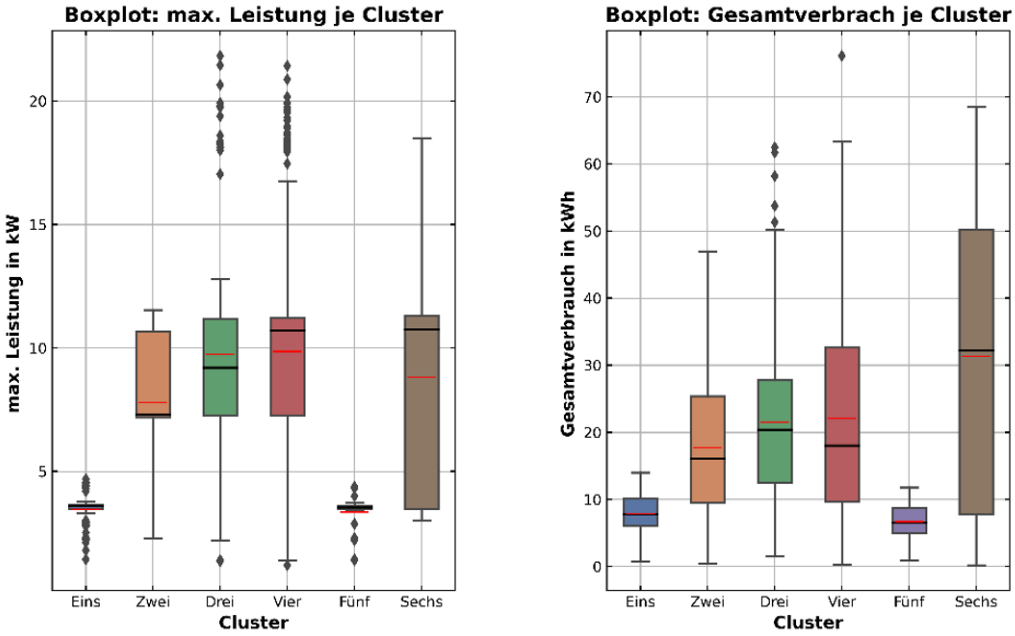


Abbildung 7.3: Boxplot: max. Leistung (links) Gesamtverbrauch (rechts)

In der linken Grafik von Abbildung 7.3 ist die Verteilung der maximalen Leistung für die jeweiligen Cluster dargestellt. Die Cluster Eins und Vier weisen eine kaum sichtbare Streuung auf und besitzen eine geringe Menge an Ausreißern. Der Mittelwert liegt bei etwa 3.5 kW. Die Cluster Zwei und Drei decken einen ähnlichen Leistungsbereich ab, wobei Cluster Drei eine höheren Mittelwert aufweist, der vermutlich durch die betragsmäßig hohen Ausreißer zustande kommt. Der Cluster Sechs deckt den gesamten Wertebereich der anderen Cluster ab.

Die rechte Grafik zeigt, dass die Cluster Eins und Zwei einen vergleichbaren Wertebereich des Gesamtverbrauchs mit einer geringen Streuung abdecken. Die Cluster Zwei, Drei und Vier zeigen ein ähnliches Verhalten im Bereich um 20 kWh. Der Cluster Sechs streut in einem weiten Wertebereich mit einem maximum von ca. 69 kWh.

Unter Berücksichtigung der statistischen Ergebnisse lässt sich annehmen, dass der Cluster Sechs überwiegend Ladevorgänge enthält, die abends beginnen und am darauffolgenden Tag bis zum Vormittag andauern. Durch die lange Standzeit besitzen diese einen geringen Ladeanteil, mit dennoch hohem Gesamtverbrauch. Die Kunden werden als **„Nachtlader“** bezeichnet. Diese Kunden besitzen jedoch nur einen geringen Anteil von 4 %.

Die Cluster Eins und Zwei enthalten überwiegend Kunden, die als **„Arbeitslader“** charakterisiert werden können. Die Kunden kommen am Morgen und fahren am Nachmittag. Der Hauptunterschied liegt in der maximalen Ladeleistung, die im Durchschnitt bei 3.5 kW

(Eins) und 7.8 kW (Zwei) liegen. Die höhere Ladeleistung im Cluster Zwei führt dazu, dass Fahrzeuge schneller vollgeladen sind, weshalb sich eine höhere Leerlaufzeit ergibt. Zusammen machen die Kunden einen Anteil von ca. 50 % aller Ladevorgänge aus.

Die Cluster Drei und Fünf besitzen nahezu keine Leerlaufzeit und eine geringe Standzeit. Die Ankunftszeit und Abfahrtszeit verteilen sich über den gesamten Tag. Die Cluster spiegeln Kunden wider, die als **„Zwischendurchlader“** bezeichnet werden. Diese machen einen Anteil von 25 % aller Ladevorgänge aus. Davon besitzen wiederum etwa 10 % eine maximale Ladeleistung im Bereich um die 3.7 kW, die einen wesentlich geringeren Gesamtverbrauch bei gleicher Standzeit aufweisen.

Der Cluster Vier beinhaltet Kunden, die als **„Nachmittagslader“** charakterisiert werden. Die Ladevorgänge besitzen eine höhere Standzeit als die der „Zwischendurchlader“ und eine geringere als die der „Arbeitslader“. Die Kunden besitzen eine maximale Ladeleistung im Bereich um die 10 kW und erreichen den höchsten Gesamtverbrauch. Der Ladeanteil ist dementsprechend geringer als beim „Arbeitslader“. Die Ladevorgänge dieser Kunden besitzen einen Anteil von etwa 10 %.

Abschließend wird die Verteilung der Cluster in Abhängigkeit der Wochentage untersucht.

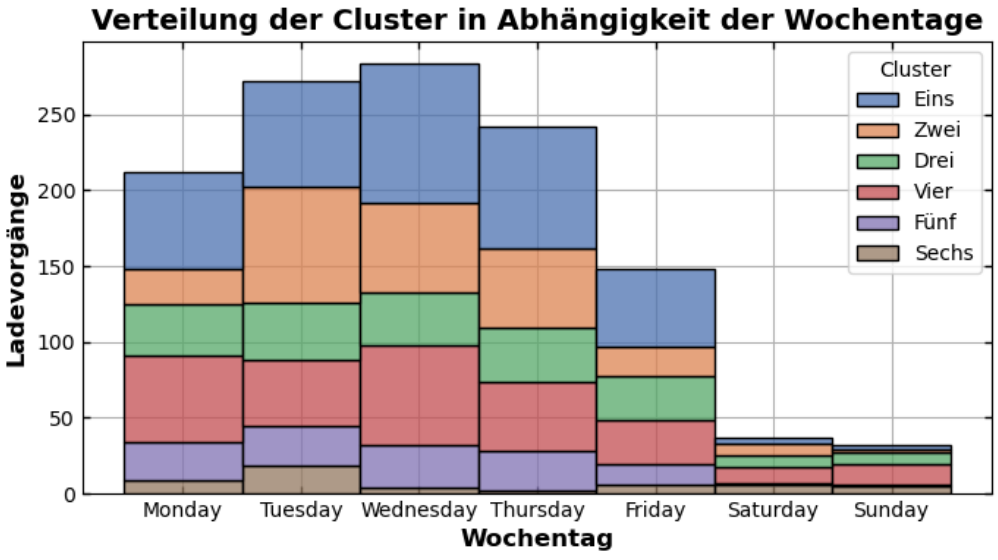


Abbildung 7.4: Verteilung der Cluster in Abhängigkeit der Wochentage

Wie in Abbildung 7.4 zu erkennen, verteilen sich fast alle Cluster gleichmäßig über die Werkstage. Der Cluster Zwei weist weniger Ladevorgänge am Montag und Freitag auf, im Vergleich zu Dienstag, Mittwoch und Donnerstag.

Letztendlich müssen die Ergebnisse dennoch kritisch betrachtet werden. Durch die geringe Anzahl von Ladevorgängen und dem sehr dynamischen Verhalten an der Hochschule, lassen die Daten nur eine grobe Einschätzung über das Verhalten der Kunden zu und eignen sich nur begrenzt für weiterführende Maßnahmen.

8 Zusammenfassung und Ausblick

Im Rahmen der vorliegenden Arbeit wurde das unüberwachte Lernen aus dem Bereich des maschinellen Lernens für die Kategorisierung von Nutzertypen an den Ladesäulen der Ostfalia eingesetzt. Der besondere Schwerpunkt lag dabei auf der Anwendung der Clusteranalyse. Zur Lösung dieser Aufgabe wurde zu Beginn ein kurzer Überblick über das Laden an Ladesäulen mit typischen Kenngrößen gegeben. Darauffolgend wurde das maschinelle Lernen mit den unterschiedlichen Lernverfahren erläutert. Dabei stand die Clusteranalyse im Fokus der Ausführung. Neben wichtigen Grundlagen zu Distanz- und Ähnlichkeitsmaßen, wurde ein Übersicht über verschiedene Clusterverfahren geben. Die in dieser Arbeit verwendeten Clusterverfahren (K-Means, DBSCAN und GMM) mit den dazugehörigen Validierungsmethoden wurden näher erläutert.

Im weiteren Verlauf wurden einige verwandte Arbeiten aus dem Bereich der Ladeinfrastrukturplanung aufgezählt und deren Ergebnisse beschrieben. Nachfolgend wurde die Methodik für die Clusteranalyse vorgestellt, die den roten Faden für das weitere Vorgehen bildet. Diese beinhaltet zudem wichtige Überlegungen zur Vorverarbeitung von Daten, sowie zur Auswahl eines Clusterverfahrens.

In Kapitel 5 wurden die vorhanden Daten von den Ladevorgängen der Ostfalia analysiert und im Hinblick auf die Clusteranalyse vorverarbeitet. Im Anschluss wurden K-Means, DBSCAN und GMM für die Clusteranalyse ausgewählt. Diese Clusterverfahren unterscheiden sich grundlegend in ihrer Wirkungsweise und bieten somit unterschiedliche Betrachtungsweisen der Daten. Durch die experimentelle Umsetzung der Clusterverfahren wurden neben den mehrdimensionalen Daten auch zweidimensionale Beispiele für die Visualisierung der Clusterergebnisse untersucht. Das GMM zeigt anhand der Visualisierung die besten Clusterbildungen, weshalb die Ergebnisse der mehrdimensionalen Clusteranalyse vom GMM für die Interpretation genutzt wurden.

Die Interpretation der Clusterergebnisse von GMM zeigte vier mögliche Nutzertypen, die sich durch verschiedene Charakteristiken innerhalb der Cluster unterscheiden. Die Nutzertypen wurden in „Nachtlader“, „Arbeitslader“, „Zwischendurchlader“ und „Nachmittagslader“ kategorisiert. Die Ergebnisse zeigen nur eine Tendenz zur den einzelnen Nutzertypen. Durch die geringe Anzahl der verfügbaren Ladedaten und durch das sehr dynamischen Verhalten, kann keine eindeutige Aussage gemacht werden.

Das GMM besitzt dennoch sehr gute Voraussetzungen für eine erfolgreiche Clusteranalyse. Besonders im Hinblick auf Standorte mit einer hohen Auslastung, die große Datenmengen zur Verfügung stellen, könnten eindeutige Nutzertypen festgestellt werden. Die wiederum für eine effektive Planung der Ladeinfrastruktur genutzt werden können.

Die Ergebnisse dieser Arbeit zeigen, dass die Clusteranalyse ein hohes Potenzial für die Analyse der Rohdaten von Ladesäulen bietet. Die daraus gewonnen Informationen können einen hohen Mehrwert für einen effizienten Ausbau der Ladeinfrastruktur bieten. Zudem kann die Auslastung von vorhanden Ladesäulen unter Berücksichtigung der Nutzertypen optimiert werden. Beispielsweise wären tageszeitabhängige Preisstrategien denkbar, um somit Kunden mit niedrigem Ladeanteil daran zu hindern die Ladesäulen weiter zu belegen.

Im Hinblick auf zukünftige Projekte, sollten dennoch weitere Clusterverfahren untersucht werden. Das GMM zeigt ein hohes Potenzial, dennoch könnten andere Verfahren, wie z.B. das spektrale Clustern, ebenfalls gute Ergebnisse erzielen. Außerdem sollten weitere interne Validierungsmethoden untersucht werden. Diese sind von entscheidender Bedeutung bei der Auswahl der Hyperparameter für die Clusterverfahren und maßgeblich dafür verantwortlich, wie vertrauenswürdig die gebildeten Cluster bei nicht vorhanden Labels sind. Zudem könnten auch Methoden der Dimensionsreduktion in Kombination mit der Clusteranalyse genutzt werden. Diese ermöglichen eine Betrachtung der Daten im niederdimensionalen Raum und könnten für die Einschätzung von möglichen Clusterformen verwendet werden. Die Erkenntnisse könnten wiederum zur Auswahl eines passenden Clusterverfahrens dienen.

Abkürzungsverzeichnis

GMM	Gaussian Mixture Modell
DBSCAN	Density-Based Spatial Clustering ob Applications with Noise
k	Anzahl der Cluster
minPts	minimale Anzahl von Datenpunkten (DBSCAN)
ε	Epsilon (Bereich um einen Datenpunkte (DBSCAN))
BIC	Bayessche-Informationskriterium
AIC	Akaike-Informaitonskriterium

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[Ab21] Abdulhafedh, A.: Incorporating K-Means, Hierarchical Clustering and PCA in Customer Segmentation. Journal of City and Development 1/3, S. 12–30, 2021.

[AV16] Arthur, D.; Vassilvitskii, S.: K-Means++: The Advantages of Careful Seeding. Stanford, 2016.

[Ba21] Backhaus, K. et al.: Multivariate Analysemethoden. Eine anwendungsorientierte Einführung. Springer Gabler, Wiesbaden, 2021.

[BD21] DKE; VDA-FNN; VDE; ZVEH; ZVEI; BDEW: Ladeinfrastruktur Elektromobilität. Technischer Leitfaden. Version 4. https://www.vde.com/resource/blob/988408/87ed1f99814536d66c99797a4545ad5d/technischer-leitfaden-ladeinfrastruktur-elektromobilitaet---version-4-data.pdf, Stand: 24.08.2022.

[Bi06] Bishop, C. M.: Pattern Recognition and Machine Learning. Springer New York, NY, 2006.

[BM21] Bhattacharjee, P.; Mitra, P.: A survey of density based clustering algorithms. Frontiers of Computer Science 1/15, 2021.

[CH13] Charu, C. A.; Chandan, K. R.: Data Clustering. Algorithms and Applications. Chapman and Hall/CRC Boston, MA, 2013.

[CL20] Cleve, J.; Lämmel, U.: Data Mining. Walter de Gruyter GmbH, Berlin/Bosten, 2020.

[De16] Develder, C. et al.: Quantifying flexibility in EV charging as DR potential: Analysis of two real-world data sets. 2016 IEEE International Conference on Smart Grid Communications (SmartGridComm), S. 600–605, 2016.

[De18] Degele, J. et al.: Identifying E-Scooter Sharing Customer Segments Using Clustering: 2018 International Conference on Development and Application Systems (DAS). 14th edition, May 24-26, 2018, Suceava, Romania conference proceedings. IEEE, Piscataway, NJ, S. 1–8, 2018.

[De21] Deb, S.: Machine Learning for Solving Charging Infrastructure Planning: A Comprehensive Review. In (Malik, O. P.; Xiong, L. Hrsg.): The 5th

International Conference on Smart Grid and Smart Cities (ICSGSC 2021). June 18-20, 2021, Tokyo, Japan. IEEE, Piscataway, NJ, S. 16–22, 2021.

[DP16] Dabhi, D. P.; Patel, M. R.: Extensive survey on hierarchical clustering methods in data mining. International Research Journal of Engineering and Technology (IRJET) 11/3, S. 659–665, 2016.

[DP17] Dhumane, P. B.; Pande, S. R.: Grid-Based Clustering: An Overview. International Journal of Researches in Biosciences, Agriculture & Technology 5, S. 1100–1102, 2017.

[DWD22] Deutscher Wetterdienst, Climate Data Center (CDC): Tägliche Stationsmessungen der mittleren Lufttemperatur in 2 m Höhe in °C für Braunschweig, Stand: 22.07.2022.

[Es96] Ester, M. et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. kdd 34/96, S. 226–231, 1996.

[Fa96] Fayyad, U. et al.: From Data Mining to Knowledge Discovery in Databases. AI Magazine 3/17, S. 37, 1996.

[Fr19] Frochte, J.: Maschinelles Lernen. Grundlagen und Algorithmen in Python. Hanser, München, 2019.

[GCB05] Grira, N.; Crucianu, M.; Boujemaa, N.: Unsupervised and semi-supervised clustering: a brief survey. A review of machine learning techniques for processing multimedia content 1, S. 9–16, 2005.

[Gé20] Géron, A.: Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow. Konzepte, Tools und Techniken für intelligente Systeme. O'Reilly, Heidelberg, 2020.

[HBV01] Halkidi, M.; Batistakis, Y.; Vazirgiannis, M.: On Clustering Validation Techniques. Journal of Intelligent Information Systems 17/2, S. 107–145, 2001.

[HKP11] Han, J.; Kamber, M.; Pei, J.: Data Mining. Concepts and Techniques. Third Edition. Elsevier, 2011.

[HR20] H Humaira; R Rasyidah: Determining The Appropiate Cluster Number Using Elbow Method for K-Means Algorithm. Proceedings of the 2nd Workshop on Multidisciplinary and Applications (WMA), 2020.

[HTF17] Hastie, T.; Tibshirani, R.; Friedman, J. H.: The Elements of Statistical Learning. Data Mining, Inference, and Prediction, Second Edition. Springer New York, NY, 2017.

[Hu98] Huang, Z.: Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. Data Mining and Knowledge Discovery 3/2, S. 283–304, 1998.

[Ja10] Jain, A. K.: Data clustering: 50 years beyond K-means. Pattern Recognition Letters 8/31, S. 651–666, 2010.

[Je22] Jenness, C.: GitHub - DBCV: Python implementation of Density-Based Clustering Validation. https://github.com/christopherjenness/DBCV, Stand: 20.08.2022.

[JQ18] J. Quirós-Tortós et al.: Statistical Representation of EV Charging: Real Data Analysis and Applications: 2018 Power Systems Computation Conference (PSCC), S. 1–7, 2018.

[Ka21] Karle, A.: Elektromobilität. Grundlagen und Praxis. Hanser Verlag, München, 2021.

[KM14] Kameshwaran, K.; Malarvizhi, K.: Survey on Clustering Techniques in Data Mining. International Journal of Computer Science and Information Technologies 2/5, S. 2272–2276, 2014.

[Ko90] Kohonen, T.: The Self-organizing Map. Proceedings of the IEEE 9/78, S. 1464–1480, 1990.

[Ll82] Lloyd, S.: Least Squares Quantization in PCM. IEEE Transactions on Information Theory 2/28, S. 129–137, 1982.

[Ma21] Matzka, S.: Künstliche Intelligenz in den Ingenieurwissenschaften. Maschinelles Lernen verstehen und bewerten. Springer Fachmedien Wiesbaden; Imprint Springer Vieweg, Wiesbaden, 2021.

[Mi17] Miljkovic, D.: Brief Review of Self-Organizing Maps. In (Biljanovic, P. Hrsg.): 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). May 22-26, 2017, Opatija, Croatia proceedings. IEEE, Piscataway, NJ, S. 1061–1066, 2017.

[Mo14] Moulavi, D. et al.: Density-Based Clustering Validation. In (Zaki, M. et al. Hrsg.): Proceedings of the 2014 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, Philadelphia, PA, S. 839–847, 2014.

[Mo20] Moshkovitz, M. et al.: Explainable k-means and k-medians clustering. International Conference on Machine Learning PMLR, S. 7055–7065, 2020.

[Oy19] Oyelade, J. et al.: Data Clustering: Algorithms and Its Applications: 2019 19th International Conference on Computational Science and Its Applications (ICCSA). IEEE, S. 71–81, 2019.

[Pl21] Plaue, M.: Data Science. Grundlagen, Statistik und maschinelles Lernen. Springer Spektrum, Berlin, Heidelberg, 2021.

[PS16] Pandya, S.; Saket J, S.: An Overview of Partitioning Algorithms in Clustering Techniques. International Journal of Electrical and Computer Engineering 6/5, 2016.

[PT21] Pradana, M.; Thi Ha, H.: Maximizing Strategy Improvement in Mall Customer Segmentation using K-means Clustering. Journal of Applied Data Sciences 1/2, S. 19–25, 2021.

[RB16] Renato Cordeiro De Amorim; Boris Mirkin: A clustering based approach to reduce feature redundancy. Knowledge, Information and Creativity Support Systems: Recent Trends, Advances and Solutions. Springer, Cham, S. 465–475, 2016.

[Ri19] Richter, S.: Statistisches und maschinelles Lernen. Gängige Verfahren im Überblick. Springer Berlin Heidelberg, Berlin, Heidelberg, 2019.

[Ro87] Rousseeuw, P. J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20, S. 53–65, 1987.

[RS16] Rahmah, N.; Sitanggan, I. S.: Determination of Optimal Epsilon (Eps) Value on DBSCAN Algorithm to Clustering Data on Peatland Hotspots in Sumatra. OP Conference Series: Earth and Environmental Science 1/31, 2016.

[Sa19] Sartorius, G.: Erfassen, Verarbeiten, Zuordnen multivariater Messgrößen. Neue Rahmenbedingungen für das Nächste-Nachbarn-Verfahren. Springer Vieweg, Wiesbaden, Heidelberg, 2019.

[Sa19] Sauer, D. U. et al.: Speicherung der elektrischen Energie. In (Tschöke, H.; Gutzmer, P.; Pfund, T. Hrsg.): Elektrifizierung des Antriebsstrangs. Grundlagen - vom Mikro-Hybrid zum vollelektrischen Antrieb. Springer Vieweg, Berlin, Heidelberg, S. 61–98, 2019.

[Sa98] Sander, J. et al.: Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. Data Mining and Knowledge Discovery 2/2, S. 169–194, 1998.

[Sc22] Schulze, O.: Elektromobilität – ein Ratgeber für Entscheider, Errichter, Betreiber und Nutzer. Facetten zu Ladeinfrastruktur, Subventionsregeln, Kosten und Handling. Springer, Wiesbaden, 2022.

[Se22] Selenium: Web-Scraping. https://selenium-python.readthedocs.io/, Stand: 24.08.2022.

[Sh20] Shen, Y. et al.: EV Charging Behavior Analysis Using Hybrid Intelligence for 5G Smart Grid. Electronics 1/9, S. 80, 2020.

[SL19] Schacht, S.; Lanquillon, C. Hrsg.: Blockchain und maschinelles Lernen. Wie das maschinelle Lernen und die Distributed-Ledger-Technologie voneinander profitieren. Springer Vieweg, Berlin, Heidelberg, 2019.

[UNO18] 68% of the world population projected to live in urban areas by 2050, says UN | UN DESA | United Nations Department of Economic and Social Affairs. https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html, Stand: 22.05.2022.

[Xi18] Xiong, Y. et al.: Electric Vehicle Driver Clustering using Statistical Model and Machine Learning, 2018.

[XT15] Xu, D.; Tian, Y.: A Comprehensive Survey of Clustering Algorithms. Annals of Data Science 2/2, S. 165–193, 2015.

[Xy16] Xydas, E. et al.: A data-driven approach for characterising the charging demand of electric vehicles: A UK case study. Applied energy 162, S. 763–771, 2016.

[YY19] Yuan, C.; Yang, H.: Research on K-Value Selection Method of K-Means Clustering Algorithm. J 2/2, S. 226–235, 2019.

A Anhang

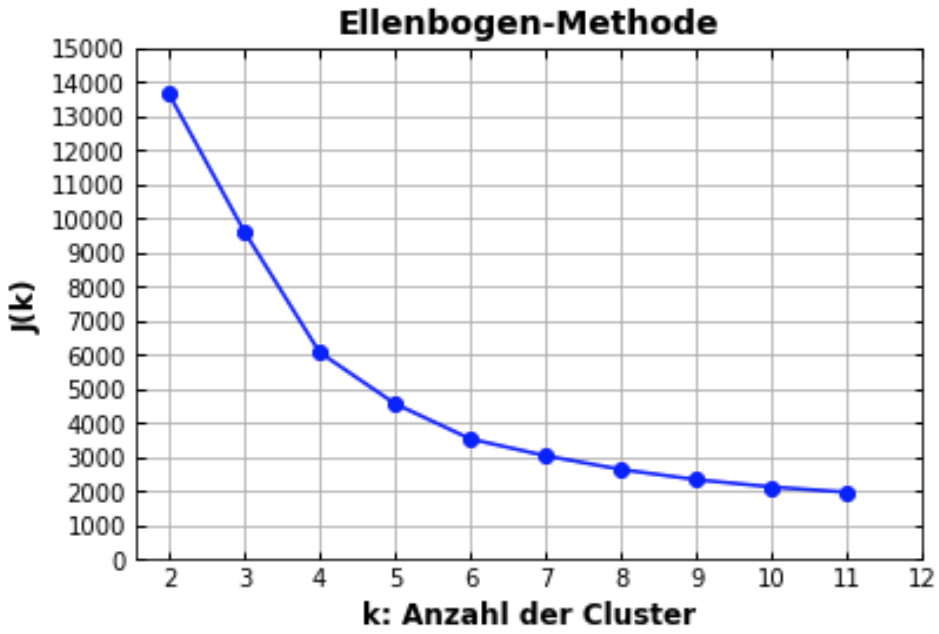
A.1 Struktur der beigefügten Zip-Datei

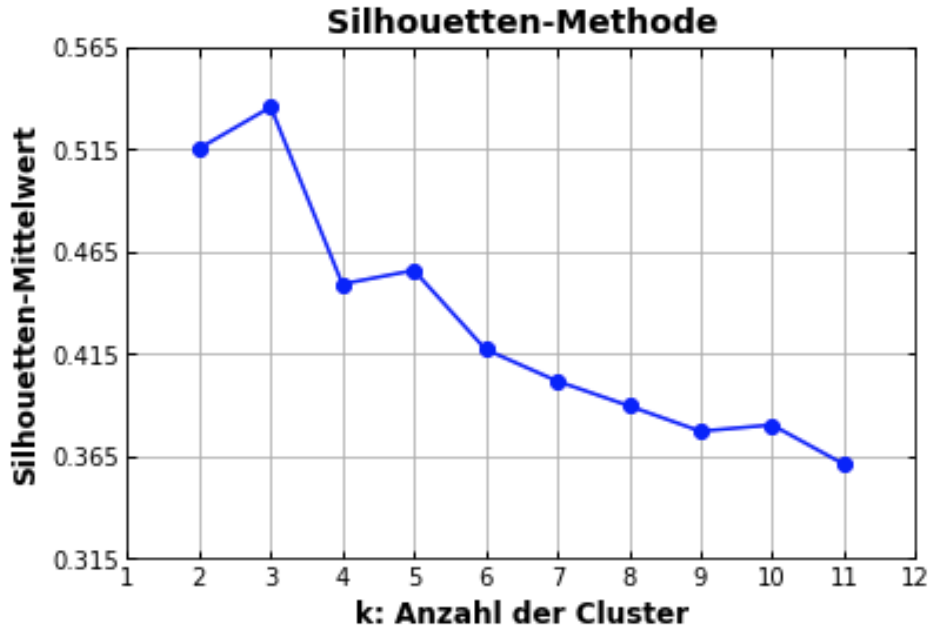
- Dokumentation_BA.pdf
- Quellcodes:
    - 01_Web-Scraping_Datenexport_Ladekurven.ipynb
    - 02_Erstellung_Rohdaten.ipynb
    - 03_Vorverarbeitung_Einfluss_Temperatur.ipynb
    - 04_Vorverarbeitung_Rohdaten.ipynb
    - 05_Clusteranalyse_KMeans.ipynb
    - 06_Clusteranalyse_DBSCAN.ipynb
    - 07_Clusteranalyse_GMM.ipynb
    - DVCV.py

Verwendete Eingabeumgebung: Jupyter Nodebook

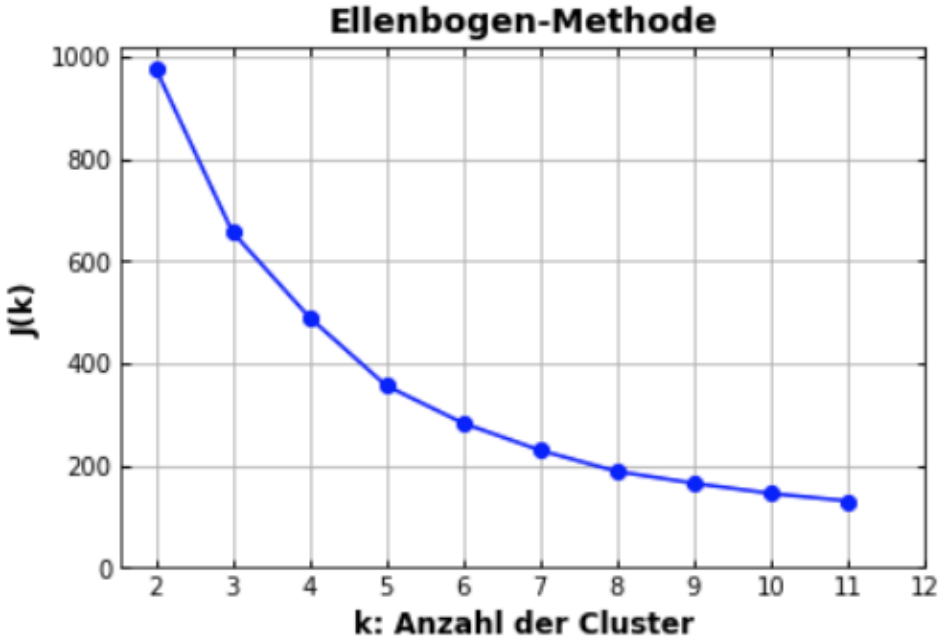
A.2 Validierungsergebnisse der Beispiele

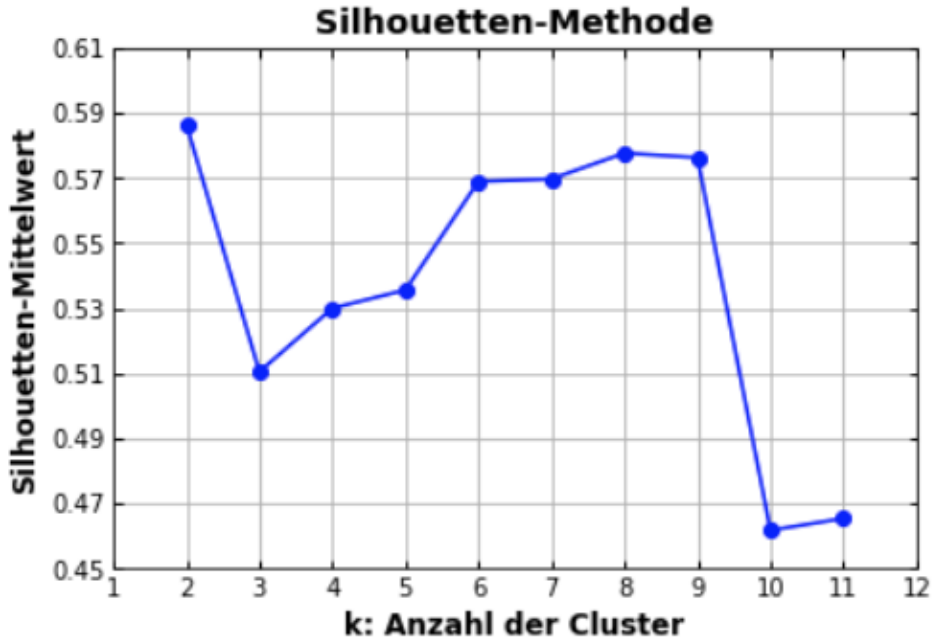
A.2.1 Validierung zum Beispiel mit Ankunft- und Abfahrzeit (K-Means)





A.2.2 Validierung zum Beispiel mit max. Leistung und Gesamtverbrauch (K-Means):



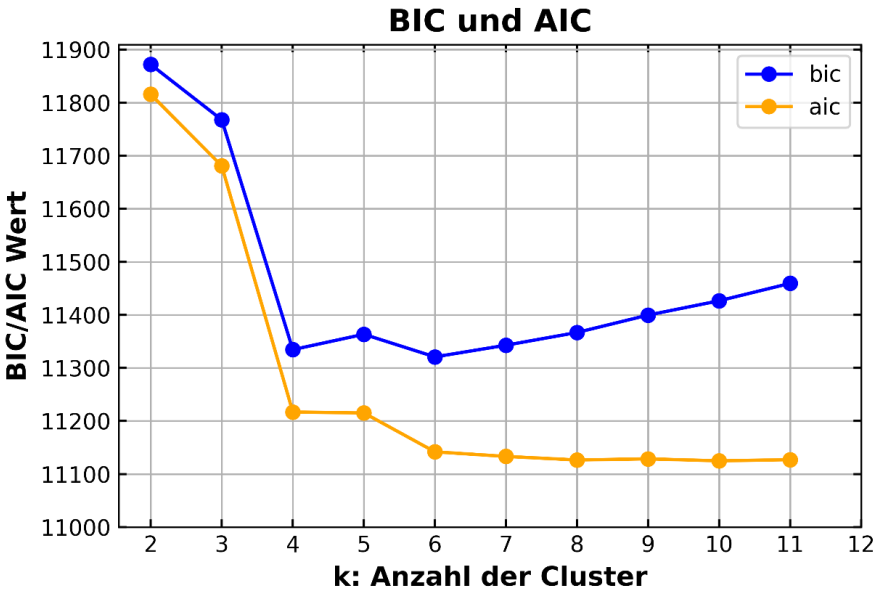


A.2.3 Validierung zum Beispiel mit Ankunft- und Abfahrtszeit, sowie für max. Leistung und Gesamtverbrauch (DBSCAN)

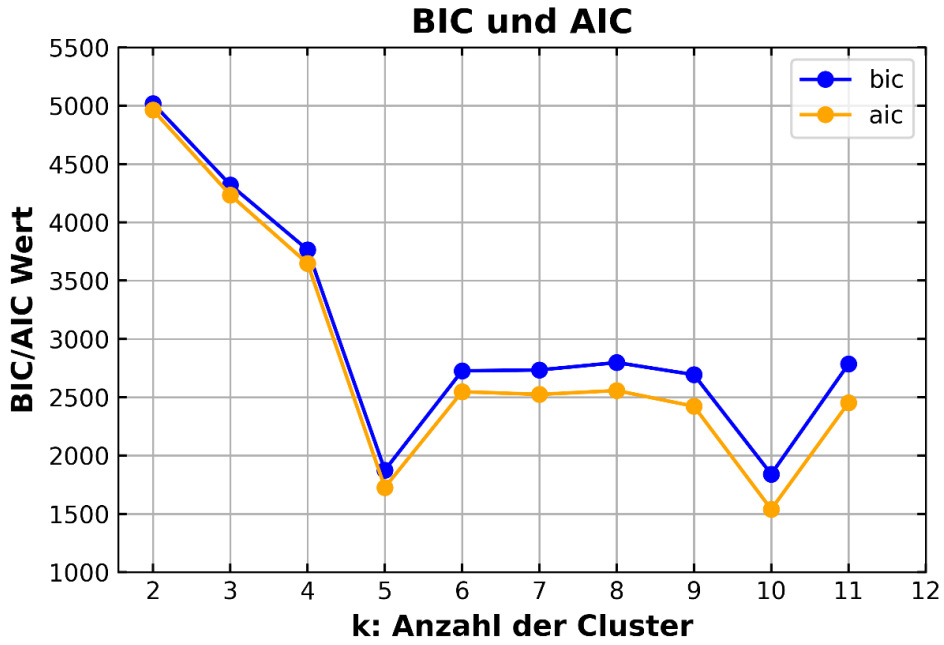
Ankunfts- und Abfahrtszeit		
DBCV	Epsilon	minPts
0,0816	0.1	4
0,0831	0.12	3
-0,1397	0.12	4
0,1652	0.14	3
-0,0698	0.14	4
0,1328	0.16	3
0,1177	0.16	4
0,0506	0.18	3
0,0366	0.18	4
0,1117	0.2	3
0,1	0.2	4
0,059	0.22	3
0,0476	0.22	4
0,2277	0.24	3
0,204	0.24	4
0,1902	0.26	3
0,1324	0.26	4
0,1795	0.28	3
0,1765	0.28	4
0,1819	0.3	3
0,1819	0.3	4
0,0276	0.32	3
0,0276	0.32	4
0,1432	0.34	3
0,1033	0.34	4
0,1397	0.36	3
0,1397	0.36	4
0,142	0.38	3
0,142	0.38	4
0,1727	0.4	3
0,1355	0.4	4

Max. Leistung und Gesamtverbrauch		
DBCV	Epsilon	minPts
-0,3258	0.1	4
-0,0487	0.12	3
-0,3973	0.12	4
-0,0172	0.14	3
-0,2051	0.14	4
0,0519	0.16	3
-0,1572	0.16	4
0,0093	0.18	3
-0,022	0.18	4
-0,0206	0.2	3
-0,1115	0.2	4
-0,0118	0.22	3
0,0129	0.22	4
0,2218	0.24	3
0,1175	0.24	4
0,2987	0.26	3
0,1521	0.26	4
0,3011	0.28	3
0,2166	0.28	4
0,232	0.3	3
0,0867	0.3	4
0,1786	0.32	3
0,0225	0.32	4
0,1788	0.34	3
0,0281	0.34	4
0,1792	0.36	3
0,0306	0.36	4
0,0609	0.38	3
-0,1529	0.38	4
0,4705	0.4	3
0,2398	0.4	4
0,4687	0.42	3
0,2398	0.42	4
0,5326	0.44	3
0,2444	0.44	4
0,538	0.46	3
0,2666	0.46	4
0,5028	0.48	3
0,5027	0.48	4

A.2.4 Validierung zum Beispiel mit Ankunft- und Abfahrzeit (GMM):



Validierung zum Beispiel mit max. Leistung und Gesamtverbrauch (GMM):



A.3 Quellcode

01_Web-Scraping_Datenexport_Ladekurven

September 13, 2022

1 Web-Scraping für Datenexport der Ladekurven

[2]:
```
import os
import time
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
```

[ ]:
```
## Aufruf des Dashboard von Chargecloud über Chrome-Webdriver

driver = webdriver.Chrome(r'Dateipfad1\chromedriver.exe')
driver.get("https://app.chargecloud.de/login")
```

[ ]:
```
## Automatische Anmeldung bei Chargecloud

mandant = WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH,␣
 ↪'//*[@id="login_client"]')))
mandant.send_keys("Mandant")
email = WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '/
 ↪/*[@id="login_email"]')))
email.send_keys("E-Mail")
passwort = WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.
 ↪XPATH, '//*[@id="login_password"]')))
passwort.send_keys("Passwort")
WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
 ↪*[@id="login_submit"]'))).click()

## Danach manuelles Vorgehen:  Standort --> Details(SZ,WF_H oder WF_R) -->␣
 ↪Ladevorgänge --> nächsten Codeabschnitt ausführen
```

[ ]:
```
## Automatisierter Datenexport der Messwerte

## Anzahl der Seiten die durchlaufen werden
Seiten = 50

for n in range(Seiten):
```

```
## Jeder Codeblöck steht für einen Ladevorgang je Seite
##
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
 ↪*[@id="1"]/td[15]/div/a[1]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
 ↪td[contains(@id, "emobility_chargepoint_transaction_metervalue__")]/table/
 ↪tbody/tr/td[4]/div/span'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
 ↪*[@id="export_exportCsv"]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
 ↪div[contains(@class, "ui-dialog-titlebar")]/button/span[2]'))).click()
    time.sleep(1)
    id_ = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.
 ↪XPATH, '//*[@id="1"]/td[1]'))).text
    ## Messwerte werden unter Downloads abgespeichert.
    name = r"C:\Users\Benutzer\Downloads\export_messwerte.csv"
    ## Verschiebung der Messwerte und unbenennung in "ID".csv
    os.rename(name, r"Dateipfad\\" + id_ + ".csv")
##
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
 ↪*[@id="2"]/td[15]/div/a[1]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
 ↪td[contains(@id, "emobility_chargepoint_transaction_metervalue__")]/table/
 ↪tbody/tr/td[4]/div/span'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
 ↪*[@id="export_exportCsv"]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
 ↪div[contains(@class, "ui-dialog-titlebar")]/button/span[2]'))).click()
    time.sleep(1)
    id_ = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.
 ↪XPATH, '//*[@id="2"]/td[1]'))).text
    name = r"C:\Users\Benutzer\Downloads\export_messwerte.csv"
    os.rename(name, r"Dateipfad\\" + id_ + ".csv")

    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
 ↪*[@id="3"]/td[15]/div/a[1]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
 ↪td[contains(@id, "emobility_chargepoint_transaction_metervalue__")]/table/
 ↪tbody/tr/td[4]/div/span'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
 ↪*[@id="export_exportCsv"]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
 ↪div[contains(@class, "ui-dialog-titlebar")]/button/span[2]'))).click()
    time.sleep(1)
    id_ = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.
 ↪XPATH, '//*[@id="3"]/td[1]'))).text
```

```
    name = r"C:\Users\Benutzer\Downloads\export_messwerte.csv"
    os.rename(name, r"Dateipfad\\" + id_ + ".csv")

    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="4"]/td[15]/div/a[1]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪td[contains(@id, "emobility_chargepoint_transaction_metervalue__")]/table/
↪tbody/tr/td[4]/div/span'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="export_exportCsv"]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪div[contains(@class, "ui-dialog-titlebar")]/button/span[2]'))).click()
    time.sleep(1)
    id_ = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.
↪XPATH, '//*[@id="4"]/td[1]'))).text
    name = r"C:\Users\Benutzer\Downloads\export_messwerte.csv"
    os.rename(name, r"Dateipfad\\" + id_ + ".csv")

    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="5"]/td[15]/div/a[1]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪td[contains(@id, "emobility_chargepoint_transaction_metervalue__")]/table/
↪tbody/tr/td[4]/div/span'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="export_exportCsv"]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪div[contains(@class, "ui-dialog-titlebar")]/button/span[2]'))).click()
    time.sleep(1)
    id_ = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.
↪XPATH, '//*[@id="5"]/td[1]'))).text
    name = r"C:\Users\Benutzer\Downloads\export_messwerte.csv"
    os.rename(name, r"Dateipfad\\" + o + ".csv")

    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="6"]/td[15]/div/a[1]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪td[contains(@id, "emobility_chargepoint_transaction_metervalue__")]/table/
↪tbody/tr/td[4]/div/span'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="export_exportCsv"]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪div[contains(@class, "ui-dialog-titlebar")]/button/span[2]'))).click()
    time.sleep(1)
    id_ = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.
↪XPATH, '//*[@id="6"]/td[1]'))).text
    name = r"C:\Users\Benutzer\Downloads\export_messwerte.csv"
```

```
    os.rename(name, r"Dateipfad\\" + id_ + ".csv")

    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="7"]/td[15]/div/a[1]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪td[contains(@id, "emobility_chargepoint_transaction_metervalue__")]/table/
↪tbody/tr/td[4]/div/span'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="export_exportCsv"]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪div[contains(@class, "ui-dialog-titlebar")]/button/span[2]'))).click()
    time.sleep(1)
    id_ = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.
↪XPATH, '//*[@id="7"]/td[1]'))).text
    name = r"C:\Users\Benutzer\Downloads\export_messwerte.csv"
    os.rename(name, r"Dateipfad\\" + id_ + ".csv")

    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="8"]/td[15]/div/a[1]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪td[contains(@id, "emobility_chargepoint_transaction_metervalue__")]/table/
↪tbody/tr/td[4]/div/span'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="export_exportCsv"]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪div[contains(@class, "ui-dialog-titlebar")]/button/span[2]'))).click()
    time.sleep(1)
    id_ = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.
↪XPATH, '//*[@id="8"]/td[1]'))).text
    name = r"C:\Users\Benutzer\Downloads\export_messwerte.csv"
    os.rename(name, r"Dateipfad\\" + id_ + ".csv")

    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="9"]/td[15]/div/a[1]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪td[contains(@id, "emobility_chargepoint_transaction_metervalue__")]/table/
↪tbody/tr/td[4]/div/span'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="export_exportCsv"]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪div[contains(@class, "ui-dialog-titlebar")]/button/span[2]'))).click()
    time.sleep(1)
    id_ = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.
↪XPATH, '//*[@id="9"]/td[1]'))).text
    name = r"C:\Users\Benutzer\Downloads\export_messwerte.csv"
    os.rename(name, r"Dateipfad\\" + id_ + ".csv")
```

```
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="10"]/td[15]/div/a[1]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪td[contains(@id, "emobility_chargepoint_transaction_metervalue__")]/table/
↪tbody/tr/td[4]/div/span'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="export_exportCsv"]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪div[contains(@class, "ui-dialog-titlebar")]/button/span[2]'))).click()
    time.sleep(1)
    id_ = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.
↪XPATH, '//*[@id="10"]/td[1]'))).text
    name = r"C:\Users\Benutzer\Downloads\export_messwerte.csv"
    os.rename(name, r"Dateipfad\\" + id_ + ".csv")

    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="11"]/td[15]/div/a[1]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪td[contains(@id, "emobility_chargepoint_transaction_metervalue__")]/table/
↪tbody/tr/td[4]/div/span'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="export_exportCsv"]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪div[contains(@class, "ui-dialog-titlebar")]/button/span[2]'))).click()
    time.sleep(1)
    id_ = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.
↪XPATH, '//*[@id="11"]/td[1]'))).text
    name = r"C:\Users\Benutzer\Downloads\export_messwerte.csv"
    os.rename(name, r"Dateipfad\\" + id_ + ".csv")

    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="12"]/td[15]/div/a[1]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪td[contains(@id, "emobility_chargepoint_transaction_metervalue__")]/table/
↪tbody/tr/td[4]/div/span'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="export_exportCsv"]'))).click()
    WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪div[contains(@class, "ui-dialog-titlebar")]/button/span[2]'))).click()
    time.sleep(1)
    id_ = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.
↪XPATH, '//*[@id="12"]/td[1]'))).text
    name = r"C:\Users\Benutzer\Downloads\export_messwerte.csv"
    os.rename(name, r"Dateipfad\\" + id_ + ".csv")
```

```
   WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="13"]/td[15]/div/a[1]'))).click()
   WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪td[contains(@id, "emobility_chargepoint_transaction_metervalue__")]/table/
↪tbody/tr/td[4]/div/span'))).click()
   WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="export_exportCsv"]'))).click()
   WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪div[contains(@class, "ui-dialog-titlebar")]/button/span[2]'))).click()
   time.sleep(1)
   id_ = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.
↪XPATH, '//*[@id="13"]/td[1]'))).text
   name = r"C:\Users\Benutzer\Downloads\export_messwerte.csv"
   os.rename(name, r"Dateipfad\\" + id_ + ".csv")

   WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="14"]/td[15]/div/a[1]'))).click()
   WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪td[contains(@id, "emobility_chargepoint_transaction_metervalue__")]/table/
↪tbody/tr/td[4]/div/span'))).click()
   WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪*[@id="export_exportCsv"]'))).click()
   WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪div[contains(@class, "ui-dialog-titlebar")]/button/span[2]'))).click()
   time.sleep(1)
   id_ = WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.
↪XPATH, '//*[@id="14"]/td[1]'))).text
   name = r"C:\Users\Benutzer\Downloads\export_messwerte.csv"
   os.rename(name, r"Dateipfad\\" + id_ + ".csv")

   ##Wechsel auf nächste Seite
   WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.XPATH, '//
↪td[contains(@id, "next_emobility_chargepoint_transaction_location__")]/
↪span'))).click()
```

02_Erstellung_Rohdaten

September 13, 2022

1 Erstellung der Rohdaten für die Vorverarbeitung

- Extraktion der Merkmale aus den Ladekurven (Leistung und Ladezeit)
- Zusammenfassen der Ladevorgänge mit den extrahierten Merkmalen

[ ]:
```
import pandas as pd
import os
import matplotlib.pyplot
import matplotlib.dates
from datetime import datetime
```

[ ]:
```
# Extraktion der Leistungsgrößen und der Ladezeit aus allen Ladekurven.

path = r'Datenpfad\\'

daten = []

for file in [f for f in os.listdir(path) if '.csv' in f]:
    try:
        fname = file.split('.')[0]
        df = pd.read_csv(path+file, sep=';', engine='python')
        df2 = df[df["&empty; Leistung (kW)"] > 0]
        lz = pd.to_timedelta(df2["Zeitspanne"]).sum()
        lz = lz.total_seconds()
        kwmin = df2["&empty; Leistung (kW)"].min()
        kwmax = df2["&empty; Leistung (kW)"].max()
        kwmean = df2["&empty; Leistung (kW)"].mean()
        data = {'kW_min':[kwmin],'kW_max':[kwmax], 'kW_mean':[kwmean],␣
 ↪'Ladezeit':[lz]}
        df = pd.DataFrame(data)
        daten.append(df.assign(ID=fname))
    except:
        pass

df = pd.concat(daten)
df.to_csv("Leistung_Ladezeit.csv", sep=";", index=False)
```

[ ]:
```
## Zusammenfassen der Ladevoränge von WF und SZ.

df1 = pd.read_csv(r'Datenpfad\Ladevorgänge_WF_H.csv', sep=";")
df2 = pd.read_csv(r'Datenpfad\Ladevorgänge_WF_R.csv', sep=";")
df3 = pd.read_csv(r'Datenpfad\Ladevorgänge_SZ.csv', sep=";")
concatenated = pd.concat([df1, df2, df3])
concatenated.to_csv("Ladevorgänge_H_R_SZ.csv", sep=";", index=False)
```

[ ]:
```
## Zusammenfassen der Ladevorgänge mit den Leistungsgrößen und der Ladezeit.

df1 = pd.read_csv(r'Datenpfad\Ladevorgänge_H_R_SZ.csv', sep=";")
df2 = pd.read_csv(r'Datenpfad\Leistung_Ladezeit.csv', sep=";")
df3 = pd.merge(df1, df2, on='ID')
df3.to_csv("Rohdaten_Ladesäulen.csv", sep=";", index=False)
```

03_Vorverarbeitung_Einfluss_Temperatur

September 13, 2022

1 Einfluss von Temperatur auf Verbrauch

[ ]:
```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter
import matplotlib.dates as mdates
```

[ ]:
```
df = pd.read_csv(r"Dateipfad\Ladevorgänge_H_R_SZ.csv",sep=';
 ↪',encoding='latin-1')
df2 = pd.read_csv("Temperatur_mittel.csv",sep='","', encoding='latin-1')
df["Gestartet"] = pd.to_datetime(df["Gestartet"])
```

[ ]:
```
df["Date"] = df["Gestartet"].dt.date
df2 = df2.rename(columns={"Zeitstempel":"Date", "Wert":"Temp_mittel"})
```

[ ]:
```
df.set_index(pd.to_datetime(df['Date']), inplace=True)
df2.set_index(pd.to_datetime(df2['Date']), inplace=True)
```

[ ]:
```
df_fertig = df.merge(df2, left_index=True, right_index=True, how='outer')
```

[ ]:
```
df_fertig = df_fertig.dropna()
```

[ ]:
```
df_fertig = df_fertig.reset_index()
```

[ ]:
```
df_fertig["Jahr"] = df_fertig["Gestartet"].dt.strftime("%Y")
df_fertig["Monat"] = df_fertig["Gestartet"].dt.strftime("%m")
df_fertig["Wochentag"] = df_fertig["Gestartet"].dt.strftime("%A")
```

[ ]:
```
df_fertig = df_fertig[['Gestartet','Verbrauch␣
 ↪(kWh)','Temp_mittel','Jahr','Monat', 'Wochentag']]
```

[ ]:
```
df_fertig["Ladesitzung"] = 1
df_fertig['Verbrauch (kWh)'] = df_fertig['Verbrauch (kWh)'].str.replace(',', '.
 ↪').astype(float)
df_fertig = df_fertig[df_fertig['Verbrauch (kWh)'] > 0]
```

[ ]:
```
data = df_fertig
data['Gestartet'] = pd.to_datetime(data['Gestartet'], format='%d.%m.%Y %H:%M')

df1 = data['Temp_mittel'].groupby([data['Gestartet'].dt.month_name(),␣
 ↪data['Gestartet'].dt.year], sort=False).mean()
df2 = data["Verbrauch (kWh)"].groupby([data['Gestartet'].dt.month_name(),␣
 ↪data['Gestartet'].dt.year], sort=False).mean()
temp = pd.DataFrame(df1)
verbrauch = pd.DataFrame(df2)

temp.index.rename(['month','year'], inplace=True)
verbrauch.index.rename(['month','year'], inplace=True)
temp.reset_index( inplace=True)
verbrauch.reset_index( inplace=True)

fig,ax = plt.subplots(figsize=(10, 5))

ax.plot(temp.index,temp. Temp_mittel, color="red", marker="o")
ax.set_xlabel("Monate (2018-2022)", fontsize = 14)
ax.set_ylabel(" \u00D8 Temperatur (°C)", color = "red",fontsize=14)

ticks = np.arange(0, 43, 1)
labels = ['December', '2019-January', 'February*', 'March*', 'April', 'May',␣
 ↪'June',
       'July*', 'August*', 'September*', 'October', 'November', 'December',
       '2020-January', 'February*', 'March*', 'April', 'May', 'June', 'July*',
       'August*', 'September*', 'October', 'November', 'December',
       '2021-January*', 'February*', 'March', 'April', 'May', 'June', 'July*',
       'August*', 'September', 'October', 'November', 'December',
       '2022-January', 'February*', 'March', 'April', 'May', 'June']

plt.xticks(ticks,labels)
plt.xticks(rotation=90)
ax2 = ax.twinx()
ax2.plot(temp.index , verbrauch["Verbrauch (kWh)"] , color="blue", marker="o")
ax2.set_ylabel(" \u00D8 Verbrauch (kWh)",color = "blue", fontsize=15)
ax2.set_ylim(ymin = 0)
ax.grid(b = True, which='major', linestyle = '-')

plt.show()

# fig.savefig("Temperatur_Verbrauch")
```

04_Vorverarbeitung_Rohdaten

September 13, 2022

1 Vorverarbeitung der Rohdaten für die Clusteranalyse

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[ ]:
```
d_parser = lambda x: pd.datetime.strptime(x, '%Y-%m-%d %H:%M:%S')
df = pd.read_csv(r"Dateipfad\Rohdaten_Ladesäulen.csv",sep=';',␣
 ↪encoding='latin-1', decimal= ',', parse_dates=['Gestartet', 'Beendet'],␣
 ↪date_parser=d_parser)
```

[ ]:
```
df.columns
```

[ ]:
```
df = df[['Ladepunkt','Gestartet', 'Beendet','Standzeit','Ladezeit','Verbrauch␣
 ↪(kWh)','kW_max', 'kW_mean', 'kW_min']]
```

[ ]:
```
## Enfternung der leeren Einträge

df = df.dropna()
```

[ ]:
```
## Anpassung des Datenformats

df["Standzeit"] = df["Standzeit"].astype(float).astype(int)
df["Verbrauch (kWh)"] = df["Verbrauch (kWh)"].astype(float)
df["kW_min"] = df["kW_min"].astype(float)
df["kW_max"] = df["kW_max"].astype(float)
df["kW_mean"] = df["kW_mean"].astype(float)
df["Ladezeit"] = df["Ladezeit"].astype(float).astype(int)
```

[ ]:
```
## Entfernung der Ladevorgänge ohne Ladekurven

df = df[df["kW_min"] != df["kW_max"]]
```

[ ]:
```
## Untersuchung der Merkmale mit statistischen Größen

df.describe()
```

[ ]:
```
df["Monat"] = df["Gestartet"].dt.month_name()
df["Wochentag"] = df["Gestartet"].dt.day_name()
df["Gestartet_hour"] = df["Gestartet"].dt.hour
df["Gestartet_minute"] = df["Gestartet"].dt.minute
df["Gestartet_sekunde"] = df["Gestartet"].dt.second
df["Beendet_hour"] = df["Beendet"].dt.hour
df["Beendet_minute"] = df["Beendet"].dt.minute
df["Beendet_sekunde"] = df["Beendet"].dt.second
```

[ ]:
```
df["Leerlaufzeit"] = df["Standzeit"] - df["Ladezeit"]
```

[ ]:
```
df['Gestartet_Stunden'] = df["Gestartet_hour"] + df["Gestartet_minute"]/60 +␣
↪df["Gestartet_sekunde"]/3600
df['Beendet_Stunden'] = df["Beendet_hour"]  + df["Beendet_minute"]/60 +␣
↪df["Beendet_sekunde"]/3600
```

[ ]:
```
df['Standzeit'] = df['Standzeit']/3600
df['Ladezeit'] = df['Ladezeit']/3600
df['Leerlaufzeit'] = df['Leerlaufzeit']/3600
```

[ ]:
```
df = df.rename(columns={'Gestartet_Stunden': 'Ankunftszeit'})
df = df.rename(columns={'Beendet_Stunden': 'Abfahrtszeit'})
df = df.rename(columns={'Leerlauf': 'Leerlaufzeit'})
df = df.rename(columns={'kW_max': 'Leistung_max (kW)'})
```

[ ]:
```
df = df[['Ladepunkt','Ankunftszeit','Abfahrtszeit','Standzeit','Ladezeit'
         ,'Leerlaufzeit','Leistung_max (kW)','Verbrauch␣
↪(kWh)','Monat','Wochentag',]]
```

[ ]:
```
df.describe()
```

[ ]:
```
df = df[df["Standzeit"] > 0]
df = df[df["Ladezeit"] > 0]
df = df[df["Leerlaufzeit"] >= 0]
df = df[df["Verbrauch (kWh)"] > 0]
df = df[df["Leistung_max (kW)"] > 0]
df = df[df["Standzeit"] > 0.1]
```

[ ]:
```
df.describe()
```

[ ]:
```
df.columns
```

[ ]:
```
df = df[['Ladepunkt','Ankunftszeit','Abfahrtszeit','Standzeit','Ladezeit',
         'Leerlaufzeit','Leistung_max (kW)','Verbrauch␣
 ↪(kWh)','Monat','Wochentag',]]
```

[ ]:
```
df = df.reset_index(drop=True)
```

[ ]:
```
df.shape
```

[ ]:
```
## Untersuchung der Korrelation

plt.figure(figsize = (16,8))
korrelation = df.corr()
sns.heatmap(korrelation, cmap= "GnBu", annot = True)

# plt.savefig('Korrelationsmatrix.png', bbox_inches='tight')
```

[ ]:
```
df["Ladeanteil"] = (df["Ladezeit"]/df["Standzeit"])*100
```

[ ]:
```
def Musterplot(ax, title, xlabel, ylabel, xticks, yticks):
    ax.spines['top'].set_visible(True)
    ax.spines['right'].set_visible(True)

    ax.xaxis.set_tick_params(top='on', direction='in', width=1)
    ax.yaxis.set_tick_params(right='on', direction='in', width=1)

    ax.set_title(title,fontweight='bold',fontsize=14)

    ax.set_xlabel(xlabel,fontweight='bold',fontsize=12)
    ax.set_ylabel(ylabel,fontweight='bold',fontsize=12)

    ax.set_xticks(xticks)
    ax.set_yticks(yticks)
```

[ ]:
```
## Histogramm von Ankunfts- und Abfahrtszeit

fig, ax = plt.subplots(1,1,figsize=(8,4))

xlabel = "Zeit in h"
ylabel = "Anzahl der Ladevorgänge"
title = "Ladevorgänge in Abhängigkeit der Ankunfts- und Abfahrtszeit"
yticks = np.arange(0, 200, step=20)
xticks = np.arange(0, 25, step=1)
plt.grid()
ax.set_axisbelow(True)
ax.hist(df['Ankunftszeit'],bins = 40, alpha=0.5, edgecolor = "black",␣
 ↪label='Ankunftszeit')
```

```
ax.hist(df['Abfahrtszeit'],bins = 40, alpha=0.5, edgecolor = "black",␣
 ↪label='Abfahrzeit')
plt.legend(loc='upper right')
Musterplot(ax, title, xlabel, ylabel, xticks, yticks)
plt.show()

# fig.savefig("hist_AN_AB", dpi=600)
```

[ ]:
```
## Streudiagramm von Ankunfts- und Abfahrzeit

fig, ax = plt.subplots(1,1,figsize=(7,5))

xlabel = "Ankunftszeit in h"
ylabel = "Abfahrtszeit in h"
title = "Zusammenhang zwischen Ankunfts- und Abfahrtszeit"
yticks = np.arange(0, 25, step=1)
xticks = np.arange(0, 25, step=1)
plt.grid()
ax.set_axisbelow(True)
ax.scatter(df['Ankunftszeit'],df['Abfahrtszeit'])
Musterplot(ax, title, xlabel, ylabel, xticks, yticks)
plt.show()

# fig.savefig("Scatter_AN_AB")
```

[ ]:
```
## Entfernung der Ausreißer bei Ankunfts- und Abfahrtszeit

df = df[df["Ankunftszeit"] > 5]
df = df[df["Abfahrtszeit"] > 5]
```

[ ]:
```
## Streudiagramm von max. Leistung und Gesamtverbrauch

fig, ax = plt.subplots(1,1,figsize=(7,5))

xlabel = "max. Leistung in kW"
ylabel = "Gesamtverbrauch in kWh"
title = "Zusammenhang zwischen Verbrauch und max. Leistung"
yticks = np.arange(0, 80, step=5)
xticks = np.arange(0, 25, step=1)
plt.grid()
ax.set_axisbelow(True)
ax.scatter(df['Leistung_max (kW)'],df['Verbrauch (kWh)'])
Musterplot(ax, title, xlabel, ylabel, xticks, yticks)
plt.show()

# fig.savefig("Scatter_kW_kWh")
```

[ ]:
```
## Histogramm von max. Leistung

fig, ax = plt.subplots(1,1,figsize=(8,4))

xlabel = "max. Leistung in kW"
ylabel = "Anzahl der Ladevorgänge"
title = "Ladevorgänge in Abhängigkeit der max. Leistung"
xticks = np.arange(0, 24, step=1)
yticks = np.arange(0, 480, step=40)
plt.grid()
ax.set_axisbelow(True)
ax.hist(df['Leistung_max (kW)'], bins = 25, edgecolor = "black",rwidth=0.8)
Musterplot(ax, title, xlabel, ylabel, xticks, yticks)
plt.show()

# fig.savefig("hist_Leistung")
```

[ ]:
```
## Histogramm von Ladeanteil

fig, ax = plt.subplots(1,1,figsize=(8,4))

xlabel = "Ladeanteil in %"
ylabel = "Anzahl der Ladevorgänge"
title = "Ladevorgänge in Abhängigkeit des Ladeanteils"
xticks = np.arange(0, 110, step=10)
yticks = np.arange(0, 400, step=40)
plt.grid()
ax.set_axisbelow(True)
ax.hist(df['Ladeanteil'], bins = 25, edgecolor = "black",rwidth=.
 ↪8,align='right')
Musterplot(ax, title, xlabel, ylabel, xticks, yticks)
plt.show()

# fig.savefig("hist_Ladeanteil")
```

[ ]:
```
## Histogramm von Wochentagen

df['Wochentag'] = pd.Categorical(df['Wochentag'],␣
 ↪['Monday','Tuesday','Wednesday','Thursday', 'Friday','Saturday','Sunday'])

fig, ax = plt.subplots(figsize=(8,4))

ax = sns.histplot(data = df,
            x = 'Wochentag',
            shrink = 0.8,
            discrete = True,
            )
```

```
plt.grid()

ax.set_axisbelow(True)
ax.set_title("Ladevorgänge in Abhänigkeit der Wochentage")
ax.set(xlabel = 'Anzahl der Ladevorgänge', ylabel='Wochentag')
ax.spines['top'].set_visible(True)
ax.spines['right'].set_visible(True)
ax.xaxis.set_tick_params(top='on', direction='in', width=1)
ax.yaxis.set_tick_params(right='on', direction='in', width=1)
ax.set_title(title,fontweight='bold',fontsize=14)
ax.set_xlabel(xlabel,fontweight='bold',fontsize=12)
ax.set_ylabel(ylabel,fontweight='bold',fontsize=12)

# fig.savefig("hist_Wochentag")
```

[ ]:
```
df.to_csv("Datensatz_Clusteranalyse.csv", sep = ";",index=False)
```

05_Clusteranalyse_KMeans

September 13, 2022

1 Clusteranalyse mit K-Means

[ ]:
```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
```

[ ]:
```
def Musterplot(ax, title, xlabel, ylabel, xticks, yticks):
    ax.spines['top'].set_visible(True)
    ax.spines['right'].set_visible(True)

    ax.xaxis.set_tick_params(top='on', direction='in', width=1)
    ax.yaxis.set_tick_params(right='on', direction='in', width=1)

    ax.set_title(title,fontweight='bold',fontsize=14)

    ax.set_xlabel(xlabel,fontweight='bold',fontsize=12)
    ax.set_ylabel(ylabel,fontweight='bold',fontsize=12)

    ax.set_xticks(xticks)
    ax.set_yticks(yticks)
```

[ ]:
```
df = pd.read_csv("Datensatz_Clusteranalyse.csv", sep=";")
```

[ ]:
```
df2 = df[['Ankunftszeit','Abfahrtszeit','Ladeanteil','Leistung_max␣
 ↪(kW)','Verbrauch (kWh)']]
```

[ ]:
```
# Skalierung der Merkmale

scaler = StandardScaler()
X = scaler.fit_transform(df2)
```

```
X = pd.DataFrame(X, columns=df2.columns)
```

[ ]:
```
## Ermittlung der Clusteranzahl mit Ellenbogen- und Silhoutten-Methode

inertia = []
silhouette = []

for k in range(2, 12):
    kmeans = KMeans(n_clusters=k, init = 'k-means++', random_state=42)
    labels = kmeans.fit_predict(X)
    labels2 = kmeans.fit(X)
    u = kmeans.inertia_
    inertia.append(u)
    score = silhouette_score(X, labels)
    silhouette.append(score)

fig, ax = plt.subplots()

xlabel = "k: Anzahl der Cluster"
ylabel = "J(k)"
title = "Ellenbogen-Methode"

yticks = np.arange(1000, 5000, step=500)
xticks = np.arange(2, 13, step=1)

# plt.annotate('Ellenbogen', xy=(5,2600),
#              xytext=(6, 3500),fontweight='bold',␣
 ↪arrowprops=dict(facecolor='red',shrink=0.1))

plt.grid()
ax.plot(range(2, 12), inertia, color='blue', marker='o')
Musterplot(ax, title, xlabel, ylabel, xticks, yticks)
plt.show()

fig2, ax2 = plt.subplots()

xlabel = "k: Anzahl der Cluster"
ylabel = "Silhouetten-Mittelwert"
title = "Silhouetten-Methode"

yticks = np.arange(0.26, 0.32, step=0.005)
xticks = np.arange(1, 13, step=1)
plt.grid()
ax2.plot(range(2, 12), silhouette, color='blue', marker='o')
Musterplot(ax2, title, xlabel, ylabel, xticks, yticks)

plt.show()
```

```
# fig.savefig("KMeans_Ellenbogen_5")
# fig2.savefig("KMeans_Silhoutte_5")
```

[ ]:
```
## Initalisierung K-Means

kmeans = KMeans(n_clusters=5, init = 'k-means++', random_state=None)
y_kmeans = kmeans.fit_transform(X)
Labels = kmeans.labels_
```

[ ]:
```
## Statistische Analyse der Cluster durch Mittelwert

df2['Labels'] = Labels
df2['Cluster'] = df2['Labels'].map({0:'Eins', 1:'Zwei',2:'Drei', 3:'Vier',4:
 ↪'Fünf'})
df2['Cluster'] = df2['Cluster'].astype('category')
df2['Cluster'] = df2['Cluster'].cat.
 ↪reorder_categories(['Eins','Zwei','Drei','Vier','Fünf'])

df3 = df2.groupby('Cluster').agg(
    {
        'Labels':'count',
        'Ankunftszeit': 'mean',
        'Abfahrtszeit': 'mean',
        'Ladeanteil': 'mean',
        'Leistung_max (kW)':'mean',
        'Verbrauch (kWh)':'mean'
    }
).reset_index()
```

[ ]:
```
df3
```

[ ]:
```
## Plot für zweidimensionales Beispiel (Ankunfts- und Abfahrtszeit)

# fig, ax = plt.subplots(1,1,figsize=(7,5))

# xlabel = "Ankunftszeit in h"
# ylabel = "Abfahrtszeit in h "
# title = "Clusterergebnis mit K-Means (Ankunfts- und Abfahrtszeit)"
# yticks = np.arange(5, 25, step=1)
# xticks = np.arange(5, 25, step=1)
# plt.grid()
# ax.set_axisbelow(True)
# sns.scatterplot(df2['Ankunftszeit'], df2['Abfahrtszeit'], hue =␣
 ↪df2['Labels'],palette="deep")
# # ax.scatter(df['Ankunftszeit'],df['Abfahrtszeit'], c = df2['Labels'])
```

```
# ax.scatter(kmeans.cluster_centers_[:,0] , kmeans.cluster_centers_[:,1]␣
 ↪,marker='x', s = 200, color = 'k')
# Musterplot(ax, title, xlabel, ylabel, xticks, yticks)

# plt.show()

# fig.savefig("Scatterplot_A_A_Beispiel_KMeans")
```

[ ]:
```
## Plot für Zweidimensionales Beispiel (max. Leistung und Gesamtsverbrauch)

# X['Labels'] = Labels

# fig, ax = plt.subplots(1,1,figsize=(7,5))

# xlabel = "max. Leistung (skaliert)"
# ylabel = "Verbrauch (skaliert)"
# title = "Clusterergebnis mit K-Means (max. Leistung und Verbrauch )"
# yticks = np.arange(-2, 5.1, step=0.4)
# xticks = np.arange(-2, 4, step=0.4)
# plt.grid()
# ax.set_axisbelow(True)
# sns.scatterplot(X['Leistung_max (kW)'], X['Verbrauch (kWh)'], hue =␣
 ↪X['Labels'],palette="deep")
# # ax.scatter(df['Ankunftszeit'],df['Abfahrtszeit'], c = df2['Labels'])
# ax.scatter(kmeans.cluster_centers_[:,0] , kmeans.cluster_centers_[:,1]␣
 ↪,marker='x', s = 200, color = 'k')
# Musterplot(ax, title, xlabel, ylabel, xticks, yticks)

# plt.show()

# fig.savefig("Scatterplot_L_V_Beispiel_KMeans")
```

[ ]:
```
## Naheliegenste Datenpunkte an den jeweiligen Zentroiden

df3 = df2.drop("Labels",axis =1)
Cluster_min = np.argmin(y_kmeans, axis=0)
Cluster_min = df3.to_numpy()[Cluster_min]
Cluster_min = pd.DataFrame(Cluster_min)
Cluster_min.columns = ['Ankunftszeit','Abfahrtszeit','Ladeanteil','Leistung_max␣
 ↪(kW)','Verbrauch (kWh)','Cluster']
Cluster_min
```

06_Clusteranalyse_DBSCAN

September 13, 2022

1 Clusteranalyse mit DBSCAN

[ ]:
```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from itertools import product
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
```

[ ]:
```
## Aufruf von DBCV aus Dateiordner

from scipy.spatial.distance import euclidean
%run DBCV.py
```

[ ]:
```
def Musterplot(ax, title, xlabel, ylabel, xticks, yticks):
    ax.spines['top'].set_visible(True)
    ax.spines['right'].set_visible(True)

    ax.xaxis.set_tick_params(top='on', direction='in', width=1)
    ax.yaxis.set_tick_params(right='on', direction='in', width=1)

    ax.set_title(title,fontweight='bold',fontsize=14)

    ax.set_xlabel(xlabel,fontweight='bold',fontsize=12)
    ax.set_ylabel(ylabel,fontweight='bold',fontsize=12)

    ax.set_xticks(xticks)
    ax.set_yticks(yticks)
```

[ ]:
```
df = pd.read_csv("Datensatz_Clusteranalyse.csv", sep=";")
```

[ ]:
```
df2 = df[['Ankunftszeit','Abfahrtszeit','Ladeanteil','Leistung_max␣
 ↪(kW)','Verbrauch (kWh)']]
df1 = df[['Leistung_max (kW)','Verbrauch (kWh)']]
```

[ ]:
```
## Skalierung der Merkmale

scaler = StandardScaler()
X = scaler.fit_transform(df2)
```

[ ]:
```
## Bestimmung von Bereich für Epsilon

minPts = 10

nachbarn = NearestNeighbors(n_neighbors=minPts)
nachbarn_fit = nachbarn.fit(X)
distanz, index = nachbarn_fit.kneighbors(X)
distanz = np.sort(distanz, axis=0)
distanz = distanz[:,1]

fig, ax = plt.subplots(1,1,figsize=(7,5))
xlabel = "Datenpunkte mit minPts"
ylabel = "minimale Distanz"
title = "Bereich von Epsilon"
yticks = np.arange(0, 2.6, step=0.2)
xticks = np.arange(0, 1300, step=100)
plt.grid()
ax.set_axisbelow(True)
ax.plot(distanz)
Musterplot(ax, title, xlabel, ylabel, xticks, yticks)

plt.show()

fig.savefig("Bestimmung_Epsilon_DBSCAN")
```

[ ]:
```
## Validierung von allen Kombination von Epsilon und minPts

eps_werte = np.round(np.arange(1.4,1.45,0.05),2)
min_punkte = np.arange(3,4)
dbscan_params = list(product(eps_werte, min_punkte))

cluster = []
DBCV_score = []
epsi= []
minPts = []

for i in dbscan_params:
    dbscan = DBSCAN(eps=i[0], min_samples=i[1])
```

```
    Labels = dbscan.fit_predict(X)
    epsi.append(i[0])
    minPts.append(i[1])
    cluster.append(len(np.unique(Labels)))
    DBCV_score.append(DBCV(X, Labels, dist_function=euclidean))
    eps_min = list(zip(cluster, DBCV_score, epsi, minPts))

eps_min_df = pd.DataFrame(eps_min, columns=['cluster','DBCV_score','epsilon',␣
 ↪'minPts'])
```

[ ]:
```
## Auswahl des Ergebnis mit dem höchsten DBCV

erg = eps_min_df.iloc[eps_min_df['DBCV_score'].idxmax()]
```

[ ]:
```
## Initalisierung von DBSCAN mit bester Kombination von minPts und Epsilon

eps = erg.epsilon
min_samples = int(erg.minPts)

dbscan = DBSCAN(eps = eps, min_samples=min_samples)
Labels = dbscan.fit_predict(X)
```

[ ]:
```
## Zweidimensionales Beispiel (Ankunfts- und Abfahrtszeit)

# fig, ax = plt.subplots(1,1,figsize=(7,5))

# xlabel = "Ankunftszeit in h"
# ylabel = "Abfahrtszeit in h "
# title = "Clusterergebnis mit DBSCAN (Ankunfts- und Abfahrtszeit)"
# yticks = np.arange(5, 25, step=1)
# xticks = np.arange(5, 25, step=1)
# plt.grid()
# ax.set_axisbelow(True)
# sns.scatterplot(df2['Ankunftszeit'], df2['Abfahrtszeit'], hue =␣
 ↪df2['Labels'],palette="deep")
# # ax.scatter(df['Ankunftszeit'],df['Abfahrtszeit'], c = df2['Labels'])
# # ax.scatter(kmeans.cluster_centers_[:,0] , kmeans.cluster_centers_[:,1]␣
 ↪,marker='x', s = 200, color = 'k')
# Musterplot(ax, title, xlabel, ylabel, xticks, yticks)

# plt.show()

# fig.savefig("Scatterplot_An_Ab_Beispiel_DBSCAN")
```

[ ]:
```
## Zweidimensionales Beispiel (max Leistung und Gesamtverbrauch)

# fig, ax = plt.subplots(1,1,figsize=(7,5))
```

```
# xlabel = "max. Leistung in kW"
# ylabel = "Verbrauch in kWh"
# title = "Clusterergebnis mit DBSCAN (Leistung und Verbrauch)"
# yticks = np.arange(0, 60, step=5)
# xticks = np.arange(0, 22, step=1)
# plt.grid()
# ax.set_axisbelow(True)
# sns.scatterplot(df2['Leistung_max (kW)'], df2['Verbrauch (kWh)'], hue =␣
 ↪df2['Labels'],palette="deep")
# # ax.scatter(df['Ankunftszeit'],df['Abfahrtszeit'], c = df2['Labels'])
# # ax.scatter(kmeans.cluster_centers_[:,0] , kmeans.cluster_centers_[:,1]␣
 ↪,marker='x', s = 200, color = 'k')
# Musterplot(ax, title, xlabel, ylabel, xticks, yticks)

# plt.show()

# fig.savefig("Scatterplot_L_V_Beispiel_DBSCAN")
```

07_Clusteranalyse_GMM

September 13, 2022

1 Clusteranalyse mit GMM

[ ]:
```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.mixture import GaussianMixture as GMM
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
```

[ ]:
```
def Musterplot(ax, title, xlabel, ylabel, xticks, yticks):
    ax.spines['top'].set_visible(True)
    ax.spines['right'].set_visible(True)

    ax.xaxis.set_tick_params(top='on', direction='in', width=1)
    ax.yaxis.set_tick_params(right='on', direction='in', width=1)

    ax.set_title(title,fontweight='bold',fontsize=14)

    ax.set_xlabel(xlabel,fontweight='bold',fontsize=12)
    ax.set_ylabel(ylabel,fontweight='bold',fontsize=12)

    ax.set_xticks(xticks)
    ax.set_yticks(yticks)
```

[ ]:
```
df = pd.read_csv("Datensatz_Clusteranalyse.csv", sep=";")
```

[ ]:
```
df2 = df[['Ankunftszeit','Abfahrtszeit','Ladeanteil','Leistung_max␣
 ↪(kW)','Verbrauch (kWh)']]
```

[ ]:
```
## Skalierung der Merkmale

scaler = StandardScaler()
X = scaler.fit_transform(df2)
```

```
X = pd.DataFrame(X, columns=df2.columns)
```

[ ]:
```
## Ermittung von BIC und AIC für die Bestimmung der Clusteranzahl (k)

bic = []
aic = []

for k in range(2, 12):
    gmm = GMM(n_components=k, init_params='k-means++', random_state=None)
    gmm.fit(X)
    prediction_gmm = gmm.predict(X)
    bic.append(gmm.bic(X))
    aic.append(gmm.aic(X))

fig, ax = plt.subplots()

xlabel = "k: Anzahl der Cluster"
ylabel = "BIC/AIC Wert"
title = "BIC und AIC"

yticks = np.arange(6000, 16000, step=1000)
xticks = np.arange(2, 13, step=1)
plt.grid()
ax.plot(range(2, 12), bic, color='blue', marker='o',label='bic')
ax.plot(range(2, 12), aic, color='orange', marker='o',label='aic')
plt.legend(loc='upper right')
Musterplot(ax, title, xlabel, ylabel, xticks, yticks)

plt.show()

# fig.savefig("BIC_AIC_L_V")
```

[ ]:
```
## Berechnung von AIC für 500 Initialisierung von GMM

aic = []

for k in range(1, 501):
    gmm = GMM(n_components = 6, init_params = 'k-means++', random_state = k).
 ↪fit(X)
    prediction_gmm = gmm.predict(X)
    aic.append(gmm.aic(X))

erg = pd.DataFrame(aic,  columns=['aic'])
```

[ ]:
```
## Initialisierung mit niedrigstem AIC wird ausgewählt

aic_min = erg.iloc[erg['aic'].idxmin()]
```

[ ]:
```
## Beste Initialisierung von GMM über random_state

random_state = aic_min.name

gmm = GMM(n_components = 6, init_params = 'k-means++', random_state =␣
 ↪random_state).fit(X)
prediction_gmm = gmm.predict(X)
```

[ ]:
```
# Statistische Analyse der Cluster durch Mittelwert

df2['Labels'] = prediction_gmm
df2['Cluster'] = df2['Labels'].map({0:'Eins', 1:'Zwei', 2:'Drei',3:'Vier',4:
 ↪'Fünf',5:'Sechs'})
df2['Cluster'] = df2['Cluster'].astype('category')
df2['Cluster'] = df2['Cluster'].cat.
 ↪reorder_categories(['Eins','Zwei','Drei','Vier','Fünf','Sechs'])

df3 = df2.groupby('Cluster').agg(
    {
        'Labels':'count',
        'Ankunftszeit': 'mean',
        'Abfahrtszeit': 'mean',
        'Ladeanteil':'mean',
        'Leistung_max (kW)': 'mean',
        'Verbrauch (kWh)': 'mean',
    }
).reset_index()
```

[ ]:
```
df3
```

[ ]:
```
## Plot für zweidimensionales Beispiel (Ankunfts- und Abfahrtszeit)

# fig, ax = plt.subplots(1,1,figsize=(7,5))

# xlabel = "Ankunftszeit in h"
# ylabel = "Abfahrtszeit in h "
# title = "Clusterergebnis mit GMM (Ankunfts- und Abfahrtszeit)"
# yticks = np.arange(5, 25, step=1)
# xticks = np.arange(5, 25, step=1)
# plt.grid()
# ax.set_axisbelow(True)
# sns.scatterplot(df2['Ankunftszeit'], df2['Abfahrtszeit'], hue =␣
 ↪df2['Labels'],palette="deep")
# # ax.scatter(df['Ankunftszeit'],df['Abfahrtszeit'], c = df2['Labels'])
# ax.scatter(gmm.means_[:,0] , gmm.means_[:,1] ,marker='x', s = 200, color =␣
 ↪'k')
# Musterplot(ax, title, xlabel, ylabel, xticks, yticks)
```

```
# plt.show()

# fig.savefig("Scatterplot_AN_AB_Beispiel_GMM")
```

[ ]:
```
## Plot für Zweidimensionales Beispiel (max. Leistung und Gesamtsverbrauch)

# X['Labels'] = prediction_gmm
# fig, ax = plt.subplots(1,1,figsize=(7,5))

# xlabel = "max. Leistung (skaliert)"
# ylabel = "Verbrauch (skaliert)"
# title = "Clusterergebnis mit GMM (max. Leistung und Verbrauch)"
# yticks = np.arange(-2, 5.1, step=0.4)
# xticks = np.arange(-2, 4, step=0.4)
# plt.grid()
# ax.set_axisbelow(True)
# sns.scatterplot(X['Leistung_max (kW)'], X['Verbrauch (kWh)'], hue =␣
 ↪X['Labels'], palette="deep")
# # ax.scatter(df['Ankunftszeit'], df['Abfahrtszeit'], c = df2['Labels'])
# ax.scatter(gmm.means_[:,0] , gmm.means_[:,1] , marker='x', s = 200, color =␣
 ↪'k')
# Musterplot(ax, title, xlabel, ylabel, xticks, yticks)

# plt.show()

# fig.savefig("Scatterplot_L_V_Beispiel_GMM", dpi=600)
```

[ ]:
```
## Boxplot für die Interpretation der Cluster

fig, ax = plt.subplots(1,1,figsize=(5,7))

xlabel = "Cluster"
ylabel = "Merkmal"
title = "Boxplot: "

sns.boxplot(x='Cluster', y='Merkmal',data=df2, palette="deep",
                    medianprops={"color": "black"}, width=0.5, showmeans=True,␣
 ↪meanprops={"marker":"_",
                                                                              ␣
 ↪          "markerfacecolor":"red",
                                                                              ␣
 ↪          "markersize": "20",
                                                                              ␣
 ↪          "markeredgecolor":"red"})
plt.grid()
ax.set_axisbelow(True)
```

```
ax.set_title(title)
ax.set_xlabel(xlabel)
ax.set_ylabel(ylabel)
ax.spines['top'].set_visible(True)
ax.spines['right'].set_visible(True)
ax.xaxis.set_tick_params(top='on', direction='in', width=1)
ax.yaxis.set_tick_params(right='on', direction='in', width=1)
ax.set_title(title,fontweight='bold',fontsize=14)
ax.set_xlabel(xlabel,fontweight='bold',fontsize=12)
ax.set_ylabel(ylabel,fontweight='bold',fontsize=12)

plt.show()

# fig.savefig("Boxplot_Merkmal")
```