



Masterarbeit

Titel der Arbeit // Title of Thesis

Konzeption und Realisierung einer Data Lake Architektur für Open Data Analysen

Akademischer Abschlussgrad: Grad, Fachrichtung (Abkürzung) // Degree

Master of Science (M. Sc.)

Autorenname, Geburtsort // Name, Place of Birth

Alexander Meinert, Herford

Studiengang // Course of Study

Wirtschaftsinformatik

Fachbereich // Department

Informatik und Kommunikation

Erstprüferin/Erstprüfer // First Examiner

Herr Prof. Dr. Henning Ahlf

Zweitprüferin/Zweitprüfer // Second Examiner

Frau Daniela Nicola

Abgabedatum // Date of Submission

10.04.2019



Eidesstattliche Versicherung

Meinert, Alexander

Name, Vorname // Name, First Name

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem Titel

Konzeption und Realisierung einer Data Lake Architektur für Open Data Analysen

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Gelsenkirchen, 10.04.2019

Ort, Datum, Unterschrift // Place, Date, Signature

Vorwort

Zum ersten Mal begegneten mir die Konzepte von Big Data und Business Intelligence in den Master-Vorlesungen meiner Professoren Dr. Siegbert Kern und Dr. Henning Ahlf an der Westfälischen Hochschule in Gelsenkirchen. Diese Vorlesungen haben bei mir ein großes Interesse an dieser Materie geweckt. Durch den Besuch der DOAG Data Analytics Messe in Brühl konnte ich Eindrücke gewinnen, die sich auf Themenbereiche rund um Machine Learning konzentrierten. Daher ist die Entscheidung gefallen, das Thema meiner Abschlussarbeit der Kombination von Big Data und Machine Learning zu widmen.

Die vorliegende Masterarbeit ist in einem Beratungsunternehmen (MT AG, mit Hauptsitz in Ratingen: <http://www.mt-ag.com/>) angefertigt worden.

An dieser Stelle möchte ich mich recht herzlich bei meinem betreuenden Professor Dr. Henning Ahlf, meinen betrieblichen Betreuern Daniela Nicola und Simon Frank sowie meinem betrieblichen Vorgesetzten Ralf Böhme und Jürgen Günter für ihre Unterstützung bedanken.

Abstract

In dieser Masterarbeit werden die besonderen Eigenschaften von Open Data herausgestellt und Anforderungen für die Speicherung und Verarbeitung von Open Data abgeleitet. Es wird gezeigt, dass viele Merkmale von Big Data auch auf Open Data zutreffen.

Da traditionelle Systeme wie Data Warehouses oft nicht geeignet sind, um mit der Heterogenität von Open Data umzugehen, wird das Data Lake Konzept vorgestellt. Dabei werden die grundlegende Idee und die Ziele eines Data Lakes erläutert. Darauf aufbauend wurde eine Data Lake Architektur entworfen werden, die eine Organisation von Daten und Prozessen vorsieht, die auf die Anforderungen von Open Data abgestimmt ist. Danach wird eine Implementierung mit Spark, Hadoop und verschiedenen Python Bibliotheken vorgestellt und die Speicherung und Verarbeitung praktisch durchgeführt.

Da Open Data in unterschiedlichen Datenqualitäten vorliegt, wird hier ein Vorgehen vorgestellt, um eine solide Datenbasis zu erzeugen. Hierbei werden beispielsweise Verfahren zur Behandlung fehlender Werte und die Zusammenführung von Daten ohne gemeinsames Schlüsselattribut angewendet. Auf den aufbereiteten Daten werden Analysen mit Hilfe von Machine Learning Verfahren durchgeführt.

In dieser Arbeit wird ein Anwendungsfall einer Filialnetzoptimierung eines Einzelhändlers betrachtet, in dem Umsätze für Regionen ohne Filialstandort prognostiziert werden sollen. Hierfür werden Bevölkerungs- und Infrastrukturdaten aus Open Data und unternehmensinternen Filialdaten verwendet. Die Umsatzprognose wird mit einem Random Forest und neuronalen Netz durchgeführt, die anschließend miteinander verglichen werden.

This master thesis highlights the special features of open data and derives requirements for storage and processing. It is shown that many big data features apply to open data as well.

Because traditional systems such as data warehouses are often inadequate to deal with the heterogeneity of open data, the data lake concept is introduced. The basic idea and the goals of a data lake are explained. Based on these concepts, a data lake architecture will be designed that organizes data and transformation processes in alignment with the open data requirements. Afterwards, an implementation with Spark, Hadoop, and various Python libraries will be presented and the storage and processing of open data demonstrated.

Since open data is available in different data qualities, a procedure is presented to create an improved representation of the data. For example, methods for handling missing values and merging data without a common key attribute are used. The processed data is analyzed using machine learning techniques.

In this paper, we will look at a retail store location optimization use case where you want to forecast sales for locations that don't have a store yet. For this purpose, population and infrastructure data from Open Data and in-house branch data are used. The sales forecast implemented using a with a random forest and neural network, which are then compared with each other.

Inhaltsverzeichnis

Vorwort	1
Abstract	2
Abbildungsverzeichnis	6
Tabellenverzeichnis	9
1. Einleitung	10
1.1. Zielsetzung und Vorgehensweise der Arbeit	12
1.2. Abgrenzung	13
2. Konzeption einer Data Lake Architektur für Open Data	14
2.1. Open Data	14
2.1.1. Datenquellen und Formen	15
2.1.2. Open Data als Form von Big Data	19
2.1.3. Eignung traditioneller Data Warehouses	24
2.2. Data Lake	29
2.2.1. Begriffsdefinition und Kerneigenschaften	29
2.3. Datenmanagement im Data Lake	33
2.3.1. Landing und Raw	35
2.3.2. Curated	40
2.3.3. Working und Sandbox	43
2.4. Gefahren und Metadatenmanagement	44
3. Implementation der Data Lake Architektur	51
3.1. Anwendungsfall	51
3.1.1. Zielstellung	51
3.1.2. Daten	52
3.2. Systemarchitektur	58
3.3. Integration und Speicherung	59
3.3.1. Lokales Dateisystem	60
3.3.2. Hadoop	62
3.3.3. Verzeichnisstruktur im Data Lake	64
3.4. Verarbeitung	65
3.4.1. Spark	66
3.4.2. Pandas	68
3.5. Entwicklungsumgebung und Visualisierung	69
3.6. Datenanreicherung	72
3.6.1. Verknüpfung von Daten	72
3.6.2. Ableiten von geografischen Koordinaten aus Adressinformationen	75
3.6.3. Verknüpfung von geografischen Punkten und Regionen mit Spark	77
3.7. Datenbereinigung	79
3.8. Behandlung fehlender Werte	81
3.8.1. Eliminierung von Merkmalsspalten mit fehlenden Werten	82
3.8.2. Schätzwerte für fehlende Werte in hierarchischen Daten	85
3.8.3. Schätzung fehlender Werte einer Zeitreihe	87
4. Praktische Anwendung von Maschine Learning Verfahren	90
4.1. Verarbeitungsprozesse in der Working Zone des Data Lakes	91
4.1.1. Feature Extraction / Selection	91
4.1.2. Feature Transformation	93
4.1.3. Maschine Learning	96
4.2. Untersuchung der vorhandenen Daten	101

4.2.1.	Analyse der Merkmale	102
4.2.2.	Feature Selection	111
4.2.3.	Zusammenfassung und Auswahl der Verfahren	119
4.3.	Random Forest mit Spark ML	121
4.3.1.	Konzept	121
4.3.2.	Durchführung	123
4.3.3.	Diskussion der Ergebnisse	127
4.4.	Neuronales Netz mit Tensorflow und Keras	131
4.4.1.	Konzept	132
4.4.2.	Durchführung	133
4.4.3.	Diskussion der Ergebnisse	137
4.5.	Vergleich der Verfahren	139
5.	Fazit	143
	Literaturverzeichnis	147
	Anhang	149
A.	1 Beigelegte CD	149
A.	2 Programmcode zur geografischen Verknüpfung der Filialstandorte mit Regionsumrissen	149
A.	3 Programmcode zur geografischen Verknüpfung der Filialstandorte mit Regionsumrissen	151
A.	4 Programmcode zur hierarchischen Schätzung von Regionsmerkmalen	152
A.	5 Programmcode zur Zeitreihenschätzung von Regionsmerkmalen	153
A.	6 Programmcode zur Schätzung des Umsatzes, Verkaufsfläche und Mitarbeiteranzahl der Filialen	153

Abbildungsverzeichnis

Abbildung 1 Ergebnisse der McKinsey Studie zum ökonomischen Mehrwert durch Bereitstellung von Open Data	11
Abbildung 2 Suchergebnis aller Lidl Filialen in Deutschland mit entsprechendem HTML Code einer Filiale	17
Abbildung 3 Konzeptueller Aufbau einer DWH in vier Schichten	25
Abbildung 4 Logische Struktur des Data Lakes	34
Abbildung 5 Initiale Prozesse der Landing und Raw Zone im Data Lake	36
Abbildung 6 Mitgelieferte Metadaten des Zensus.....	38
Abbildung 7 Daten und Metadaten die vom Bundesamt für Kartographie und Geodäsie bereitgestellt werden	39
Abbildung 8 Data Lake erweitert um Curated Zone für die Datenbereinigung und -anreicherung.	41
Abbildung 9 Data Lake erweitert um Bereiche Working und Sandbox	44
Abbildung 10 Temperaturklassen mit entsprechender Farbskala für die Klassifizierung von Daten	46
Abbildung 11 Datenorganisation der unterschiedlichen Temperaturklassen über die Speicherbereiche DISK und ARCHIVE	46
Abbildung 12 Identifikation von potenziell ertragreichen Regionen ohne bestehenden Filialstandort.....	52
Abbildung 13 Administrative Gliederung der Verwaltungseinheiten in Deutschland (Quelle: Wikipedia: Verwaltungsgliederung Deutschland).....	53
Abbildung 14 Aufbau des Rechnerclusters und technische Realisierung des Data Lake	59
Abbildung 15 Physische Speicherung der einzelnen Schichten im Data Lake über ein lokales Dateisystem und HDFS	60
Abbildung 16 Für Menschen visuell aufbereitetes Excel Dokument über Arbeitslosigkeit vom Bundesamt für Statistik.	60
Abbildung 17 CSV Tabelle über Arbeitslosigkeit vom Bundesamt für Statistik nach Strukturierung und Formatierung	61
Abbildung 18 Verzeichnisstruktur im Data Lake	64
Abbildung 19 Darstellung eines Spark DataFrame mit Angaben zu Filialstandorten	67
Abbildung 20 Jupyter Umgebung mit bestehender Verbindung zum Spark System und Betriebssystem Kommandos.....	70
Abbildung 21 Einheitliche String-Typisierung aller Spalten im einzulesenden Datensatz (Zensus Haushalte)	73
Abbildung 22 Einlesen des Datensatzes über Spark. Der Dateipfad und Dateiname wurden zuvor Variablen zugewiesen.....	73
Abbildung 23 Erzeugung einer temporären SQL Tabelle in Spark	74
Abbildung 24 Durchführung einer Join Operation mit Spark SQL	74
Abbildung 25 Schreibvorgang der zusammengeführten Datei ins HDFS mit Spark	74
Abbildung 26 Ablauf der Anreicherung von geografischen Informationen eines Filialstandorts über die Google API.....	76
Abbildung 27 Verortung eines geografischen Knotenpunktes im Polygon einer Region	77
Abbildung 28 Ersetzung Funktion zur Datenbereinigung mit Python.....	80
Abbildung 29 Zuweisung der definierten Funktion in eine UDF	80
Abbildung 30 Bereinigung des Zensusdatensatzes über eine Schleifenkonstruktion	81

Abbildung 31 Anzahl fehlender Werte für jedes Merkmal aus dem Zensusdatensatz	82
Abbildung 32 Pandas dropna() Funktion zur Entfernung von Spalten die nicht eine Mindestanzahl an Werten besitzen	83
Abbildung 33 Zusammenhang von Spalten im Datensatz zu gegebenen Schwellwerten für die Nullwerttoleranzen	84
Abbildung 34 Hierarchische Schätzung der Arbeitslosenquote für Bund, Kreis und Gemeindeebene	86
Abbildung 35 Inter- und Extrapolation von Zeitreihen am Beispiel der Arbeitslosenquote ...	88
Abbildung 36 Interpolation und Extrapolation über pchip und spline am Beispiel der Arbeitslosenquote	89
Abbildung 37 Überführung eines nominalen Merkmals in Dummy Variablen.....	93
Abbildung 38 Überführung eines nominalen Merkmals in einen binären Vektor	94
Abbildung 39 Aufbau des Analysemodells über die Ebenen Input, Verarbeitung und Output	96
Abbildung 40 Einordnung von ML Algorithmen in Clustering, Klassifizierungs- und Regressionsverfahren.....	98
Abbildung 41 Prozess der Modellbewertung über die Fehlermetriken	100
Abbildung 42 Vorgehensweise im Machine Learning.....	101
Abbildung 43 Statistische Profil des Merkmals Einwohneranzahl einer Region (mit Pandas erzeugt)	102
Abbildung 44 Visualisierung der Verteilung von Merkmalen über Boxplots.....	103
Histogramm	104
Verlaufsplot	104
Abbildung 47 Unterer Extremwertbereich der 10 bevölkerungsärmsten Regionen.....	105
Abbildung 48 Oberer Extremwertbereich der 10 bevölkerungsreichsten Regionen	105
Abbildung 49 Boxplot für Haushalte einer Region vor Reduzierung des Datensatzes (links) und danach (rechts)	107
Abbildung 50 Boxplot für geografische Knotenpunkte in einer Region vor Reduzierung des Datensatzes (links) und danach (rechts)	107
Abbildung 51 Boxplot für Familien ohne Kinder in einer Region vor Reduzierung des Datensatzes (links) und danach (rechts)	108
Abbildung 52 metrische und visuelle Darstellung der Lageparameter der Verteilung	108
Abbildung 53 Missverhältnis zwischen Anzahl Gemeinden und Regionen mit und ohne Filialen	109
Abbildung 54 Verhältnis von Städten und Gemeinden im Trainingsdatensatz (links) und Vorhersagedatensatz (rechts)	110
Abbildung 55 Visualisierung der Verteilung des Umsatzmerkmals vor der Reduzierung des Datensatzes (links) und danach (rechts) über Verlaufsplots	110
Abbildung 56 Zusammenhang von Korrelationskoeffizienten und Datenverteilung	112
Abbildung 57 Steigung von Korrelationskoeffizienten von 1 und -1	112
Abbildung 58 Komplexe Zusammenhänge eines KK von 0	113
Abbildung 59 Scatterplot für den Zusammenhang Anteil Arbeitslose und Einkommen je Einwohner	113
Abbildung 60 Ausschnitt über die mit Pandas erzeugte Korrelationsmatrix.....	114
Abbildung 61 Zusammenhang von Haushalten und Bevölkerungsanzahl in einer Region über einen Scatterplot	115

Abbildung 62 Zusammenhang von Wohnungen und Bevölkerungsanzahl in einer Region über einen Scatterplot	115
Abbildung 63 Zusammenhänge von Anteil arbeitslose Ausländer und Einkommen (links) sowie arbeitslose Ausländer und Anteil Arbeitslose (rechts) über Scatterplots	116
Abbildung 64 Merkmale mit den höchsten Scoringwerte nach dem Mutual Information Verfahren.....	117
Abbildung 65 Entscheidungsverfahren mit mehreren Entscheidungsbäumen	122
Abbildung 66 Spark ParamGrid mit Kombinationen von Hyperparametern	123
Abbildung 67 Spark CrossValdidator für die automatische Suche der besten Parameterkombinationen für eine Schätzung	124
Abbildung 68 Feature Transformation mit Spark	125
Abbildung 69 Feature Vektoren Spark	125
Abbildung 70 Aufteilung Trainings- und Testdaten sowie Durchführung der Trainingsphase in Spark.....	126
Abbildung 71 Random Forest - regionsunabhängiges Modell: Fehlerkennzahlen und R^2	127
Abbildung 72 Random Forest - regionsunabhängiges Modell. Statistisches Profil von Real und Schätzwert.....	127
Abbildung 73 Random Forest - regionsunabhängiges Modell: Vergleich Real- und Schätzwerte	128
Abbildung 74 Visualisierung der Importance Faktoren des Random Forest	129
Abbildung 75 Schichten eines künstlichen neuronalen Netzes	132
Abbildung 76 Schematischer Aufbau eines künstlichen Neurons	133
Abbildung 77 Erzeugung eins neuronalen Netzes in Keras.....	134
Abbildung 78 Compile() Funktion um das Netzwerk endgültig zu definieren	135
Abbildung 79 Standardisierung der Features mit Scikit-learn	135
Abbildung 80 Anlernen des neuronalen Netzes Verfahrens und abspeichern der Trainingsschritte über Keras	136
Abbildung 81 Verlauf des Fehlerwertes über die Trainingseinheiten	136
Abbildung 82 Neuronale Netz - regionsunabhängiges Modell: Vergleich Real- und Schätzwerte	137
Abbildung 83 Visuelle Darstellungen des MAE der Modelle beider Verfahren.....	141
Abbildung 84 Darstellung der Abweichung von wahren und prognostizierten Umsatzwert über das Neuronale Netz.....	141
Abbildung 85 Darstellung der Abweichung von wahren und prognostizierten Umsatzwert über das RandomForest Verfahren.....	142

Tabellenverzeichnis

Tabelle 1 Exemplarische Auflistung einiger Open Data Quellen.	18
Tabelle 2 Prüfung der Umsatzbarkeit von Anforderung für Open Data mit einem DWH	28
Tabelle 3 Prüfung der Umsatzbarkeit von Anforderung für Open Data mit einem DWH und Data Lake.....	33
Tabelle 4 Beispiel für ein Metadatenschema für Open Data	47
Tabelle 5 Beschreibung der Datenquelle des Bundesamtes für Statistik	55
Tabelle 6 Beschreibung der Datenquelle des Zensus	56
Tabelle 7 Beschreibung der Datenquelle von OpenStreetMap.....	57
Tabelle 8 Beschreibung der unternehmensinternen Daten	57
Tabelle 9 Beschreibung der Datenquelle des Bundesamtes für Kartographie und Geodäsie	58
Tabelle 10 Übersicht über die in dieser Arbeit verwendeten Technologien	70
Tabelle 11 Feature Set für die Analyse	118
Tabelle 12 Parameterausprägungen der einzelnen Modelle für das Random Forest Verfahren	130
Tabelle 13 Fehlerwerte der Modelle für das Random Forest Verfahren	130
Tabelle 14 Parameterausprägungen der einzelnen Modelle für das neuronale Netz ...	138
Tabelle 15 Fehlerwerte der Modelle über das Random Forest Verfahren	138
Tabelle 16 Fehlerwerte der Modelle über RandomForest Verfahren.....	140
Tabelle 17 Fehlerwerte der Modelle über das neuronale Netz	140

Abkürzungsverzeichnis

API	Application programming interface
CPU	Central processing unit Central processing unit
CSV	Comma-separated values
FTP	File Transfer Protocol
http	Hypertext Transfer Protocol
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
RAM	Random-Access Memory
REST	REpresentational State Transfer
SQL	Structured Query Language
SSH	Secure shell host
SSL	Secure socket layer
VPN	Virtual private network
XML	Extensible Markup Language

1. Einleitung

Durch die Digitalisierung werden Aktivitäten aus Wirtschafts-, Gesellschafts- und Privatleben mit Hilfe von Daten abgebildet. Die Messung und Verwendung dieser Daten stellt sich als strategischen Ressource heraus, die positiven Einfluss auf den Geschäftserfolg eines Unternehmens haben kann, vgl. (Fasel & Meier, 2016, S. 40).

Waren es in der Vergangenheit vorrangig interne Daten, die bei der Analyse unternehmerischen Aktivitäten berücksichtigt wurden, sind in jüngster Zeit zunehmend unternehmensexterne Daten von Interesse.

Dafür wurden zunächst kommerzielle Datendienstleister genutzt, wie etwa Marktforschungsunternehmen, Betreiber von Adressdatenbanken oder Anbieter von Wetterdaten, vgl. (Gluchowski, 2016, S. 3).

In den letzten Jahren öffneten Behörden, staatliche Institutionen und teilweise sogar Unternehmen ihre Datenbestände und stellten sie zur freien Verfügung. Dafür hat sich der Begriff Open Data etabliert. Das Datenangebot und deren Zugänglichkeit verbessert sich durch Gesetze und Standards auf nationaler und internationaler Ebene stetig.

Durch die Verknüpfung von unternehmensinternen und externen Daten können neue Erkenntnisse gewonnen und Geschäftsmodellen optimiert werden.

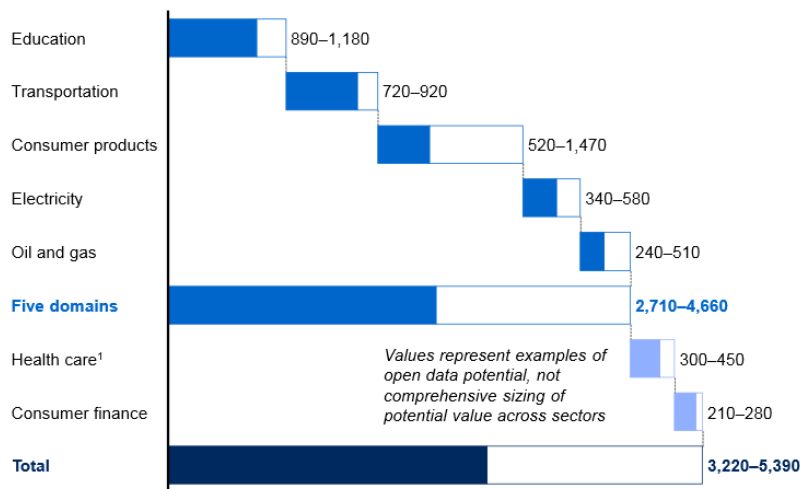
In einer Studie von McKinsey¹ wird das jährliche wirtschaftliche Potential von Open Data weltweit auf bis zu 5,4 Billionen Dollar geschätzt, die in verschiedensten Bereichen wie Logistik, Konsumgüter, Finanzen oder Bildung erreicht werden können (siehe Abbildung 1). Die EU schätzt den ökonomischen Mehrwert von Open Data für Europa jährlich auf 40 Milliarden Euro²

¹ McKinsey Studie: <https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/opendata-unlocking-innovation-and-performance-with-liquid-information> (letzter Zugriff 29.11.2018)

² Pressemitteilung der EU: http://europa.eu/rapid/press-release_IP-11-1524_en.htm?locale=en (letzter Zugriff 29.11.2018)

Open data can help unlock \$3.2 trillion to \$5.4 trillion in economic value per year across seven “domains”

\$ billion



¹ Includes US values only.

NOTE: Numbers may not sum due to rounding.

SOURCE: McKinsey Global Institute analysis

Abbildung 1 Ergebnisse der McKinsey Studie zum ökonomischen Mehrwert durch Bereitstellung von Open Data

Doch der Nutzung von Open Data in einem Unternehmen stehen Hürden im Weg. Es gibt kaum etablierte oder flächendeckende Standards. Jeder Datenbereitsteller benutzt eigene Schnittstellen mit unterschiedlichen Formaten und Nutzungsbedingungen. Typischerweise werden Daten für analytische Zwecke in einem Data Warehouse gespeichert. Diese Systeme sind jedoch meist platzmäßig beschränkt und mit hohen Lizenzkosten verbunden. Daher sind sie für Open Data ungeeignet, da der Nutzen und die genaue Verwendung dieser Daten nicht immer von Anfang an feststehen. Außerdem sind die meisten Data Warehouses historisch gewachsen und beinhalten ein komplexes Gefüge aus voneinander abhängigen Tabellen und Verarbeitungsketten. Durch diese starre Struktur erfordert die Einbindung und Analyse neuer Datenquellen viel Zeit und Aufwand.

In dieser Arbeit wird eine Data Lake Architektur vorgestellt, die eine geeignete Alternative für die Speicherung und Verarbeitung für Open Data darstellt.

1.1. Zielsetzung und Vorgehensweise der Arbeit

Das Ziel dieser Arbeit ist es, die Besonderheiten von Open Data herauszustellen und eine geeignete Architektur für die Speicherung und Verarbeitung im unternehmerischen Umfeld zu entwickeln. Anhand eines Anwendungsbeispiels soll ein Verfahren vorgestellt werden, das Open Data mit unternehmensinternen Daten verbindet. Darauf aufbauend soll gezeigt werden, wie eine Auswertung dieser verknüpften Daten mit Methoden des Machine Learnings durchgeführt werden kann.

Um ersteres Ziel zu erreichen, wird im **ersten Kapitel** ein generelles Verständnis des Begriffs Open Data vermittelt und einige Beispiele von etablierten Bereitstellern aufgezeigt. Außerdem wird auf die technischen und rechtlichen Besonderheiten im Open Data Bereich eingegangen.

Für die Analyse der Daten wird zunächst eine Architektur benötigt in dem die verschiedenen Datenbestände und Verarbeitungsprozesse organisiert werden, um flexibel auf die Anforderungen von Open Data reagieren zu können. Zu diesem Zweck soll im **zweiten Kapitel** das Data Lake Konzept vorgestellt werden.

Anhand dieses Konzepts soll dann im **dritten Kapitel** die technische Ausgestaltung für die Integration von Open Data mit unternehmensinternen Daten gezeigt werden. Dafür wird der Anwendungsfall des Einzelhändlers beschrieben und alle verwendeten Daten vorgestellt und für die spätere Analyse vorbereitet.

Darauf aufbauend wird im **vierten Kapitel** ein Prognosemodell konstruiert, das optimale neue Standorte für den Einzelhändler identifizieren soll. Dafür werden die Daten zunächst mit verschiedenen statistischen Methoden analysiert und visualisiert. Danach werden geeignete Machine Learning Verfahren ausgewählt und durchgeführt.

Im **fünften Kapitel** werden die Erkenntnisse der gesamten Arbeit zusammengefasst und ein Ausblick gegeben.

1.2. *Abgrenzung*

Da Open Data vergleichsweise neu ist, wird es in der wissenschaftlichen Literatur bisher nicht einheitlich definiert. In dieser Arbeit wird keine tiefergehende Auseinandersetzung mit Open Data erfolgen. Für eine detaillierte Betrachtung des Open Data Begriffs siehe (Meinert, 2018). Stattdessen soll hier lediglich auf die Kerneigenschaften von Open Data eingegangen werden, um Anforderungen zu formulieren und eine geeignete Architektur zu entwickeln.

Konzepte rund um Data Governance und Sicherheit werden nicht detailliert beschrieben, da in dieser Arbeit ein hypothetischer Anwendungsfall betrachtet und keine konkreten organisatorischen Gegebenheiten bekannt sind.

Es erfolgt keine Gegenüberstellung einer on-premise Lösung in Vergleich zu einer Cloud Umsetzung der vorgeschlagenen Architektur, da lediglich die methodische Vorgehensweise zur Konzeption und zum Aufbau einer Data Lake Architektur im Vordergrund der Betrachtung steht.

Im vierten Kapitel liegt der Fokus auf der Anwendung von Machine Learning Verfahren. Es findet keine tiefergehende Diskussion der verwendeten Verfahren statt, hierzu sei auf die entsprechende Fachliteratur verwiesen.

Zur Abgrenzung des betrachteten Anwendungsfalls werden strukturierte und semi-strukturierte Daten verwendet. Unstrukturierte Daten wie Bild, Ton und Text erfordern andere Prozesse und Analyseverfahren, die hier nicht weiter berücksichtigt werden.

2. Konzeption einer Data Lake Architektur für Open Data

In diesem Kapitel erfolgt eine Vorstellung des konzeptuellen Aufbaus einer Data Lake Architektur für Open Data. Dafür wird zunächst Open Data thematisiert und auf die Entwicklung und Begriffsdefinition eingegangen. Dabei sollen die Eigenschaften von Open Data herausgestellt werden. Hierfür werden konkrete Beispiele für Open Data gegeben, die auch in dem behandelten Anwendungsfall Anwendung finden. Anschließend sollen die Anforderungen für eine Verwendung dieser Daten abgeleitet und darauf aufbauend Architekturüberlegungen formuliert werden. Danach wird das Data Lake Konzept vorgestellt. Hierbei sollen die Kerneigenschaften erläutert und mit den Anforderungen von Open Data verglichen werden. Darauf aufbauend kann der Data Lake für Open Data definiert werden.

2.1. Open Data

Nach Bewegungen wie Open Access (Öffnung von Wissenschaft und Lehre) und Open Source (Öffnung von Software und Programmbibliotheken), präsentiert sich Open Data als weitere Erscheinung der Open-Bewegung. Open Data besitzt zwei Perspektiven, gesellschaftlicher und technischer Natur (Baack, 2013, S. 5). Gesellschaftlich stehen Ziele wie Partizipation und Transparenz, allgemein „Datendemokratie“ im Vordergrund, die durch Offenlegung von Datenbeständen verschiedenster Institutionen gefördert wird. Technisch definiert sich der Offenheitsgedanke wie folgt:

Kontrollmechanismen gibt, die den Zugang, die Weiterverarbeitung und die Weiterverbreitung dieser Daten einschränken. Der Zugang, die Weiterverarbeitung und die Weiterverbreitung soll jedermann und zu jeglichem Zweck, auch kommerziellem, ohne Einschränkungen und Diskriminierung und ohne Zahlung von Gebühren möglich

3

Diese Definition ist im Vergleich zu anderen Definitionsversuchen sehr umfassend und kann auch als Forderung interpretiert werden. Längst nicht alle Bereitsteller können alle

³ Zitat aus <http://www.bpb.de/gesellschaft/digitales/opendata/64055/was-sind-offene-daten>

Punkte aus der Definition einhalten, weswegen der Open Data Begriff in dieser Arbeit an der jeweiligen Stelle vergrößert werden muss.

Bei Open Data handelt es sich um verschiedenste Daten aus den Bereichen Umwelt, Bevölkerung, Bildung, Gesundheit, Wirtschaft, u.v.m. die über Plattformen von unterschiedlichen Institutionen bereitgestellt werden.

Da im Grundsatz die kommerzielle Nutzung nicht ausgeschlossen wird, ergeben sich einige Möglichkeiten für wirtschaftliche Akteure. Für eine mögliche Verwendung dieser Datenbestände, ist es jedoch zunächst notwendig zu verstehen, wie es um die Beschaffenheit dieser Daten steht und welche Datenbestände wo und in welcher Form zugänglich sind. Nach dieser begrifflichen Einordnung werden Beispiele für Datenquellen gegeben. Dabei soll auf Eigenschaften wie Zugangsformen, rechtliche Bedingungen und Datenqualität eingegangen werden.

2.1.1. Datenquellen und Formen

Die prominentesten Beispiele für Open Data in Deutschland ist das Datenportal GovData, der Deutsche Wetterdienst (DWD) sowie die Datenbank des Statistischen Bundesamtes (destatis) für amtliche Statistik. Die Datenquellen sollen im Folgenden kurz vorgestellt werden.

Verwaltungsdaten von Bund, Ländern und Kommunen werden zentral über das **Open Data Portal GovData** veröffentlicht, bzw. in einem Metadatenkatalog referenziert. Das Portal bietet verschiedene Suchfunktionen zur Recherche von Daten an. Unter anderem kann auf www.govdata.de über ein Suchfeld auf mehr als 20.000 Datensätze⁴ zugegriffen und nach inhaltlichen Kriterien, Datenformaten oder zeitlicher Abdeckung gefiltert werden. Beispiele der im GovData referenzierten Datensätze sind Finanzhaushaltsdaten, Wahldaten, Kriminalstatistiken, Bebauungspläne oder auch Ergebnisse aus der amtlichen Lebensmittelüberwachung.

GENESIS (Gemeinsames neues statistisches Informationssystem) ist die Haupt-Datenbank des **Statistischen Bundesamtes (destatis)**. Über GENESIS-Online⁵ hat ein Besucher die Möglichkeit, Statistiken über eine Stichwortsuche aufzurufen, oder eigene

⁴ Stand Dezember 2018

⁵ <https://www-genesis.destatis.de/genesis/online>

Abfragen und Auswertungen vorzunehmen. Diese können anschließend in Excel, CSV oder HTML exportiert werden. Das Datenangebot erstreckt sich über Themenbereiche wie Demografie, Bildung, oder Wirtschaft. Hierzu zählen Angaben zur Arbeitslosigkeit, Insolvenzstatistik, oder Einkommensverteilungen.

Eine bekannte und häufig verwendete Open Data Quelle sind Messwerte von meteorologischen Dienstleistern, wie sie beispielsweise vom **Deutschen Wetterdienst (DWD)** veröffentlicht werden. Der DWD bietet verschiedene Datensätze zu Wettervorhersagen sowie aktuellen und historischen Wetterdaten an. Dabei können die Daten manuell von einem FTP Server heruntergeladen werden.

Über einen Download werden auch Geodaten von **OpenStreetMap (OSM)** angeboten. Dabei gibt es mehrere Anbieter die regelmäßig Kartendaten aus der OSM Datenbank extrahieren und auf bestimmten Webseiten zum Download anbieten. So können beispielsweise komplette Infrastrukturdaten für ganz Deutschland über eine Datei heruntergeladen werden⁶. Dabei werden die Dateien in einem stark komprimierten und XML-ähnlichen Format angeboten.

Eine weitere Methode um an frei verfügbare Datenbestände zu gelangen, ist das sog. **Webscraping**. Dabei kann der Inhalt einer Webseite über den HTML Quelltext ausgelesen werden. Auf Abbildung 2 ist die Webseite www.meinprospekt.de für digitale Prospekte verschiedenster Discounter (Aldi, Lidl, usw.) dargestellt. Die Seite bietet zusätzlich eine Übersicht über Standorte von Filialen des jeweiligen Discounters an. Die Adresse einer Filiale ist in der HTML Struktur eingebettet und kann mit entsprechenden Programmen extrahiert werden.

⁶ Download OSM Daten für Deutschland: <http://download.geofabrik.de/europe/germany.html>

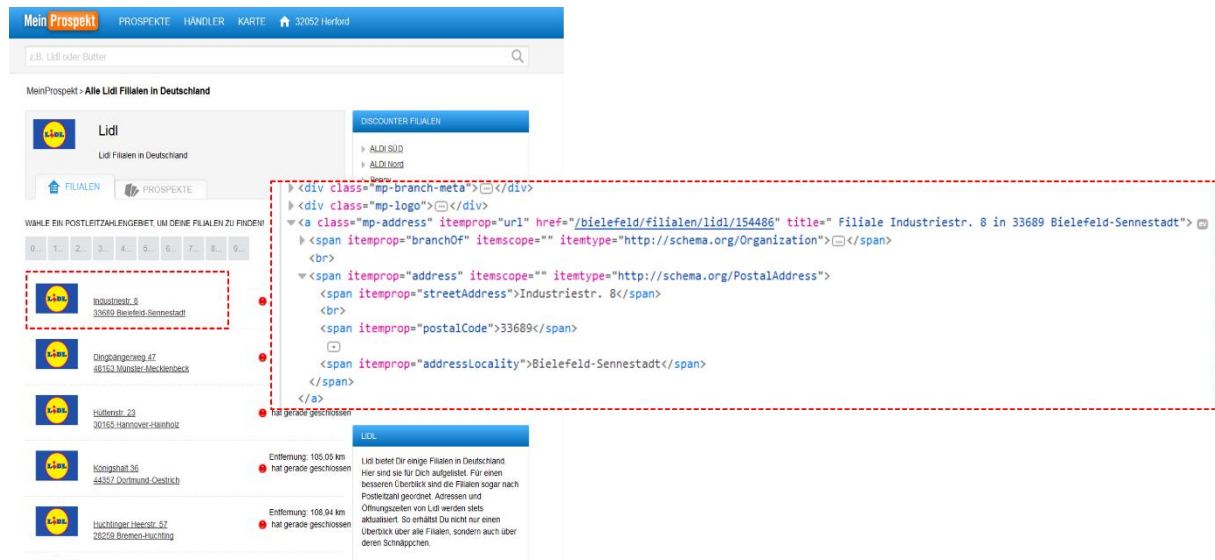


Abbildung 2 Suchergebnis aller Lidl Filialen in Deutschland mit entsprechendem HTML Code einer Filiale

In der vorgestellten Definition von Open Data wird gefordert, dass *jedermann* ohne technische Hürden auf veröffentlichte Daten zugreifen sollte. Da erforderlichen Kenntnisse für das Webscraping nicht vorausgesetzt werden können, zählt diese Art von veröffentlichten Daten nicht zu der Definition. Daher wird in dieser Arbeit die Definition von Open Data um das Webscraping sodass diese Daten verwenden zu können.

In Tabelle 1 werden zuvor genannten Open Data Quellen zusammengefasst. Dabei werden auch die jeweiligen Lizenzen genannt, unter denen die Daten veröffentlicht werden. Für nähere Informationen zu gebräuchlichen Lizenzmodellen von Open Data siehe (Meinert, 2018, S. 36 ff.). An dieser Stelle sei darauf hingewiesen, dass die genannten Datenquellen eine Verwendung der Daten im kommerziellen Sinne uneingeschränkt erlauben.

Tabelle 1 Exemplarische Auflistung einiger Open Data Quellen.

Open Data Bereitsteller	Inhalt	Datenzugang	Format	Bereitstellungs- intervall	Lizenz
Zensus	Bevölkerung, Gebäude und Haushalte	Manueller Download	.csv .xlsx	Alle 10 Jahre	Datenlizenz Deutschland
OpenStreet- Map	Geodaten über Wege/ Straßen, Areale, POIs)	Manueller Download	.osm .shp	täglich	Open Database Licence
Deutscher Wetter- dienst (DWD)	Klima und Prognosen	FTP-Server, Webservice	.zip .txt	jährlich bis stündlich	GeoNutzV
Statistisches Bundesamt (destatis)	Bevölkerung, Infrastruktur, Wirtschaft	Manueller Download	.csv .xlsx .html	jährlich bis täglich	Datenlizenz Deutschland
GovData Datenportal für Deutschland	Bevölkerung, Kultur, Umwelt, Gesundheit, Finanzen, usw.	Je nach Datenquelle	.pdf .csv .json	je nach Datenquelle	Je nach Datenquelle
Bundesamt für Geodäsie	Geodaten	Manueller Download	.shp .geojson	jährlich	GeoNutzV
Webseiten	Alle Themenbereiche	Webscraping	.html	je nach Datenquelle	-

Wie in diesem Kapitel vorgestellt, gibt es verschiedene Anbieter von Open Data. Dabei werden die Daten über unterschiedliche Schnittstellen sowie Formate angeboten und decken viele inhaltliche Themen ab. Das nachfolgende Kapitel soll nun einen Eindruck geben, welche Implikationen diese Vielfältigkeit für eine Verwendung der Daten hat.

2.1.2. *Open Data als Form von Big Data*

Wurde im vorherigen Kapitel die Vielfalt nur angedeutet, soll in diesem Kapitel die Eigenschaften an Beispielen veranschaulicht werden. Eine ähnliche Heterogenität von Daten wird auch in der Definition von Big Data adressiert. Da es für Big Data viele technische und methodische Lösungsansätze gibt, soll nachfolgend überprüft werden, inwieweit Open Data als Big Data angesehen werden kann. Die Anforderungen für eine Architektur lassen sich dann präziser ableiten. Daraufhin kann auf bestehende Konzepte für Big Data zurückgegriffen werden, um mit Open Data zu arbeiten.

Die Herausforderungen von Big Data wird üblicherweise über eine variable Anzahl von Eigenschaften, deren Begriffe mit V beginnen, eingeleitet. Es sollen im Folgenden Volumen, Variety, Velocity, Veracity und Value verwendet werden, um Parallelen zwischen Open Data und Big Data herauszustellen. Hier sei auf die Definitionen von (Fasel & Meier, 2016, S. 6) und (Gadatsch & Landrock, 2017, S. 3) verwiesen, auf die sich in diesem Abschnitt bezogen wird.

Volume: Big Data adressiert in erster Linie die Herausforderung der wachsenden Datenmenge. Es gibt alleine in Deutschland 75 Open Data Portale (Stand 2018), die seit 2014 immer mehr Daten anbieten. Durch bestehende Open Data Initiativen und Gesetzesgrundlagen, wie dem Informations-Freiheits-Gesetz (IFG) und E-Government-Gesetz (EGovG)⁷ ist ein stetiges Wachstum von Datenanbietern und Datenmenge zu erwarten.

Je mehr Open Data angeboten wird, desto mehr potenziell interessante Datenbestände könnte es für ein Unternehmen geben. Dabei variiert je nach Anbieter und Inhalt die Größe der bereitgestellten Daten zwischen Kilobyte und Gigabyte. Die Statistische Bundesamt stellt Wirtschaftsdaten über Deutschland in exportierbaren Tabellen bereit, die lediglich mehrere Hundert Kilobyte groß sind. Kartendaten von OpenStreetMap können für verschiedene Regionen heruntergeladen werden, die dabei mehrere Gigabyte umfassen. Um unter dieser Entwicklung stetig Open Data in unternehmensinterne Analysen mit einbeziehen zu können, benötigt man ein System zur Datenhaltung, der mit den wachsenden Datenmengen umgehen kann. Ein verbreiteter Big Data Ansatz ist die Verwendung von skalierbaren Datenspeicher, vgl. (Marz, 2016, S. 25).

⁷ IFG und EGovG bilden die rechtliche Grundlage für die Veröffentlichung von Daten öffentlicher Behörden, wie z.B. dem Bundesamt für Statistik und Deutschen Wetterdienst

Anforderung: Skalierbarer Speicher damit stetig weitere Open Data Ressourcen berücksichtigt werden können.

Variety: Die schiere Vielfalt von Datenanbietern geht mit der Herausforderung einher, dass unterschiedliche Schnittstellen und Datenformate berücksichtigt werden müssen. Zudem gibt es die gesamte Bandbreite an Strukturierungsgraden – von strukturierten Statistikdaten im CSV-Format, oder geografischen Daten mit semistrukturierten Formaten (XML/JSON), aber auch Bilddaten, welche beispielsweise von Satelliten der europäischen Raumfahrtbehörde (ESA) geliefert werden⁸. Auf der Suche nach geeigneten externen Datenquellen kann daher nicht festgelegt werden, welche Datenformate und technische Zugänglichkeit zu erwarten sind. Verschärft wird dieses noch durch die Tatsache, dass sich Schnittstellen Formate über die Zeit ändern können. Dabei können sich die Granularitäten des Dateninhalts oder die Qualität ändern. Daher ist es notwendig bezüglich Ablage und Verarbeitung der Daten flexibel zu sein. Diese Flexibilität kann nur erreicht werden, die Daten zunächst in ihrem ursprünglichen Format gespeichert. Diese Daten werden als „Rohdaten“ bezeichnet. Die Rohdaten durchlaufen dann individuelle Prozesse, in denen Transformationsschritte notwendig sind, um eine einheitliche Datenqualität zu erreichen. Danach können die Daten verknüpft und zusammenhängend analysiert werden.

Durch die individuelle Behandlung der Rohdaten müssen die Verarbeitungsprozesse in organisiert werden. In einer Architektur für Open Data müssen daher Qualitätszustände und Prozessschritte nachvollziehbar strukturiert werden.

Anforderungen:

Möglichkeit zur Ablage von Rohdaten.

Verarbeitung unterschiedlicher Formate.

Organisation der Daten, die Verarbeitungsschritte und Qualitätszustände nachvollziehbar machen.

Velocity: In vielen Big Data Definitionen wird diese Eigenschaft mit einer „sehr schnellen Verarbeitungszeit“ übersetzt. Hierbei ist gemeint, dass die Verarbeitung und Auswertung von Datenströmen nahezu in Echtzeit oder sog. „Near-time“ erfolgen soll. Beispielsweise werden bei einer Meinungsbild-Analyse von Twitter Posts aktuelle und schnelle Ergebnisse erwartet, um auf Ereignisse adäquat reagieren zu können. Die

⁸ Siehe Open Data Portal ESA: <http://open.esa.int/copernicus-sentinel-satellite-data/>

Echtzeitbereitstellung von Daten ist im Open Data Umfeld nicht weit verbreitet⁹, da es meistens eine Verzögerung zwischen Entstehung und Bereitstellung gibt. Beispielsweise wird der Kartenbestand von OpenStreetMap täglich aktualisiert, die Volkszählungsdaten des Zensus dagegen alle 10 Jahre. Nach weiteren Überlegungen wird jedoch klar, dass dies eine sehr enge Betrachtungsweise ist und einen Großteil von Open Data als Big Data ausklammern würde, die keine Echtzeitsysteme erfordern. Dabei basieren viele Big Data Technologien auf eine Batchverarbeitung, wie z.B. Hadoop, die für einen nicht unwesentlichen Teil der Anwendungsfälle im Big Data Bereich konzipiert wurden, vgl. (Marz, 2016, S. 109).

Alternativ kann die Eigenschaft Velocity (dt. Geschwindigkeit) als verkürzte „Time-to-Analytics“ verstanden werden und beschreibt dabei die Zeitspanne von Einbindung einer Datenquelle in ein System, bis zur Verwertung in der jeweiligen Analyse, vgl. (Fasel & Meier, 2016, S. 309). Bei anwachsenden Datenbeständen, wie unter Volume beschrieben wurde, erhöht sich die Datenmenge, die es zu verarbeiten gilt. Die Verarbeitungszeit erhöht sich bei herkömmlichen Systemen linear zu der Datenmenge. Um die Rechenzeit zu reduzieren, werden muss die Leistung der Rechereinheiten um CPU und RAM Kapazitäten erweitert werden. Um die Verarbeitungszeit mit anwachsenden Datenmengen zu skalieren, haben sich bei Big Data verteilte Verarbeitungskonzepte etabliert, vgl. (Marz, 2016, S. 121). So können Verarbeitungs- und Analyseprozesse auch mit großen Datenbeständen effizient gestaltet werden.

Anforderung: Skalierbare Verarbeitungssysteme für wachsende Datenbestände von Open Data.

Veracity (oder auch Validity): Wird meist mit „Wahrhaftigkeit“ übersetzt. Im Zusammenhang mit Big Data wird damit ausgedrückt, dass Daten in unterschiedlicher Qualität vorliegen und bei einer Verarbeitung und Analyse berücksichtigt werden muss. Da kein direkter Einfluss auf die Datenqualität von externen Daten wie Open Data genommen werden kann, kann diese für jede Datenquelle anders sein. Die Qualität der bereitgestellten Daten des Deutschen Wetterdienstes ist tendenziell hoch, die Klimadaten

⁹ Echtzeitverarbeitung ist vor allem im Kontext der sozialen Medien zu erwarten, falls sofern diese unter die Open Data Definition fallen

auch für interne Zwecke verwendet werden und Prüfverfahren durchlaufen¹⁰. Die Kartendaten von OpenStreetMap werden von vielen Nutzern der Community erzeugt. Hier liegt keine umfangreiche Qualitätssicherung¹¹ vor. Daher muss im Eingangsbereich einer Architektur für Open Data ein Prozess definiert werden, in denen die Daten zunächst untersucht werden, um die Datenqualität und Verwendbarkeit zu prüfen. Erst nach dieser Prüfung qualifiziert sich die Daten für die nachgelagerte Bereiche in der Architektur.

Anforderung: Da die Datenqualität einer Datenquelle vorher nicht bekannt ist, ist ein Prozess im Eingangsbereich erforderlich, in dem eine Prüfung durchgeführt werden kann.

Value: Laut Definition sollen Daten in einem Unternehmen einen Mehrwert generieren und den Unternehmenswert steigern. Daten über Finanzen, Lagerbestände, oder Stammdaten über Artikel und Kunden, sind für ein Unternehmen von großer Bedeutung. Diese sind für betriebliche Abläufe unerlässlich. Im Zusammenhang mit Open Data muss jedoch erst für jeden Datenbestand bewiesen werden, dass dieser einen Nutzen für ein Unternehmen hat. Da jedoch vorab nicht definiert werden kann, welcher Teil eines Datenbestandes relevant sein könnte, müssen die Daten zunächst in ihrer ursprünglichen Form als Rohdaten abgelegt werden. Dieses Argument ergänzt die Anforderung, die unter Variety für Open Data abgeleitet wurde. Die beschriebene Problematik kann auch aus einem Blickwinkel betrachtet werden. Für die Speicherung von Open Data (oder generell Daten) entstehen zunächst Kosten. Für Open Data, deren Mehrwert noch nicht ersichtlich ist, empfiehlt es sich eine günstige Speicherform aus dem Open Source Bereich zu verwenden. Andererseits sind Mehrwerte nicht auf Anhieb erkennbar. Testweise müssen Verknüpfungen mit bestehenden Daten erstellt werden. So können auch Anwendungsszenarien erzeugt oder getestet werden. Für diesen Zweck sollte ein Bereich definiert werden, in dem eine solche Verprobung durchgeführt werden kann. In der Literatur wird häufig der Begriff „Sandbox“ verwendet, der einen gekapselten Bereich darstellt für explorative Analysen¹².

¹⁰ Siehe Dokumentation DWD:

https://www.dwd.de/DE/forschung/wettervorhersage/met_fachverfahren/weterradarverfahren/qualitaetssicherung_weterradardaten_node.html

¹¹ Siehe Dokumentation OSM: <https://wiki.openstreetmap.org/wiki/DE:Qualit%C3%A4tssicherung>

¹² Siehe Definition: <https://www.itwissen.info/Sandbox-sandbox.html>

Anforderungen:

Die Speicherform sollte kostengünstig sein, da der Nutzen einiger Datenbestände im Vorhinein nicht absehbar ist und im Zweifel hohe Kosten durch die Datenhaltung entstehen.

Um eine Wertsteigerung zu erreichen wird ein Sandbox Bereich benötigt, um ggf. Dinge zu probieren, wie die Verknüpfung neuer Daten mit bestehenden Datenbeständen.

Diese Darstellung über die 5 V's sollte zeigen, dass für Open Data klassische Big Data Herausforderungen übertragbar gelten. Die daraus abgeleiteten Anforderungen für eine Architekturlösung soll nun zusammengefasst werden.

1. **Skalierbarer Speicher und Verarbeitungskonzept** um weitere Datenquellen einzubinden und auf das Wachstum einzelner Datenbestände reagieren zu können.
2. Die Speicherlösung sollte **kostengünstig** sein, da der Nutzen einiger Datenbestände im Vorhinein nicht absehbar ist und im Zweifel hohe Kosten durch die Datenhaltung entstehen.
3. Die **Speicherung und Verarbeitung** der Daten sollte **über eine Struktur** organisiert werden, in denen die Verarbeitungsschritte und Qualitätszustände ersichtlich sind. Somit können nicht nur Daten, sondern Prozessschritte logisch partitioniert werden.
4. Es wird eine Möglichkeit benötigt, die Daten im **Rohformat** abzulegen. Einerseits ermöglicht dies die Flexibilität, Open Data jeglichen Formates zu berücksichtigen. Andererseits steht vorher nicht fest, welcher Teil der Daten von Nutzen sein können.
5. Da die Datenqualität einer Datenquelle vorher nicht bekannt ist, wird ein **Prozess im Eingangsbereich** erforderlich, in dem eine Prüfung durchgeführt werden kann. Danach kann entschieden werden, ob sich die Daten für die weitere Verwendung eignen.
6. Um den Nutzen von Open Data zu testen, ist **Sandbox Bereich** erforderlich, um Analysen oder Verknüpfung neuer Daten mit bestehenden Datenbeständen durchzuführen.

2.1.3. *Eignung traditioneller Data Warehouses*

Es wurde im vorangegangenen Kapitel festgestellt, dass eine Verwendung von Open Data mit Herausforderungen aus dem Big Data Kontext einhergehen. Es soll nun geprüft werden, inwieweit klassische relationale Systeme, wie das des Data Warehouses (DWH), für Open Data geeignet sind.

In einem Unternehmen ist es üblich, dass Daten zentral in einem **Data Warehouse (DWH)** gespeichert werden. Sie unterstützen Entscheidungsprozesse und können für analytische Zwecke (z.B. Reports) verwendet werden. Diese Daten werden in regelmäßigen Abständen von den operativen Datenquellen ins DWH geladen vgl. (Kimball & Ross, 2013, S. 21).

Nach (Kimball & Ross, 2013, S. 19 ff.) kann der Aufbau eines DWH in vier Schichten beschrieben werden (siehe Abbildung 3). Die **externen Datenquellen**, wie Customer-Relationship-Management (CRM), Enterprise-Resource-Planning (ERP), oder andere operative Systeme, bilden die Grundlage der im DWH gehaltenen Daten. Um diese Daten zu extrahieren, werden diese zunächst in eine vorgeschaltete Schicht, der sog. **Ladebereich** geladen. Damit die Daten in ein bestimmtes Zielmodell überführt werden können, müssen Bereinigungs- und Transformationsprozesse durchgeführt werden. Ist dies erfolgt, können die Daten in eine zentrale Datenhaltungsschicht geladen werden, dem sog. **Kernschicht**. Das grundlegende Vorgehen wird dabei mit Extract-Transform-Load (ETL) beschrieben. In Kernschicht liegen Daten eines Unternehmens in normalisierter und standardisierter Form vor. Um Fachabteilungen (Rechnungswesen, Finanzen, etc.) eines Unternehmens nicht den gesamten Datenbestand bereitzustellen, sondern der für die individuellen Zwecke relevanten, werden Datenausschnitte, sog. **Data Marts** gebildet. Die Kernschicht sowie Data Marts zählen nach (Kimball & Ross, 2013, S. 19) zur **Präsentationsschicht**. Auf dieser können analytische Abfragen durchgeführt und z.B. in Leistungskennzahlen aggregiert werden. Die daraus abgeleiteten Informationen können in verschiedenen Anwendungen in der **Applikationsschicht** verwendet werden.

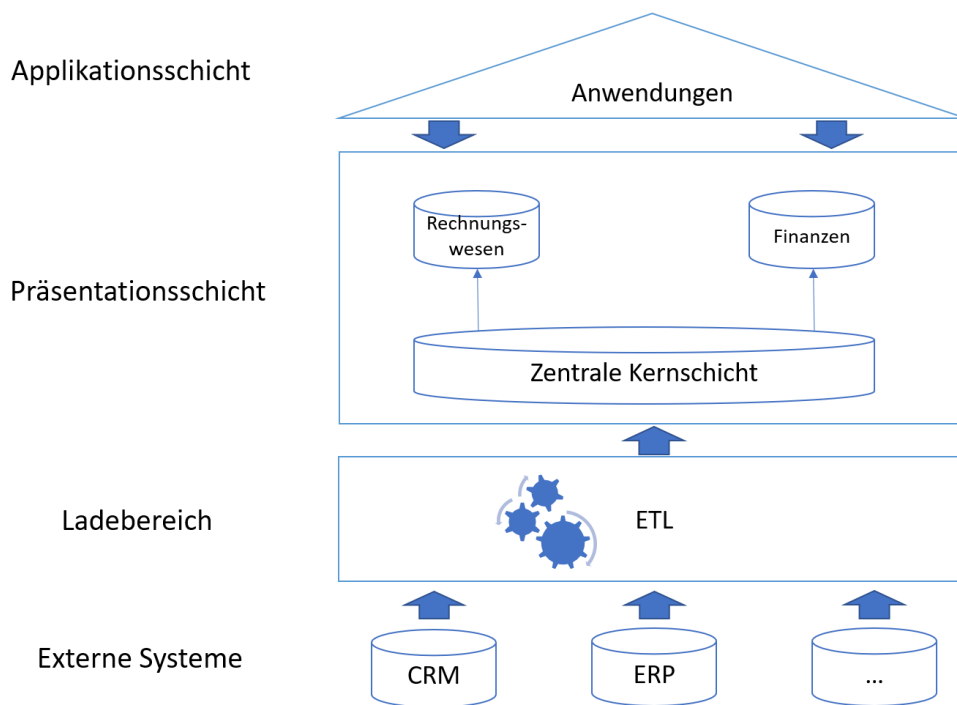


Abbildung 3 Konzeptueller Aufbau einer DWH in vier Schichten

Im Folgenden soll anhand der zuvor abgeleiteten Anforderungen überprüft werden, inwieweit ein DWH System für Open Data geeignet ist.

Skalierbarer Speicher und Verarbeitungskonzept um weitere Datenquellen einzubinden und auf das Wachstum einzelner Datenbestände reagieren zu können.

Die Speicherlösung sollte **kostengünstig** sein, da der Nutzen einiger Datenbestände im Vorhinein nicht absehbar ist und im Zweifel hohe Kosten durch die Datenhaltung entstehen.

Falls kontinuierlich weitere Open Data Quellen eingebunden werden sollen, wächst der Datenbestand im DWH und wird durch die relationale Darstellung immer komplexer. Es ist intuitiv klar, dass bei wachsenden Datenmengen und einem komplexeren Datenmodell die Verarbeitungsgeschwindigkeit jeglicher Operationen mehr Rechenzeit erfordern. Die zusätzlichen Spalten oder Tabellen können beispielsweise bei Verknüpfungsoperationen zu immensen Berechnungen führen, vgl. (Grover, 2015, S. 312). Für eine gleichbleibende Verarbeitungszeit und Speicherkapazität müsste daher die Infrastruktur stetig angepasst werden. Hierbei können höhere Kosten für Lizenzen entstehen, für Daten, dessen Nutzen noch nicht erkennbar ist.

Aus Gründen der Wirtschaftlichkeit ist daher ein skalierbarer Speicher geeigneter, der nicht von Lizenzvereinbarungen kommerzieller Anbieter von DWH (wie Oracle, IBM oder Teradata) über Speicherbedarf und Rechenleistung abhängig ist, vgl. (Grover, 2015, S. 309). Daher werden nur geschäftsrelevante Daten ins DWH aufgenommen, um die Kosten möglichst gering zu halten.

Die **Speicherung und Verarbeitung** der Daten sollte **über eine Struktur** organisiert werden, in denen die Verarbeitungsschritte und Qualitätszustände ersichtlich sind. Somit können nicht nur Daten, sondern Prozessschritte logisch partitioniert werden.

In erster Linie existieren zwei Qualitätsstufen im DWH – im Bereich der externen Datenquelle und Präsentationsschicht.

Das Ziel des DWH Konzept ist eine zentrale Datenbasis zu schaffen, die konsistent ist und verschiedene Qualitätsanforderungen erfüllt. Da nur bereinigte und standardisierte Daten in die Kernschicht gelangen, haben die Daten einen einheitlichen Qualitätszustand. Die Datenqualität vor der Präsentationsschicht, entspricht im Allgemeinen nicht den definierten Qualitätsstandards. Das bedeutet, dass alle Bereinigungs- und Transformationsprozesse zwischen diesen beiden Bereichen konzentriert werden. Da Open Data in unterschiedlichen Qualitäten vorliegt, werden mitunter sehr viele Transformationsschritte benötigt, um die Qualitätsanforderungen zu genügen. Für Open Data besteht in Vorhinein kein Wissen um die Struktur und den Inhalt der Daten vollständig zu verstehen. Daher die vollständige Verarbeitungskette erst definiert werden, wenn mit den Daten gearbeitet wird. Deswegen empfiehlt es sich, die Transformationsschritte zu entzerren und auf mehrere Bereiche aufzuteilen.

*Es wird eine Möglichkeit benötigt, die Daten im **Rohformat** abzulegen. Einerseits ermöglicht dies die Flexibilität, Open Data jeglichen Formates zu berücksichtigen.*

Die Datenquellen eines DWH sind meist semi-strukturiert (JSON oder XML) oder strukturiert (JDBC Tabellen, CSV), da sie aus relationalen und NoSQL Systemen stammen, vgl. (Grover, 2015, S. 311). Im Open Data Bereich werden meist vielfältige

Datei-Formate verwendet, die unstrukturiert bis strukturiert sein können, oder nicht üblichen Formaten entsprechen. Beispielsweise verwendet OpenStreetMap ein spezielles komprimiertes und XML-ähnliches Dateiformat, welches von einem Programm zunächst dekomprimiert und anschließend in eine flache Struktur gebracht werden kann. Daten aus dem öffentlichen Sektor, wie dem Bundesamt für Statistik, liegen häufig in einem Excel-Dokument vor. Da Excel-Dateien in einem binären Format vorliegen, müssen diese zunächst in eine gängige Struktur wie CSV konvertiert werden.

Um die Daten dieser Formate in einem DWH zu speichern, müsste die Formatierung in einem externen Dateisystem erfolgen, die im konzeptuellen Aufbau des DWH nicht definiert wird. Für Open Data wäre eine Architektur sinnvoll, die diesen Strukturierungs- bzw. Formatierungsprozess abdeckt und jegliche Datenformate berücksichtigen kann.

*Um den Nutzen von Open Data zu testen, ist **Sandbox Bereich** erforderlich, um Analysen oder Verknüpfung neuer Daten mit bestehenden Datenbeständen durchzuführen. Da die Datenqualität einer Datenquelle vorher nicht bekannt ist, wird ein **Prozess im Eingangsbereich** erforderlich, in dem eine Prüfung durchgeführt werden kann. Danach kann entschieden werden, ob sich die Daten für die weitere Verwendung eignen.*







Bevor Daten ins DWH geladen werden, muss zunächst ein Schema entworfen oder ergänzt werden – nach dem Prinzip *schema-on-write*, vgl. (Grover, 2015, S. 1). Hierfür ist es notwendig die Struktur der Daten vollständig zu verstehen. Liegt dieses nach Absprache mit diversen Fachabteilungen vor, können die Daten zunächst ins System geladen werden.

Da es sich bei Open Data um externe, gegebenenfalls auch schlecht dokumentierte Daten handelt, muss zunächst ein Verständnis für die Daten entwickelt werden. Hierfür müssen die Daten genauer analysiert werden. Nach (Grover, 2015, S. 310) sieht der theoretische Aufbau des DWH keinen Bereich vor, in denen eine vorgeschaltete explorative Analyse erfolgen könnte. Entweder müssten die Daten direkt in die Kernschicht geladen, oder auf separate Systeme ausgewichen werden. Ersteres erfordert die Definition neuer ETL-Strecken, was wiederum Zeit in Anspruch nehmen würde. Die Betreibung von weiteren Systemen zur Datenexploration würde zusätzliche Kosten verursachen. Üblicherweise ist das Vorgehen im Rahmen des DWH anders. Hier werden Anforderungen von Business Analysten definiert (z.B. weitere Produktdaten werden benötigt), daraufhin werden

entsprechende Daten von den externen Datenquellen bezogen und über ETL beschafft, vgl. (Grover, 2015, S. 309). Danach können die neuen Daten in die zentrale Kernschicht geladen und anschließend in die Data Mart Schicht propagiert werden. Erst zu diesem Zeitpunkt stehen die Daten den Analyseinstrumenten zur Verfügung und können explorativ untersucht werden. Dieses strikt organisierte Vorgehen bietet nicht die Flexibilität, um effizient weitere Daten aus dem Open Data Bereich zu berücksichtigen. Dadurch wären Analysen mit Open Data unter Umständen gar nicht möglich.

Die Anforderungen für Open Data können nicht, oder nur mit erhöhtem Aufwand über eine DWH Lösung umgesetzt werden (siehe Tabelle 2), jedoch müssten viele Aktivitäten auf externe Systeme ausgelagert oder spezielle Lösungen herbeigeführt werden. Die Stärke des DWH ist konsistente Datenhaltung mit standardisierten Prozessen, die jedoch eine Verwendung von Open Data erschweren. Durch die mangelnde Flexibilität kann nicht auf die dynamische Entwicklung des Open Data Bereich reagiert werden. Für eine flexiblere Lösung wird daher im nächsten Abschnitt das Data Lake Konzept vorgestellt und auf bestehenden Anforderungen hin geprüft.

Tabelle 2 Prüfung der Umsetzbarkeit von Anforderung für Open Data mit einem DWH

Anforderungen	DWH
Skalierbarer Speicher	 ¹³
Speicherung und Verarbeitung über eine Struktur organisiert	
Rohdaten mit unterschiedlichen Formaten	
Eingangsbereich für die Erstablage von Daten für weitere Untersuchungen	 ¹⁴
Kostengünstig skalierbare Ressourcen	
Bereich für schnelle Analysen (Sandbox)	

¹³ Ein DWH könnte ebenfalls mit einem skalierbaren Speicher umgesetzt werden. In der Praxis ist dies jedoch seltener der Fall.

¹⁴ Nicht im DWH Konzept berücksichtigt, technisch jedoch umsetzbar

2.2. *Data Lake*

Es wurde gezeigt, dass die Anforderungen, die mit einer Verwendung von Open Data einhergehen, nicht oder nur schwierig mit traditionell relationalen Ansätzen zu lösen sind. Daher wird im Folgenden das Konzept eines Data Lakes vorgestellt, welches Lösungen für die genannten Herausforderungen bietet. Zunächst wird eine Begriffsdefinition vorgenommen und Kerneigenschaften des Data Lakes vorgestellt. Danach können die definierten Anforderungen für Open Data mit den Ansätzen des Data Lakes verglichen werden.

2.2.1. *Begriffsdefinition und Kerneigenschaften*

In der Literatur findet eine sehr undifferenzierte Betrachtung des Data Lake Begriffs statt, die auch unter dem Namen „Big Data Lake“ (Coiera, 2014), „Enterprise Data Lake“ (Terrizzano, 2015), oder „Raw Data Lake“ zu finden sind. Hierbei werden häufig Definitionen formuliert, die zum Themenfokus der jeweiligen Arbeit passen, oder nur relevante Teilaspekte betrachten. Zum einen existieren zu grobe Definition wie aus (Gadatsch & Landrock, 2017, S. 9), in dem der Data Lake als

„IT-Infrastruktur zur Sammlung und Aufbereitung von großen D

angesehen wird. Darunter könnte ebenfalls ein DWH verstanden werden. Zum anderen sind einige Betrachtungsweisen zu sehr an eine gewisse Technologie gebunden, wie nach (Coiera, 2014)

The Big Data Lake has been conceptualized as a single data repository for an enterprise, typically designed to work in conjunction with Hadoop.

Außerdem wird unter einem Data Lake auch ein Rohdatenspeicher verstanden, welcher als Archiv für Applikationsdateien angedacht werden kann, vgl. (Fasel & Meier, 2016, S. 388). Diese Betrachtung stellt zwar den Grundgedanken heraus, jedoch werden hier nicht die weiterführenden Speicher- und Verarbeitungsansätze genannt, die den Data Lake zu anderen Lösungskonzepten unterscheidet. Diese unscharfe Betrachtung trägt dazu bei, dass kein einheitliches Verständnis unter dem Begriff besteht.

Die Eigenschaften eines Data Lakes wurden aus vielen Definitionen, die in der Literatur zu finden sind, zusammengetragen und verdichtet.

In dieser Arbeit wird der Data Lake als ein Konzept für das **zentrale Datenmanagement** verstanden und dient zur **Entscheidungsunterstützung** – so wie das DWH auch, das Vorgehen ist dabei jedoch anders.

Im Data Lake werden heterogene **Rohdaten** in einem **skalierbaren (meist verteilten) Speicher** abgelegt, in welcher Form auch immer sie von einem Quellsystem zur Verfügung gestellt werden. Der Fokus liegt dabei auf der möglichst **agilen und schnellen Analyse**. Dabei sieht die Architektur **verschiedene logische Partitionen** vor, um Prozesse und Datenbestände über verschiedene Ebenen zu organisieren.

Die Kerneigenschaften, welche in der Literatur zu finden sind, werden im Folgenden nochmal zusammengefasst und die zugrundeliegende Idee dahinter erläutert.

Zentraler und skalierbarer Speicher (Miloslavskaya & Tolstoy, 2016, S. 301)

Der Data Lake sollte mit als zentraler Speicher für unternehmensweite Datenbestände aus strukturierten und unstrukturierten Daten betrachtet werden. Die technische Ausgestaltung sollte mit allen Anwendungen aus jeglichen Bereichen in einem Unternehmen integriert sein, sodass von überall die Daten zugänglich sind.

Dies soll Datensilos vermeiden, in denen die jeweiligen Fachabteilungen ihre eigene Datenhaltung unabhängig verwalten und organisieren. Nur so kann das volle Potenzial einer zentralen Lösung ausgeschöpft werden.

Da in einem Unternehmen immer mehr Daten anfallen und auch externe Datenbestände wie Open Data herangezogen werden, wird der Speicherbedarf sukzessive steigen. Hierfür werden oft Lösungen von verteilten Systemen angewendet, die sich horizontal über mehrere Rechnerknoten skalieren lassen. Nach (LaPlante & Sharma, 2016, S. 8) ist dieser verteilte Ansatz häufig kostengünstiger als herkömmliche Speicherkonzepte.

Rohdaten (Terrizzano, 2015, S. 1)

Im Vergleich zum klassischen Data Warehouse werden in einem Data Lake, die eingehenden Daten nicht mit komplexen Datenqualitäts- und Integrationsverfahren in definierte Strukturen überführt, sondern direkt in ihrer Ursprungsform abgelegt. Das Ablegen von ungefilterten und unbearbeiteten Rohdaten hat den Vorteil, dass die weitere Verarbeitung der Daten flexibel gestaltet werden kann, da die Daten in der größtmöglichen Granularität vorliegen. Somit bleiben alle Möglichkeiten offen, neue Informationen und Erkenntnisse zu generieren. Der Nachteil ist hierbei, dass die Rohdaten viel umfangreicher sind – dies wird jedoch über die zuvor genannte Eigenschaft des skalierbaren Speichers abgefangen.

Wie in (Terrizzano, 2015, S. 1) formuliert, wird die Schematisierung vom „point of entry“ auf den „point of use“ verlagert. Demnach wird die Struktur der Daten erst definiert, wenn diese für bestimmte Anwendungen gebraucht werden. Das entsprechende Paradigma dazu ist das sog. *schema-on-read*.

Im Gegensatz zu der Vorgehensweise in einem DWH nach ETL, wird hier die Reihenfolge auf ELT geändert. Die Daten werden zunächst ins System geladen damit sie allen Anwendern zur Verfügung stehen, bevor diese individuell transformiert werden können.

Datenorganisation in Bereichen (LaPlante & Sharma, 2016, S. 8)

Durch die Speicherung von Rohdaten aus verschiedenen Quellen kommen unweigerlich Qualitätsunterschiede der einzelnen Daten auf, was viele Autoren als Problem dieses Ansatzes sehen. Daher ist eine Organisation sinnvoll, welche Datenherkunft sowie Qualitätsunterschiede berücksichtigt.

Die von der Datenquelle extrahierten Rohdaten werden in einem sog. **Raw Data** Bereich abgelegt. Danach werden die Daten mithilfe von Verfahren zur Datenbereinigung sowie Anreicherung aufbereitet. Die aufbereiteten Daten (**Curated Data**) werden dann einer großen Gruppe von Anwendern bereitgestellt, die mit verschiedenen Abfragewerkzeugen auf die behandelten Daten zugreifen können. Der aufbereitete Teil des Data Lakes entspricht dabei vom Konzept her der Kernschicht des DWH.

Für die anwendungsspezifische Weiterverarbeitung und Analyse, werden verschiedenen Arbeitsbereiche definiert, auch **Working Zones** genannt, vgl. (M. & S., 2017). Hier können auf den Daten klassische statistische Verfahren zu Machine Learning durchgeführt werden. Dieser Bereich kann analog zu der Applikationsschicht im DWH gesehen werden.

Bevor die Daten in die jeweiligen Bereiche gelangen, müssen zuvor Transformationsschritte durchlaufen werden, die diese in einen einheitlichen Zustand bringen. Dadurch kann sichergestellt werden, dass die Daten in den jeweiligen Schichten erwartbare Eigenschaften beispielsweise bezüglich Standards und Validität besitzen.

In (Miloslavskaya & Tolstoy, 2016, S. 303) wird außerdem eine Trennung von internen und externen (*third parties*) Datenbeständen vorgeschlagen. Diese Unterteilung kann sinnvoll sein, da beide Bereiche unterschiedliche rechtliche und organisatorischen Richtlinien unterliegen können. Beispielsweise erfordern Kundendaten eine Anonymisierung oder Pseudonymisierung der Daten. Die Verwendung von externen Daten wie Open Data orientiert sich an die jeweilige Lizenzbedingung. Die Zusammenführung dieser Bestände muss dann zunächst überprüft werden. Ebenso gibt es einen Ansatz zur zeitlichen Aufteilung von aktuellen (*current data*) und historischen Daten (*historical data*) sowie einem Bereich zur Ablage von temporären Daten (*temporary data*). Diese Einteilung soll dabei die Such nach relevanten Daten erleichtern.

Hier sind gewisse Freiheitsgrade bei der Gestaltung des Data Lakes zu erkennen, die je nach Anforderungen angepasst werden können. Trotz guter Organisation können sich jedoch einige Fallstricke bei Anwendung des Data Lakes Konzepts ergeben, die in Kapitel 2.4 vorgestellt werden.

Agile und schnelle Analyse (Miloslavskaya & Tolstoy, 2016, S. 303) und (LaPlante & Sharma, 2016, S. 3 ff.)

Wie vorherigen Punkte andeuten lassen, ist das Konzept so ausgelegt, dass der Fokus auf einer schnellen und unkomplizierten Integration von Daten in das System liegt. Hierfür erzeugt man größtmögliche Flexibilität die letztendlich zu einer schnelleren Entscheidungsunterstützung verhelfen soll. In der Literatur wird dabei ein weiterer unabhängiger Bereich genannt, der als analytischen Sandbox bezeichnet wird (LaPlante & Sharma, 2016, S. 52). Daten aus allen restlichen Bereichen des Data Lakes können hier für weitere Untersuchungen abgelegt werden.

So kann bei Bedarf in kurzer Zeit unternehmenseigene Daten mit den neuen Daten (z.B. Open Data) analysiert und kombiniert werden.

Data Lakes adressieren üblicherweise eher explorative Anwendungsfälle, bei denen neue Fragestellungen in der Sandbox unter Einbeziehung bisher nicht genutzter Daten

untersucht werden. Im Erfolgsfall werden diese dann in einen etablierten Verarbeitungsprozess bereitgestellt. Im Falle von Open Data kann hier überprüft werden, wie valide oder relevant die Daten einer Datenquelle sind und ob diese einen Nutzen für das Unternehmen stiften.

Ziel ist vor allem aber die schnellere Umsetzung neuer Anforderungen, um auf dynamische Veränderungen im Geschäftsumfeld reagieren zu können. Die Tabelle 3 soll den direkten Vergleich des Data Lakes mit dem klassischen DWH gegenüberstellen.

Tabelle 3 Prüfung der Umsatzbarkeit von Anforderung für Open Data mit einem DWH und Data Lake

Anforderungen	DWH	Data Lake
Skalierbarer Speicher	—	✓
Speicherung und Verarbeitung über eine Struktur organisiert	✓	✓
Rohdaten unterschiedlicher Formate	✗	✓
Eingangsbereich für die Erstablage von Daten für weitere Untersuchungen	—	✓
Kostengünstig skalierbar	✗	✓ ¹⁵
Bereich für schnelle Analysen (Sandbox)	✗	✓

2.3. Datenmanagement im Data Lake

Der Data Lake lässt sich in verschiedene Bereiche einteilen (auch *Zonen* genannt), welche Auskunft über den Qualitätszustand der Daten geben und in denen verschiedene Verarbeitungsprozesse organisiert werden. Wie in der Kerneigenschaft „Datenorganisation in Bereichen“ bereits angeklungen, werden in der Literatur diesbezüglich verschiedenste Lösungsansätze diskutiert. Daher besitzt man gewisse Freiheiten beim Entwurf des Data Lakes und muss je nach Anforderungen individuelle Architekturentscheidungen treffen.

¹⁵ Kann aber auch von der technischen Umsetzung abhängen. Beispielsweise werden Data Lakes in der Cloud mit Lizenzmodellen angeboten.

Vorliegend wird der Data Lake in die Bereiche Landing, Raw, Curated und Working sowie Sandbox Zone eingeteilt¹⁶. Diese beinhalten jeweils unterschiedliche Transformationsschritte um zum einen die Rohdaten in eine verwertbare Form zu bringen und zum anderen daraufhin anwendungsspezifische Analysen durchzuführen. Im Gesamtbild wird der Prozess von der Datenbeschaffung bis zur Datenanalyse über die genannten Bereiche vollständig abgebildet. Begleitend werden die Metadaten übergreifend zu den entsprechenden Daten erstellt und angepasst. In diesem Kapitel werden die einzelnen Schritte und Aktivitäten detailliert definiert, bevor sie im Kapitel 3 und 4 implementiert und angewendet werden.

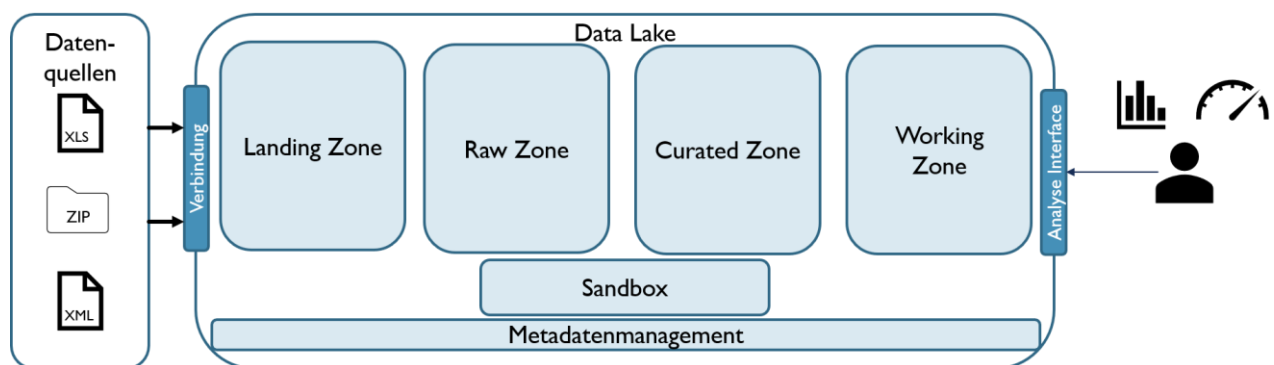


Abbildung 4 Logische Struktur des Data Lakes

Der auf Abbildung 4 dargestellte Data Lake gliedert sich in folgende Bereiche:

Datenquellen: Als Datenquellen stehen interne und externe Datenbestände zur Verfügung. Interne Unternehmensdaten aus analytischen oder operativen Systemen können gezielt extrahiert werden. Der Bereich von externen Daten wird um Open Data erweitert und stehen als Datenquellen zur Verfügung.

Landing Zone: Im Eingangsbereich werden zunächst potenziell interessante Datenbestände identifiziert, in Rohformat abgelegt und mit Metadaten beschrieben.

Raw Zone: Der Rohdatenspeicher enthält Daten, die zuvor in ein Format gebracht wurden, die von üblichen Programmen ausgelesen werden können. Derweil die Daten in der Landing Zone im z.B. Excel-Format vorlagen, liegen diese in der Raw Zone als CSV Datei vor.

¹⁶ Zu berücksichtigen ist, dass es sich hier um eine logische Partitionierung handelt. Die physische Speicherform ist hiervon unabhängig und wird in Kapitel 3 näher beleuchtet.

Curated Zone: Bildet die zentrale Schicht für die Daten eines Unternehmens. Analog zur Kernschichten in einem DWH, liegen hier konsolidierte und integrierte Daten vor. In weiteren Bereichen können je nach Anwendung Teildaten aus dem zentralen Bereich extrahiert werden und weiterverarbeitet werden (ähnlich zum Data Mart).

Working Zone: Aus den Daten im anwendungsspezifischen Bereich der Curated Zone, können verschiedene Analysen durchgeführt werden, um Erkenntnisse und Informationen abzuleiten. Hierbei können Transformationen angewendet werden, die für das jeweilige Analyseverfahren von Nöten sind, z.B. die Vorverarbeitung für Machine Learning Algorithmen.

Sandbox: Eine Zone zur generellen Datenexploration. Hier können Ressourcen aus unterschiedlichen Kontexten und Schichten eingebracht werden. Beispielsweise kann für bestehende Anwendung geprüft werden, inwieweit weitere Datenquellen die Analyse ergänzen könnten.

Metadatenmanagement: Um sicherzustellen, dass die Daten über den Gesamtprozess hinweg auffindbar und nutzbar sind, werden beschreibende Metadaten zu den entsprechenden Daten gepflegt.

Die Bereiche und Vorgänge werden im Weiteren detaillierter beschrieben, um eine möglichst genaue Vorstellung über die in dem Data Lake stattfindenden Prozesse zu vermitteln.

2.3.1. *Landing und Raw*

Nachdem Daten identifiziert werden, werden diese in ihrer ursprünglichen Form mit der höchstmöglichen Granularität im Data Lake gespeichert. Daher wird zunächst ein Bereich vorgesehen, in dem die Daten im Rohdatenformat abgelegt werden können. Diese Schicht wird in der Literatur meist als „Raw“ bezeichnet. Nicht aufbereitete Rohdaten können von der Struktur und dem Inhalt nur schwer zugänglich und sollten daher in eine Form gebracht werden, bei dem diese mit reduziertem Zeitaufwand konsumiert werden können, vgl. (Terrizzano, 2015, S. 2). Daher kann es sinnvoll sein, gewisse vorgeschaltete Transformationen an den Rohdaten durchzuführen, die eine weitere Verarbeitung erleichtern. Im Open Data Bereich kann dies beispielsweise bedeuten, dass die für Bürger/Innen visuell aufbereiteten Excel-Dateien in eine strukturierte CSV Datei

umgeformt werden müssen, damit diese leichter von Programmen eingelesen werden können.

Daher sieht die Architektur für Open Data eine „Landing Zone“ vor, in der die Rohdaten in ihrem Urzustand zeitweise abgelegt werden können, bevor diese nach einer geringen Vorverarbeitung in die „Raw Zone“ gelangen. Auch wenn hierdurch eine gewisse Änderung an den Daten vorgenommen wird, befinden sich diese weiterhin in ihrer detailliertesten Form. Organisatorisch sollte der Zugriff auf die Bereiche Landing und Raw nur durch spezielle Nutzergruppen möglich sein. So kann gewährleistet werden, dass die Einbettung weiterer Datenquellen und Vorverarbeitung nicht willkürlich stattfindet und eventuell schützenswerte Daten nicht in unqualifizierten Anwendungen eingebunden werden. Für einen Überblick der Prozesse und einzelnen Bereiche siehe Abbildung 5.

Meist sind die Daten einer Datenquelle hinsichtlich ihres technischen Formats und rechtlichen Nutzbarkeit ähnlich. Daher werden diese in der Raw Zone für jeden Bereitsteller in separate Bereiche organisiert.

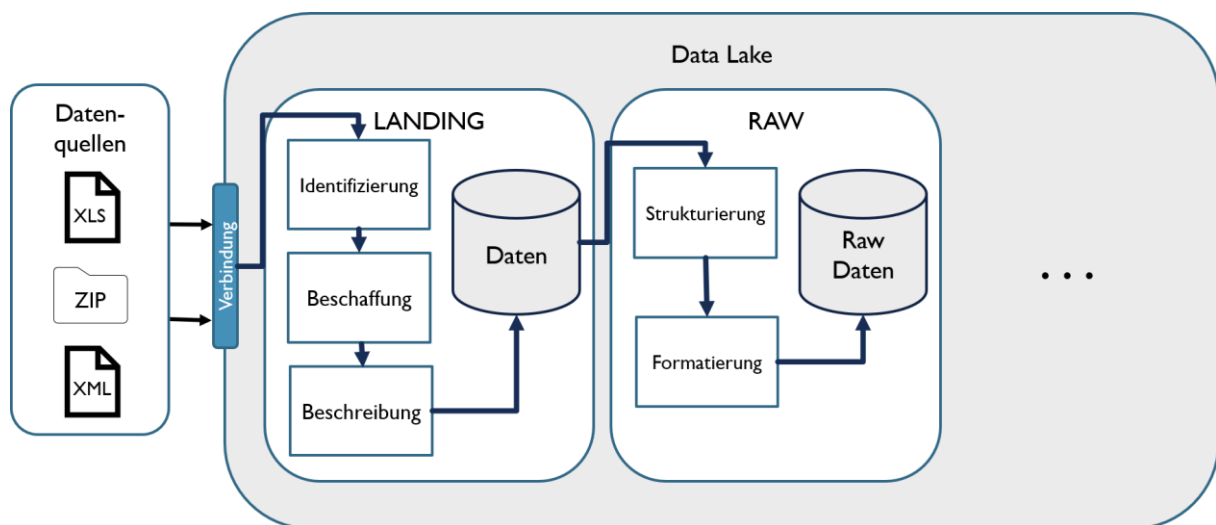


Abbildung 5 Initiale Prozesse der Landing und Raw Zone im Data Lake

Im Folgenden werden die genauen Prozesse von der Selektion der Datenquellen bis zur Abspeicherung im Raw Data Bereich aufgezeigt und erläutert.

Identifizierung – vgl. (Terrizzano, 2015, S. 4)

Bevor neue Daten (Open Data oder unternehmensintern) in den Data Lake gelangen, müssen zunächst verfügbare Datenquellen identifiziert werden.

In diesem Zusammenhang wird häufig das Prinzip von „garbage in, garbage out“ erwähnt, welches das Problem adressiert, dass die Qualität der eingehenden Daten die Qualität der

Analyseergebnisse beeinflussen. Demnach ist es wichtig bei der Auswahl von Datenquellen, insbesondere im Open Data Umfeld, bestimmte Aspekte zu berücksichtigen. Diese sollen über folgende Leitfragen formuliert werden, die individuell beantwortet werden müssen.

Welche Daten stehen zur Verfügung und welche sind bereits im Data Lake enthalten?

Passen die Daten aus den möglichen Datenquellen und meinem derzeitigen Bestand zeitlich und räumlich zueinander?

Welche rechtlichen Anforderungen müssen berücksichtigt werden? Sind die Intervalle (periodisch oder einmalig) in denen die Daten zur Verfügung gestellt werden geeignet für bestehende Anwendungsfälle? Wie lange bleibt das Datenangebot bestehen?

Liefern mir die Daten einen potenziellen Mehrwert in möglichen oder vorhandenen Anwendungen?

Beschaffung (physischer Transfer)

Halten die Daten den Selektionskriterien stand und haben sich für eine Aufnahme in den Data Lake qualifiziert, ist der nächste Schritt, diese technisch zu transferieren. Die Bereitstellung von Daten kann dabei über unterschiedliche Schnittstellen an erfolgen. Typischerweise stellen die Open Data Bereitsteller Ressourcen in einer großen Datei sog. *bulk data* zur Verfügung (OpenStreetMap Daten). Wahlweise auch in vielen kleinen (Wetterdaten vom DWD), die zum Download angeboten werden. Einige Anbieter, wie das Open Data Portal von GovData, bieten eine Abfragefunktionalität an, wodurch gezielt nach Daten gesucht werden kann. Dabei sollte die bulk data Form bevorzugt werden, da so der größtmögliche Datenumfang geladen und je nach Anwendungsfall gefiltert werden kann.

Beschreibung in Form von Metadaten - vgl. (LaPlante & Sharma, 2016, S. 45)

Die Daten allein sind wenig nützlich, wenn diese nicht entsprechend beschrieben werden. Nach der Ablage der Daten müssen daher Metainformationen zu den Daten erstellt werden. Je aussagekräftiger die Beschreibungen sind, desto besser ist die Wiederverwendbarkeit, was bei einem zentralen Datenbestand mit organisationsweitem Zugriff von großer Bedeutung ist.

Nach (Grover, 2015, S. 31 ff) gibt unter anderem zwei Formen von Metainformationen:

Technische bzw. strukturelle Metadaten über Format, Anzahl Datensätze und Attribute, oder Bereitstellungsintervalle.

Semantische Metadaten die auf den Inhalt der Daten schließen lassen. Beispielsweise welchen Themenkomplex die Daten abbilden, wie Bevölkerung oder Finanzen. Zusätzlich kann die zeitliche und räumliche Granularität der Daten erfasst werden.

Meistens kann ein Großteil der semantischen Informationen vom Bereitsteller direkt bezogen werden. Dieser dokumentiert seine Datensätze beispielsweise in Form von PDF Dokumenten, oder Beschreibungstexte, die über die Webseite bereitgestellt werden. Diese können als Metadaten direkt im Data Lake abgelegt werden. Dies trifft auch auf die Ressourcen zu, die vom Zensus angeboten werden, die auf nachfolgender Abbildung dargestellt wird.




 Datensatzbeschreibung_Bevoelkerung.xlsx	Microsoft Excel-Arbeitsblatt
 Hinweise_CSV-Datensatz.pdf	PDF-Datei
 Zensus11_Datensatz_Bevoelkerung.csv	Microsoft Excel-CSV-Datei

Abbildung 6 Mitgelieferte Metadaten des Zensus

Eine umfassende Dokumentation liefert auch der Datensatz des Bundesamtes für Geodäsie und Kartographie (Abbildung 7). Neben den Daten (Verzeichnis *gn250*) werden Beschreibungen zum Dateninhalt, rechtliche Nutzbarkeit, letzte Aktualisierung des Datenbestandes, oder primärer Datenerzeuger bereitgestellt.







 dokumentation	Dateiordner
 gn250	Dateiordner
 aktualitaet.txt	Textdokument
 geonutzv.pdf	PDF-Datei
 geonutzv_eng.pdf	PDF-Datei
 quellenvermerk.txt	Textdokument

Abbildung 7 Daten und Metadaten die vom Bundesamt für Kartographie und Geodäsie bereitgestellt werden

Ein Teil der Metainformation stellt auch der Dateiname dar, welcher im ursprünglichen Zustand meistens sehr technischer Natur ist und nicht ohne Weiteres auf den Inhalt schließen lässt (siehe vorherige Abbildung „gn250“). Daher empfiehlt es sich, einen Standard für die Benennung der Dateien über alle Zonen hinweg zu etablieren. Die Ausgestaltung der Metadatenstruktur in dieser Arbeit wird in Kapitel 2.4 vorgestellt.

Strukturierung und Format Standardisierung

Da Open Data ein breites Spektrum an Zielgruppen abdeckt, werden entweder menschenlesbare (visuell ansprechend) oder maschineninterpretierbare (leicht auslesbare Struktur) Formen von Daten zur Verfügung gestellt. Die in einem Excel-Dokument bereitgestellten Daten des Bundesamtes für Statistik, werden menschenlesbar aufbereitet und folgen nicht unbedingt keiner klaren Struktur einer flachen Tabelle. Ein entsprechendes Beispiel wird in Kapitel 3.1.1 gegeben.

Außerdem wird bei der Bereitstellung großer Datenmengen oftmals ein Kompressionsverfahren benutzt, um die Datenübertragung zu verkürzen. So werden die Ressourcen vom Zensus in zip-Dateien ausgeliefert. Dieses durchaus sinnvolle Vorgehen macht es jedoch notwendig, die entsprechenden Dateien vor der Verwendung zu dekomprimieren, was oftmals nur mit zusätzlichen Programmen durchzuführen ist.

Sind die Daten dokumentiert, strukturiert und formatiert, können diese in der Raw Zone abgelegt werden. Von hier aus können die Dateien leicht eingelesen und in der Curated Zone weiterverarbeitet werden.

2.3.2. *Curated*

Da der Data Lake als zentraler Datenlieferant für unternehmensweite Anwendungen und Analysen dienen soll, ist es notwendig einen Bereich für eine solide Datenbasis zu schaffen – ähnlich zu der Kernschicht im DWH. Daher werden mit den Datenbeständen, die in der Raw Zone minimal aufbereitet vorliegen, weitere Bereinigungs- und Anreicherungsprozesse durchgeführt, wonach die Daten dann als aufbereitet gelten.

Dafür ist es notwendig, einen einheitlichen Qualitätszustand zu definieren, in denen sich verschiedene Standards etablieren. Aufbereitete Daten erfüllen dabei folgende Eigenschaften, vgl. (Maletic & Marcus, 2000):

Validität: Richtige Datentypen und korrekte Repräsentationen von Werten.

Einheitlichkeit: Konsistente Darstellung von Größenordnungen (z.B. Meilen oder Kilometer) oder Dezimaltrennzeichen (Komma vs. Dezimalpunkt).

Vollständige Daten: Behandlung von fehlenden Werten oder gänzlich fehlender Objekte

Die Curated Zone teilt sich analog zur Kernschicht im DWH in zwei Bereiche. Zum einen wird eine umfassende und **zentrale Datenbasis** erzeugt, die valide und einheitliche Daten bereithält. Aus diesem Datenbestand können je nach Anwendungsfall unterschiedliche **Teildatenbestände** extrahiert und kopiert werden – im DWH werden diese auch *Data Marts* genannt. Je nach Anforderungen kann hier die erforderliche Datenqualität gezielt angepasst werden, wie z.B. durch die Behandlung fehlender Werte. Aus beiden Bereichen, zentraler Datenbasis und anwendungsspezifische Ausschnitte, können sich verschiedene Anwender bedienen, um darauf aufbauend regelmäßige Reportings abzuleiten oder explorative Analysen durchzuführen (mehr dazu in der Working Zone).

Im Folgenden werden drei Prozesse definiert, um die aufbereitete Datenbasis zu erzeugen. Zuerst werden inhaltlich zusammengehörigen Datenbestände miteinander verknüpft und erweitert (Anreicherung). Danach können im Rahmen der Datenbereinigung Transformationen durchgeführt werden, um eine zentrale Datenbasis mit einheitlicher Datenqualität zu erzeugen. Die Daten können dann für die jeweilige Anwendung vervollständigt werden, indem fehlende Werte behandelt werden. Die Prozesse aus der Curated Zone sind in der nachstehenden Grafik abgebildet.

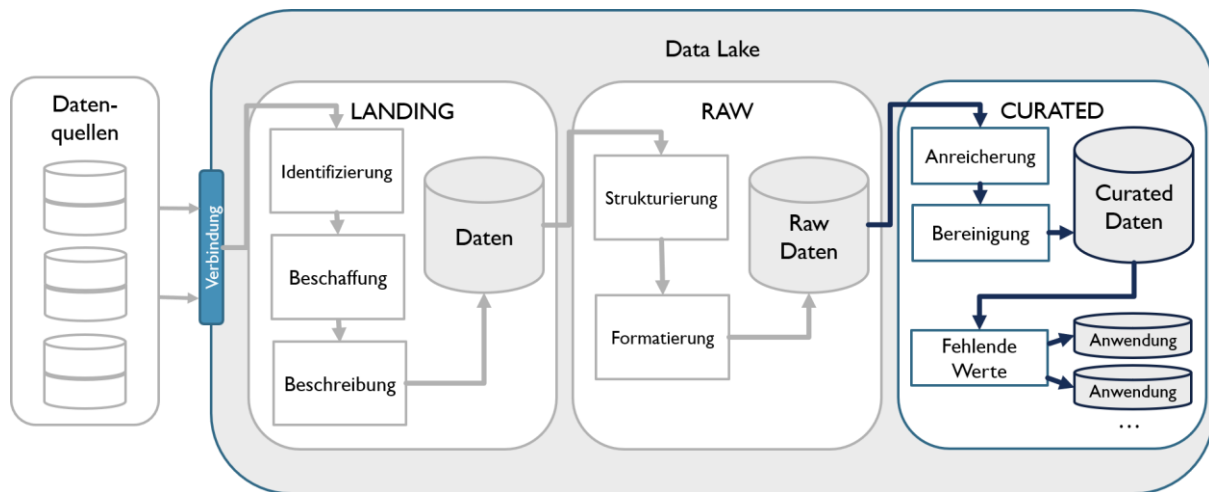


Abbildung 8 Data Lake erweitert um Curated Zone für die Datenbereinigung und -anreicherung.

Anreicherung von Datenbeständen

In dieser Arbeit beschreibt Datenanreicherung den Prozess der Erweiterung und Verfeinerung von Rohdaten vgl. (Terrizzano, 2015, S. 7). Die *Erweiterung* sieht eine kontextuelle Anreicherung eines Objektes in einem Datenbestand vor, bei dem der Informationsgehalt über das Objekt steigt. Bei bestehenden Unternehmensdaten wird meist auf externe Datenquellen, z.B. aus dem Open Data Bereich, zurückgegriffen um eine Anreicherung von internen Geschäftsobjekten zu erreichen. Beispielsweise könnte für eine Filiale eines Discounter Informationen zu beschäftigten Mitarbeitern oder erzielten Jahresumsatz vorliegen. Um Informationen über Kunden aus dem näheren Umkreis der Filiale zu erhalten, könnten die Filialdaten um demografische Daten erweitert werden. Dies wird über die Verknüpfung mehrerer Datenbestände erreicht.

Eine *Verfeinerung* beschreibt ein Ableiten von Informationen aus bestehenden Daten, wie das schlussfolgern des Geschlechts aus dem Kundennamen.

Diese Art der Anreicherung kann beispielsweise dadurch erreicht werden, dass aus der Adressinformation einer Filiale, die geografischen Koordinaten (Längen- und Breitengrad) abgeleitet werden.

Die kontextuelle Verknüpfung von Datenbeständen hat auch den Vorteil, dass nachfolgende Bereinigungsprozesse auf einer größeren Datenmenge durchgeführt werden anstatt an vielen kleineren Datensätzen.

Bereinigung

Die Datenbereinigung ist eines der wichtigsten Schritte, um die Daten auf ein bestimmtes Qualitätsniveau zu bringen. Eine einheitliche Auflistung der Vorgänge, die unter der Datenbereinigung fallen, ist in der Literatur nicht zu finden. Allgemein ist das Ziel, die enthaltenen Informationen des Datenbestandes korrekt darzustellen, indem Fehler identifiziert und behandelt werden, (Maletic & Marcus, 2000, S. 3). Ein Fehler (auch Anomalie genannt) kann dabei ein inkorrekt, unvollständiger, ungenauer oder irrelevanter Teil des Datensatzes und muss je nach Anforderungen zunächst definiert werden.

Vorliegend werden folgende Aktivitäten unter der Datenbereinigung gefasst: Das Definieren, Identifizieren und Behandlung von

- Statistischen Ausreißern
- Inkorrekten Werten
- Redundante Daten (Duplikate)
- Unrelevanten Daten

In Kapitel 3.7 wird die Datenbereinigung praktisch am Anwendungsbeispiel durchgeführt.

Nach (Maletic & Marcus, 2000, S. 2) kann ein fehlender Wert ebenfalls als Fehler interpretiert werden und fällt damit unter die Datenbereinigung. Das Behandeln von fehlenden Werten ist ein umfangreicher Themenkomplex und wird daher im Folgenden gesondert betrachtet.

Behandlung fehlender Werte

Die Ursachen für das Aufkommen von fehlenden Daten können vielfältig sein. Meistens entstehen diese im Datenentstehungsprozess und können behandelt werden. Im Open Data Fall ist dies nicht möglich, da die Erzeugung der Daten den Bereitstellern obliegt. Aus diesem Grunde ist der Umgang mit fehlenden Werten ein fester Bestandteil im vorliegenden Data Lake.

Dabei gibt es drei grundlegende Strategien im Umgang mit fehlenden Werten, die je nach Ausmaß angewendet werden können, vgl. (Géron, 2018, S. 60).

- Entfernen des Objekts mit fehlenden Werten.
- Eliminierung des entsprechenden Datenattributs.
- Schätzung fehlende Werte über bestimmten Verfahren

Da das Bestreben ist, möglichst viele Informationen beizubehalten, sind die ersten beiden Varianten sehr gravierend. Diese können jedoch sinnvoll sein, wenn der Umfang der fehlenden Werte eines Elements oder Merkmals besonders hoch ist und dadurch eine Verwendung für den jeweiligen Zweck nicht mehr sinnvoll ist.

Bei einem geringeren Umfang an fehlenden Werten, bietet sich alternativ eine Schätzung an. Die gebräuchlichsten Verfahren sind hierbei die Ersetzung der fehlenden Werte durch den Mittelwert des zugehörigen Merkmals oder einer Konstanten. Für den Mittelwert eignen sich Maße wie das arithmetische Mittel, der Median, oder der Modus (höchsthäufigster Wert).

Dieses durchaus einfache Vorgehen sollte jedoch nur durchgeführt werden, wenn die wahre Verteilung der zugrundeliegenden Daten bekannt und beispielsweise einer Normalverteilung folgen¹⁷. Ist diese Annahme nicht gegeben, muss mit Verzerrungen gerechnet werden. Alternativ können weitaus komplexeren Methoden angewendet werden, wie beispielsweise auf stochastische Verfahren basieren.

2.3.3. *Working und Sandbox*

Die vorangegangenen Prozesse waren danach ausgerichtet einen aufbereiteten Datenbestand zu erzeugen. Im Arbeitsbereich des Data Lakes, der sog. **Working Zone**, liegt der Fokus auf die Gewinnung von Informationen und Erkenntnissen über die Daten. Die Working Zone orientiert sich dabei an den Anwendungen aus dem Curated Bereich. Mit den anwendungsbezogenen Daten können verschiedene Analysen durchgeführt werden. Beispielsweise können übliche Berichtserstattungen abgeleitet werden, wie sie im DWH vorkommen. Daneben können auch erweiterte Analyseverfahren angewendet werden, die Algorithmen aus dem Machine Learning verwenden. Für jede Analyse ist eine individuelle Vorverarbeitung der Daten erforderlich, die für jedes Verfahren anders aussehen kann. Daher werden für jedes Analyseprojekt eigene Working Zonen definiert, um die Prozesse und Daten voneinander abzugrenzen. In dieser Arbeit wird eine Working Zone definiert, welche ausschließlich auf Machine Learning Prozesse angepasst wird. Die Vorverarbeitung der Daten, umfasst die Feature Selection bzw. Extraction, die Feature Transformation und weitere Kernaktivitäten des Machine Learnings (Abbildung 9). Im

¹⁷ Unter den Umständen entspricht der Erwartungswert (theoretisches Mittel) dem empirischen Mittelwert. Siehe hierzu <https://de.wikipedia.org/wiki/Erwartungswert>

Rahmen der Feature Selection bzw. Extraction werden geeignete Merkmale für die Analyse ausgewählt. Liegen diese vor, müssen die Daten in der Feature Transformationen insoweit aufbereitet werden, sodass ein Machine Learning Algorithmus Berechnungen mit diesen durchführen kann. Liegen selektierte und transformierte Merkmale vor, kann ein Machine Learning Algorithmus angewendet und optimiert werden. Die Beschreibung und Durchführung dieser Prozesse werden in Kapitel 4 detailliert beschrieben.

Neben den anderen Zonen im Data Lake wird ein **Sandbox** Bereich für experimentelle Aktivitäten reserviert, in denen Daten kurzfristig abgelegt werden, die aus jeglichen Zonen des Data Lake stammen können. So können Anwender verschiedene Untersuchungen durchführen. Die Sandbox ist dabei ein temporärer Bereich, in dem die verwendeten und erzeugten Daten oder Visualisierungen, regelmäßig gelöscht werden (z.B. alle 3 Monate).

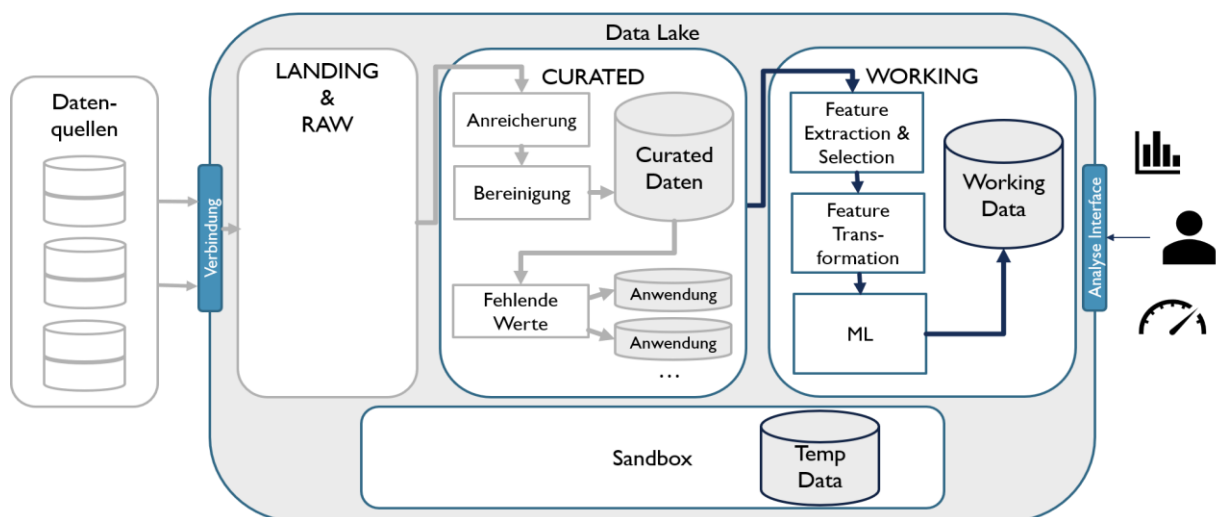


Abbildung 9 Data Lake erweitert um Bereiche Working und Sandbox

2.4. Gefahren und Metadatenmanagement

Wie in den vorangegangenen Kapiteln herausgestellt wurde, eröffnet das Data Lake Konzept nicht nur neue Möglichkeiten durch die gewonnene Skalierbarkeit und Flexibilität des Datenmanagements. In einem Artikel aus dem Jahre 2015 der EMC Dell¹⁸ heißt es, dass 2018 etwa 90% aller Data Lakes nutzlos sind, da diese mit Daten geflutet werden, für noch kein Anwendungsfall definiert wurde.

¹⁸ Siehe https://infocus.dellemc.com/william_schmarzo/achieving_big_data_maturity/

Auch wenn die Aussage im Jahre 2019 etwas drastisch scheint, soll diese auf die Risiken des Konzepts hindeuten, die metaphorisch mit einer **Data Swamp** Problematik beschrieben werden vgl. (LaPlante & Sharma, 2016, S. 34). Die Entwicklung von einem See (*Lake*) in ein Moor (*Swamp*) begründet sich vor allem durch die unzureichende Organisation der Daten. Der Anspruch, dass potenziell alle Daten jeglicher Herkunft und jeden Formats im Data Lake abgelegt werden können, führt zwangsläufig zu einer vielfältigen Datensammlung mit unterschiedlichen Qualitäten, die von vielen verschiedenen Anwendern benutzt und verarbeitet werden. Die Daten unterscheiden sich in ihrer Aktualität, Relevanz, Inhalt sowie rechtlicher und organisatorischer Anforderungen. Die Verwaltung großer heterogener Datenmengen sollte im Einklang mit einer Data Governance Strategie erfolgen. Nach (LaPlante & Sharma, 2016, S. 34) gehören zu Data Governance unternehmensweite Standards bzgl. Datenqualität, Prozesse, Sicherheit sowie die Definition des Datenlebenszyklus.

Im Folgenden werden einige Ansätze vorgestellt, die ins Data Governance integriert werden können, um einen Data Swamp zu vermeiden.

Problem: Daten sind irgendwann nicht mehr relevant und nehmen weiterhin Speicherplatz ein.

Ansatz: ebay temperature tiering

Die größte Herausforderung bei Verwendung einer Data Lake Architektur ist, nicht die Übersicht über die Daten im Eingangsbereich (Landing und Raw) zu verlieren.

Im Zeitverlauf werden einige der im Data Lake enthaltenen Daten für mögliche Anwendungsfälle zunehmend irrelevant und nehmen weiterhin im Speicherplatz ein, was wiederum mit höheren Kosten verbunden ist. Um die Datenorganisation diesbezüglich effizient zu gestalten, bietet sich eine Separierung nach Aktualität und Nutzungsfrequenz von Datenbeständen an. Eine mögliche Lösung ist das „Temperature Tiering“, welches in einem Blogbeitrag¹⁹ von ebay beschrieben wird.

Durch Kombination von Zugangszeitpunkt des Datensatzes und Nutzungsstatistik, lassen sich vier verschiedene Klassen definieren. Hot, Warm, Cold und Frozen, welche symbolisch für die Temperatur stehen sollen.

¹⁹ Siehe <https://www.ebayinc.com/stories/blogs/tech/hdfs-storage-efficiency-using-tiered-storage/>



Abbildung 10 Temperaturklassen mit entsprechender Farbskala für die Klassifizierung von Daten

Daten die mit „Hot“ bezeichnet werden, werden sehr häufig verwendet und sind aktuell. Daten die als „Frozen“ gekennzeichnet werden, gelten als ungenutzt und veraltet. Zwischen diesen Zuständen reihen sich Warm und Cold ein. Werden alle im Data Lake enthaltenen Daten mit entsprechenden Temperaturklassen markiert, können diese im Speicher umorganisiert werden. Hierzu wird der verfügbaren Speicher in eine „DISK“ und „ARCHIVE“ Partition eingeteilt. Der DISK Bereich ist ein hoch verfügbarer Speicher, welcher mit hoher Rechenleistung ausgestattet ist (CPU und RAM). Die vorgehaltene Rechenleistung im ARCHIVE Teil wird dagegen insoweit reduziert, sodass geringe Kosten für die Datenhaltung entstehen. Die Daten können dann je nach Temperaturklasse auf diese Speicherpartitionen verteilt werden (Abbildung 11). So befinden sich aktuelle und häufig benutzte Daten in der DISK Partition und veraltete sowie ungenutzte im Archiv-Bereich.

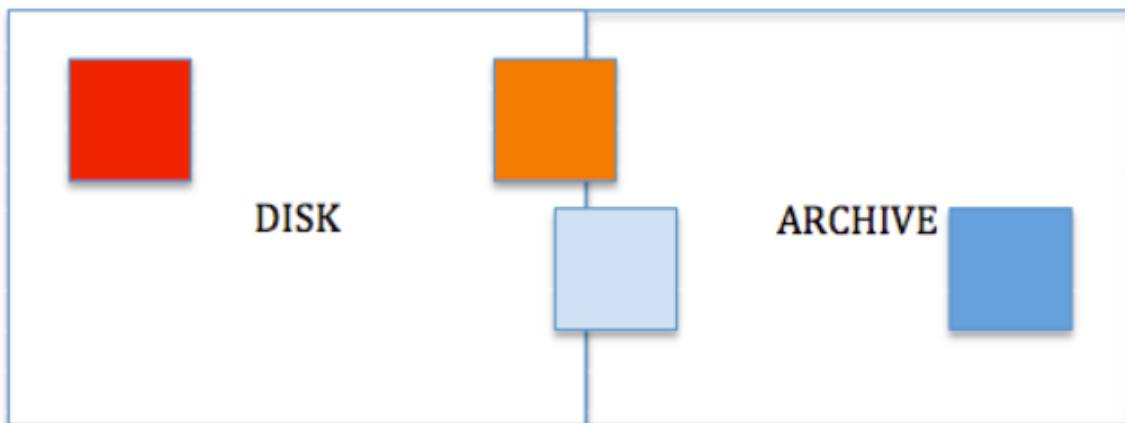


Abbildung 11 Datenorganisation der unterschiedlichen Temperaturklassen über die Speicherbereiche DISK und ARCHIVE

Da die Klassifizierung der Datensätze im Zeitverlauf angepasst werden, müssen die Daten dynamisch über diese beiden Speicherbereiche verschoben werden.

Problem: Das Datenangebot wird über die Zeit redundant und wichtige Ressourcen gehen dabei unter. Durch die Datenmasse bleiben viele Datensätze ungenutzt, da diese schwerer zu finden sind.

Ansatz: Metadatenchema und Datenkatalog – vgl. (LaPlante & Sharma, 2016, S. 37)

Je umfangreicher und vielfältiger die Datenmenge im Data Lake ist, desto bedeutender wird ein geeignetes Metadatenmanagement. Grundlage hierfür ist ein standardisiertes Metadatenchema, anhand dessen die Daten im Data Lake charakterisiert und unterschieden werden können. Im vorliegenden Anwendungsfall wurde ein Metadatenchema definiert, welches Informationen über Datenquelle, Dateninhalt und technische Eigenschaften enthält. Da die Ressourcen sehr heterogen sein können, sind einige Attribute optional und nur wenige obligatorisch. Ein Beispiel für eine Metadatenstruktur kann aus nachfolgender Tabelle entnommen werden. Dieses Schema wird im betrachteten Anwendungsfall dieser Arbeit verwendet.

Tabelle 4 Beispiel für ein Metadatenchema für Open Data

Metadaten-bezeichnung	Beschreibung	Beispiele
Inhalt (Tags)	Kategorie, auf der sich der Dateninhalt bezieht	Bevölkerung, Geografie, Wirtschaft, Infrastruktur
Zeitbezug	Zeitraum oder Zeitpunkt, für die die Daten gelten	2011
Raumbezug	Art des geografischen Bezugs.	Logisch (Regionsname oder Adressen), geografisch (Polygone oder Koordinatenpunkt)
Granularität	Die kleinste dargestellte Einheit, die in den Daten vorkommen	Kreise und Kreisfreie Städte, Gemeindeverbände, Gemeinden, Koordinaten
Bereitsteller	Name der Datenquelle	Zensus2011, Bundesamt für Statistik, OpenStreetMap, etc.
Datenbezug	Technische Schnittstelle, über die die Daten bereitgestellt werden	Manueller Download

URL zur Datenquelle	URL zur Datenquelle	www.opengov.de
Letzter Datenbezug	Zeitpunkt, an dem die Daten das letzte Mal bezogen wurden.	28.09.2018
Durchgeführte Verarbeitungsschritte	Durchgeführte Verarbeitungsschritte, die sich auf die Prozesse im Data Lake orientieren	Strukturiert, bereinigt, angereichert
Dateipfad	Absolute Pfadangabe bei der die Ressourcen im Speicher zu finden sind	/curated/daten/
Dateiname	Name der Datei	ZENSUS_BEVOELKERUNG
Dateiformat	Technisches Format der Datei	CSV
Schlüsselattribut	Attribut, an dem die Datensätze eindeutig identifizierbar sind	Filial-ID, Regionalschlüssel (RS)
Anzahl Elemente	Anzahl der Objekte/Datensätze die in der Datei enthalten sind	12544
Anzahl Attribute	Höchste Anzahl an Attributen ein Objekt im Datensatz beschreiben	200
Dateigröße (MB)	Speicherplatz der für die Datei eingenommen wird	70
Kommentar	Wichtige Vermerkung zur Datei	
Lizenz	Lizenz, die rechtliche oder organisatorische Nutzungsrichtlinien definiert	Deutschlandlizenz 2.0

Werden die Daten mit Metadaten beschrieben, können diese in einem übergreifenden **Metadatenkatalog** gespeichert werden. Über diesen Katalog kann der Datenbestand nach bestimmten inhaltlichen und strukturellen Merkmalen durchsucht werden. Je mehr Attribute zur Beschreibung von Daten in dem Schema vorhanden sind, desto gezielter

kann nach Daten im Data Lake gesucht werden. Weitere Informationen bezüglich der Ausgestaltung und technischen Umsetzung des Metadatenkatalogs siehe (LaPlante & Sharma, 2016, S. 37 ff.).

Ansatz: Data Profiling und Ähnlichkeitsanalyse der Daten – (Alserafi, Abello, Romero, & Calders, 2016, S. 1)

Data Profiling bezeichnet den Prozess zur Analyse vorhandener Daten durch unterschiedliche Analysetechniken. Dabei wird die Struktur der Daten erfasst und die daraus resultierenden Informationen in den Metadaten aufgenommen. Im Datenprofil eines Datensatzes kann beispielsweise festgehalten werden, wie viele Attribute enthalten, oder wie viele Attribute eines Datentyps vorhanden sind²⁰. Zudem können die Attribute separat untersucht werden, indem statistische Kennzahlen gebildet werden. Beispielsweise kann die Verteilung des Merkmals „Anzahl Einwohner“ über den Minimal-, Maximal- und Durchschnittswert beschrieben werden. Somit über kann überprüft werden, ob dieses Merkmal in anderen Datenbeständen in dieser Form vorhanden ist. Die Redundanzen können damit beseitigt werden, was zu einer Verdichtung der Daten im Data Lake führt.

Problem: Die Kombination von Daten unterschiedlicher Herkunft kann bei langen Verarbeitungsketten nicht mehr nachvollzogen werden. Dadurch geht das Verständnis für die Daten verloren

Ansatz: Data Lineage – vgl. (LaPlante & Sharma, 2016, S. 38).

Im Data Lake werden häufig Daten kombiniert, verändert oder aggregiert. Hier besteht die Gefahr, dass Transformationen durchgeführt werden, die semantisch nicht sinnvoll sind. Ein semantischer Bruch bei einer Kombination von Daten ist sehr schwer auszumachen, falls das domänenspezifische Wissen fehlt. Das Ziel des Data Lineage (dt. Datenherkunft) ist es zu einem gegebenen Datenbestand, die ursprünglichen Daten zu bestimmen. Hierfür wird ein Transformationslinie über ein sog. *Tracing* erzeugt, anhand derer die Kette der Verarbeitungsschritte zurückverfolgt werden kann. Data Lineage ist ein umfangreiches Konzept mit vielen Methoden, die ursprünglich für DWH konzipiert wurden.

²⁰ Diese Analyse kann in der Sandbox Zone des Data Lakes stattfinden

Im Data Lake für Open Data ein Data Lineage ebenfalls sinnvoll sein, um zu überprüfen, unter welchen Lizenzbedingungen die ursprünglichen Daten bereitgestellt wurden. Somit kann sichergestellt werden, dass die Daten der jeweiligen Anwendungen rechtlich konform genutzt werden.

Es wurden einige Lösungsansätze aufgezeigt, die die Gefahr eines Data Swamps minimieren sollen. Im nächsten Kapitel geht es um die technische Umsetzung des Data Lakes. Dabei erfolgt eine Auswahl an technischen Komponenten, die den Prozess von Datenbeschaffung bis zur Datenanalyse abdecken.

3. Implementation der Data Lake Architektur

In diesem Kapitel wird die Realisierung des zuvor erstellten Architekturentwurfs beschrieben.

Hierbei wird vor allem der konkrete Aufbau des Systems thematisiert. Bei der technischen Umsetzung musste berücksichtigt werden, dass keine Kosten durch Installation und Betrieb entstehen. Außerdem sollten die eingesetzten Komponenten einen hohen Funktionsumfang aufweisen, um möglichst viele Prozesse, die im Data Lake definiert sind, abzudecken. Dadurch konnten die Anzahl der Programme und Frameworks möglichst klein gehalten werden.

In den nachfolgenden Unterkapiteln wird zunächst ein Überblick über den Anwendungsfall gegeben und darauf basierend die Softwarekomponenten für Integration, Speicherung, Verarbeitung, Analyse und Visualisierung vorgestellt. Insbesondere sollen hierbei die zuvor beschriebenen Prozessschritte aus dem vorherigen Kapitel in der praktischen Anwendung nachvollzogen werden. Hierzu zählt die Einbindung der verschiedenen Datenquellen, die Datenbereinigung und -anreicherung sowie der Umgang mit fehlenden Werten. Die darauffolgende Analyse wird im 4. Kapitel gesondert behandelt.

3.1. Anwendungsfall

In dieser Arbeit wird die Umsetzung der theoretischen Konzepte, wie Datenorganisation und -verarbeitung in einem Data Lake, anhand eines konkreten Anwendungsfalls gezeigt. Zunächst wird ein Überblick über die Zielstellung gegeben. Danach werden die verwendeten Daten des Anwendungsfalls vorgestellt, die sich aus Open Data und internen Unternehmensdaten zusammensetzen.

3.1.1. Zielstellung

Ein Einzelhändler verfügt über 3348 Filialen in Deutschland und sucht nach optimalen Standorten, um sein bestehendes Filialnetz zu erweitern. Dabei sollen Regionen in Deutschland identifiziert werden, welche noch keine Filialen haben und von den Standorteigenschaften ertragreich sein könnten.

Für jede Filiale liegen Informationen aus einem unternehmensinternen DWH vor, wie z.B. Mitarbeiteranzahl, Verkaufsfläche und Umsatz. Zusätzlich stehen frei zugängliche externe Daten von verschiedenen deutschen Open Data Anbietern zur Verfügung. Hierzu zählen Bevölkerungsdaten (Bevölkerungsdichte, Einkommensverteilung, Familienstand, etc.) und Infrastrukturdaten (Anzahl Gebäude, Verstädterungsgrad, etc.), die für alle Regionen in Deutschland vorliegen. Die Daten beziehen sich auf das Jahr 2011. Die Daten werden von verschiedenen Bereitstellern (z.B. öffentliche Ämter oder freie Kartenanbieter wie OpenStreetMap) extrahiert und in eine Data Lake Architektur mit entsprechenden Metadatenbeschreibungen abgelegt, bevor weitere Verarbeitungsschritte für die Analyse durchgeführt werden.

Danach soll der wirtschaftliche Erfolg von noch nicht erschlossenen Regionen mit Machine Learning Verfahren prognostiziert werden. Die Ergebnisse können dann einem Entscheider vorgelegt werden, welcher darauf basierend eine Strategie für die Anpassung des Filialnetzes ableiten kann.

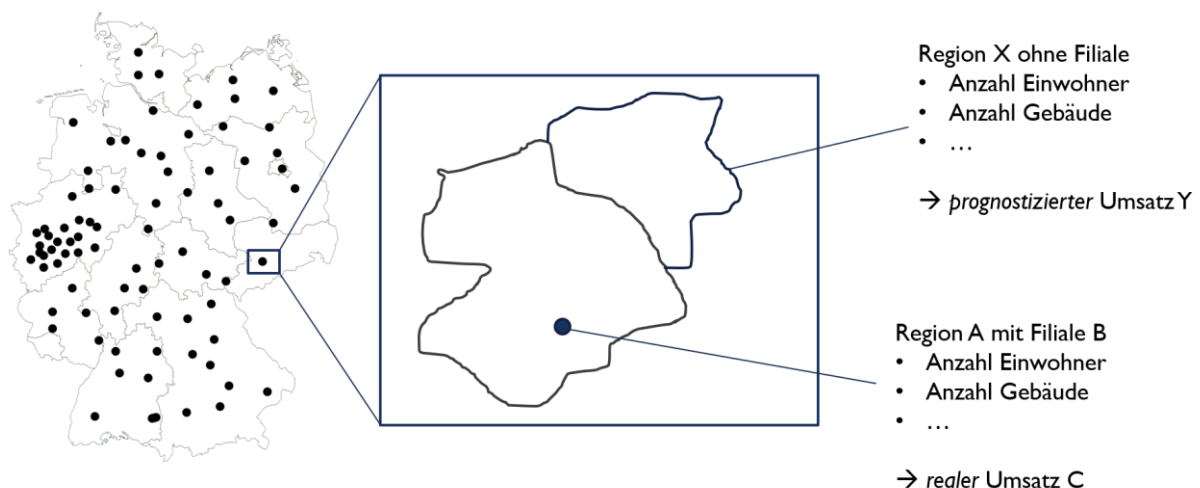


Abbildung 12 Identifikation von potenziell ertragreichen Regionen ohne bestehenden Filialstandort

3.1.2. Daten

Jede Filiale liegt in einer Region in Deutschland. Eine Region ist ein abgrenzbarer Ausschnitt einer geografischen Fläche. Jeder Region können bestimmte Kennzahlen aus Bevölkerungs- und Infrastrukturmerkmalen zugeordnet werden. Ein Großteil der Daten stammt von öffentlichen Verwaltungsstellen, die eine Einteilung in administrative

Verwaltungseinheiten²¹ verwenden. Die Verwaltungseinheiten sind hierarchisch in Bundesländer, Regierungsbezirke, Landkreise und Gemeinden sowie kreisfreie Städte unterteilt (siehe Abbildung 13).

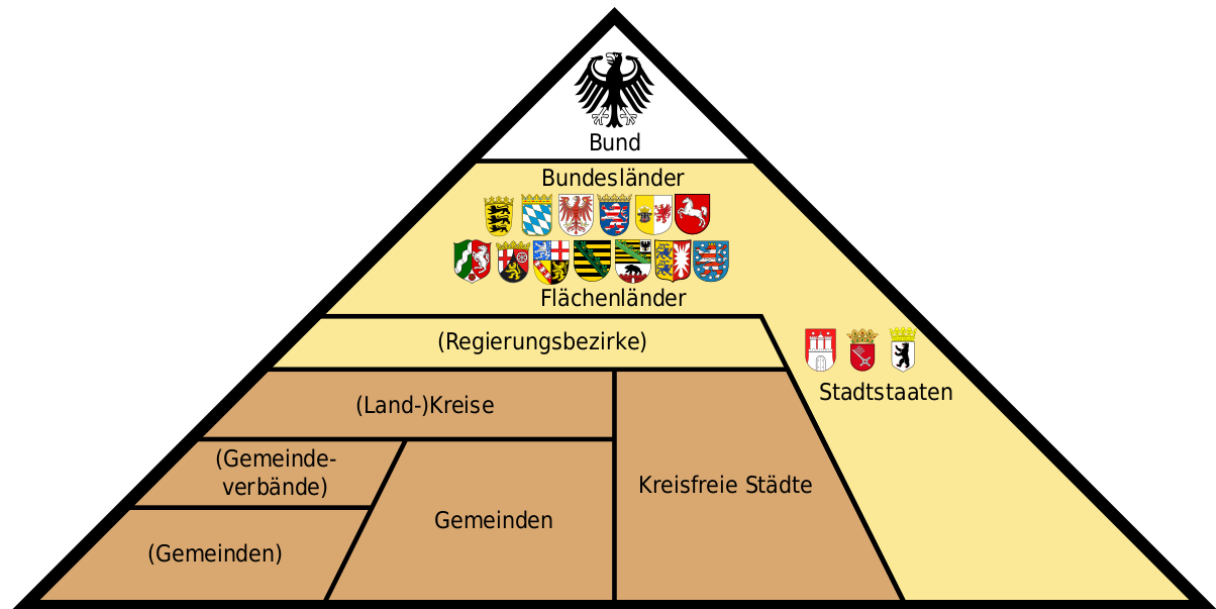


Abbildung 13 Administrative Gliederung der Verwaltungseinheiten in Deutschland (Quelle: Wikipedia: Verwaltungsgliederung Deutschland)

Beispielsweise besteht das Bundesland Nordrhein-Westfalen aus den Regierungsbezirken Düsseldorf, Detmold, Arnsberg, Köln und Münster, die jeweils wieder Kreise und Gemeinden unter sich gliedern. Zu berücksichtigen ist, dass einige Bundesländer wie Thüringen oder Schleswig-Holstein keine Regierungsbezirke besitzen, sodass diese Ebene für einige Bundesländer optional ist. Außerdem können einige Verwaltungseinheiten auf mehreren Ebenen auftreten. So ist z.B. Düsseldorf zugleich Landkreis, kreisfreie Stadt und Gemeinde.

Die Verwaltungseinheiten aller Ebenen werden in dieser Arbeit vereinfacht als Regionen bezeichnet. In Kapitel 4 werden die Verwaltungseinheiten der untersten Hierarchieebene (Gemeinden und kreisfreie Städte) für Machine Learning Verfahren verwendet.

Regionen in der Verwaltungsstruktur werden über einen 12-stelligen amtlichen Regionalschlüssel (RS) identifiziert, welcher ebenso hierarchisch strukturiert ist. Über die

²¹ Die logische Gliederung über die Verwaltungseinheiten ist von der Aufteilung über die Postleitzahl zu unterscheiden. Die Abgrenzung der Gebiete nach beiden Konzepten ist nur bedingt vereinbar.

Stellen im Schlüssel kann abgeleitet werden, welche Region welcher Verwaltungseinheit untergeordnet ist:

Bundesland (1. und 2. Stelle),
Regierungsbezirk (3. Stelle),
Kreis (4. und 5. Stelle),
Gemeindeverband (6. bis 9. Stelle) und
Gemeinde (10. bis 12. Stelle) an.

Nach diesem Schema kann beispielsweise die Verwaltungseinheit Ratingen über folgenden Schlüssel aufgelöst werden:

05 1 58 0028 028 (05=NRW, 1=Düsseldorf, 58=Kreis Mettmann,
0028=Gemeindeverband²², 028=Ratingen)

Einige Bereitsteller aus dem öffentlichen Sektor verwenden alternative Formen eines Identifikationsschlüssels, welche jedoch alle auf dem Regionalschlüssel basieren. Hierzu zählt der 8-stellige amtliche Gemeindeschlüssel (AGS), welcher dem RS entspricht, jedoch um die vier Stellen, welche den Gemeindeverband identifizieren, verkürzt ist. Demnach sieht der AGS für Ratingen wie folgt aus:

05 1 58 028 (Land=NRW, Regbez= Düsseldorf, Landkreis=Mettmann,
Gemeinde=Ratingen)

Zu den verschiedenen Formen der Schlüssel kommt noch hinzu, dass die Verwaltungsgliederungen ständigen Änderungen durch Reformen unterliegen. Dabei können z.B. bestimmte Regionen neu entstehen oder wegfallen sowie neu zusammengefasst werden. Dadurch kann es passieren, dass zwei verschiedene Ausprägungen eines Schlüssels für ein und dieselbe Region vorliegen, was im weiteren Verlauf zu Verknüpfungsproblemen führt. Dies lässt sich zum Teil durch die verkürzte Darstellung des AGS abfangen, da hier die Gemeindeverbände nicht berücksichtigt werden. Häufig werden Regierungsbezirke im Laufe der Zeit strukturell verändert, oder fallen ganz weg, wie es 2012 für Sachsen der Fall ist. Dabei können die daraus resultierenden Verknüpfungsprobleme nicht mit den bereitgestellten Informationen aufgelöst werden.

²² Gemeindeverbände werden lediglich über 4-stellige Kennzahlen definiert und besitzen keine eigene Bezeichnung

Aufbauend auf die Beschreibung der Datenquellen aus Kapitel 2.1.1 (Datenquellen und Formen), soll hier eine Erläuterung verwendeten Daten erfolgen und warum diese für den Anwendungsfall als relevant erachtet wurden.

Das Bundesamt für Statistik besitzt einen umfangreichen Datenbestand aus den Bereichen Infrastruktur, Bevölkerung und Wirtschaft. Bei der Betrachtung des Konsumenten, welcher in einer Filiale einkaufen geht, wird vermutet, dass der Faktor Einkommen einen Einfluss auf den Umsatz hat. Es wird angenommen, dass das zur Verfügung stehende Budget eines arbeitstätigen Konsumenten höher ist, als das der arbeitslosen Bevölkerung. Um diesen Zusammenhang mit einzubeziehen, wird außerdem die Arbeitslosenquote betrachtet.

Ebenso wird vermutet, dass die Umgebung eines Standorts Einfluss auf den wirtschaftlichen Erfolg einer Filiale haben könnte. Die Insolvenzstatistik kann einen Hinweis geben, wie die Entwicklung des lokalen Marktes aussieht. Ebenso könnten die infrastrukturellen Gegebenheiten ein Standortpotenzial begründen. Beispielsweise hätte eine Filiale in einem Siedlungsviertel eine höhere Konsumentenzahl in Reichweite, als in einer durch Waldfläche geprägten Region.

Tabelle 5 Beschreibung der Datenquelle des Bundesamtes für Statistik

Datenquelle	Bundesamt für Statistik (destatis)
Räumlicher Bezug	Merkmale sind auf Verwaltungseinheiten bezogen, die über den 8-stelligen amtlichen Gemeindeschlüssel (AGS) identifiziert werden
Zeitlicher Bezug	2010 bis 2016
Granularität	Landkreis (Gemeindeverzeichnis besitzt alle Ebenen)
Datensatz Einkommen	Durchschnittliches verfügbares Einkommen (nach Abzug von Lebenshaltungskosten) im Jahr pro Einwohner
Datensatz Arbeitslosigkeit	Anteile von Arbeitslosen (gesamt) sowie in den Alterssegmenten 15 bis 24 und 55 bis 64. Außerdem Anteil Langzeitarbeitslose und Arbeitslose Ausländer am Gesamtbestand
Datensatz Gebietsart	Flächenanteile (in %) von Siedlungen, Erholungsfläche, Landwirtschaft und Waldfläche
Datensatz Insolvenz	Anzahl beantragter, eröffneter, abgewiesener Verfahren sowie Anzahl betroffener Beschäftigte und Forderungsbetrag (von Unternehmen aller Sparten)
Datensatz Gemeindeverzeichnis	Gesamtfläche, geografischer Mittelpunkt, Postleitzahl und Verstädterungsgrad (städtisch/halbstädtisch/ländlich) der Regionen. Zusätzlich noch Anzahl Bevölkerung (weiblich/männlich) und Einwohner je km ² .

Im Rahmen des Microzensus wurden im Jahre 2011 umfangreiche Daten über Bevölkerungs- und Infrastruktur erhoben, welche ein genaueres Bild über die Umgebung eines Filialstandortes geben kann. Hierzu zählt nicht nur die demographische Struktur in einer Standortumgebung, sondern auch die Familiengröße der verschiedenen Haushalte. Interessante Beispiele werden in der nachstehenden Tabelle gegeben. Dabei werden jegliche Merkmale über alle Verwaltungseinheiten angeboten – von der Gemeinde bis zum Bundesland aggregiert.

Tabelle 6 Beschreibung der Datenquelle des Zensus

Datenquelle	Zensus 2011
Räumlicher Bezug	Merkmale sind auf Verwaltungseinheiten bezogen, die über den 12-stelligen Regionalschlüssel (RS) identifiziert werden
Zeitlicher Bezug	2011
Granularität	Verwaltungseinheiten aller Ebenen
Datensatz Bevölkerung	Beispiele: Anzahl Einwohner, Altersgruppen, Erwerbstätige, Familienstand, oder Migration.
Datensatz Gebäude und Wohnungen	Beispiele: Wohnfläche, Baujahre von Gebäuden, Anzahl Wohnungen in Gebäuden, oder Anzahl leerstehender Wohnungen.
Datensatz Familie und Haushalt	Beispiele: Haushaltsgröße, Haushalte mit Senioren, Haushalte mit Kindern, Haushalte mit ledigen Personen.

Der OpenStreetMap Datenbestand beinhaltet viele geografische Informationen. Darunter befinden sich auch geografische Punkte (Nodes), mit denen beispielsweise Bildungseinrichtungen, öffentliche Gebäude oder Einkaufsmöglichkeiten verortet werden. Diese Objekte geben einen Eindruck über die Beschaffenheit der Infrastruktur in der jeweiligen Region. Beispielsweise könnte gemutmaßt werden, dass die Anzahl an geografischen Punkten in einer Region, Auskunft über den Verstädterungsgrad geben. Die OSM Daten bieten somit eine sinnvolle Ergänzung zu den Wohnungs- und Gebäudedaten aus dem Zensusdatensatz.

Es wurde der aktuellste Datenbestand von OSM verwendet, da folgender Umstand berücksichtigt werden muss: Jedes geografische Objekt besitzt einen physikalischen Entstehungszeitpunkt (Baujahr) und einen digitalen Erfassungszeitpunkt durch einen Nutzer. Objekte, die im Jahre 2011 existierten, könnten erst Jahre später in den OSM Daten erfasst werden. Aus diesem Grunde wird die aktuellste Version des Datenausschnittes genommen, auch wenn hierdurch eine leichte Verzerrung entstehen könnte.

Tabelle 7 Beschreibung der Datenquelle von OpenStreetMap

Datenquelle	OpenStreetMap
Räumlicher Bezug	Merkmale auf geographische Objekte bezogen
Zeitlicher Bezug	2018
Granularität	Geografische Lage über Koordinaten
Datensatz Germany	Das grundlegende Objekt im OSM Modell ist ein Knoten (Node). Dieser Knoten besitzt Attribute wie Koordinaten, Identifikationsnummer, Versionsnummer, Zeitstempel und Angaben des Benutzers, der dieses Objekt erfasst hat. Außerdem werden Nodes über Tags charakterisiert (z.B. shopping, building, education, etc.). Nach unterschiedlichen Eigenschaften können die Nodes gefiltert werden.

Handelte es sich zuvor um offene Daten von öffentlichen Bereitstellern, stammen die Unternehmensdaten aus einem DWH. Hierbei wurden für jeden Filialstandort bestimmte Eigenschaften extrahiert, die in nachstehender Tabelle aufgelistet werden.

Tabelle 8 Beschreibung der unternehmensinternen Daten

Datenquelle	Unternehmen
Räumlicher Bezug	Merkmale sind auf Filialstandort bezogen, welche über eine Adresse lokalisiert wird
Zeitlicher Bezug	2011
Granularität	Filiale
Datensatz Filialen	Pro Filiale sind Standortangaben wie Region, Straße und Hausnummer gegeben sowie Verkaufsfläche, Mitarbeiteranzahl und Jahresumsatz.

Die Filialstandorte und die Daten von OpenStreetMap werden über die Adresse bzw. über geografischen Punkten lokalisiert. Um diese Objekte den einzelnen Verwaltungseinheiten zuordnen zu können, werden die geografischen Umriss jeder Region benötigt. Diese werden vom Bundesland für Kartographie und Geodäsie bereitgestellt.

Tabelle 9 Beschreibung der Datenquelle des Bundesamtes für Kartographie und Geodäsie

Datenquelle	Bundesamt für Kartographie und Geodäsie
Räumlicher Bezug	Merkmale sind auf Verwaltungseinheiten bezogen, die über den 12-stelligen Regionalschlüssel (RS) identifiziert werden
Zeitlicher Bezug	2010 bis 2016
Granularität	Landkreis und Gemeinden
Datensatz VG250 Kreisgrenzen	Geographische Angaben von allen Kreisgebieten wie Umrisse, Mittelpunkte, Geofaktor (Land, Wasser, etc.). Geodaten liegen im UTM32 Referenzformat vor.
Datensatz VG250 Gemeinden	Angaben wie in VG250 Kreisgrenzen, jedoch auf Gemeinden bezogen.

Für eine Untersuchung eines potenziellen wirtschaftlichen Standorts müssen alle vorhandenen Merkmale aus dem Datenbestand auf die jeweiligen Regionen übertragen werden. Damit können Regionen hinsichtlich Bevölkerungs-, Infrastruktur- und Wirtschaftsmerkmale verglichen werden.

3.2. *Systemarchitektur*

Den technische Rahmen des vorliegenden Systems bildet ein Cluster, auf welches über ein VPN Zugang zugegriffen werden kann. Das Cluster bezeichnet im Rahmen dieser Arbeit den Zusammenschluss von mehreren Rechnern, bzw. virtuellen Maschinen, über die parallel eine Verarbeitung von Daten erfolgen kann. Das Cluster besteht aus einem Masterknoten, über den sich Verarbeitungsaufgaben auf drei verschiedene Arbeitsknoten (Slaves) verteilen lassen werden können. Auf den jeweiligen Knoten ist ein Linux Betriebssystem (CentOS) installiert sowie weitere Programme für die Speicherung und Verarbeitung. Für die Speicherung von Daten wird das verteilte System von Hadoop verwendet, welches ein Open Source Projekt von Apache ist. Zur Verarbeitung wird Spark genutzt, ein Open Source Framework, das ebenfalls verteilt operiert. Der Masterknoten besitzt zusätzlich eine Entwicklungsumgebung, die mit Jupyter Notebooks realisiert wurde. Da die virtuellen Rechnerknoten in einem Netzwerk organisiert sind, lassen sich alle Maschinen über eine bestimmte IP Adresse über eine SSL Verbindung ansprechen, etwa um weitere Komponenten zu installieren, oder Fehler in den Logs nachzuvollziehen. In Abbildung 14 wird die Gesamtarchitektur des Systems dargestellt. Der Client greift über die VPN/SSH Verbindung auf den Masterknoten des Clusters zu, der die Prozesse auf drei Arbeitsknoten verteilt.

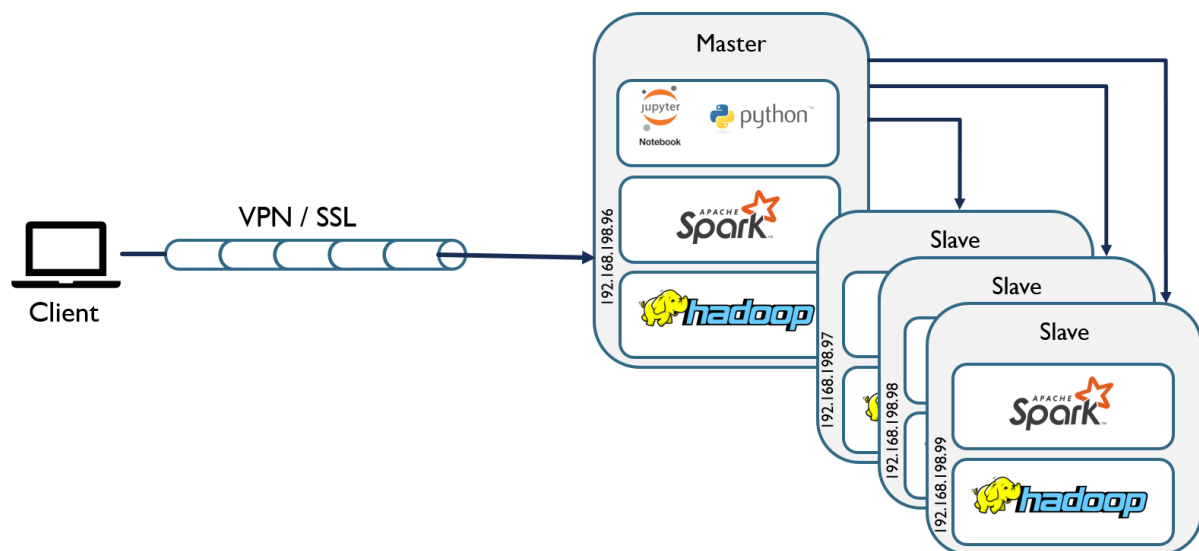


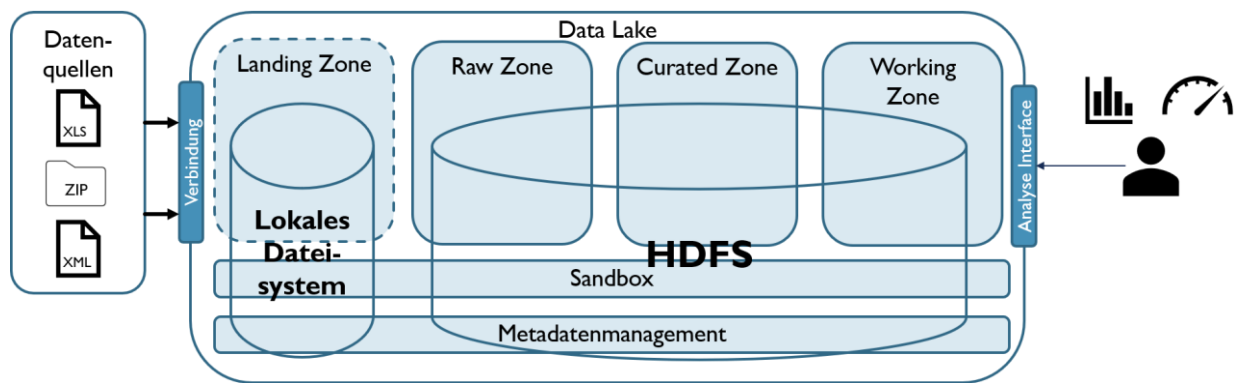
Abbildung 14 Aufbau des Rechnerclusters und technische Realisierung des Data Lake

3.3. Integration und Speicherung

Da die extrahierten Rohdaten in der **Landing Zone** noch manuell mit entsprechender Software strukturiert werden müssen, werden die Dateien im **lokalen Dateisystem** der Linux-VM gehalten. Von hier aus können diese z.B. über Excel bearbeitet und nachfolgend in einem strukturierten und nicht-proprietären Format (CSV, XML oder JSON) im HDFS gespeichert werden. Von der **Raw Zone bis zur Working Zone** aus werden alle Dateien im **Hadoop-Speicher** abgelegt.

Die **Metadaten** werden im Data Lake zu den Speicherorten abgelegt, in denen die jeweiligen Daten gespeichert werden. Demnach befinden sich diese teilweise im lokalen Dateisystem und im Hadoop-Speicher. Dies stellt eine sehr einfache Form der Metadatenverwaltung dar, die jedoch an den Umfang dieser Arbeit angepasst ist. Alternativ wäre ein zentraler Metadatenpeicher für die im Data Lake befindlichen Daten vorstellbar, der beispielsweise über den Apache Atlas realisiert werden könnte. Für nähere Informationen siehe offizielle Dokumentation²³.

²³ Dokumentation Apache Atlas: <https://atlas.apache.org/>



3.3.1. Lokales Dateisystem

Die Daten werden in der Landing Zone in ihrer ursprünglichen Form abgelegt. Bevor die Daten in die nächste Zone gelangen, werden die Datenbestände einer Quelle zunächst in einen Zustand gebracht, der eine weitere Verarbeitung zulässt. Beispielsweise werden hier die für Menschen visuell aufbereiteten Excel-Dokumente strukturiert und formatiert, sodass diese leichter zu verarbeiten sind. Dabei werden Begleittexte eliminiert und Spaltennamen angepasst, was an dem Beispiel der Datei über die Arbeitslosenquote stellvertretend gezeigt werden soll (siehe Abbildung 16).

Automatisches Speichern

1 AI008-1.xlsx - Excel Meinert, Alexander

Datei Start Einfügen Seitenlayout Formeln Daten Überprüfen Ansicht Hilfe Was möchten Sie tun?

Einfügen

Zwischenablage

Schriftart

Ausrichtung

Zahl

Formatvorlagen

Zellen

Bearbeiten

J13

2

	A	B	C	D	E	F	G	H	I	J	K
1	GENESIS-Tabelle: AI008-1										
2	Regionalatlas Deutschland										
3	Indikatoren des Themenbereichs "Arbeitslosigkeit"										
4	Regionalatlas Deutschland										
5											
6				Arbeitslosen Prozent	Anteil Arbeitslosen Prozent	Anteil Arbeitslosen Prozent	Anteil Langzeitarbeitslose Prozent	Anteil arbeitslose Ausländer an Arbeitslosen insg. Prozent			
7	2010	DG	Deutschland	7,7	10	16,4	35,2	15,5			
8	2010	1	Schleswig-Ho	7,5	11,3	14,7	32,3	10,5			
9	2010	1001	Flensburg,	12,7	11,8	12,2	28,9	10,6			
10	2010	1002	Kiel, Lande	10,8	8,4	10,3	34,8	16,6			

3

Folgende Anpassungen wurde an der Datei vorgenommen:

1. Der Dateiname wird nach einem übergreifenden Standardschema umbenannt. Dabei soll Datenquelle und Dateninhalt ersichtlich werden. Zusätzlich wird die Datei von einem Excel Format (.xlsx) zu einer CSV Datei konvertiert. Der Dateiname ändert sich von *AI008-1.xlsx* zu *DESTATIS_ARBEITSLOSIGKEIT.csv*.
2. Um eine flache Tabellenstruktur zu erzeugen, muss der Beschreibungstext entfernt werden. Dieser kann als Vorlage für die Metadatenbeschreibung dienen. Beispielsweise ist die Bezeichnung der GENSIS-Tabelle zu vermerken, da hierüber die aktuellste Tabelle aus der Statistikdatenbank von destatis gefunden und geladen werden kann.
3. Die Spaltenbeschreibungen erstrecken sich über zwei Zeilen. Außerdem enthalten diese Umlaute (Anteil arbeitslose Ausländer) und muss daraufhin angepasst werden.

Wie auf der nachstehenden Abbildung zu erkennen ist, ist das Ergebnis eine flache Tabellenstruktur, die von Programmbibliotheken leicht auszulesen ist.

JAHR	RS	GEBIET	ANTEIL_ARBE	ANTEIL_ARBE	ANTEIL_ARBE	ANTEIL_LANG	ANTEIL_ARBEITSLOSE_AUSLAENDER
2010	0	Deutschland	7,7	10	16,4	35,2	15,5
2010	1	Schleswig-Holstein	7,5	11,3	14,7	32,3	10,5
2010	1001	Flensburg, dän. Inseln	12,7	11,8	12,2	28,9	10,6
2010	1002	Kiel, Land	10,8	8,4	10,3	34,8	16,6
2010	1003	Lübeck, Hamburg	11,2	11,7	12,7	41,1	14,3
2010	1004	Neumünster	11,5	12,7	12,3	31,4	13,4
2010	1051	Dithmarschen	8,5	14,6	14	28,5	5

Abbildung 17 CSV Tabelle über Arbeitslosigkeit vom Bundesamt für Statistik nach Strukturierung und Formatierung

Für das beschriebene Vorgehen wird kein separates und umfangreiches Datenintegration-System verwendet. Technisch wird dies im lokalen Verzeichnis des Masterknotens mit Hilfe von Excel umgesetzt. Das Herunterladen der Ressourcen wird dabei mit dem Linux-eigenen Kommandozeilenprogramm *wget* realisiert, welches die Ressourcen über FTP und http lädt. Im zweiten Schritt werden die Eingangsdaten verteilt auf das den Hadoop-Speicher kopiert.

3.3.2. *Hadoop*

Das zugrundeliegende verteilte Dateisystem von Hadoop ist das **HDFS (Hadoop Distributed File System)**. Das HDFS besteht zum einen aus einer Verwaltungseinheit, dem sog. **NameNode** und mehreren **DataNodes**, auf welchen die Daten persistiert sind, vgl. (Marz, 2016, S. 80). Dabei liegen die Daten nicht als Ganzes, sondern in **Blöcken** vor, welche über alle Datenknoten verteilt werden. Jede in HDFS gespeicherte Datei, wird demnach zunächst in gleich große Blöcke²⁴ partitioniert und daraufhin über die DataNodes verteilt. Dabei liegen die Blöcke nicht einfach vor, sondern werden um einen bestimmten Faktor repliziert, sodass jeder Datenblock auf unterschiedlichen Knoten redundant vorliegt. Im NameNode sind die Speicherorte Blöcke einer Datei hinterlegt. Bei Bedarf können die Blöcke wieder zusammengeführt oder darauf parallele Operationen ausgeführt werden. Wird ein Datensatz über die DataNodes verteilt gespeichert, ist dieser nicht mehr veränderbar und damit „**immutable**“ (Grover S. 60). Das System ist für Leseoperationen optimiert, da mehrere Knoten Zugriffe zur gleichen Datei umsetzen können. Zudem können auch mehrere Anfragen parallel ausgeführt werden. Das entsprechende Paradigma nennt sich **write once, read many** – vgl. (Marz, 2016, S. 28). Um Datensätze zu aktualisieren, müssen diese also erneut ins HDFS geladen werden.

Die physikalischen Ressourcen (Speicher, RAM und CPU) der einzelnen Knoten sind unabhängig voneinander. Dieser sog. **Shared Nothing** Ansatz hat dabei mehrere Vorteile. Zum einen erhält man eine gewisse **Fehlertoleranz**, da die Daten-Fragmente im HDFS repliziert vorliegen und das Ausfallen eines DataNodes nicht zum Verlust aller Daten führt. Durch die Informationen, die im NameNode enthalten sind, kann der ursprüngliche Zustand wiederhergestellt werden vgl. (Marz, 2016, S. 20). Hierdurch wird außerdem eine **hohe Verfügbarkeit**, der im HDFS gespeicherten Daten erreicht. Um zu verhindern, dass auch diese Informationen bei einem Ausfall des NameNode verloren gehen, wird ein weiterer Backup-Knoten (Secondary NameNode) integriert, um einen „Single Point of Failure“ zu vermeiden. Zum anderen kann Speicherplatz und Rechenleistung **skaliert** werden, indem weitere Knoten hinzugefügt werden. **MapReduce** beschreibt das Verarbeitungs-Framework, welches in Hadoop verwendet wird um die performante Verarbeitung großer Datenmengen zu ermöglichen. Durch das zugrundeliegende Vorgehen (Map, Shuffle und Reduce) kann die Verarbeitung parallel auf mehreren Rechnerknoten durchgeführt werden vgl. (Grover, 2015, S. 80). Eine Stärke von Hadoop

²⁴ Die voreingestellte Blockgröße ist 64 MB.

ist die Fähigkeit, Daten in jeglichen Formaten zu speichern und zu verarbeiten. Diese Flexibilität ist bei der Verwendung von Open Data notwendig, da unterschiedliche Datenformate angeboten werden. Es empfiehlt sich jedoch bei größeren Dateien Hadoop-spezifische Formate zu verwenden, welche für eine verteilte Verarbeitung optimiert wurden. Es bieten sich unter anderem Formate wie Avro oder Parquet an, welche unterschiedliche Eigenschaften besitzen und je nach Anwendungsfall eingesetzt werden können. Beide Formate unterstützen eine Datenkompression, welche zum einen den Speicherplatz minimiert, als auch die Datenmenge, die bei der Verarbeitung zwischen den Rechnerknoten ausgetauscht werden. Zusätzlich sind beide Datenformate partitioniert. Derweil Avro zeilenorientiert ist, ermöglicht Parquet eine spaltenorientierte Speicherung. Der Datenbestand von OpenStreetMap wurde im Parquet konvertiert, um so eine schnellere Verarbeitung zu ermöglichen. Dieses Vorgehen wird in Kapitel 3.6.3 detaillierter beschrieben.

Kleinere Dateien werden in dieser Arbeit im CSV oder JSON Format abgelegt, da die Verarbeitung dieser Formate sehr gut durch eine Vielzahl von Bibliotheken und Frameworks unterstützt wird.

Die genannten Bestandteile der Datenhaltung sind Teile eines umfangreichen Hadoop Ökosystems, in denen viele Lösungen für Big Data Anwendungen existieren. Der Vorteil ist hier, dass Komponenten aus diesem Bereich gut miteinander integriert sind. So wäre es beispielsweise möglich ein Tool wie Sqoop zu installieren, welches den Datentransfer von einem relationalen System (wie einem DWH) in das HDFS steuert und durchführt. Denkbar wäre auch ein Workflow Manager wie Oozie um gewisse regelmäßige Integrations- und Verarbeitungsprozesse zu automatisieren und um Abfolgen von Skripten zu verschalten. Somit könnte auf neu auftretende Anforderungen reagiert werden.

3.3.3. Verzeichnisstruktur im Data Lake

In dieser Arbeit orientiert sich die Verzeichnisstruktur am Aufbau des Data Lakes. Auf der ersten Ebene sind Verzeichnisse für Landing, Raw, Curated und Working angelegt. Dabei werden die einzelnen Bereiche unterschiedlich strukturiert. Die Verzeichnisstruktur, die hier verwendet wird, ist auf Abbildung 18 zu entnehmen.



Abbildung 18 Verzeichnisstruktur im Data Lake

Die Daten im Landing Verzeichnis wird nach Datenquellen unterteilt. Dies hat den Hintergrund, dass die Daten einer Datenquelle ähnliche Eigenschaften aufweisen, wie technische Beschaffenheit oder rechtliche Verwendungsmöglichkeit gemäß der Lizenz. Für jede Datenquelle gibt es jeweils einen Bereich für die Dateien und Metadaten. In der Metadaten Sektion können die vom Bereitsteller zur Verfügung gestellten Informationen, oder selbst definierte Metadatenmodelle abgelegt werden. Der Raw Bereich ist ebenfalls nach Datenquellen strukturiert, da hier lediglich die Daten aus dem Landing Bereich vorliegen, die minimal aufbereitet sind. Der Curated Bereich unterteilt sich in einen allgemeinen und einen anwendungsbezogenen Bereich. Der allgemeine Bereich enthält die zentrale Datenbasis, die alle bereinigten und angereicherten Daten enthält. Neben den Dateien und Metadaten können Programmskripte zur Verarbeitung der Daten in einem Ordner abgelegt werden²⁵. Der anwendungsbezogene Bereich ist über ein separates Verzeichnis in der Curated Zone zu erreichen. Dieses enthält alle Anwendungen, die auf die zentrale Datenbasis aufbauen. Die jeweiligen Anwendungsverzeichnisse enthalten ebenfalls Ordner zur Ablage von Daten, Metadaten und Programmskripten. Zusätzlich gibt es einen separaten Sandbox Ordner, wo temporär Ressourcen und Ergebnisse abgelegt werden können. Diese werden in einem bestimmten Intervall gelöscht. Der Sandbox Bereich besitzt dabei eine variable Struktur, die beispielsweise über Nutzerverzeichnisse unterteilt werden kann. Jeder Anwender eines Projekts so individuelle Untersuchungen durchführen und die Ergebnisse bei Bedarf zu teilen.

3.4. *Verarbeitung*

In diesem Kapitel sollen die Frameworks zur Verarbeitung der Daten im Data Lake vorgestellt werden. Für eine Verarbeitung kleiner bis mittelgroße Datenbestände wird Pandas verwendet. Pandas ist eine häufig verwendete Bibliothek für Python, die für Datenmanipulation und -analyse benutzt wird, vgl. (VanderPlas, 2018, S. 116). Für rechenintensivere Operationen mit großen Datenmengen wird Spark eingesetzt, welches im Gegensatz zu Pandas verteilt und parallelisiert auf mehreren Rechneinheiten

²⁵ Dies hat außerdem den Hintergrund, dass die Daten mehrere Datenquellen in verknüpfter Form vorliegen, sodass eine Unterscheidung nach Bereitsteller nicht mehr möglich wäre

arbeitet. Damit die praktische Durchführung der Transformationen in der Curated Zone nachvollzogen werden kann, werden beide Technologien in den nächsten Abschnitten kurz vorgestellt.

3.4.1. *Spark*

Wie bei Hadoop gibt es für Spark ein ganzes Ökosystem an Komponenten, die für jegliche Anwendungen mit Big Data Anforderungen geeignet sind. Im Folgenden werden die für diese Arbeit relevanten Bestandteile von Spark erläutert²⁶, die auch im Anwendungsfall verwendet wurden:

Spark Core: Ist die Kernkomponente für die Datenverarbeitung im Spark System. Die Kernschicht übernimmt die Speicherverwaltung, die Fehlertoleranz und die Steuerung sowie Überwachung der Verarbeitungsaufgaben, sog. Spark Jobs. Auf diesen Kern bauen weitere Bestandteile von Spark auf.

Spark SQL: Spark SQL ist ein Modul um relationale Operationen. Es unterstützt dabei traditionelle relationale Abfragen der Daten über SQL²⁷.

Spark ML: Ermöglicht eine Datenanalyse mit verschiedensten Machine Learning Verfahren. Hierüber können Modelle trainiert und validiert sowie Daten transformiert werden. Die Auswahl an Verfahren reicht von der klassischen Regressionsanalyse, bis hin zu erweiterten Verfahren wie Random Forest oder Support Vector Machines. In Kapitel 4.3 erfolgt die Anwendung von Spark ML mit tieferen Einblicken über Struktur und Funktionen.

Daneben bietet Spark noch Bibliotheken für Echtzeitverarbeitung (Spark Streaming) und Graph Processing (GraphX)

Die Repräsentation von Daten erfolgt in Spark über **Resilient Distributed Datasets (RDD)**, und sind unveränderbare Datensammlungen (*resilient*), welche über mehrere Rechnerknoten verteilt (*distributed*) parallelisiert verarbeitet werden können vgl. (Grover, 2015, S. 97). Die Daten werden, ähnlich wie in Hadoop, in Blöcke partitioniert, die redundant vorliegen. Seit der 2. Version von Spark werden die RDDs durch sog.

²⁶ Für nähere Information siehe Spark Dokumentation <https://spark.apache.org/docs/latest/index.html>

²⁷ Spark SQL ist konform zu ANSI SQL 2011

DataFrames abgelöst. Ein DataFrame kann als eine flache Tabelle interpretiert werden und besitzt dieselben Eigenschaften wie ein RDD.

Ein DataFrame organisiert sich in Zeilen und Spalten. Die Spalte (in Spark sog. Column) eines DataFrames kann neben den nativen Datentypen auch komplexere Strukturen²⁸ abbilden. Ein Beispiel für eine Darstellung eines DataFrames ist auf Abbildung 19 zu entnehmen. Hier wurden alle Informationen zu einer Filiale abgefragt.

ID	Strassenname	PLZ	Region
1	Musestieg 5	06502	Thale
2	Oeringer Str. 20	06484	Quedlinburg
3	Felsenkellerweg 21	06493	Ballenstedt
4	Kyffhäuserstr. 64	06567	Bad Frankenhausen
5	Weinbergstr. 2	06526	Sangerhausen
6	Hauptstr. 55/56	06577	Heldrungen
7	Geschwister-Schol...	06449	Aschersleben
8	Kachstedter Str. 4	06556	Artern
9	Klausstr. 1A	06343	Mansfeld

Abbildung 19 Darstellung eines Spark DataFrame mit Angaben zu Filialstandorten

Über die Spark SQL Schnittstelle können auf einem DataFrame relationale Abfragen und Operationen durchgeführt werden. Die Abfolge von Verarbeitungsschritten auf DataFrames wird über einen sog. **Directed Acyclic Graph (DAG)** abgewickelt, vgl. (Grover, 2015, S. 95). Über diesen werden Abfolgen von Operationen logisch abgebildet, sodass aufeinander aufbauende Verarbeitungsschritte optimiert werden können. Damit verbunden begründet sich auch die Fehlertoleranz von Spark, da bei einem Ausfall von Rechnerknoten Transformationsschritte notfalls zurückverfolgt und anschließend neu berechnet werden können. So brauchen – im Gegensatz zu dem MapReduce Verfahren in Hadoop - keine Zwischenergebnisse persistiert werden, sondern lediglich der DAG neu ausgeführt werden.

Die parallelisierte Verarbeitung von Daten erfolgt in Spark über **Worker** und **Executer**. Der Worker ist die Rechneinheit, bzw. Hardware. Ein Worker kann mehrere Executer-Instanzen erzeugen und führt darüber Verarbeitungsaufgaben (sog. **Tasks**) aus. Die

²⁸ Über sog. UserDefinedTypes können Java-Klassen abbilden. Diese werden in Python jedoch nicht unterstützt

Datenverarbeitung kann über Programmcode geschrieben werden. Eine Kette von Verarbeitungsanweisungen wird in Spark als Anwendung bezeichnet. Im Programmcode kann eine Verbindung zum Spark Cluster über eine **SparkSession**²⁹ aufgebaut werden. Über die SparkSession werden die jeweiligen Anwendungen im Cluster ausgeführt und Ergebnisse zurückgeliefert. Hadoop und Spark sind insoweit miteinander integriert, dass über eine Spark Anwendung direkt auf die im HDFS befindlichen Dateien zugegriffen und ins Dateisystem geschrieben werden kann.

Spark bietet zusätzlich Schnittstellen zu vielen etablierten Programmiersprachen an, wie Java, Scala, R und Python. Somit kann in diesem Projekt die komplette Verarbeitung in Python durchgeführt werden.

3.4.2. *Pandas*

Python hat sich als Programmiersprache für Datenverarbeitung und -analyse etabliert (VanderPlas, 2018, S. 15). Dies liegt unter anderem an der Vielzahl an freien Bibliotheken – hierzu zählt das Modul Pandas (Python and data analysis). Über Pandas können neben klassischen Datentransformationen auch Auswertungen über verschiedene Verfahren durchgeführt und visualisiert werden. Zu den Verfahren zählen beispielsweise unterschiedliche Methoden zur Schätzung von fehlenden Werten, welche in dieser Arbeit verwendet werden. Die zugrundeliegende Datenstruktur von Pandas ist das sog. **DataFrame**, welches ähnlich zu dem Spark DataFrame³⁰, eine flache Tabelle mit Zeilen und Spalten darstellt. Seit Version 23.4 ist es möglich einen Datensatz direkt vom HDFS in ein Pandas DataFrame zu laden. Da jedoch zu Beginn dieser Arbeit Version 22 aktuell war, muss das Ladevorgang über Spark durchgeführt werden. Die Konvertierung eines Spark DataFrames in ein Pandas DataFrame kann dabei über die Methode toPandas() erfolgen. Jedoch führt dies häufig zu Problemen mit der Datentypenkonvertierung und der Repräsentationen von Nullwerten.

²⁹ Vor Spark Version 2.0 wurde hierfür der SparkContext genutzt.

³⁰ So wie das Spark DataFrame, ist auch ein Pandas DataFrame immutable. Es wird daher bei jeder Transformation ein neues DataFrame Objekt erzeugt.

3.5. *Entwicklungsumgebung und Visualisierung*

Der Data Lake wird auf einem Linux-System implementiert, auf dem Hadoop, Spark und Python installiert sind. Für eine Steuerung und Verwaltung von Aufgaben, wird eine Entwicklungsumgebung benötigt, die mit diesen Komponenten integriert werden kann. Als Entwicklungsumgebung wurde daher Jupyter Notebooks gewählt, einer browserbasierten grafischen Schnittstelle, über die einerseits Programmcode für Julia, Python und R ausgeführt werden kann und andererseits Interaktionen mit dem zugrundeliegenden System erlaubt vgl. (VanderPlas, 2018, S. 20). Dies geschieht über einen sog. **Kernel**, welches die Laufzeitumgebung der entsprechenden Programmiersprache darstellt. Über Jupyter kann auf die System-Shell zugegriffen werden, um beispielsweise im Systemspeicher oder HDFS Verzeichnisse anzulegen, Dateien herunterzuladen sowie zu kopieren. Außerdem kann so die verteilte Infrastruktur von Spark und Hadoop gesteuert werden.

Die erstellten Programmskripte bzw. sog. Notebooks strukturieren sich dabei in sog. **Zellen**, welche ausführbare Codefragmente darstellen. Jupyter erlaubt darüber auch die Erstellung einer Dokumentation und die Darstellung verschiedenster Visualisierungen, wie eingebettete Bilddateien oder von Pandas erzeugte Grafiken. Dies ist gerade im Zusammenhang mit dem Data Lake Konzept von Vorteil, da so Prozesse und Datenbestände für jeden Anwender verständlich dokumentiert werden können.

Es ist möglich, eine Instanz von Jupyter auf einem leistungsstarken entfernten Rechner laufen zu lassen, und sich von einem Clientrechner per Browser zu verbinden. In dieser Arbeit wird eine entfernte Instanz von Jupyter über einen SSL Tunnel vom lokalen Arbeitsrechner gesteuert.

In der nachstehenden Grafik ist ein Notebook abgebildet, welches zum einen die Verbindung mit dem Spark Cluster über die SparkSession zeigt und zum anderen die Zugriffsmöglichkeit auf das HDFS veranschaulicht.

The screenshot shows a Jupyter Notebook titled 'osm_geodaesie_contains_node'. The top bar includes the Jupyter logo, the title, and a status message 'Last Checkpoint: vor ein paar Sekunden (autosaved)'. On the right, there is a 'Logout' button. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. A 'Trusted' badge and 'Python 3' are also visible. The toolbar contains icons for file operations, running cells, and code execution. The notebook content consists of two input cells. The first cell, labeled 'In [1]:', contains a Spark configuration command: `spark.sparkContext.getConf().getAll()`. The output, labeled 'Out[1]:', is a list of 18 Spark configuration properties and their values, such as `('spark.dynamicAllocation.minExecutors', '1')`. The second cell, labeled 'In [2]:', contains a bash command: `hdfs dfs -ls /meinert/raw/osm_geo/`. The output shows the results of the command, listing three items: `osm_germany_nodes.osm`, `osm_germany_nodes_tags.osm`, and `osm_germany_nodes_tags.parquet`.

Abbildung 20 Jupyter Umgebung mit bestehender Verbindung zum Spark System und Betriebssystem Kommandos

In nachfolgender Tabelle sollen die verwendeten Technologien aus den Aufgabenfeldern Integration, Speicherung, Verarbeitung, Analyse, Visualisierung nochmal zusammenfassend beschrieben werden. Dabei wird auch auf die Verwendung im Anwendungsfall eingegangen.

Tabelle 10 Übersicht über die in dieser Arbeit verwendeten Technologien

Name	Beschreibung	Anwendung
<i>Integration und Speicherung</i>		
Lokales Dateisystem der Linux Maschine	Das lokale Arbeitsverzeichnis der virtuellen Maschine	Speicherung von Open Data im ursprünglichen Format zur weiteren Strukturierung in der Landing Zone.
Hadoop Distributed File System (HDFS)	Skalierbares und verteiltes Speichersystem. 3 Datanodes 1 Namenode 1 Secondary Node	Ablage aller Daten aus der Raw, Curated und Working Schicht des Data Lakes.

<i>Verarbeitung</i>		
Spark	Framework zur Verarbeitung und Analyse großer Datenmengen. Vorliegend wird die Python Schnittstelle verwendet. Struktur: 3 Worker mit jeweils skalierbarer Anzahl an Executer	Geeignet für Anwendungen auf großen Datenmengen von heterogenen Formaten. Die Parallelität wird hier z.B. für das Verorten von geografischen Informationen auf Regionen benötigt (siehe Kapitel 3.6)
Python & Pandas	Python ist zum einen eine schnell erlernbare Programmiersprache und besitzt viele Bibliotheken für Datenverarbeitung und Analyse. Hierzu zählt auch die Bibliothek Pandas.	Pandas wird für die Verarbeitung von kleinen und mittelgroßen Datenmengen benutzt. Außerdem werden aus dieser Bibliothek Schätzverfahren von fehlenden Werten verwendet (siehe Kapitel 3.8)
<i>Analyse & Visualisierung</i>		
Jupyter's Notebooks	Integrierte und browserbasierte Entwicklungsumgebung u.a. für Python	Ausführen von Python Programmen und Interaktion Softwarekomponenten wie Spark, Hadoop und dem Linux Betriebssystem. Zusätzlich erfolgt darüber die Dokumentation von verschiedenen Verarbeitungs- und Analyseprozessen
Spark ML	Bestandteil von Spark für die Durchführung von Machine Learning Verfahren	Durchführung der Umsatzschätzung im Rahmen des Anwendungsfalls

Da nun alle Softwarekomponenten für Integration, Speicherung, Verarbeitung und Visualisierung definiert wurden, können die im Data Lake erforderlichen Prozesse praktisch umgesetzt werden. Daher wird im Folgenden mit der Anwendung der Phasen Datenanreicherung, -bereinigung sowie Behandlung fehlender Werte gestartet.

3.6. *Datenanreicherung*

Als Anreicherung wurde in Kapitel 2.3.2 die Erweiterung eines Datenbestandes um zusätzliche Informationen definiert. Zum einen kann dies über eine Zusammenführung von Datenbeständen erfolgen, indem beispielsweise Bevölkerungs- und Infrastrukturdaten mit einem Filialstandort verknüpft werden. Zum anderen durch das Ableiten von bestehenden Daten in neue Informationen, wie die Adressinformationen in geografische Koordinaten. In den folgenden Abschnitten werden Beispiele beider Varianten für die Datenanreicherung gegeben.

3.6.1. *Verknüpfung von Daten*

Bevor großflächig Transformationen an den Datensätzen durchgeführt werden, ist es sinnvoll, zuerst semantisch zusammengehörige Datenbestände zu verknüpfen.

Der Zensus stellt separate Dateien der Bereiche Bevölkerung, Familie und Haushalt sowie Gebäude und Wohnung bereit. In jedem Datensatz werden die Attribute auf eine Region bezogen, welche wiederum mit dem zwölfstelligen Regionalschlüssel identifiziert werden. So lassen sich alle Datenbestände über diesen Schlüssel verknüpfen, um nachfolgend Bereinigungsverfahren über die gesamten Daten durchzuführen.

Hierfür werden alle Datensätze aus HDFS in ein Spark DataFrame geladen. In Spark muss jede Spalte typisiert werden, damit bei der Verarbeitung korrekt mit diesen Daten gearbeitet werden kann. Es gibt dabei zwei Methoden, um die Spalten des zu ladenden Datensatzes zu typisieren – über ein Metadatenschema, oder über eine automatische Typerkennung. Die Definition eines Metadatenschema ist dabei eine sehr sichere Methode, um Fehler in der Typisierung zu vermeiden. Bei Datensätzen mit vielen Merkmalen ist der manuelle Aufwand jedoch sehr hoch. Die Datensätze vom Zensus besitzen jeweils über 100 Merkmale, in deren Schemadefinition folglich über 300 Datentypen definiert werden müssten. Daher bietet sich die automatische Typerkennung an, welche in Abhängigkeit der ersten 10% aller Werte je Spalte eine Entscheidung über den Datentyp trifft.

Der Einlesevorgang kann je nach Größe des Datensatzes einige Zeit in Anspruch nehmen und muss nicht unbedingt zu dem erwünschten Ergebnis führen. Beispielsweise kann der Regionalschlüssel einer Region mit einer führenden 0 beginnen. Spark typisiert diese Spalte als Integer, was grundsätzlich richtig ist, jedoch dazu führt, dass die führende 0 eliminiert wird. Da die Verknüpfung der Datensätze über den Regionalschlüssel erfolgen,

ist es notwendig diesen richtig darzustellen. Um die umfangreiche Schemadefinition per Hand zu umgehen, gibt es eine Alternative: Es wird über eine Schleifenkonstruktion ein Metadatenschema konstruiert, welches jedes Merkmal als String typisiert. Dadurch kann sichergestellt werden, dass alle Werte nach dem Einlesen und schließlich beim Abspeichern, korrekt dargestellt werden. Das Problem dabei ist, dass keine Berechnungen mit numerischen Spalten durchgeführt werden können und zudem eine String-Repräsentation mehr Speicherplatz erfordert als für einen Integer oder Float. Da in dieser Phase keine Berechnungen vorgenommen werden und ausreichend physikalische Ressourcen vorhanden sind, können diese Punkte vernachlässigt werden.

Um eine einheitliche Typung zu erhalten, wird zunächst die erste Kopfzeile aus der CSV Datei kopiert und in Python in einen String überführt. Diese enthält alle Spaltenbezeichnungen, die mit einem Semikolon separiert werden. Teilt man diesen Ausdruck an dem Semikolon auf, kann über eine for-Schleifenkonstruktion – genauer eine List Comprehension³¹ – auf die einzelnen Elemente zugegriffen werden. Die Typisierung wird in Spark über ein sog. StructField definiert, welches u.a. den Spaltennamen und den Datentyp (hier StringType()) beinhaltet. Wird so jeder StructField aller Spalten definiert, können diese in einer Liste zusammengefasst und als Metadatenschema definiert werden. Das Metadatenschema wird in Spark über einen StructType umgesetzt, der beim Einlesen des Datensatzes angegeben werden kann. Auf der nachfolgenden Abbildung ist das beschriebene Vorgehen in Python Code dargestellt.

```
schema_string_hh = "AGS_12;RS_Land;RS_RB_NUTS2;RS_Kreis;RS_VB;RS_Gem;Name;Reg_Hier;HH_1.1;HH_1.2;HH_1.3;HH_1.4;HH_1.5;HH_1.6;
fields_hh = [StructField(field_name.replace(".", "_"), StringType(), True) for field_name in schema_string_hh.split(";")]
schema_hh = StructType(fields_hh)
```

Abbildung 21 Einheitliche String-Typisierung aller Spalten im einzulesenden Datensatz (Zensus Haushalte)

Das Einlesen des Datensatzes erfolgt über die Angabe des Dateinamens und Speicherortes im HDFS sowie dem soeben definierten Metadatenschema, wie auf folgender Abbildung verdeutlicht wird.

```
spark.read.csv(dateipfad + datei_HAUSHALTE, sep=";", header=True, encoding="ISO-8859-1", schema=schema_hh)
```

Abbildung 22 Einlesen des Datensatzes über Spark. Der Dateipfad und Dateiname wurden zuvor Variablen zugewiesen

³¹ Für nähere Informationen siehe https://www.python-kurs.eu/list_comprehension.php

Wurden alle Datensätze eingelesen und in ein DataFrame überführt, können diese über eine Join-Operation zusammengeführt werden. Für diesen Zweck wird die Spark SQL Schnittstelle benutzt, in der eine entsprechende SQL Abfrage formuliert werden kann. Dafür müssen sogenannte Views erzeugt werden, welche als schlichte SQL Tabellen interpretiert werden können. Neben der Tabellenbezeichnung kann die Tabelle zusätzlich als temporär gekennzeichnet werden, sodass diese nach Beendigung der Spark Session gelöscht wird. Dies wird mit folgender Codezeile erreicht:

```
# Erzeugung der SQL Tabellen
df_hh.createOrReplaceTempView("hh")
df_geb.createOrReplaceTempView("gebaeude")
df_bev.createOrReplaceTempView("bev")
```

Abbildung 23 Erzeugung einer temporären SQL Tabelle in Spark

Sind alle DataFrames in Tabellen ausgelagert, kann mit einer SQL-Anweisung eine Zusammenführung der Datensätze zu Bevölkerung, Haushalte und Gebäude über den Regionalschlüssel (RS) erfolgen. Die Verknüpfung ist auf der nachstehenden Abbildung über das Spark SQL Modul durchgeführt worden.

```
df_join = spark.sql("""
    SELECT hh.*, bev.*, gebaeude.*
    FROM hh
        JOIN bev
            ON hh.AGS_12 = bev.bev_AGS_12
        JOIN gebaeude
            ON gebaeude.geb_AGS_12 = bev.bev_AGS_12
    """)
```

Abbildung 24 Durchführung einer Join Operation mit Spark SQL

Schließlich kann der so verknüpfte Zensusdatensatz im HDFS persistiert werden.

```
df_join.write.csv(dateipfad + "ZENSUS_ANGEREICHERT.csv", sep=";")
```

Abbildung 25 Schreibvorgang der zusammengeführten Datei ins HDFS mit Spark

Auf dem zusammengeführten Datensatz können in der nächsten Phase Bereinigungsverfahren durchgeführt werden. Die nachfolgenden Abschnitte beschreiben eine spezielle Form der Datenanreicherung, deren Vorgehen im Folgenden erläutert wird.

3.6.2. Ableiten von geografischen Koordinaten aus Adressinformationen

Eine Verknüpfung von Datenbeständen kann unter Umständen komplizierter sein als oben beschreiben, z.B. wenn kein gemeinsamer Schlüssel vorliegt. Ein Beispiel ist das Lokalisieren der Filialen in den jeweiligen Regionen. Hierfür müssen die Adressen der Filiale auf die Fläche einer Region verortet werden. Derzeit wird die Adresse über Postleitzahl, Straße und Hausnummer logisch referenziert, liegen für die Regionen geografische Umrisse über Koordinaten vor, die vom Bundesamt für Geodäsie und Kartographie stammen. In dieser Form sind beide geografischen Angaben nicht vereinbar. Hinter jeder Adressangabe verbirgt sich jedoch ein geografischer Punkt, welcher mit Längen- und Breitengrad beschrieben werden kann. Um diese Koordinatenangaben aus der Adressinformation abzuleiten, gibt es Dienste, welche ein sogenanntes „Geocoding“ anbieten. Die Google Maps API verfügt über einen sehr großen Kartendatenbestand und bietet diesen Service bis zu einer gewissen Grenze kostenlos an³². Um die Koordinaten einer Adresse zu erhalten, muss diese in eine http-Anfrage an die REST-Schnittstelle des Dienstes eingebettet werden. Diese sieht wie folgt aus:

Adresse = Heinrich-Böll-Weg 12, 32120 Hiddenhausen

Anfrage =

*<https://maps.googleapis.com/maps/api/geocode/json?address={32120%20Heinrich-Böll-Weg%2012}>*³³

Der Google Server gibt dann ein JSON Objekt zurück, welches umfassende Informationen über diesen Standort enthält. Darin enthalten sind geografische Informationen wie der Längen- und Breitengrad, die hier mit Longitude und Latitude bezeichnet werden. Die Koordinaten können aus dem JSON Objekt extrahiert und zum Datensatz einer Filiale hinzugefügt werden. Der Vorgang wird auf Abbildung 26 verdeutlicht.

³² Im Juli 2018 waren dies 2500 Anfragen pro Tag auf 50 Anfragen die Sekunde limitiert

³³ Im Jahre 2019 können keine anonymen Anfragen durchgeführt werden. Über ein Google Benutzerkonto muss ein eindeutiger API-Key generiert werden, der bei jeder Anfrage übergeben werden muss

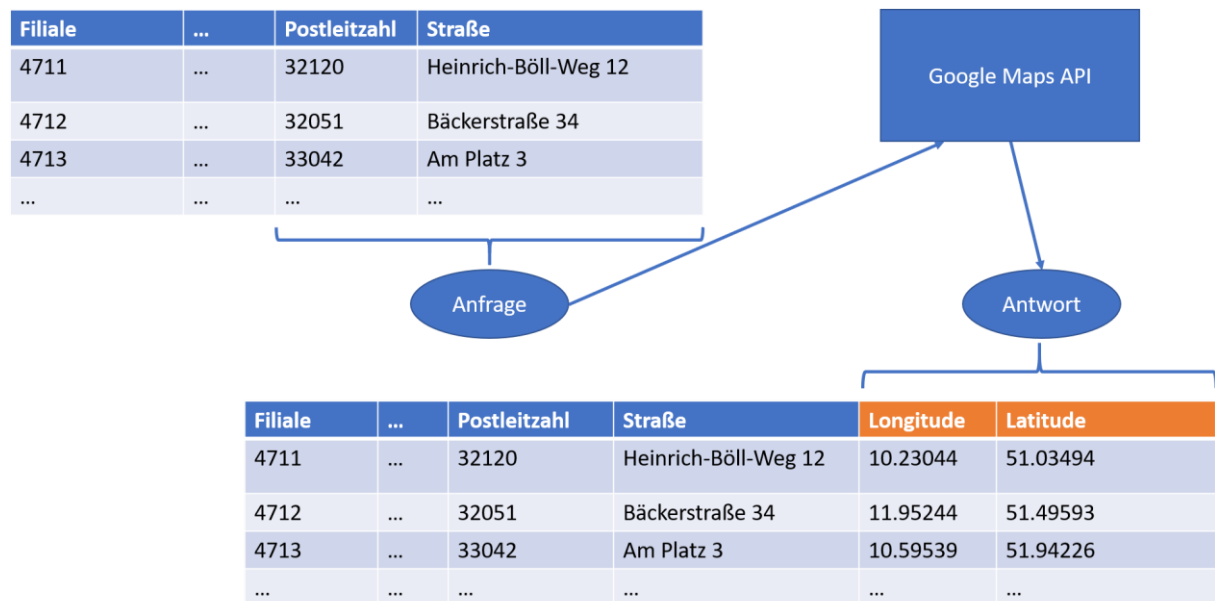


Abbildung 26 Ablauf der Anreicherung von geografischen Informationen eines Filialstandorts über die Google API

Der so gewonnene geografische Punkt einer Filiale kann daraufhin mit den Umrissen einer Region verglichen werden. Liegt dieser Punkt innerhalb der Fläche einer Region, wird die Identifikationsnummer der Filiale in der jeweiligen Region hinterlegt. Darüber kann der Umsatz einer Filiale mit den Bevölkerungs- und Infrastrukturdaten in Zusammenhang gebracht werden. Sind alle Filialen mit den Regionen verknüpft, kann für jede Region ein durchschnittlicher Umsatz der darin enthaltenen Filialen berechnet werden. Die Analyse der Umsatzdaten sowie Standorteigenschaften einer Filiale wird im 4. Kapitel dieser Arbeit durchgeführt. Der Programmcode des beschriebenen Vorgehens ist aus dem Anhang 2 zu entnehmen.

Diese Art der Verknüpfung besitzt einen aufwändigen Berechnungsprozess, in dem ein Abgleich von vielen Koordinaten erfolgen muss. Eine Region kann eine vielfältige Struktur besitzen, wodurch viele Koordinatenpunkte von Nöten sind um den Umriss akkurat zu beschreiben. Eine Abfrage, ob eine Koordinate innerhalb dieser Grenzen liegt, kann daher rechen- sowie zeitintensiv sein. Da es sich bei 3348 Filialen um eine verhältnismäßig kleine Menge an Daten handelt, kann die Verknüpfung mit Pandas durchgeführt werden. Die Berechnung wurde auf dem Masterknoten des Clusters ausgeführt und belief sich dabei etwa 5 Stunden. Im nächsten Beispiel wird die Verknüpfung mit einer großen Datenmenge durchgeführt, bei der das verteilte Architektur von Spark genutzt wird, um die Verarbeitungsdauer zu verringern. Hierfür werden 1,6 Millionen geografische Punkte auf die knapp 11.000 Regionen verortet.

3.6.3. Verknüpfung von geografischen Punkten und Regionen mit Spark

Im Datensatz von OpenStreetMap (OSM) werden u.a. Bildungseinrichtungen, Einkaufsmöglichkeiten oder öffentliche Gebäude in Deutschland verortet. Diese Infrastruktureigenschaften können zu jeder Region hinzugefügt werden. Die geografischen Umriss einer Region werden in einem GeoJSON Format bereitgestellt, in denen die Strukturen als Polygone bezeichnet werden. Ein Polygon, oder auch „Vieleck“ ist eine in sich geschlossene Fläche³⁴, welche über eine Liste von Koordinatenpunkten definiert wird. Dabei gilt, je komplexer das Polygon, desto länger ist die Liste an Koordinaten. Der Umriss einer Region kann dabei mehrere tausend Koordinatenpunkte beinhalten. Gegeben einer Koordinate eines OSM Objektes, wird so für jede Region überprüft, ob sich dieser Punkt innerhalb des Umrisses einer Region befindet. Der Zusammenhang wird auf folgender Abbildung verdeutlicht.

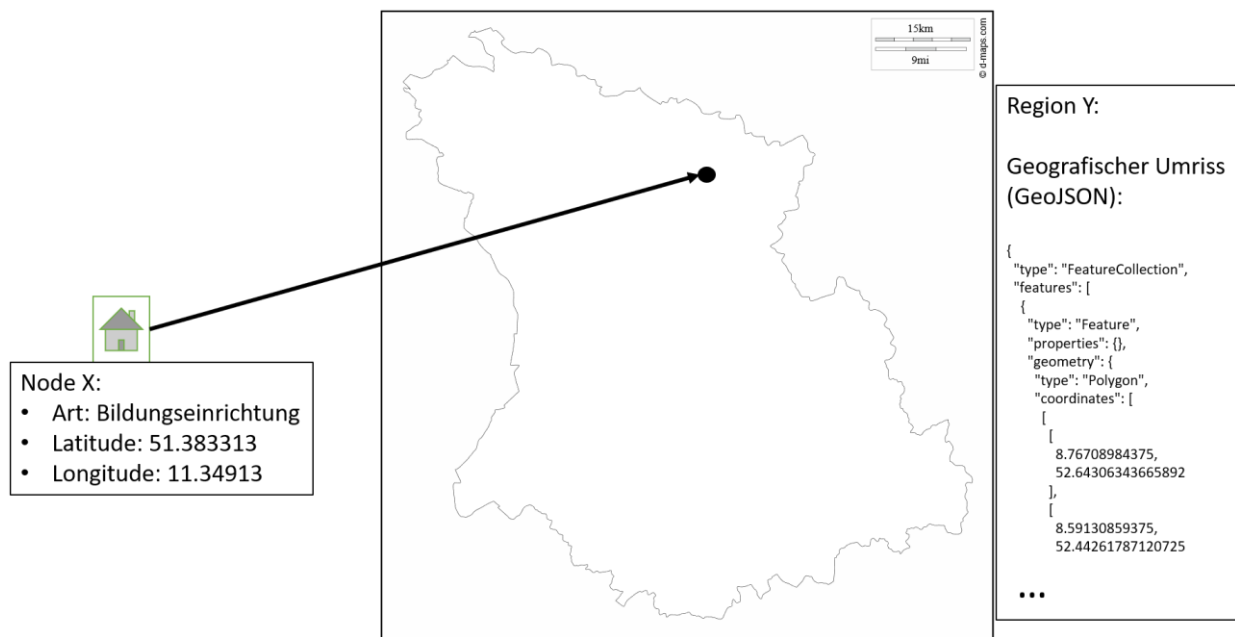


Abbildung 27 Verortung eines geografischen Knotenpunktes im Polygon einer Region

³⁴ Siehe Wikipedia: <https://de.wikipedia.org/wiki/Polygon>

Wird die Verknüpfung der 1,6 Millionen OSM Objekte diese Art umgesetzt, müssten bei 12000 Regionen etwa 19 Milliarden Abfragen erfolgen. Dabei würde die Durchführung mit Pandas hochgerechnet etwa 50 Tage dauern. Um die Verarbeitungsdauer zu verkürzen, bietet sich das verteilte Konzept von Spark an, um Berechnungen über das Rechnercluster durchzuführen. Bei der Verknüpfung soll für jede Region festgehalten werden, wie viele OSM-Objekte enthalten sind.

Spark verfügt über einige Konfigurationsmöglichkeiten um die Berechnungszeit zu verkürzen. Diese sollen kurz genannt werden. Eine Übersicht über sog. Tuning Parameter können aus der Dokumentation³⁵ von Spark entnommen werden.

Der OSM Datensatz wurde in eine **Parquet** Datei überführt, ein spaltenorientiertes und komprimiertes Format, welches einen effizienteren Zugriff auf Spalten erlaubt. Spark arbeitet im Hauptspeicher der jeweiligen Maschinen des Clusters. Dabei bietet Spark sog. **Caching**³⁶ an, um schon vor der Durchführung einer Operation Daten in den Hauptspeicher zu laden. Darüber wurden die Regionsumrisse vorab geladen.

Da die Daten in Spark über mehrere Rechneinheiten, bzw. Worker verteilt werden, können Verarbeitungen durch Instanziierung von Executer parallelisiert ausgeführt werden. Dadurch arbeitet jeder Worker auf einen anderen Teil der Daten. Um ein Matching mit diesen Teilen der Daten durchzuführen, müssen alle Worker auf dieselbe Liste mit den Umrissen aller Regionen zugreifen können. Die Liste selbst darf nicht verteilt vorliegen, sondern muss alle ca. 11.000 Regionen enthalten um ein Matching durchzuführen. Für diesen Zweck können in Spark sog. **Shared Variables**³⁷ verwendet werden. Wird die Regionsliste als „verteilte Variable“ deklariert, können alle Worker auf diese zugreifen und so mit den Fragmenten der OSM Daten ein Matching durchführen.

Über das beschriebene Vorgehen konnte die Verarbeitungszeit von 50 Tagen mit Pandas mit Spark auf 7 Stunden reduziert werden. Der Programmcode kann aus dem Anhang 3 entnommen werden.

³⁵ Siehe Spark Dokumentation: <https://spark.apache.org/docs/latest/sql-performance-tuning.html>

³⁶ Für weitere Informationen siehe Spark Dokumentation: <https://spark.apache.org/docs/latest/rdd-programming-guide.html#rdd-persistence>

³⁷ Für weitere Informationen siehe Spark Dokumentation: <https://spark.apache.org/docs/latest/rdd-programming-guide.html#shared-variables>

Die Beispiele sollen zeigen, dass es unterschiedliche Wege gibt Daten miteinander zu verknüpfen. Liegen die Daten verknüpft vor, können nun Bereinigungsverfahren angewendet werden, die nun vorgestellt werden.

3.7. *Datenbereinigung*

Um eine aufbereitete zentrale Datenbasis in der Curated Zone zu erzeugen, müssen Fehler bereinigt und Standards eingeführt werden. Die Fehler und Standards werden zunächst definiert.

Um die gesetzlichen Vorgaben des Personendatenschutzes umzusetzen, werden die Daten des Zensus über ein sog. Geheimhaltungsverfahren geändert. Damit in bevölkerungsarmen Regionen keine Merkmale auf eine Person (z.B. Herkunft oder Religionszugehörigkeit) eindeutig zurückzuführen ist, werden daher einige Angaben stark erhöht, mit einem negativen Vorzeichen oder mit Klammern versehen. Die im Zuge des Geheimhaltungsverfahrens geänderten Werte werden als Fehler definiert, da diese nicht die Wirklichkeit abbilden. Daher werden sie im Bereinigungsprozess entfernt und als fehlende Werte behandelt.

Die Standards sehen beispielsweise einen einheitlichen Umgang mit der Repräsentation von fehlenden Werten vor. Fehlende Werte werden je nach Datenquelle auf unterschiedlichste Weise gekennzeichnet. Im Zensusdatensatz werden diese durch Zeichen wie „-“, „/“ oder einem Leerzeichen repräsentiert. Da diese von Anwendungen wie Spark oder Pandas als Zeichenkette interpretiert werden, werden auch numerische Merkmale als String typisiert. Um den numerischen Merkmalen den korrekten Datentyp zuzuweisen, werden die mit fehlenden Werten in Python mit *None* ersetzt. *None* ist das Äquivalent zu *Null*, welches in Programmiersprachen wie Java, C++, C# verwendet wird. In Python ist *None* kompatibel mit allen Datentypen.

Im Folgenden wird die Bereinigung des Fehlers und die Umsetzung des Standards über eine einzige Funktion abgehandelt. In der Python-Funktion wird der aktuelle Wert abgefragt und durch *None* ersetzt, falls dieser einem maskierten oder fehlenden Wert entspricht. Diese Abfrage wird in Python mit der eingebauten Funktion *set()* umgesetzt, welche als mathematische Menge interpretiert werden kann. Mit Mengen von Zahlen

oder Zeichen können verschiedene Operationen durchgeführt werden. Werden jeweils der aktuelle Wert und die Zeichenkette für maskierte/fehlende Werte in Sets überführt, kann die Schnittmenge beider Sets mit *intersection()* gebildet werden. Ist die Schnittmenge nicht leer, so handelt es sich bei dem aktuellen Wert um einen maskierten/fehlenden Wert und wird mit None überschrieben. Falls nicht, wird der Wert unverändert zurückgegeben. Das beschriebene Vorgehen wird wie folgt umgesetzt:

```
def ersetzung_funktion(wert):  
    """ Ersetzt fehlende Werte maskiert mit None """  
  
    zeichen_fehlende_und_maskierte_werte = set("-/ ()")  
  
    schnittmenge = set(wert).intersection(zeichen_fehlende_und_maskierte_werte)  
  
    if(len(schnittmenge) > 0):  
        wert_neu = None  
  
    else:  
        wert_neu = wert  
  
    return wert_neu
```

Abbildung 28 Ersetzung Funktion zur Datenbereinigung mit Python

In Kapitel 3.6.1 wurde über Spark eine Anreicherung der Zensusdaten durchgeführt. Darauf aufbauend kann nun die Bereinigung mit der definierten Funktion durchgeführt werden. Da Spark verteilt arbeitet, muss die Funktion zunächst zu den einzelnen Knoten transportiert und anschließend parallel ausgeführt werden. In Spark müssen selbst geschriebene Methoden als User-Defined-Function (UDF) deklariert werden, die als Wrapper fungieren. Der UDF wird dann die selbst definierte Funktion und der Datentyp des Rückgabewertes übergeben. Der tatsächliche Datentyp ist von dem jeweiligen Merkmal des DataFrames abhängig. Daher müsste für jeden Datentyp eine eigene UDF definiert werden. Durch die Vielzahl an Spalten im Zensusdatensatz wurde zuvor ein Metadatenschema konstruiert, welches jedes Merkmal als String typisiert. Die verwendete UDF braucht sich daher nur einen Rückgabebetyp beschränken, wie folgende Programmzeile veranschaulicht.

```
ersetzung_funktion_udf = udf(ersetzung_funktion, StringType())
```

Abbildung 29 Zuweisung der definierten Funktion in eine UDF

Die UDF kann nun auf die entsprechenden Spalten des Datensatzes anwenden werden. Um in Spark eine Transformation an einer Spalte vorzunehmen, wird diese in der DataFrame Methode `withColumn()` unter der Angabe der definierten UDF übergeben. Da das DataFrame unveränderlich ist, wird bei der Anpassung einer Spalte stets neues DataFrame erzeugt, sodass dieses neu zugewiesen werden muss. Mit einer for-Anweisung wird über die Spalten des Datensatzes iteriert, die fehlende und maskierte Werte enthalten, wie folgende Darstellung veranschaulicht.

```
## Fehlende und maskierte Werte ersetzen  
for spalte in zensus.columns[8:388]:  
    zensus = zensus.withColumn(spalte, ersetzung_funktion_udf(zensus[spalte]))
```

Abbildung 30 Bereinigung des Zensusdatensatzes über eine Schleifenkonstruktion

Damit wurde ein einheitlicher Standard für maskierte und fehlende Werte umgesetzt. Das gezeigte Vorgehen wurde auf die Datenbestände von destatis und Geodäsie in angepasster Weise übertragen, um in der Curated Zone eine aufbereitete Datenbasis mit einheitlicher Datenqualität zu erzeugen.

Innerhalb der Curated Zone können anwendungsbezogene Daten aus der zentralen Datenbasis in einen separaten Bereich überführt werden, um je nach Anforderungen weitere Verarbeitungsschritte durchzuführen. Beispielsweise können hier im Rahmen der Filialnetzoptimierung, fehlende Werte behandelt werden, wie im nachfolgenden Kapitel beschrieben wird.

3.8. *Behandlung fehlender Werte*

Wie in Kapitel 3.1 beschrieben, soll in dieser Arbeit eine Filialnetzoptimierung über ein Machine Learning Verfahren durchgeführt werden. Für diesen Zweck ist es erforderlich, sich mit dem Problem der fehlenden Daten zu beschäftigen, da nach (Géron, 2018, S. 60) die meisten Machine Learning Algorithmen nicht mit unvollständigen Daten arbeiten können.

Die Behandlung fehlender Werte kann über zwei Arten erfolgen. Entweder durch Löschung des Objektes bzw. Merkmals aus dem Datenbestand, oder durch Ersetzung mit einem Schätzwert. Für beide Varianten werden in diesem Kapitel Beispiele gegeben. Der

nachfolgende Abschnitt beschreibt das Vorgehen der Löschung der Merkmale mit fehlenden Daten.

3.8.1. *Eliminierung von Merkmalsspalten mit fehlenden Werten*

Unabhängig von den Anforderungen einiger Machine Learning Algorithmen bzgl. fehlender Werte, kann das Problem auch aus einem anderen Blickwinkel betrachtet werden. Ab einem bestimmten Ausmaß an fehlenden Werten, werden die Merkmale für eine Analyse irrelevant, da z.B. keine verlässlichen Aussagen über den Einfluss auf eine Zielvariable gemacht werden kann, vgl. (Géron, 2018, S. 26)

Im Zensusdatensatz werden etwa 12544 Regionen durch 388 Merkmalen aus Bevölkerung und Infrastruktur beschrieben. In der nachfolgenden Übersicht sind die Merkmale nach der Anzahl fehlender Werte sortiert. Es ist zu erkennen, dass einige Merkmale mit sehr wenig Werten besetzt sind.

Spaltenname	Anzahl fehlende Werte
WHG_2_9	11978
WHG_2_10	11978
WHG_2_8	11978
WHG_2_7	11978
WHG_2_6	11978
WHG_2_5	11978
WHG_2_4	11978
WHG_2_3	11978
WHG_2_2	11978
WHG_2_1	11978
ERW_4_5	10398
ERW_4_11	10384
ERW_4_9	10361
ERW_4_13	10359
ERW_4_15	10359
ERW_1_10	10358
MIG_1_9	10357
MIG_1_2	10357
MIG_1_3	10357
MIG_1_4	10357

Abbildung 31 Anzahl fehlender Werte für jedes Merkmal aus dem Zensusdatensatz

Die ersten zehn Merkmale aus der Auflistung besitzen für nicht einmal 5% aller Regionen entsprechende Werte. Für eine Analyse des Einflusses dieser Merkmale auf den Umsatz ist eine Einbeziehung dieser Merkmale wenig sinnvoll. Daher muss eine Mindestanzahl an

vorhandenen Werten für die Merkmale im Datensatz definiert werden. Dabei werden alle Merkmale verworfen, die diesen Schwellwert unterschreiten. Um nicht willkürlich eine Schwellwertgrenze (z.B. nur 10% fehlende Werte) zu definieren, empfiehlt sich hier eine visuelle Betrachtung des Zusammenhangs zwischen Mindestzahl vorhandener Werte und verbleibender Merkmale im Datensatz. Das beschriebene Vorgehen kann über Pandas mit Hilfe der vorhandenen DataFrame Funktionalität umgesetzt werden. Um auf die fehlenden Werte eines DataFrames zuzugreifen³⁸, wird eine `dropna()` Methode angeboten. Übergibt man dieser einen Schwellwert der erforderlichen Anzahl vorhandener Werte (*thresh*), wird der Datensatz um die Spalten reduziert, die diesen unterschreiten. Auf der nachfolgenden Abbildung werden die Spalten aus dem Datensatz entfernt werden, welche nicht mehr als Werte 11289 (10% fehlende Werte) besitzen.

```
zensus.dropna(axis="columns", thresh=11289)
```

Abbildung 32 Pandas dropna() Funktion zur Entfernung von Spalten die nicht eine Mindestanzahl an Werten besitzen

Die Anzahl der beibehaltenen Spalten verringert sich auf dabei auf 242. Bei 20% fehlender Werte bleiben 258 Merkmale im Datensatz vorhanden, bei 30% sind es 266. Aus diesen drei Werten ist zu erkennen, dass der Zusammenhang zwischen Nullwert-Toleranz und verbleibenden Spalten im Datensatz nicht linear ist. Um einen Schwellwert festzulegen, wird der Verlauf dieser beiden Größen grafisch betrachtet. Auf der folgenden Abbildung ist der Schwellwert auf der x-Achse dargestellt. Auf der y-Achse ist Anzahl der Spalten angegeben, die bei einem gegebenen Schwellwert noch im Datensatz existieren würden.

³⁸

rwendet, werden in Pandas fehlende Werte mit „NaN“ (Not a Number) repräsentiert. Dabei basiert NaN auf einen Datentyp aus der Python Bibliothek numpy. Numpy wird in Python dafür verwendet um mit Vektoren oder multidimensionale Listen und Matrizen zu arbeiten.

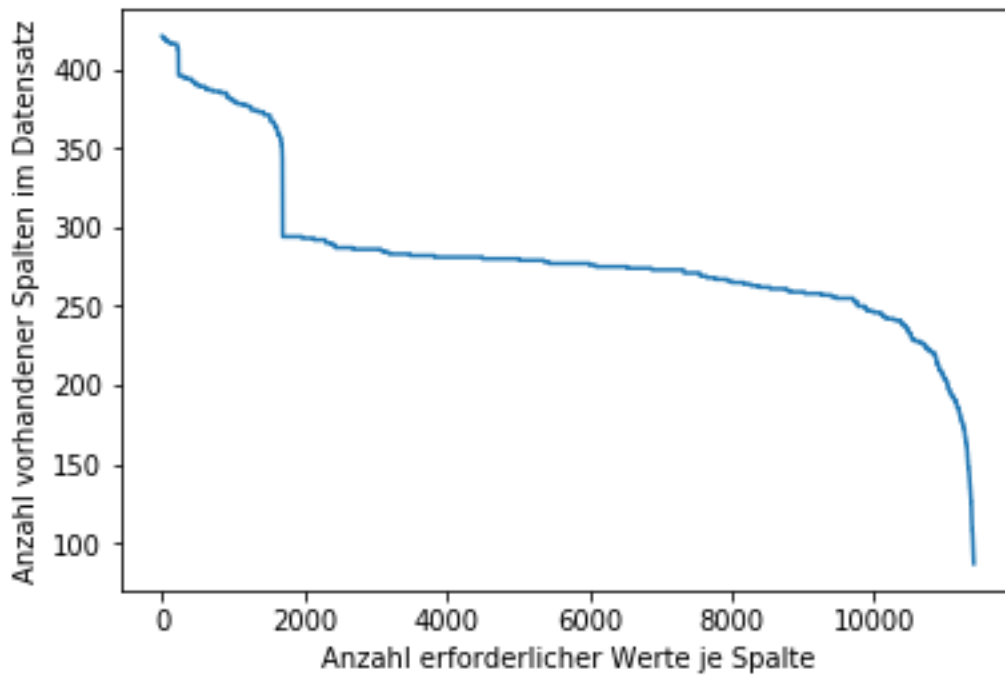


Abbildung 33 Zusammenhang von Spalten im Datensatz zu gegebenen Schwellwerten für die Nullwerttoleranzen

Das Ziel ist es, die Anzahl der fehlenden Werte einer Spalte möglichst gering zu halten und dabei die beibehaltenen Merkmalsspalten zu maximieren. Bei der grafischen Untersuchung werden daher Punkte gesucht, die möglichst weit im oberen rechten Segment gelegen sind. Der Verlauf zeigt einen drastischen Abfall von noch verbleibenden Spalten an den Schwellwerten $x_1 = 1700$ und $x_2 = 10500$. Nach diesen Punkten gehen bei einer weiteren Erhöhung des Schwellwertes sehr viele Merkmale verloren. Da eine Merkmalsspalte möglichst wenige fehlende Daten besitzen soll, wird der Schwellwert $x_2 = 10500$ gewählt. Dieser Wert entspricht etwa einer Toleranz von 16% fehlenden Werten je Merkmal. Somit können 233 Merkmale beibehalten werden, die in der Analyse zu verlässlicheren Ergebnissen führen sollen.

Der nun verdichtete Datensatz kann für die weitere Behandlung fehlender Werte verwendet werden. Als nächstes werden zwei Beispiele beschrieben, die eine Schätzung der fehlenden Werte vorsehen.

3.8.2. *Schätzwerte für fehlende Werte in hierarchischen Daten*

Im vorherigen Kapitel wurde gezeigt, dass Merkmale mit weniger als 16% Werten aus dem Datenbestand entfernt wurden. Es bleibt nun allerdings das Problem, dass den verbliebenen Merkmalen unter Umständen bis zu 16% an Werten fehlen. In Kapitel 2.3.2 wurden Verfahren genannt, um fehlende Werte zu schätzen. Die Schätzung mit Mittelwerten, wie dem arithmetischen Mittel oder dem Median, sind weit verbreitet, vgl. (Géron, 2018, S. 60). Bei einer geringen Anzahl von fehlenden Werten und unter der Annahme, dass diese Werte zufällig fehlen und unabhängig von anderen Merkmalen sind, kann dieses Verfahren zu realitätsnahen Schätzungen führen. Trifft dies nicht zu, kann es zu starken Verzerrungen kommen, die wiederum Auswirkungen auf Ergebnisse einiger statischer Verfahren haben. Bei einer näheren Betrachtung der Daten des statistischen Bundesamtes, kann festgestellt werden, dass eine Schätzung fehlender Werte mit dem Mittelwert nicht sinnvoll ist. Dafür können zwei Gründe angeführt werden.

Da im Datensatz mehrere hierarchische Ebenen vertreten sind, liegen auch aggregierte Werte für Bundesländer, Regierungsbezirke oder Kreise vor. Für eine Schätzung, beispielsweise des durchschnittlichen Einkommens je Region, müsste der Datensatz gemäß der Verwaltungsgliederung unterteilt werden. So würde man die Schätzung des Einkommens auf Kreisebene oder Gemeindeebene separat durchführen. Angenommen, es würde eine Mittelwertschätzung des Einkommens durchgeführt werden, würde diese dennoch nicht zu realitätsnahen Ergebnissen führen. Grund hierfür sind regionale Eigenheiten und Unterschiede, z.B. zwischen Ost- und West-Deutschland, die nicht berücksichtigt werden würden.

Das Fehlen der Werte unterliegt nicht dem Zufall. Das nicht Vorhandensein einiger Verwaltungsdaten ist eher systematischer Natur. Einzelne Verwaltungseinheiten sind nicht dazu verpflichtet, alle Merkmale zu messen oder zu veröffentlichen. Außerdem sind einige Merkmale nur für bestimmte hierarchische Ebenen vorhanden, wie das verfügbare Einkommen oder die Arbeitslosenquote, welche lediglich auf Kreisebene vorliegen. Eine Übertragbarkeit auf andere Ebenen (Bundesländer oder Gemeinden) nur möglich, wenn Durchschnitts- oder Verhältnisangaben vorliegen. Absolute Zahlen, wie Anzahl der Bevölkerung einer Region, können dagegen nicht über eine Verwaltungsebene hinaus geschätzt werden. Daten des Bundesamts für Statistik bzgl. Arbeitslosigkeit, Einkommen und Gebietsnutzung, liegen in relativer Form vor. Entsprechend werden hier Durchschnitte und prozentuale Anteile gemessen, wodurch eine Schätzung entlang der hierarchischen Struktur vorgenommen werden kann.

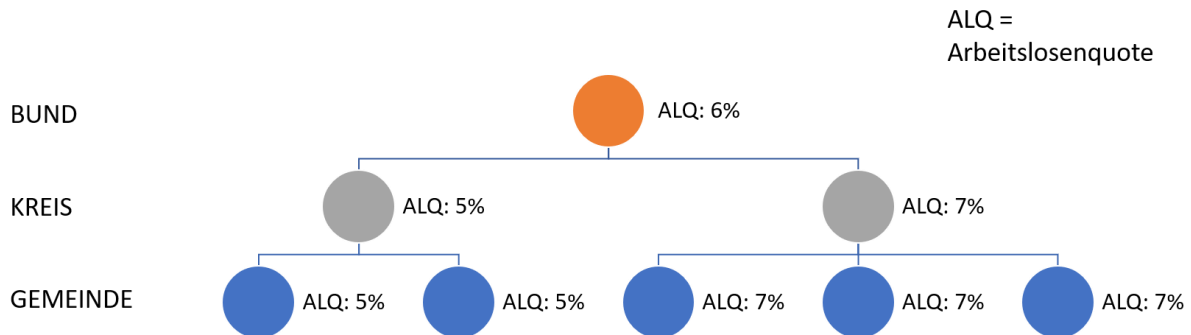


Abbildung 34 Hierarchische Schätzung der Arbeitslosenquote für Bund, Kreis und Gemeindeebene

Beispiel auf Abbildung 34: Für einen Kreis liegt eine Arbeitslosenquote von 5% vor. Für die Gemeinden wird daher auch angenommen, eine Arbeitslosenquote von 5% zu besitzen. Dabei muss berücksichtigt werden, dass der Schätzwert vom realen Wert abweichen kann, da es sich auf Kreisebene um einen Mittelwert der untergeordneten Gemeinden handelt. Dieser Fehler wird jedoch in Kauf genommen, da ansonsten Merkmale bezüglich Arbeitslosigkeit, Einkommen und Gebietsnutzung verworfen werden müssten. Um die Daten für höhere Hierarchieebenen wie Bundesländer zu übertragen, könnten die Angaben für alle Kreise innerhalb eines Bundeslandes gemittelt werden. Da in der Analyse jedoch nur die unterste Hierarchieebene für die Filialnetzoptimierung betrachtet wird, wird die Schätzung lediglich für Gemeinden und kreisfreie Städte durchgeführt.

Implementiert wird diese Schätzung mit Pandas, da sich die Ersetzung von Werten einfacher durchführen lässt als mit Spark. Zunächst wird ein sog. Look-up Datensatz gebildet, in dem alle Kreise mit vorhandenen Daten bzgl. Arbeitslosigkeit, Einkommen und Gebietsnutzung zusammengefasst werden. Die Gemeinden werden in einen separaten Datensatz überführt.

Wie in Kapitel 3.1.1 beschrieben, ist der Regionalschlüssel ebenfalls hierarchisch aufgebaut ist, sodass für eine Gemeinde, der Regionalschlüssel des zugehörigen Kreises abgeleitet werden kann. Beispielsweise sieht das für die Gemeinde Ratingen wie folgt aus:

Gemeinde Ratingen: 05 1 58 0028 028

Kreis Mettmann: 05 1 58 0000 000

Daraufhin kann in der Look-up Tabelle der zugehörige Kreis selektiert und die Werte der Merkmale bzgl. Arbeitslosigkeit, Einkommen und Gebietsnutzung für die Gemeinde übernommen werden. Nach diesem Vorgehen wird über alle Gemeinden im Datensatz iteriert und die fehlenden Werte ersetzt. Für Programmcode siehe Anhang 4.

Am Ende dieses Prozesses liegen für alle Gemeinden und kreisfreien Städte Angaben zur Arbeitslosigkeit, Einkommen und Gebietsnutzung vor und können als Merkmale in die Analyse in Kapitel 4 eingehen.

3.8.3. *Schätzung fehlender Werte einer Zeitreihe*

Ein verbreitetes Problem sind fehlende Werte in Zeitreihen. In diesem Zusammenhang gibt es zwei Arten von Schätzungen, die Interpolation und Extrapolation. Die Daten des Bundesamtes für Statistik erstrecken sich über einen Zeitraum von 2010 bis 2016. Dabei fehlen häufig Ausprägungen für bestimmte Zeitpunkte. Beispielsweise liegen Werte für die Arbeitslosenquote für Ratingen in den Jahren 2011, 2013 und 2015 vor. Eine Schätzung über eine Interpolation erfolgt dabei innerhalb von zwei vorhandenen Zeitpunkten – beispielsweise zwischen den Jahren 2011 und 2015. Die Extrapolation dagegen schätzt Datenpunkte, die sich außerhalb dieses Zeitraumes belaufen, demnach die nicht vorhandenen Angaben für 2010 sowie 2016 – wie auf Abbildung 35 mit einem Beispiel dargestellt wird.

Jahr	Arbeitslosenquote	
2010		Extrapolation
2011	2%	
2012		Interpolation
2013	3%	
2014		
2015	3,5%	
2016		

Abbildung 35 Inter- und Extrapolation von Zeitreihen am Beispiel der Arbeitslosenquote

Für eine realitätsnahe Schätzung ist es notwendig, die wahre Entwicklung eines Merkmals zu kennen. Da Werte in der Zeitreihe fehlen, kann diese jedoch nur anhand der vorhandenen Datenpunkte geschätzt werden. Man nimmt an, dass es eine Funktion $w(x)$ gibt, die den realen Verlauf eines Merkmals widerspiegelt. Daher ist es das Ziel, eine Schätzfunktion $s(x)$ zu finden, welche sich $w(x)$ möglichst genau annähert. Die Schätzfunktion $s(x)$ wird aus den vorhandenen Messzeitpunkten abgeleitet. Nun bieten sich viele Formen einer Funktion an, die je nach Anwendungsfall gewählt werden können. Ist ein linearer Verlauf zu erwarten, wird eine Funktion sog. ersten Grades (klassische Regression) gewählt, um die restlichen Zeitpunkte (hier 2010, 2012, 2014 und 2016) zu schätzen. Nicht in allen Fällen können lineare Entwicklungen in der realen Welt beobachtet werden. Die Arbeitslosenquote einer Region könnte beispielsweise konjunkturellen Schwankungen und wirtschaftspolitischen Ereignissen unterliegen, wodurch eine Vorhersage (Extrapolation) über einen linearen Verlauf nicht realistisch wäre. Daher werden nicht-lineare Funktionen mit Polynomen höherer Ordnung genutzt, die sich den vorhandenen Messzeitpunkten (sog. Stützstellen) orientieren. Ein Polynom³⁹ ist ein mehrgliedriger Term mit mind. einer Potenz, wie z.B. $5x^4 - 2x^3 + 12x + 9$. Der Grad des Polynoms entspricht dabei den Wert der höchsten Potenz – hier beispielsweise 4. Der Verlauf zwischen den Stützstellen hängt dabei vom gewählten Grad des Polynoms ab. Je höher das Polynom, desto komplexer kann der Verlauf zwischen den Stützstellen abgebildet werden. Dies führt jedoch zu unrealistischeren Schätzwerten – dies sei hier lediglich mit dem Verweis auf das *Runges Phänomen*⁴⁰ angedeutet.

³⁹ Siehe Beschreibung Polynom auf Wikipedia: <https://de.wikipedia.org/wiki/Polynom>

⁴⁰ Für detaillierte Beschreibung siehe: https://mathepedia.de/Runges_Phaenomen.html

Vorliegend wurde eine Funktion 2. Ordnung verwendet, um sowohl die Werte innerhalb, als auch außerhalb des Zeitraumes zu schätzen. Auch hier bietet es sich an, mit Pandas zu arbeiten, da eine Schätzung über verschiedenste Verfahren möglich ist. Die gängigsten Verfahren sind *pchip* (piecewise cubic interpolation) und *spline*. Für eine genaue Definition dieser beiden Verfahren siehe Dokumentation⁴¹. Beide Verfahren bilden aus den vorhandenen Datenpunkten ein Polynom höherer Ordnung. Für die Auswahl eines der beiden Verfahren, ist das Monotonieverhalten ausschlaggebend, wie anhand der Arbeitslosenquote gezeigt werden soll. Beide Schätzverfahren wurden hier angewendet, um über den Zeitraum von 2010 bis 2015 fehlende Werte zu ersetzen. Beide Verfahren kommen insbesondere im Jahre 2015 auf unterschiedliche Ergebnisse, wie nachfolgender Tabelle zu entnehmen sind.

Jahr	Arbeitslosenquote (real)	Arbeitslosenquote (spline)	Arbeitslosenquote (pchip)
2010	1%	1%	1%
2011		0,9%	1%
2012	1%	1%	1%
2013	0%	0%	0%
2014	0%	0%	0%
2015		-0.7%	0%

Abbildung 36 Interpolation und Extrapolation über *pchip* und *spline* am Beispiel der Arbeitslosenquote

Die Schätzung des fehlenden Wertes im Jahre 2011 führt in beiden Fällen zu einem ähnlichen Ergebnis. Im Jahre 2015 ist jedoch zu erkennen, dass das *spline* Verfahren eine negative Arbeitslosenquote annimmt, was jedoch nicht realistisch ist⁴². Durch das sog. robustere Monotonieverhalten von *pchip*, wird in dieser Arbeit auf dieses Schätzverfahren zurückgegriffen, um fehlende Messwerte für Zeitpunkte bzgl. Einkommen und Arbeitslosigkeit zu schätzen. Der Programmcode ist aus Anhang 5 zu entnehmen.

⁴¹ Dokumentation Pandas *interpolate*: <https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.interpolate.html#pandas.DataFrame.interpolate>

⁴² In anderen Kontexten kann der negative Wert für 2015 eine realistische Schätzung sein. Pauschal ist kein Verfahren zu bevorzugen, sondern muss in der jeweiligen Anwendung getestet werden

4. *Praktische Anwendung von Maschine Learning Verfahren*

In den vorherigen Kapiteln wurde eine Architektur entworfen um umgesetzt, um Open Data zu speichern und für eine Analyse. Auf diese Datenbasis kann nun zugegriffen werden, um eine Umsatzschätzung für Regionen durchzuführen, die noch keinen Filialstandort haben. In diesem Kapitel geht es um Anwendung verschiedener Konzepte und Methode zur Datenanalyse mit Machine Learning. In Kapitel 2.3.3 wurde die Definition für Feature Selection/Extraction, Feature Transformation und dem generellen Vorgehen im Machine Learning nicht gegeben. Diese soll daher im ersten Teil dieses Kapitels erfolgen. Bevor diese Schritte praktisch am Anwendungsfall durchgeführt werden, müssen für den Anwendungsfall geeignete Machine Learning Verfahren ausgewählt werden. Dafür ist zunächst ein tiefergehendes Verständnis der Daten erforderlich. Konkret werden zuerst die Filialstandorte und Regionen näher betrachtet um gewisse Eigenschaften herauszustellen. Daraufhin werden Abhängigkeiten zwischen den Merkmalen untersucht, um eine geeignete Auswahl von Merkmalen zu finden, welche in das Machine Learning einfließen sollen. Da sich die Schritte der Feature Transformation der verwendeten Verfahren unterscheiden, werden diese erst im jeweiligen Kapitel der Verfahren praktisch durchgeführt.

Es sei darauf hingewiesen, dass die beschriebenen Aktivitäten in diesem Kapitel zwar sequenziell dargestellt werden, jedoch eher einem zyklischen Vorgehen folgen⁴³. Die Ergebnisse werden dabei nicht nur nach vorne gespielt, sondern können auch auf vorgelagerte Prozesse im Data Lake einwirken, die von Datenbeschaffung bis zur -bereinigung reichen. So kann nach der Durchführung der Analyse, die Nachfrage nach weiteren aussagekräftigen Merkmalen bestehen. Diese können beispielsweise über eine Anreicherung von Datenbeständen, oder durch eine gezieltere Behandlung von fehlenden Werten erfolgen.

Es muss berücksichtigt werden, dass die hier verwendeten Verfahren und Konzepte nicht umfassend beschrieben werden können. Daher soll gerade so viel Wissen vermittelt werden, um das Vorgehen zu verstehen und die Ergebnisse interpretieren zu können. Das Optimierungspotenzial der einzelnen Verfahren kann in dieser Arbeit nicht diskutiert werden, da ein tiefergehendes Verständnis dafür vorausgesetzt werden muss.

⁴³ Ähnlich wie im Cross-Industry Standard Process for Data Mining (CRISP-DM) Modell, einem Standard Prozess für Data Mining.

4.1. *Verarbeitungsprozesse in der Working Zone des Data Lakes*

Zunächst ist zu erwähnen, dass der Prozess zur Vorbereitung der Daten für das Analyseverfahren in der Literatur auch unter Feature Engineering verstanden wird, vgl. (Domingos, 2017, S. 5 + 6). Die Aktivitäten die hier unter Feature Selection/Extraction und Feature Transformation verstanden gefasst werden, lassen sich dem Feature Engineering unterordnen.

4.1.1. *Feature Extraction / Selection*

Je nach Datenbasis kann die Menge vorhandenen Merkmalen (sog. *Features*) sehr groß sein. Eine Einbeziehung einer großen Anzahl an Merkmalen kann in einigen Verfahren zu Verzerrungseffekten führen. Dieses Problemfeld wird in der Literatur auch als „Fluch der Dimensionalität“ bezeichnet, da jedes Merkmal eine neue Dimension aufspannt, die das Analysemodell komplexer werden lässt, vgl. (Raschka, 2017, S. 122). Das Bestreben ist daher, den Merkmalsraum so zu verdichten, sodass dieser nur die relevantesten Merkmale für die Analyse enthält. Dieses Vorgehen hat mehrere Vorteile. Zum einen erfordern einfachere Analysemodelle weniger Rechenleistung und Rechenzeit, was insgesamt zu einer schnelleren Entscheidungsfindung führt. Zum anderen können weniger aussagekräftige Merkmale zu einem besseren Verständnis über das betrachtete Analyseproblem führen.

Um die Verdichtung der zur Verfügung stehenden Merkmale zu erreichen, bieten sich zwei Verfahren an. Ein intuitives Vorgehen beschreibt die **Feature Selection**, bei dem der Merkmalsraum sukzessive verkleinert wird, indem die „geeignetsten“ Merkmale ausgewählt werden, vgl. (Raschka, 2017, S. 140). Eine Auswahl von geeigneten Merkmalen kann über unterschiedliche Kriterien stattfinden. Beispielsweise können Merkmale entfernt werden, die eine geringe Varianz besitzen⁴⁴, oder hochgradig mit anderen Merkmalen korreliert sind⁴⁵. Für weitere Methoden der Feature Selection siehe Dokumentation Scikit-Learn⁴⁶.

⁴⁴ Es wird angestrebt, dass die Merkmale eine hohe Varianz aufweisen, damit diese die Varianz der Zielvariable möglichst genau abbilden können

⁴⁵ Der Hintergrund ist, dass die Hinzunahme eines stark abhängigen Merkmals die Varianz insgesamt nicht oder nur wenig erhöht.

⁴⁶ Dokumentation Scikit-Learn: https://scikit-learn.org/stable/modules/feature_selection.html

Eine Dimensionsreduktion kann ebenso eine sog. **Feature Extraction** durchgeführt werden, vgl. (Raschka, 2017, S. 159). Hierbei wird unter Beibehaltung aller Merkmale der Merkmalsraum mathematisch so transformiert, sodass eine geringere Dimensionalität erreicht wird. Um diese Art von Dimensionskomprimierung zu verstehen, soll ein gängiges Verfahren, die Hauptkomponentenanalyse (PCA, Principal Component Analysis) kurz vorgestellt werden. Ein Merkmalsraum mit n Merkmalen kann in ein Koordinatensystem mit n Dimensionen überführt werden. In diesem kann ein Objekt (beispielsweise eine Region) hinsichtlich der Merkmalsausprägungen (Anteil Arbeitslosigkeit, Anzahl Insolvenzverfahren, usw.) als Punkt dargestellt werden. Dabei sind auch Merkmale enthalten, welche hochgradig miteinander korrelieren. Durch das PCA Verfahren werden zwei Änderungen am Koordinatensystem vorgenommen, um die Abhängigkeiten der Merkmale aufzulösen. Dies wird durch das Zusammenfassen von Merkmalsdimensionen und Verschiebung der Achsen erreicht, sodass die verbleibenden Achsen in allen Richtungen eine hohe Varianz aufweisen. Die dadurch entstandenen Achsen bilden dann die sog. Hauptkomponenten, die nun nicht mehr als Merkmale interpretierbar sind. Die ursprüngliche Korrelation im Merkmalsraum ist durch die orthogonale Ausrichtung der Hauptkomponentenachsen (90 Grad zueinander) und in Folge der Verschiebung aufgehoben. Ein Nachteil dieser Methode ist, dass das Verständnis für die Daten und die Nachvollziehbarkeit bei der Durchführung der Analyse verloren geht. Daher wird im Rahmen dieser Arbeit nur die Feature Selection durchgeführt, da so vorab auch Eigenschaften und Muster in Daten erkannt werden können.

Das Resultat der Selection sowie Extraktion ist ähnlich – am Ende sollen Merkmalskandidaten vorliegen, sog. Feature Set, welche in die Analyse eingehen sollen.

Hier sei darauf hingewiesen, dass ein unterschiedliches Verständnis von Feature Extraction in der Literatur existiert. Neben der vorgestellten Definition wird unter dem Begriff auch eine Umformung der einzelnen Merkmale verstanden, die für Maschine Learning Algorithmen verarbeitbar ist (Guyon & Elisseeff, S. 2). Im Rahmen dieser Arbeit wird dies jedoch der Feature Transformation zugeordnet, da der Fokus auf die *Umgestaltung der jeweiligen Features* für die Analyse liegt und nicht mehr das Dimensionalitätsproblem betrachtet wird.

4.1.2. Feature Transformation

Viele Machine Learning Algorithmen haben Anforderungen an die Form der eingehenden Daten. Allgemein können die Verfahren nur mit numerischen Darstellungen arbeiten, die beispielsweise bei der Verwendung eines neuronalen Netzes, einen ähnlichen Wertebereich aufweisen sollten. Daher ist die Transformation der Features ein entscheidender Schritt im Verarbeitungsprozess. Im Folgenden werden gängige Transformationsstrategien vorgestellt, welche zu einem späteren Zeitpunkt angewendet werden.

Metrische Darstellung von nominalen oder ordinalen Merkmalen

In jeder Datensammlung gibt es Merkmale mit unterschiedlichen Datentypen. Da ein Lernalgorithmus mit den Merkmalen Berechnungen durchführt, müssen diese in numerische Wertebereiche überführt werden. Zu unterscheiden sind dabei nominale und ordinale Skalenniveaus. Nominale Merkmale, wie das Bundesland einer Region, werden mit einer Zeichenketten repräsentiert. Diese könnte durch eine Durchnummerierung von 1 bis 16 auf ein metrisches Niveau überführt werden. Dadurch würde jedoch indirekt eine Reihenfolge erzeugt werden, wie 7 (für Bayern) < 10 (für Saarland). Derweil diese Art der Darstellung bei ordinalen Merkmalen wie dem Besiedlungsschlüssel (städtisch > halbstädtisch > ländlich bzw. 3 > 2 > 1) folgerichtig ist, kann die Durchnummerierung (Indexierung) bei nominalen Skalen zu einem falschen Umgang in den Berechnungen eines Algorithmus führen. Dieses Problem kann auf zwei Arten gelöst werden. Zum einen können sog. binäre Dummy Merkmale erzeugt werden, die jede Ausprägung des nominalen oder ordinalen Merkmals in weitere Spalten auslagert. In der Dummy-Spalte wird dann die jeweilige Ausprägung mit einer „1“ angezeigt, derweil die restlichen Dummy-Spalten „0“ betragen. Beispielsweise würde das Bundesland über 16 Spalten dargestellt werden, wie nachfolgendem Ausschnitt verdeutlicht werden soll.



Filiale	Bundesland
X	Saarland
Y	Bayern
...	...

Filiale	Bundesland_Bayern	Bundesland_Saarland
X	0	1
Y	1	0
...

Abbildung 37 Überführung eines nominalen Merkmals in Dummy Variablen

Zum anderen kann das Merkmal in einen binären Vektor zu transformieren werden. Dabei wird jedes Bundesland über einen 16-stelligen Vektor repräsentiert, welcher an einer bestimmten Position eine „1“ enthält und sonst nur Nullen. Somit ist jeder Vektor eindeutig identifizierbar. Beispiel:

Filiale	Bundesland		Filiale	Vektor Bundesland
X	Saarland		X	(0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
Y	Bayern		Y	(0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0)
...

Abbildung 38 Überführung eines nominalen Merkmals in einen binären Vektor

Da diese Vektoren nicht mehr miteinander verglichen werden können⁴⁷, wäre das Reihenfolgenproblem überwunden. Auch wenn die Vektorrepräsentation dabei übersichtlicher erscheint, können diese im Gegensatz zu den Dummy-Spalten für einen Anwender nur schwer interpretiert werden.

Skalieren von Merkmalen

Einige Algorithmen können nicht optimal mit Merkmalen unterschiedlicher Skalen arbeiten. Beispielsweise kann das Merkmal „Anzahl Bevölkerung“ einer Region von 8 bis 3,5 Millionen erstrecken, dagegen ist der Wertebereich die Arbeitslosenquote auf einem Intervall von 0 bis maximal 1 abgebildet. Um alle Merkmale auf einen Wertebereich zu bringen, werden zwei Verfahren sehr häufig angewandt – Normalisierung und Standardisierung. Bei der *Normalisierung* von Merkmalen werden die Werte zwischen 0 und 1 skaliert. Bei der Berechnung fließen insbesondere Minimal- und Maximal-Wert (und) in folgender Form mit ein:

Dabei ist zu beachten, dass die Normalisierung sehr anfällig gegenüber Ausreißern ist. Wenn die maximale Bevölkerungsanzahl einer Region nicht 3,5 Millionen ist, sondern 35

⁴⁷ In einem Koordinatensystem zeigen die Vektoren zwar in unterschiedliche Richtungen, besitzen jedoch alle die Länge von 1.

Millionen, erhöht sich dadurch die Distanz zwischen Min und Max und somit wird jeder Wert durch eine noch größere Zahl geteilt.

Etwas robuster zeigt sich dagegen die *Standardisierung*. Dabei wird die ursprüngliche Verteilung des Merkmals auf die Standard-Normalverteilung abgebildet. Die Berechnungsvorschrift einer Standardisierung sieht vor, den zu transformierenden Wert um den Mittelwert zu minimieren (auch zentrieren genannt) und die Differenz durch die Standardabweichung des betrachteten Merkmals zu teilen.

Im Zuge Standardisierung wird das Merkmal einen bestimmten Wertebereich abgebildet – der Mittelwert des standardisierten Merkmals wird auf 0 transformiert und die Varianz auf 1⁴⁸

Bei der Standardisierung und Normalisierung handelt es sich um sogenannte lineare Transformationen, da die relativen Distanzen zwischen den Datenpunkten nicht geändert werden. Dagegen nehmen nicht-lineare Transformationen, wie das Logarithmieren der Ausgangswerte, direkten Einfluss auf die Lage der Datenpunkte. So kann beispielsweise der Abstand von statistischen Ausreißern zu den restlichen Datenpunkten verringert werden. Weitere Transformationsverfahren können von Scikit Learn entnommen werden⁴⁹, weiteren Python Bibliothek für Verarbeitungsprozesse rund um Machine Learning.

Kombination von Features

Die Feature Transformation kann auch über eine Kombination von vorhandenen Merkmalen durchgeführt werden. Beispiel: Es wird Gesamtumsatz für alle Filialen einer Region betrachtet. Soll die Anzahl der in der Region Filialen berücksichtigt werden, wird der Gesamtumsatz durch die Anzahl Filialen geteilt – somit erhält man den durchschnittlichen Umsatz einer Filiale in dieser Region. Die Kennzahl ist nicht nur verständlicher, sondern stellt eine Vergleichbarkeit zu anderen Regionen her. Ein weiteres Beispiel wäre der Pro-Kopf-Umsatz in einer Region

⁴⁸ Einige statistische Testverfahren erfordern eine Standardnormalverteilung. Liegt diese vor, können stochastische Verfahren über Wahrscheinlichkeiten angewendet werden.

⁴⁹ Siehe hierfür Dokumentation: <https://scikit-learn.org/stable/modules/preprocessing.html>

4.1.3. Maschine Learning

Im Laufe dieser Arbeit wurde bislang der Begriff „Analysemodell“ oder „Modell“ verwendet und wird in diesem Abschnitt detaillierter erläutert. Aus dem Anwendungsfall lässt sich die Zielgröße des Modells definieren (Output) – der mögliche Jahresumsatz einer Filiale innerhalb einer Region, welche bis zu diesem Zeitpunkt noch keinen Filialstandort besitzt. In das Modell gehen Filialumsätze sowie Standorteigenschaften bzgl. Bevölkerung und Infrastruktur aus dem Open Data Bereich ein (Input). Im Modell soll anhand der eingehenden Daten eine Umsatzschätzung über ein Machine Learning Verfahren (Verarbeitung) erfolgen.

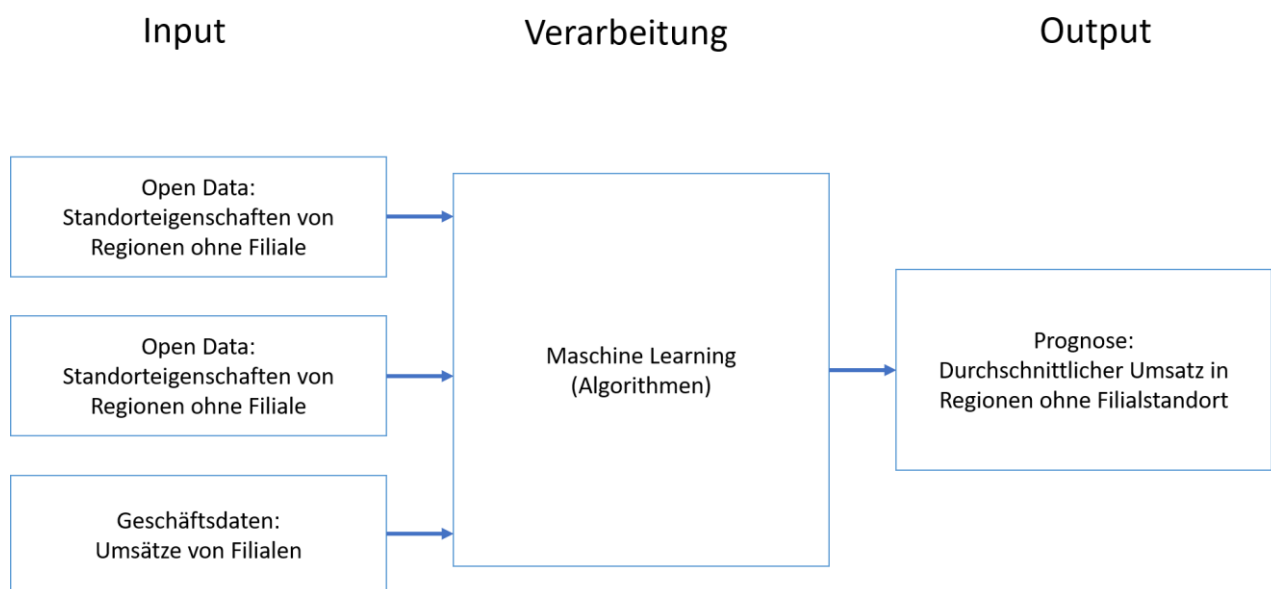


Abbildung 39 Aufbau des Analysemodells über die Ebenen Input, Verarbeitung und Output

Wie mithilfe des Machine Learning eine Prognose durchgeführt werden kann, wurde dabei nicht genauer erklärt. Um das generelle Vorgehen beim Machine Learning zu verstehen, wird das Konzept im Folgenden kurz erläutert.

Machine Learning Konzept

Nach (Géron, 2018, S. 4) ist Machine Learning die Wissenschaft, Computer so zu programmieren, dass sie anhand von Daten lernen. Machine Learning wird für Aufgaben eingesetzt, bei dem das betrachtete Problem für herkömmliche Verfahren zu komplex ist. Ein klassisches Beispiel ist ein E-Mail Spamfilter. Die Erkennung einer Spam Mail ist durch

die Vielfalt der Sprache eine schwere Aufgabe. Um hierfür kein detailliertes Regelwerk zur Klassifizierung von Mails manuell zu entwerfen, kann die Erkennung von Spam Mails durch das Machine Learning Verfahren „gelernt“ werden. Dabei wird eine Menge an Lernbeispielen benötigt, die in sog. Trainingsdaten zusammengefasst werden. Ein Machine Learning Algorithmus lernt dabei die Eigenschaften der Trainingsdatensätze, indem verfahrensinterne Parameter angepasst werden, vgl. (VanderPlas, 2018, S. 360). In dem betrachteten Anwendungsbeispiel soll der Zusammenhang zwischen Standorteigenschaften einer Region und dem Umsatz einer Filiale gelernt werden. Die Komplexität der Aufgabenstellung ergibt sich durch die Menge an unterschiedlichen Merkmalen je Region, die für die Analyse zur Verfügung stehen. Ein Machine Learning Algorithmus kann dabei automatisch erkennen, welche Standorteigenschaften den Umsatz gut vorhersagen und welche weniger relevant sind.

Es gibt unterschiedliche Arten von Machine Learning Verfahren, die je nach Problemstellung eingesetzt werden. Da im Rahmen des verwendeten Modells die Zielgröße bekannt ist, lässt sich das betrachtete Problem dem supervised learning (zu dt.: überwachten Lernen) zuordnen. Die Daten zum Anlernen des Algorithmus sind dabei mit einem sog. *Label* versehen – übertragen auf den Anwendungsfall, sind dies die Umsätze, die für die Regionen vorliegen.

Neben dem überwachten Lernen gibt es auch das unüberwachte, das sog. unsupervised learning. Verfahren aus dem unsupervised learning Bereich benötigen keine gelabelten Daten, da das Ziel der Analyse ist, Muster in den Daten zu erkennen. Ein typisches Beispiel ist das Gruppieren von Kunden in verschiedene Klassen über Clusteringmethoden – siehe hierzu (VanderPlas, 2018, S. 365).

Das überwachte Lernen teilt sich in Verfahren für Regressions- und Klassifikationsprobleme. Da der Umsatz als Zielgröße kontinuierlich ist, handelt es sich hierbei um eine Fragestellung aus dem Bereich der Regression. Dagegen besitzt die Zielgrößen eines Klassifikationsproblems diskrete Ausprägungen, wie z.B. Umsatzklassen.

Die beschriebene Einteilung von Machine Learning Verfahren wird auf nachfolgender Abbildung veranschaulicht.

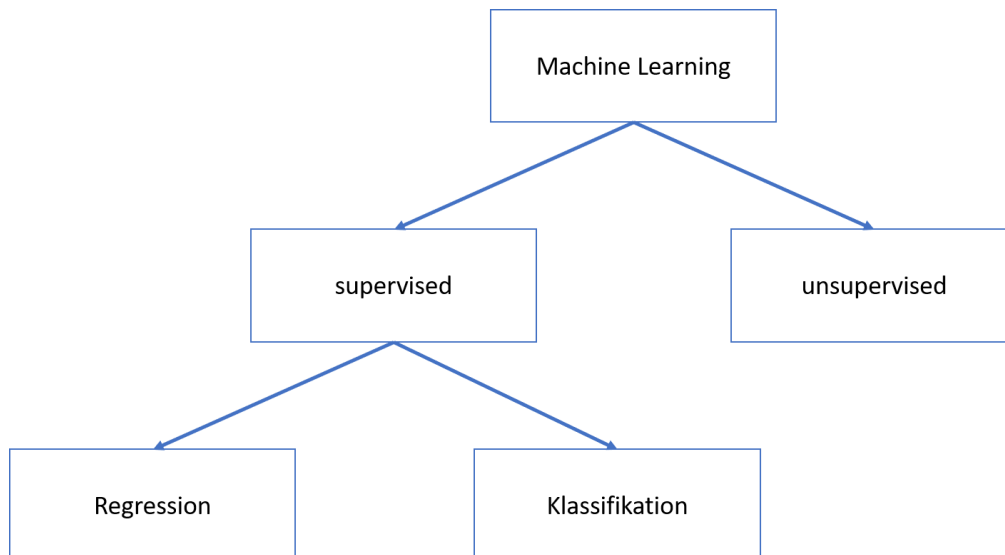


Abbildung 40 Einordnung von ML Algorithmen in Clustering, Klassifizierungs- und Regressionsverfahren

Vorbereitung

Im Machine Learning lernt das Verfahren auf Basis einer vorhandenen Datenmenge eine Zielgröße, wie vorliegend den Umsatz, so präzise wie möglich zu schätzen. Der eine Teil der Datenmenge sog. Trainingsdaten wird dabei verwendet, dass Verfahren anzulernen. Der andere Teil, die sog. Testdaten, werden benötigt um die Vorhersagequalität des angelernten Modells zu bewerten⁵⁰. Das Verhältnis von Trainings- und Testdaten beträgt häufig 80% zu 20% und kann bei Bedarf angepasst werden, vgl. (Géron, 2018, S. 30).

Die Vorbereitung umfasst auch die Feature Transformation sowie Features Selection/Extraction.

Modellauswahl

Für die Durchführung der Prognose stehen verschiedenste Algorithmen zur Auswahl. Je nach Eigenschaften der Daten kann können die dafür passende Verfahren eingegrenzt werden. In dieser Arbeit wird dies in Kapitel 4.2.3 durchgeführt. Das Ziel ist eine engere Auswahl der vielversprechendsten Verfahren – laut (Géron, 2018, S. 73) sollten dies zwei bis fünf sein. In dieser Arbeit werden lediglich zwei Verfahren näher betrachten, um sich genauer mit diesen auseinandersetzen zu können.

⁵⁰ Die Einteilung der Datenmenge wird hier nur in Trainings- und Testdaten vorgenommen. Es gibt auch Ansätze, die eine weitere Validierungsdatenmenge bilden, welcher im Grunde einen zweiten Testdatensatz darstellen. Dieser soll verwendet werden um interne Parameter des Modells zu optimieren (vgl. Geron, 2018, S. 31)

Modellbewertung

Die Qualität der Vorhersage wird über die Testdatenmenge validiert, indem die Differenz zwischen Vorhersage (*Prediction*) und tatsächlichem Wert (*Label*) bewertet wird. Dabei gibt es verschiedene Methoden diesen Fehler zu messen. Ist der Fehler gering, erzeugt das Verfahren präzise Vorhersagen. Die typischen Metriken zur Messung dieses Fehlers werden kurz vorgestellt. Dabei wird nur auf die Fehlermetriken zur Bewertung eines Regressionsverfahrens eingegangen.

Ein sehr verbreitetes Fehlermaß ist die **mittlere quadratische Abweichung (MSE)**. Dabei werden die Differenzen zwischen Schätzung und wahren Wert aufsummiert und gemittelt. Die Differenzen werden dabei quadriert, sodass große Abweichungen mit einem höheren Fehler bewertet werden, wie aus nachfolgender Formel ersichtlich ist

—

Der MSE stellt den Fehler dar, den das Modell im Mittel bei einer Vorhersage macht. Alternativ wird die Wurzel des MSE genommen und als **Wurzel der mittleren quadratischen Abweichung (RMSE)** interpretiert. Da im RMSE sowie MSE die Differenzen quadriert werden, sind diese empfindlich für extreme Abweichungen. Alternativ bietet sich das Maß des **mittleren absoluten Fehlers (MAE)** an. Im MAE geht der Betrag der Abweichung von Schätzung und wahren Wert ein, der wiederum über alle Datensätze gemittelt wird.

—

Der Wertebereich des (R)MSE und MAE ist abhängig von der Zielgröße. Im Anwendungsfall dieser Arbeit können Jahresumsätze von Filialen mehrere Milliarden betragen, sodass die vorgestellten Fehlermaße für einen Anwender nicht immer einfach zu deuten sind. Dagegen wird die Metrik des **R²** (auch **Bestimmtheitsmaß** genannt) auf einen Bereich von 0 bis 1 normiert. Das R² kann als Anteil der Varianz verstanden werden, die durch das Modell erklärt wird. Bei einem Wert von 0,9 bedeutet dies, dass 90% der

Streuung der Zielgröße durch die Streuung der eingehenden Variablen erklärt werden kann. Für entsprechende Formel und nähere Erklärung siehe (VanderPlas, 2018, S. 391).

Fehlermetriken für ein Klassifikationsproblem werden in dieser Arbeit nicht weiter betrachtet und in (Géron, 2018, S. 84) ausführlich beschrieben.

Die beschriebene Modellbewertung wird in Abbildung 41 zusammengefasst.

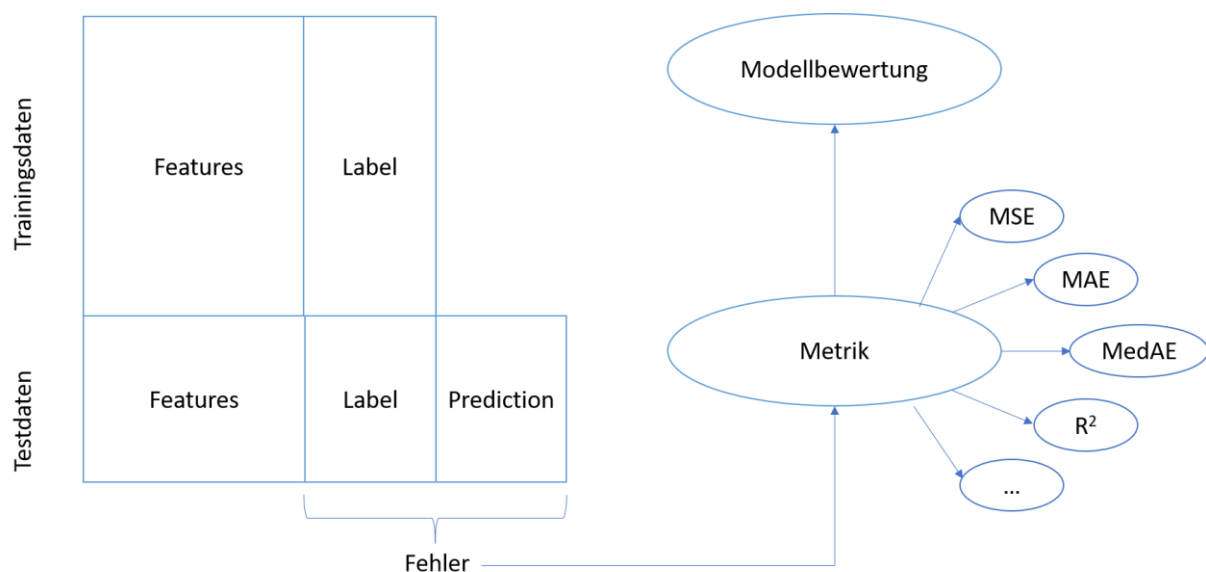


Abbildung 41 Prozess der Modellbewertung über die Fehlermetriken

Modelloptimierung

Liegen die Ergebnisse der Prognose vor, kann die Qualität der Vorhersage über die vorgestellten Metriken validiert werden. Häufig lässt sich die Vorhersage verbessern indem die Parameter des jeweiligen Algorithmus, sog. Hyperparameter verändert werden. Dabei müssen verschiedene Kombinationen von Hyperparametern ausprobiert und anschließend validiert werden. Zusätzlich kann die Aufteilung der Trainings- und Testdaten angepasst, oder das eingehende Feature Set verändert werden, um ggf. eine bessere Prognose zu erzielen.

Bei der Modellbewertung sollte berücksichtigt werden, dass ein geringer Fehlerwert lediglich bedeutet, dass das Verfahren die wahren Werte der Zielgröße in den Testdaten präzise schätzen kann. Somit können neu auftkommende Daten genau vorhergesagt

werden, sofern diese dieselben Eigenschaften haben, wie die Daten zum Anlernen des Algorithmus. Da dies nicht immer vorausgesetzt werden kann, ist ein gewisser Fehler erwünscht. In der Literatur wird in diesem Zusammenhang das *Varianz-Bias* Dilemma genannt. Es beschreibt die Suche nach dem Gleichgewicht zwischen zu präziser Schätzung und dem gewünschten Fehler. Ist die Schätzung zu präzise, wird vom *overfitting* gesprochen. Führt die Vorhersage zu hohen Fehlerwerten, handelt es sich hierbei um ein *underfitting*.– vgl. (VanderPlas, 2018, S. 390).

Prognose

Liegt das optimierte Verfahren vor, kann damit die Vorhersage für neu ankommenden Daten erfolgen.

Der dargestellte Prozess wird auf folgender Abbildung zusammengefasst.

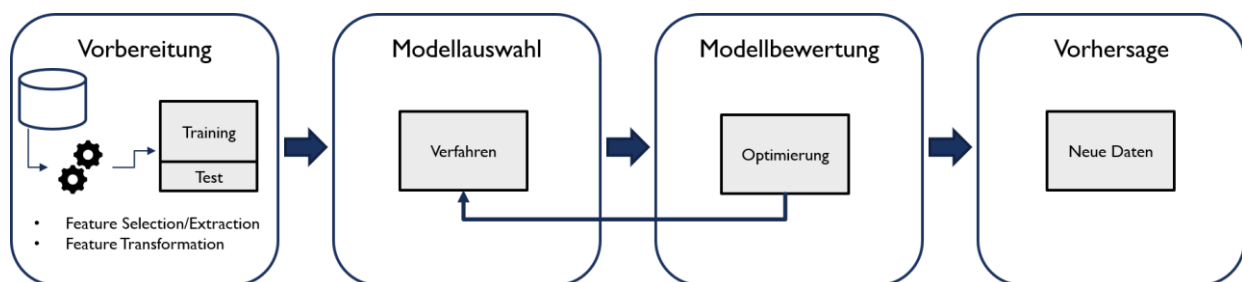


Abbildung 42 Vorgehensweise im Machine Learning

Um ein geeignetes Machine Learning Verfahren auszuwählen und die passenden Merkmale für die Vorhersage zu finden, ist ein Wissen über die Daten notwendig. Dafür müssen die Daten zunächst untersucht werden

4.2. Untersuchung der vorhandenen Daten

Da gerade für Open Data kein Fachwissen besteht, muss ein grundlegendes Verständnis für die Daten aufgebaut werden. Daher werden in diesem Kapitel die wichtigsten Eigenschaften der Daten herausgestellt. Zuerst werden die einzelnen Merkmale isoliert voneinander betrachtet und in der Feature Selection dann zusammenhängend analysiert.

Es ist zu beachten, dass längst nicht alle verwendeten Explorationswege in diesem Kapitel beschrieben werden können. Im Folgenden wird eine mögliche Vorgehensweise skizzierte, die als Beispiel dient.

4.2.1. Analyse der Merkmale

Für die Untersuchung sollen die Verteilungen der einzelnen Merkmale aus den Bereichen Bevölkerung und Infrastruktur betrachtet werden. Hierfür eignet sich Pandas, da leicht statistische Kennzahlen erstellt und visualisiert werden können. Im Folgenden soll stellvertretend die Einwohnerzahl der Regionen analysiert werden. Das Vorgehen kann auf die Untersuchung anderer Merkmale übertragen werden. Die gesamte Einwohnerzahl in einer Region (BEV_INSGESAMT) hat folgendes statistische Profil:

```
count    11222.00
mean      4956.24
std       10927.18
min         8.00
25%        627.00
50%       1650.50
75%       4891.75
max      522686.00
Name: BEV_INSGESAMT, dtype: float64
```

Abbildung 43 Statistische Profil des Merkmals Einwohneranzahl einer Region (mit Pandas erzeugt)

Die Kennzahlen (auch Lageparameter genannt) in diesem Profil, geben einen Eindruck über die Verteilung des Merkmals. Über das Minimum und Maximum ist zu erkennen, dass die betrachteten Regionen zwischen 8 und 522686 Einwohner haben. Außerdem ist anhand des 50% Quantils, bzw. dem Median zu entnehmen, dass 50% der betrachteten Regionen weniger als 1650 Einwohner haben. Entsprechend besitzt die andere Hälfte mehr als 1650 Einwohner. Werden die Werte der Quantile verglichen, ist zu erkennen, dass diese um den Faktor 3 ansteigen. Beispielsweise besitzen 75% Regionen weniger als 4891 Einwohner, was genau das 3-fache des 50% Quantils entspricht. Innerhalb des 25% bis 75% Quantils ist demnach ein gleichmäßiger bzw. linearer Verlauf zu erkennen. Wird dagegen die minimale und maximale Ausprägung des Merkmals betrachtet, so ist festzustellen, dass an den Enden des Wertebereiches kein linearer Verlauf zu beobachten ist. Im unteren Bereich entspricht der minimale Wert von 8 das 78-fache des 25% Quantils. Das Maximum von knapp 522.000 entspricht das 107-fache des 75% Quantils. Allein von der numerischen Betrachtung, kann davon ausgegangen werden, dass eine ungleichmäßige Verteilung des Merkmals besteht.

Der Mittelwert (**mean**) weicht stark vom Median ab, was ebenfalls darauf hindeutet, dass die Datenpunkte nicht gleichmäßig über den gesamten Wertebereich (min/max) aufgeteilt sind. Der Mittelwert wird durch die extremen Ausprägungen an den oberen und unteren Enden des Wertebereichs stark verzerrt und weicht daher vom Median ab. Durch die Extremwerte ergibt sich auch eine hohe Standardabweichung (**std**) für diese Verteilung. Alles deutet darauf hin, dass die Verteilung des Merkmals von Extremwerten sog. Ausreißern verzerrt wird. Rein visuell kann die Vermutung mit Hilfe einiger Plots überprüft werden, welche einen visuellen Eindruck über die Verteilung des Merkmals liefern sollen. Die Lageparameter aus dem statistischen Profil können über einen Boxplot dargestellt werden. Der Boxplots des Merkmals auf Abbildung 44 (links) ist derart gestaucht, sodass der Aufbau nicht mehr erkennbar ist. Zum Vergleich wurde der Boxplot des Durchschnittseinkommens je Einwohner gegenübergestellt (rechts).

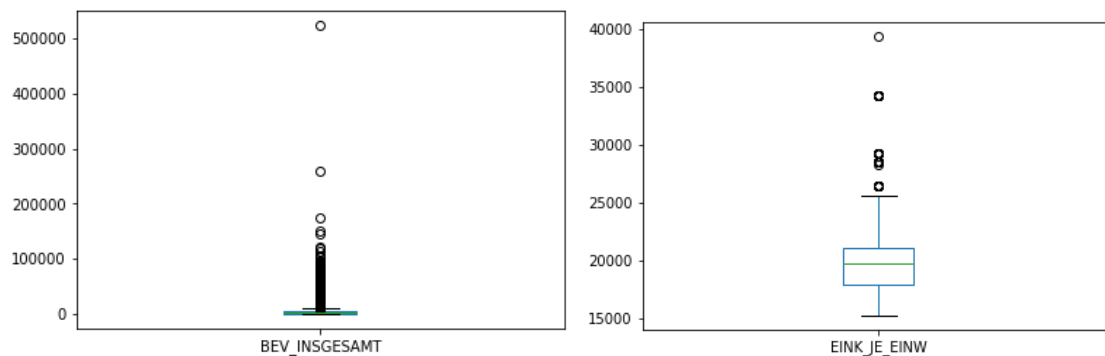


Abbildung 44 Visualisierung der Verteilung von Merkmalen über Boxplots.

Auch wenn der Großteil der Daten innerhalb der Box liegt, wird dieser durch Ausreißer (leere Kreise)⁵¹ nach oben hin visuell stark verzerrt.

Das dazugehörige Histogramm, welches die Häufigkeitsverteilung einzelner Wertebereiche darstellt, verstärkt diesen Eindruck. Der Großteil der Regionen liegt im unteren und damit bevölkerungsärmeren Segment.

⁵¹ Die Boxplot Visualisierung von Pandas kennzeichnet Datenpunkte als Ausreißer, sofern diese das 1,5-fache des Abstands zwischen dem 25% und 75% Quantils unter- oder überschreiten.

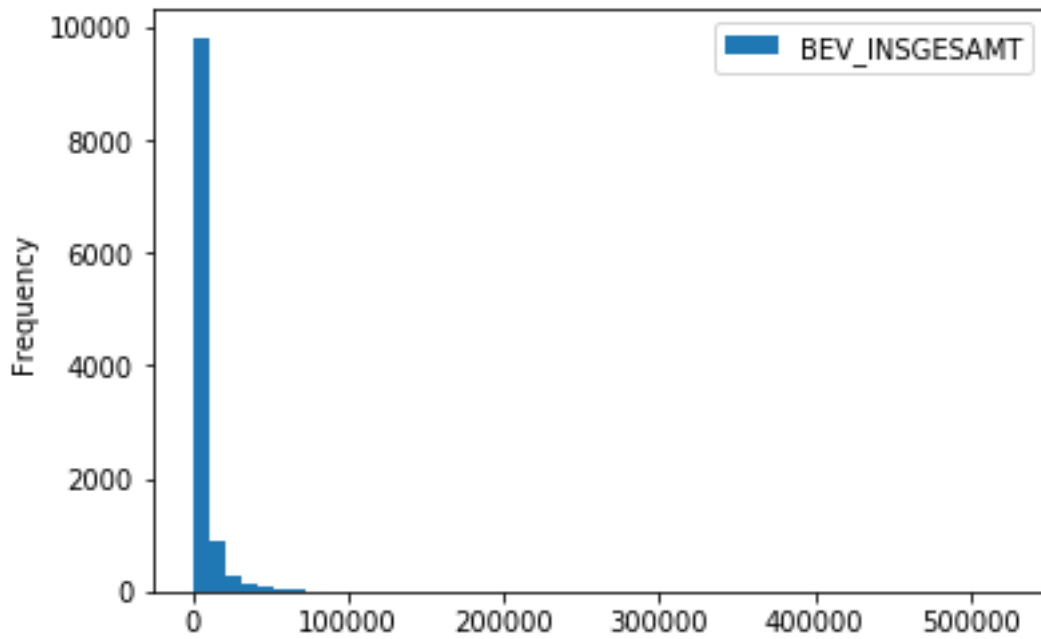


Abbildung 45

Histogramm

Ab einer Einwohnerzahl von etwa 50.000 sind die Häufigkeiten so gering, dass die Balken in diesen Bereichen bei dieser Skala kaum mehr zu erkennen sind. Werden die Regionen nach ihrer Einwohnerzahl aufsteigend sortiert, ergibt sich ein explosiver Anstieg ab etwa 50.000 Bewohnern, wie aus Abbildung 46 erkennbar ist.

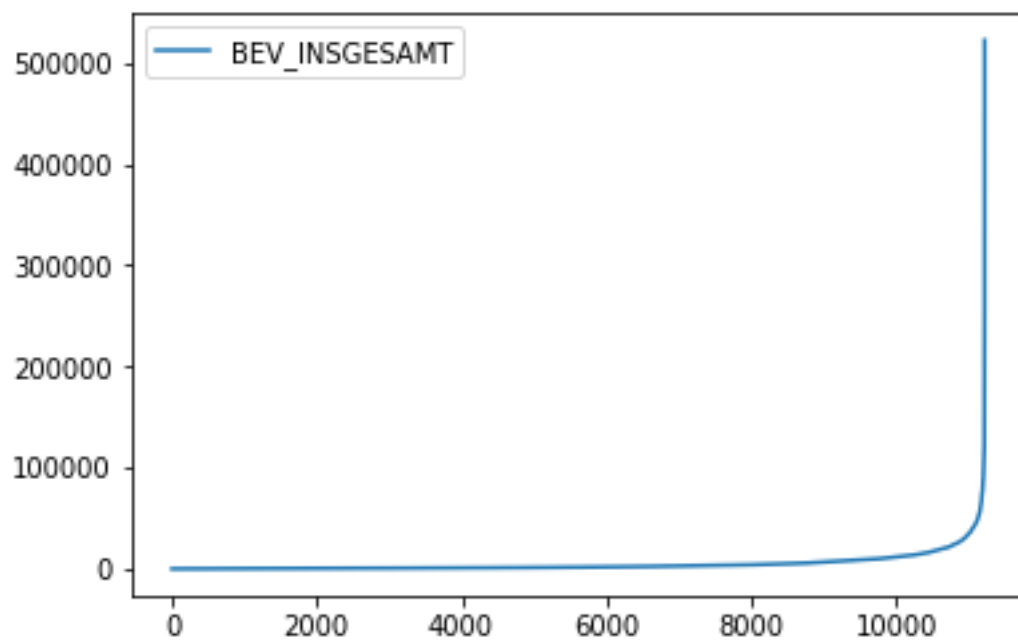


Abbildung 46

Verlaufsplot

Auf den folgenden Abbildungen sind jeweils die 10 einwohnerschwächsten (Abbildung 47) und 10 einwohnerstärksten Regionen (Abbildung 48) selektiert und ausgegeben worden.

	Name	BEV_INSGESAMT	DES
4157	Dierfeld	8	Gemeinde (gemeinschaftsangehörig)
294	Gröde	8	Gemeinde (gemeinschaftsangehörig)
1061	Wiedenborstel	9	Gemeinde (gemeinschaftsangehörig)
4367	Ammeldingen an der Our	10	Gemeinde (gemeinschaftsangehörig)
4378	Gemünd	10	Gemeinde (gemeinschaftsangehörig)
4296	Hisel	13	Gemeinde (gemeinschaftsangehörig)
4373	Burg	16	Gemeinde (gemeinschaftsangehörig)
263	Grothusenkoog	20	Gemeinde (gemeinschaftsangehörig)
4386	Keppeshausen	21	Gemeinde (gemeinschaftsangehörig)
4345	Etteldorf	22	Gemeinde (gemeinschaftsangehörig)

Abbildung 47 Unterer Extremwertbereich der 10 bevölkerungsärmsten Regionen

	Name	BEV_INSGESAMT	DES
8571	Berlin, Stadt	3460725	Stadt (kreisfrei)
1122	Hamburg, Freie und Hansestadt	1786448	Stadt (kreisfrei)
6438	München, Landeshauptstadt	1353186	Stadt (kreisfrei)
2237	Köln, Stadt	1007119	Stadt (kreisfrei)
2579	Frankfurt am Main, Stadt	679664	Stadt (kreisfrei)
5326	Stuttgart, Landeshauptstadt	606588	Stadt (kreisfrei)
2159	Düsseldorf, Stadt	588735	Stadt (kreisfrei)
2491	Dortmund, Stadt	580444	Stadt (kreisfrei)
2163	Essen, Stadt	574635	Stadt (kreisfrei)
2156	Bremen, Stadt	547340	Stadt (kreisfrei)

Abbildung 48 Oberer Extremwertbereich der 10 bevölkerungsreichsten Regionen

Die Extremwerte ergeben sich dadurch, dass gemäß der Verwaltungsgliederung Landeshauptstädte ebenfalls als kreisfreie Städte definiert sind. Dies erklärt die extreme Abweichung nach oben. Außerdem ist zu erkennen, dass allgemein mehrheitlich Städte im oberen Wertbereich vertreten sind, hingegen Gemeinden das untere Segment dominieren. Dies ist intuitiv gut nachvollziehbar, da Städte allgemein eine höhere Bevölkerungsanzahl besitzen als Gemeinden.

Für eine detaillierte Untersuchung stellt sich folgende Frage:

1. Lassen sich Städte und Gemeinden hinsichtlich ihrer Merkmale unterscheiden oder sogar gänzlich abgrenzen?

Um auszuschließen, dass die Regionen im Extremwertbereich die Untersuchung der Separierbarkeit von Gemeinden und Städte beeinflussen, sollten diese zunächst identifiziert und entfernt werden. In der Statistik sind Ausreißer unerwartete Daten, die für die zugrundeliegende Verteilung ungewöhnlich sind, vgl. (Brell, 2017, S. 30). Das Problem ist, dass diese Objekte (hier Regionen) nicht repräsentativ sind und Ergebnisse bei der Anwendung von statistischen Verfahren⁵² nicht mehr verlässlich sein könnten. Daher stellt sich zusätzlich die Frage:

2. Kann die Datenmenge insoweit eingegrenzt werden, sodass die Extremwerte nach unten sowie oben gedämpft werden können?

Es ist zunächst herauszufinden, welche Regionen für den vorliegenden Anwendungsfall der Umsatzschätzung relevant sind, um die Datenmenge zu reduzieren. Einerseits sind die Regionen von Bedeutung, die gerade so viele Einwohner besitzen, um ein wirtschaftliches Potenzial zu haben. Die einwohnerärmste Region, die mit einer Filiale besetzt ist, besitzt 374 Einwohner. Die maximale Einwohnerzahl einer Region, die als relevant gilt, definiert sich dagegen über die einwohnerstärkste Region, welche noch nicht mit einer Filiale besetzt ist. Regionen über diesen Wert sind für das Analyseziel nicht interessant, da ab diesem Wert alle Regionen mit Filialen besetzt sind⁵³. Entsprechend ist „Meerbusch“ mit 54318 Einwohnern die Region, welche den relevanten Menge nach oben hin abgrenzt.

Somit ist eine Region für den Anwendungsfall relevant, sofern diese mindestens 374 und maximal 54318 Einwohner besitzt. Hierbei verdichtet sich die Datenmenge von 11222 auf 9571 Regionen. Gleichzeitig reduziert sich die Anzahl von Regionen mit Filialstandorten von 1977 auf 1825.

⁵² Hierzu zählt beispielsweise das Neuronale Netz

⁵³ Außerdem ist es hilfreich sich vorzustellen, mit welchen Daten der Algorithmus trainiert werden soll. Dieser soll gerade die für das Analyseziel „relevante“ Menge trainieren und eben nicht den oberen Extremwertbereich, der ohnehin schon mit Filialen besetzt ist.

Hier ist zu berücksichtigen, dass die relevante Menge über der Einwohneranzahl identifiziert wurde⁵⁴. Durch Reduzierung der Datenmenge auf den relevanten Bereich, konnte eine Entzerrung einiger anderer Merkmale erreicht werden, wie die folgenden Boxplots zeigen sollen. Für jedes Merkmal ist links der Boxplot vor der Reduzierung dargestellt und rechts nach der Eingrenzung. Insbesondere sollte hier auf die verkleinerten Wertebereiche auf der Y-Achse geachtet werden.

Haushalte in einer Region insgesamt:

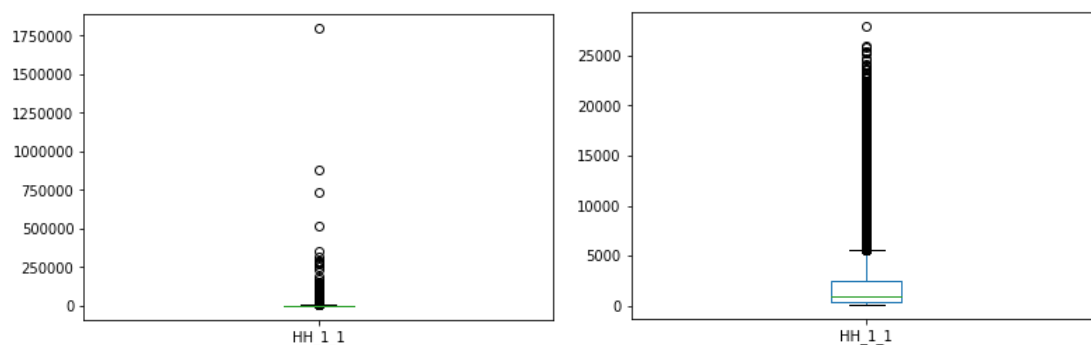


Abbildung 49 Boxplot für Haushalte einer Region vor Reduzierung des Datensatzes (links) und danach (rechts)

Anzahl geografischer Punkte aus dem OSM Datenbestand (Nodes):

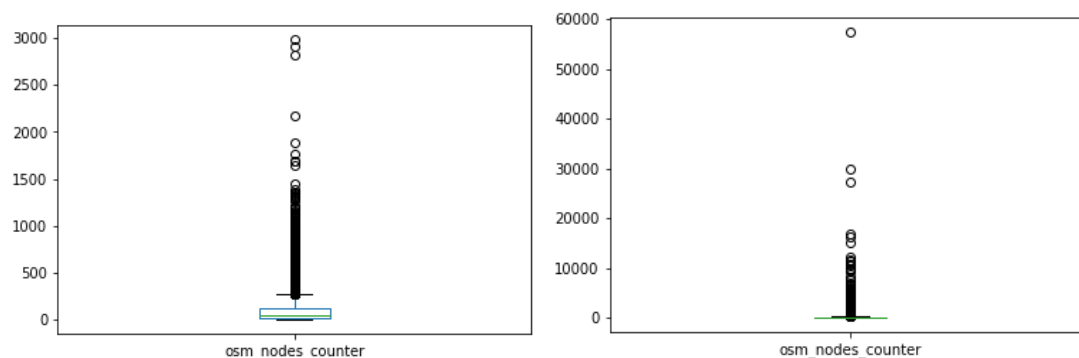


Abbildung 50 Boxplot für geografische Knotenpunkte in einer Region vor Reduzierung des Datensatzes (links) und danach (rechts)

⁵⁴ Es wurde nach der Bevölkerungsanzahl entschieden, da dies der Intuition nach eines der wichtigsten Merkmale ist, was die Wirtschaftlichkeit eines Standortes ausmacht.

Familien ohne Kinder in einer Region:

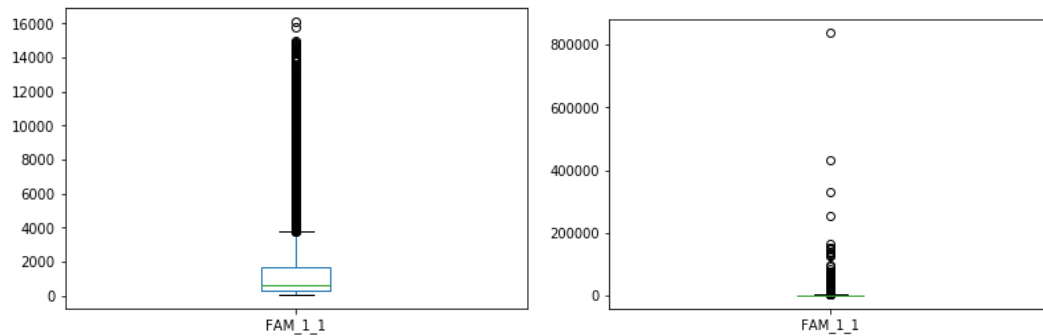


Abbildung 51 Boxplot für Familien ohne Kinder in einer Region vor Reduzierung des Datensatzes (links) und danach (rechts)

Auf der nun gebildeten Datenmenge kann eine Untersuchung der Frage erfolgen, ob sich Gemeinden und Städte grundsätzlich voneinander abgrenzen lassen. Auch hier wird die Einwohnerzahl als Indikator genommen. Die Gegenüberstellung beider statistischen Profile unterstützt weitestgehend die Hypothese, da zumindest die inneren Quantile (25% bis 75%) keine Überschneidungen in ihren Wertebereichen zeigen. Auf Abbildung 52 sind die statistischen Profile und die dazugehörigen Boxplots zusammengefasst.

Regionstyp	gem	stadt
count	7683.00	1888.00
mean	2863.58	14659.78
std	3425.18	11892.21
min	374.00	482.00
25%	811.00	5531.00
50%	1579.00	11025.00
75%	3399.50	20552.75
max	42213.00	54318.00

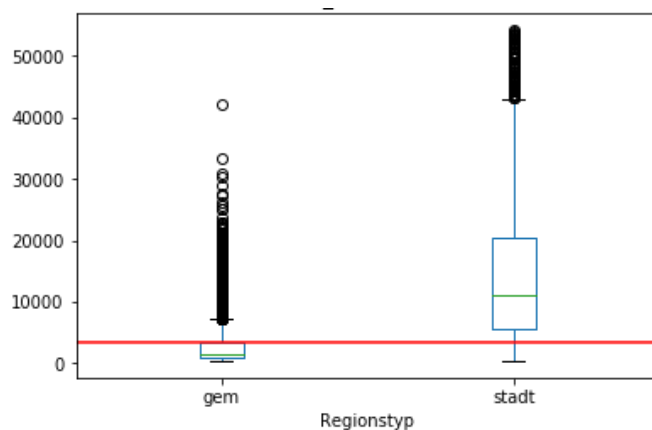


Abbildung 52 metrische und visuelle Darstellung der Lageparameter der Verteilung

Es ist zu erkennen, dass es weiterhin Ausreißer in beiden Gruppen gibt. Da die inneren Quantile bezüglich ihrer Wertebereiche überschneidungsfrei sind, kann die Hypothese formuliert werden, dass die Verteilung der Merkmale von Gemeinden und Städten unterschiedlichen Verteilungen unterliegen. Um diese Hypothese zu fundieren müsste dieses Muster in weiteren Merkmalen gefunden werden. Die Untersuchung weitere

Merkmale hat gezeigt, dass sich Gemeinden und Städte in Bevölkerungs- und Infrastrukturmerkmalen unterscheiden. Für die Analyse könnte es also sinnvoll sein, dass die Prognose für Gemeinden und Städte über zwei verschiedene Modelle durchgeführt wird. Zunächst sollte daher gemessen werden, wie viele Trainingsdaten für das Machine Learning eines „Gemeindemodells“ zur Verfügung stehen. Das Ergebnis ist, dass 604 der insgesamt 7683 Gemeinden, also etwa 8%, Umsatzdaten besitzen. Analog ergeben sich für ein „Stadtmodell“, dass für 1221 von 1888 Städte Umsätze vorliegen – das entspricht ca. 64%. Grafisch lassen sich die unterschiedlichen Verhältnisse wie folgt darstellen:

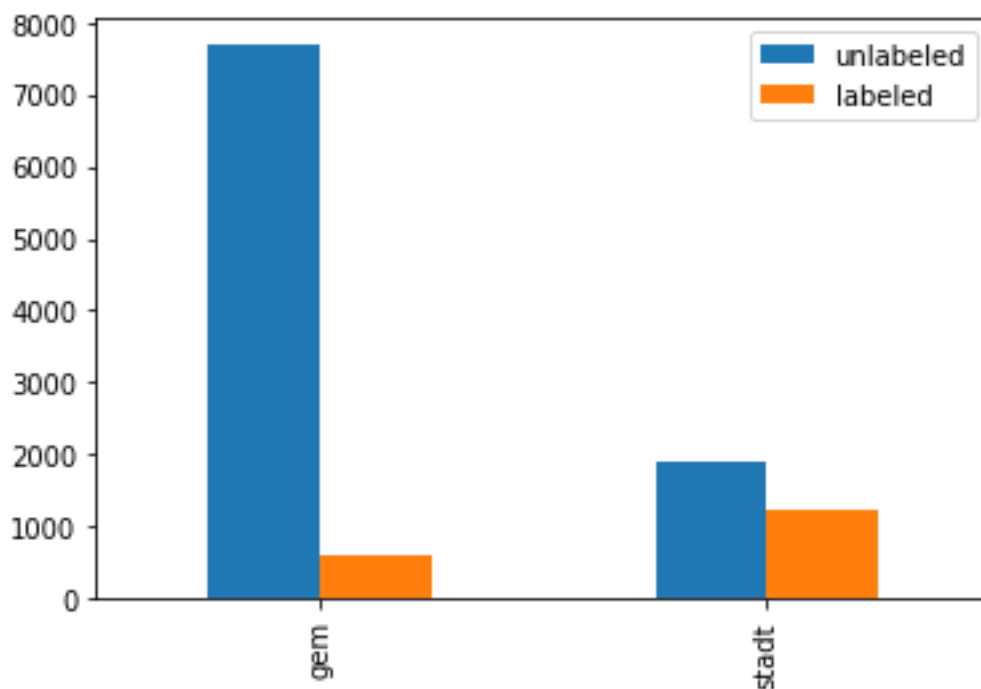


Abbildung 53 Missverhältnis zwischen Anzahl Gemeinden und Regionen mit und ohne Filialen

Bei der Verwendung eines Modells, welches Städte und Gemeinden nicht unterscheidet, würde sich jedoch ein ähnliches Missverhältnis ergeben.

Demnach würde man versuchen mit einem Modell, welches größtenteils mit Städten trainiert wird, Umsätze für Gemeinden zu schätzen.

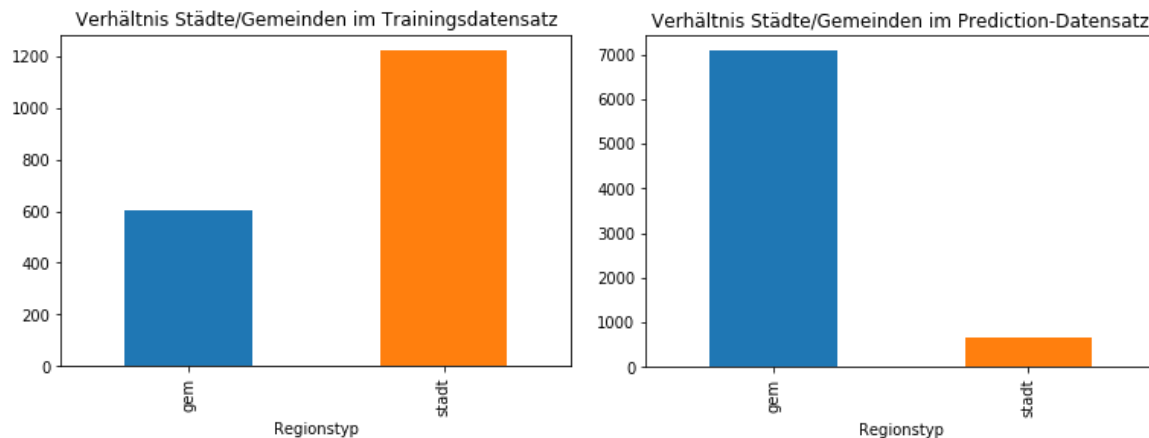


Abbildung 54 Verhältnis von Städten und Gemeinden im Trainingsdatensatz (links) und Vorhersagedatensatz (rechts)

Das Missverhältnis von Städten und Gemeinden in den beiden Datenmengen, führt also zu zwei Problemfeldern. Wenn *ein* Modell mit beiden Regionstypen trainiert werden würde, könnte dies zu einer Verzerrung führen, da Städte im Datensatz überrepräsentiert sind. Dagegen sollen die Umsätze zum Großteil von Gemeinden prognostiziert werden. Man könnte dies umgehen indem beispielsweise *zwei* Modelle trainiert werden – eines um Umsätze für Gemeinden zu schätzen und ein weiteres für Städte. Die Modelle könnten jedoch sehr unterschiedliche Schätzqualitäten besitzen, da der Umfang an Trainingsdaten für das Gemeindemodell viel kleiner ist. Auch hier würden die Modelle tendenziell bevölkerungsreiche Gemeinden bzw. Städte trainieren und Schätzung auf bevölkerungsärmere Regionen anwenden.

Die Verteilung des Umsatzmerkmals weist einen ähnlichen nicht-linearen Verlauf auf, der durch die Reduzierung des Datensatzes etwas verändert werden konnte. Auf der nachfolgenden Abbildung ist der drastische steile Anstieg vor Reduzierung abgeschwächt werden.

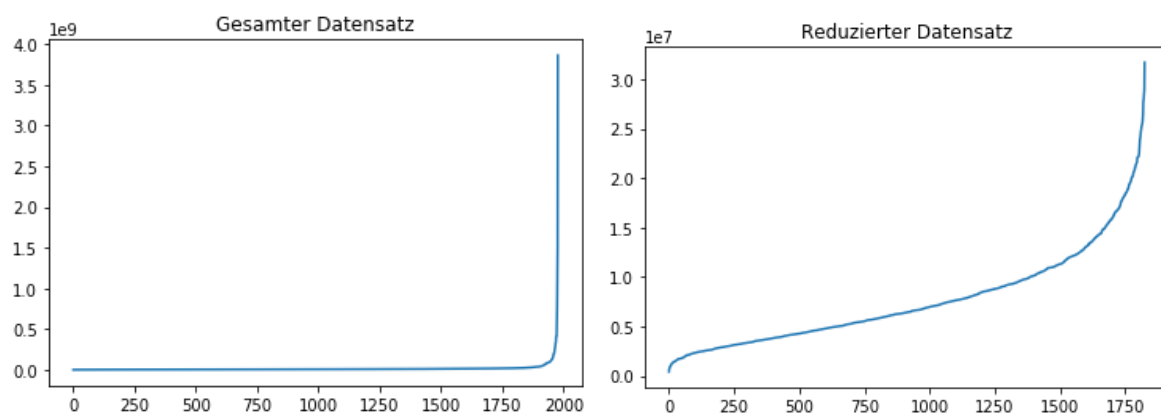


Abbildung 55 Visualisierung der Verteilung des Umsatzmerkmals vor der Reduzierung des Datensatzes (links) und danach (rechts) über Verlaufsplots

Zusammenfassend kann festgestellt werden, dass fast alle Merkmale der Städte und Gemeinden keine gleichmäßige Verteilung aufweisen. Es bestehen nicht-lineare Zusammenhänge, die zu komplex für herkömmliche statistische Verfahren, wie der linearen Regression sein könnten. Für eine Vorhersage des Umsatzes wird deswegen auf Machine Learning Algorithmen zurückgegriffen, da diese für diese Aufgabe besser geeignet sind.

Neben der hier skizzierten Analyse wurden weitere Untersuchungen durchgeführt, die an dieser Stelle nicht weiter beschrieben werden. Hieraus konnten folgende Erkenntnisse über die Daten abgeleitet werden:

Verschiedene Wertebereiche: **Relative Angaben** wie Arbeitslosenquote oder Siedlungsanteil in Prozent und **absolute Darstellungen** wie Bevölkerungsanzahl, Einkommen, etc.

Qualitative und quantitative Merkmale: **Kategoriale Merkmale** wie Besiedlungsschlüssel (städtisch/halbstädtisch/ländlich) sowie **metrische**.

Das Beispiel sollte verdeutlichen, dass es für die Exploration kein stringentes Vorgehen gibt, sondern sehr anwendungsbezogen erfolgt. Im nächsten Schritt werden Zusammenhänge zwischen den Merkmalen untersucht.

4.2.2. *Feature Selection*

Wie in Kapitel 4.1.1 beschrieben wurde, wäre es wenig sinnvoll alle vorhandenen Merkmale für eine Analyse zu verwenden. Einerseits kann durch die Reduzierung des Merkmalsraumes Berechnungszeit eingespart und andererseits Verzerrungseffekte (Fluch der Dimensionalität) gemindert werden. Das Bestreben ist, möglichst wenige und aussagekräftige Merkmale zu finden, welche ausreichen, um die Zielgröße des Modells zu erklären. Naheliegend wäre es nun, den Zusammenhang jedes Merkmals mit der Zielgröße zu messen und diejenigen Merkmale ins Feature Set aufzunehmen, welche die größte Abhängigkeit zueinander aufweisen. Um den Zusammenhang im univariaten⁵⁵ Fall zu messen, wird häufig eine Zusammenhangsanalyse über den Korrelationskoeffizienten (vorliegend nach Pearson) durchgeführt. Der Korrelationskoeffizient ist ein zwischen -1

⁵⁵ Zwischen zwei Variablen einer erklärenden (z.B. Bevölkerungsanzahl) und einer Zielvariablen (Umsatz)

und 1 normiertes Maß, welches Auskunft über die Stärke eines *linearen* Zusammenhanges zwischen zwei Variablen gibt, vgl. (Géron, 2018, S. 56). Bei einer vollständig negativen bzw. positiven (linearen) Beziehung wird der Wert -1 bzw. 1 angenommen, bei einem nicht messbaren Zusammenhang entsprechend der Wert 0.

So könnte für jedes Merkmal die Korrelation mit dem Umsatz gemessen und die Merkmale mit den stärksten Zusammenhängen ausgewählt werden. Das dieses Vorgehen nicht immer zielführend ist, soll durch die Visualisierung von verschiedenen Ausprägungen von Korrelationskoeffizienten veranschaulicht werden. Auf der Abbildung 56 sind zu den Korrelationskoeffizienten die jeweilige Verteilung der Daten dargestellt.

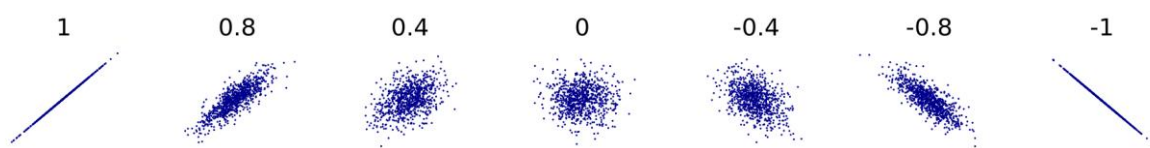


Abbildung 56 Zusammenhang von Korrelationskoeffizienten und Datenverteilung

Wie beschrieben, misst der Korrelationskoeffizienten (KK) die Stärke des *linearen* Zusammenhanges zweier Variablen – d.h. wenn x steigt, steigt/ sinkt y im Allgemeinen. Rein visuell interpretiert, formt sich die Punktwolke bei einem KK von 0 in Richtung 1 / -1 zu einer Geraden mit positiver/negativer Steigung. Dabei wird keine Aussage gemacht, wie der Zusammenhang konkret aussieht. So können starke lineare Zusammenhänge auch wie folgt aussehen:

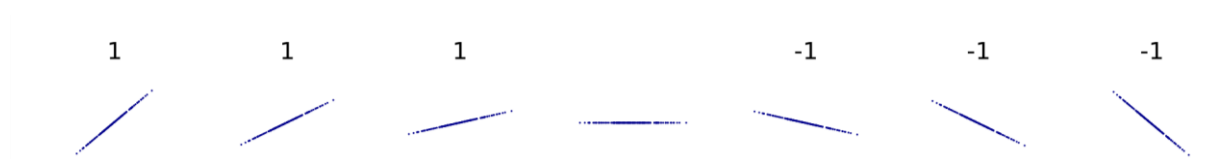


Abbildung 57 Steigung von Korrelationskoeffizienten von 1 und -1

Angenommen, es wird in KK von 1 gemessen, sagt dies nichts darüber aus, ob das eine Merkmale einen Einfluss auf ein anderes hat – siehe mittlere Darstellung auf Abbildung 57.

In der Realität können zudem Wirkungszusammenhänge durchaus komplexer sein, wie die nachfolgende Abbildung zeigen soll. Dadurch ist eine reine lineare Betrachtung über den KK nicht zielführend.

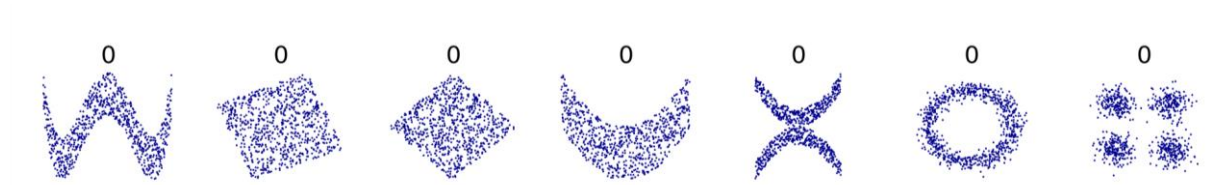


Abbildung 58 Komplexe Zusammenhänge eines KK von 0

Wird der Zusammenhang zwischen Durchschnittseinkommen je Einwohner und der Arbeitslosenanteil in einer Region betrachtet, ist zu erkennen, dass bei einem hohen Einkommensniveau die Arbeitslosenrate sehr niedrig ist und umgekehrt. Der Zusammenhang ist jedoch nicht linear, sondern zeigt einen eher konvexen Verlauf, wie auf folgender Abbildung zu sehen ist.

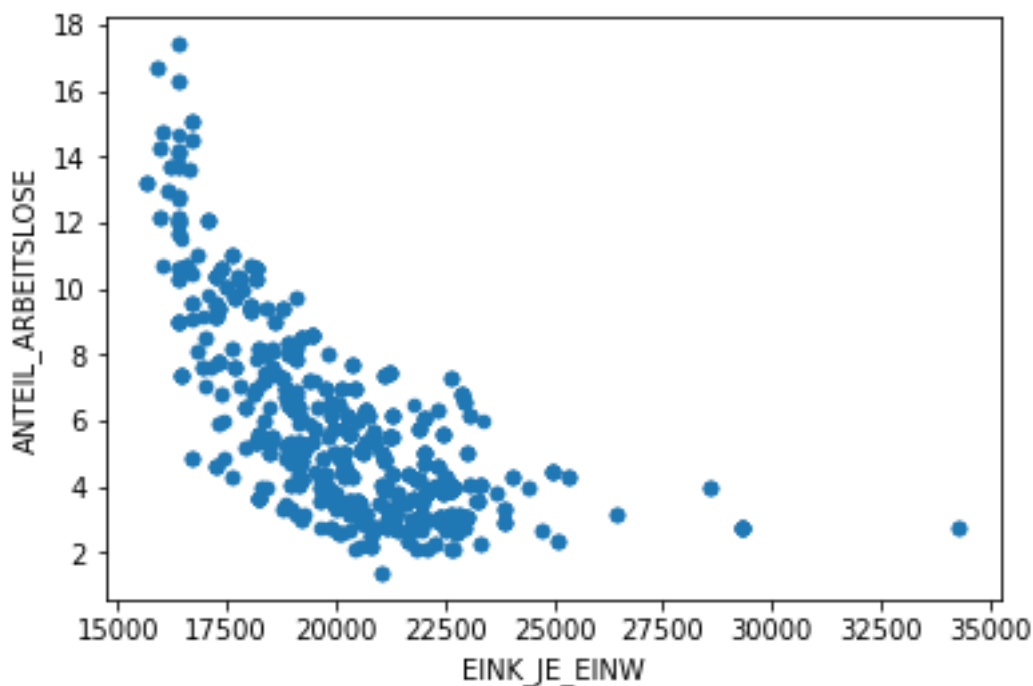


Abbildung 59 Scatterplot für den Zusammenhang Anteil Arbeitslose und Einkommen je Einwohner

Der Vorteil von Machine Learning Algorithmen ist, dass die gezeigten komplexen Zusammenhänge „gelernt“ werden können.

Die Aussage des Korrelationskoeffizienten kann jedoch für eine andere Problematik genutzt werden: Um Redundanzen zwischen den verwendeten Merkmalen zu erkennen. Redundante Merkmale erhöhen nicht nur die Komplexität des Modells, sondern können

den Erklärungsgehalt des Modells nicht erhöhen. So ist es sinnvoll diese Merkmale über den KK zu identifizieren und nachfolgend das Feature Set zu verdichten.

Hierfür wird über Pandas eine Korrelationsmatrix aufbereitet, welche für jede Kombination an Merkmalen den Koeffizienten mit entsprechender Farbskalierung zwischen grün (KK → 1) und rot (KK → -1) ausgibt. Der folgende Ausschnitt dieser Matrix zeigt, dass viele Merkmale hochgradig miteinander positiv korreliert sind.

	HH_1_1	HH_2_1	HH_3_1	HH_4_1	HH_4_4	AEWZ	DEM_1_1	DEM_1_3	DEM_2_1	DEM_2_3	DEM_2_4	DEM_2_6
HH_1_1	1	1	1	1	0.998463	0.995259	0.995259	0.996091	0.995259	0.996091	0.990986	0.989927
HH_2_1	1	1	1	1	0.998463	0.995259	0.995259	0.996091	0.995259	0.996091	0.990986	0.989927
HH_3_1	1	1	1	1	0.998463	0.995259	0.995259	0.996091	0.995259	0.996091	0.990986	0.989927
HH_4_1	1	1	1	1	0.998463	0.995259	0.995259	0.996091	0.995259	0.996091	0.990986	0.989927
HH_4_4	0.998463	0.998463	0.998463	0.998463	1	0.995679	0.995679	0.99611	0.995679	0.99611	0.994194	0.993453
AEWZ	0.995259	0.995259	0.995259	0.995259	0.995679	1	1	0.99982	1	0.99982	0.997147	0.995935
DEM_1_1	0.995259	0.995259	0.995259	0.995259	0.995679	1	1	0.99982	1	0.99982	0.997147	0.995936
DEM_1_3	0.996091	0.996091	0.996091	0.996091	0.99611	0.99982	0.99982	1	0.99982	1	0.996635	0.99589
DEM_2_1	0.995259	0.995259	0.995259	0.995259	0.995679	1	1	0.99982	1	0.99982	0.997147	0.995936
DEM_2_3	0.996091	0.996091	0.996091	0.996091	0.99611	0.99982	0.99982	1	0.99982	1	0.996635	0.99589
DEM_2_4	0.990986	0.990986	0.990986	0.990986	0.994194	0.997147	0.997147	0.996635	0.997147	0.996635	1	0.999298
DEM_2_6	0.989927	0.989927	0.989927	0.989927	0.993453	0.995935	0.995936	0.99589	0.995936	0.99589	0.999298	1
DEM_3_1	0.995259	0.995259	0.995259	0.995259	0.995679	1	1	0.99982	1	0.99982	0.997147	0.995936
DEM_3_3	0.996091	0.996091	0.996091	0.996091	0.99611	0.99982	0.99982	1	0.99982	1	0.996635	0.99589
DEM_3_13	0.986719	0.986719	0.986719	0.986719	0.991356	0.993751	0.993751	0.993041	0.993751	0.993041	0.995714	0.995229
DEM_3_15	0.982935	0.982935	0.982935	0.982935	0.987773	0.9919	0.9919	0.991432	0.9919	0.991432	0.993398	0.993904
DEM_4_1	0.995259	0.995259	0.995259	0.995259	0.995679	1	1	0.99982	1	0.99982	0.997147	0.995936
DEM_4_3	0.996091	0.996091	0.996091	0.996091	0.99611	0.99982	0.99982	1	0.99982	1	0.996635	0.99589

Abbildung 60 Ausschnitt über die mit Pandas erzeugte Korrelationsmatrix

Hierbei bestätigt sich beispielsweise die lineare Beziehung zwischen der Bevölkerungsanzahl und der Anzahl an Haushalten mit Paaren und Kindern in einer Region (Zelle BEV_INSGESAMT und HH_1_4)⁵⁶.

⁵⁶ Normalerweise wird hierbei die Signifikanz des KK mit einer Teststatistik überprüft. Voraussetzung für einen Signifikanztest ist jedoch, dass die Variablen normalverteilt sind. Da dies vorliegend nicht angenommen werden kann, sind solche Tests nicht sinnvoll.

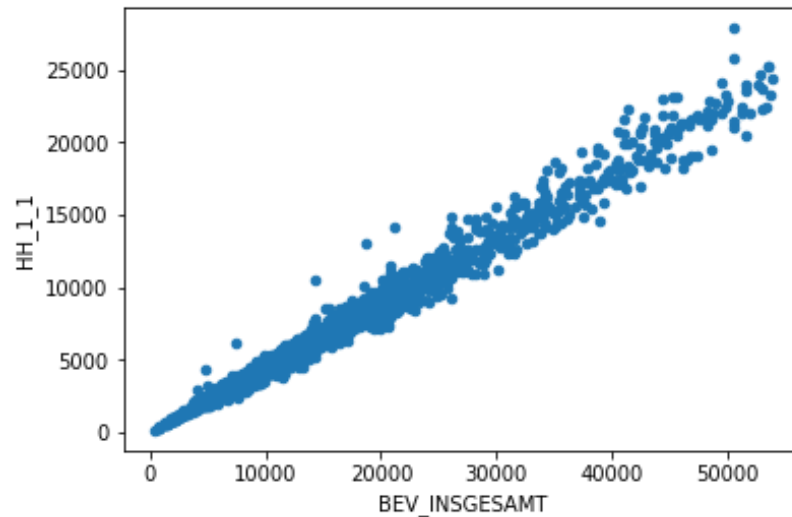


Abbildung 61 Zusammenhang von Haushalten und Bevölkerungsanzahl in einer Region über einen Scatterplot

Generell weist die Bevölkerungsanzahl quasi-lineare Zusammenhänge mit vielen betrachteten Merkmalen auf. Rein intuitiv ist dies nachzuvollziehen, da mit einem wachsenden Bevölkerungsbestand nicht nur demografische Attribute skalieren, sondern auch Infrastrukturmerkmale, wie Gesamtzahl der Wohnungen (WHG_1_1):

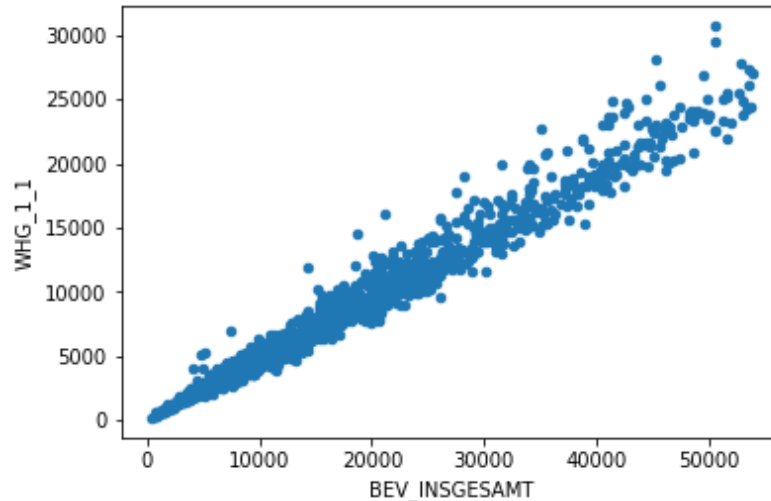


Abbildung 62 Zusammenhang von Wohnungen und Bevölkerungsanzahl in einer Region über einen Scatterplot

Generell konnte gezeigt werden, dass 177 von 257 Merkmalen stark mit der Bevölkerungsanzahl korrelieren – mit Korrelationskoeffizient zwischen 0,9 und 1 bzw. -0,9 und -1. Ein „Herausrechnen“ dieser statistischen Abhängigkeit hat auch nicht über weitere eine Feature Transformation funktioniert, indem z.B. alle abhängigen Merkmale durch die Bevölkerungsanzahl geteilt werden. Wie in der Korrelationsmatrix ersichtlich ist,

korrelieren auch andere Merkmale stark miteinander, wie z.B. die Anzahl der Wohnungen und Haushalte. Die daraus resultierenden Verhältniszahlen wie Wohnungen oder Haushalte pro Kopf würden daher weiterhin stark korrelieren.

Das Merkmal der Bevölkerungsanzahl wird daher stellvertretend für andere Merkmale ins Feature Set aufgenommen.

Das Durchschnittseinkommen je Einwohner zeigt dagegen kaum eine signifikante Korrelation mit anderen Merkmalen. Der stärkste Zusammenhang besteht mit Merkmalen aus dem Bereich Arbeitslosigkeit. Es kann ein negativer Zusammenhang mit der Arbeitslosenquote (KK = -0,74) und interessanterweise ein positiver mit dem Anteil arbeitsloser Ausländer (KK = 0,72), gemessen werden. Dieser Zusammenhang wird mit den nachfolgenden Scatterplots visuell verdeutlicht:

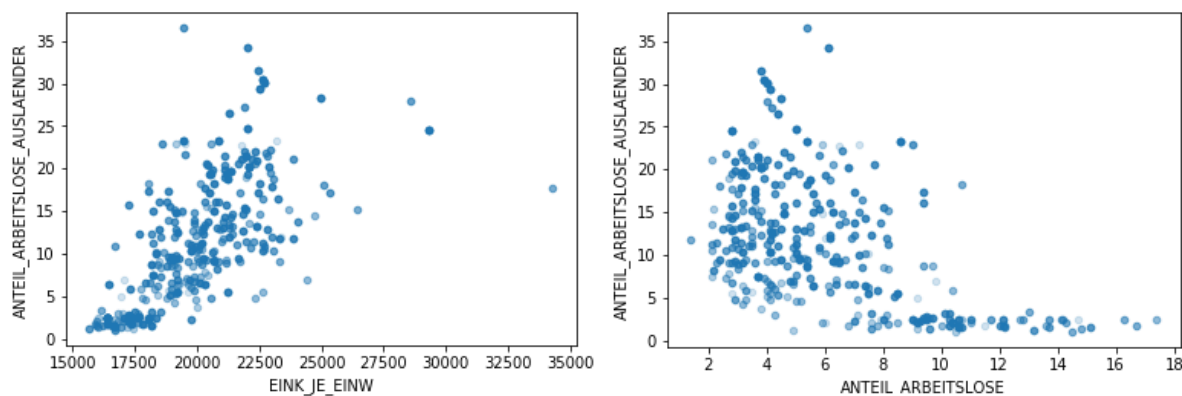


Abbildung 63 Zusammenhänge von Anteil arbeitslose Ausländer und Einkommen (links) sowie arbeitslose Ausländer und Anteil Arbeitslose (rechts) über Scatterplots

Demnach steigt der Anteil arbeitsloser Ausländer, je höher das Einkommensniveau ist. Je höher jedoch das Einkommen ist, desto niedriger ist die allgemeine Arbeitslosenquote.⁵⁷ Im Kreisschluss kann gezeigt werden, dass bei einer geringen Arbeitslosenquote der Anteil der arbeitslosen Ausländer steigt (und umgekehrt).

Durch die geringe Abhängigkeit zu anderen Merkmalen kann das Einkommensniveau, Arbeitslosigkeit, Anteil Siedlungsfläche und die Gesamtfläche einer Region ins vorläufige Feature Set aufgenommen werden.

⁵⁷ Hierbei sei auf den Unterschied zwischen Kausalität (echter Wirkungszusammenhang) und Korrelation (kalkulierter statistischer Zusammenhang) hingewiesen

Eine weitere Einschränkung ist, dass über den KK nur Zusammenhänge von einem intervallskalierende Merkmalspaar gemessen werden können. Für kategoriale Merkmale existieren andere Zusammenhangsmaße.

Um die Auswahl von stellvertretenden Merkmalen nicht zufällig vorzunehmen, wird eine weitere Methode aus der Feature Selection angewandt. Wie in Kapitel 4.1.1 bereits angeklungen, gibt es Verfahren, die den Einfluss der vorhandenen Merkmale zur Zielgröße messen und über eine Kennzahl bewerten. Dieses Scoring kann mit „*SelectKBest*“⁵⁸ durchgeführt werden. Über diese Methode werden Merkmale anhand einer Scoringfunktion bewertet und die *k* besten ausgewählt. Da es sich vorliegend um eine kontinuierliche Zielgröße handelt, wird die „*mutual information*“ Metrik verwendet, um die Abhängigkeit der vorhandenen Merkmale auf die Zielgröße zu bewerten. Die Metrik beruht auf nichtparametrischen Methoden, die auf der Entropie-Schätzung basieren. Je höher diese Abhängigkeit, desto höher ist der Scoringwert – entsprechend wird bei keiner Abhängigkeit ein Wert nahe 0 angenommen. Die Durchführung des SelectKBest Verfahrens wird mit der Python Bibliothek Scikit-learn umgesetzt. Hierbei haben folgende Merkmale die höchsten Werte erreicht.

osm_nodes_counter	0.85
WHG_4_2	0.72
GEB_4_2	0.68
GEB_6_4	0.65
WHG_7_5	0.64
WHG_5_5	0.59
WHG_1_2	0.58
WHG_6_1	0.58
WHG_9_1	0.58
WHG_3_1	0.58
WHG_5_1	0.58
WHG_7_1	0.58
WHG_4_1	0.58
WHG_8_1	0.58
WHG_1_3	0.58
WHG_1_1	0.58
HH_4_1	0.58
HH_2_1	0.58
HH_3_1	0.58
HH_1_1	0.58
DEM_3_29	0.57
....	...

Abbildung 64 Merkmale mit den höchsten Scoringwerte nach dem Mutual Information Verfahren

⁵⁸ Für nähere Information siehe folgenden Artikel: <https://towardsdatascience.com/feature-selection-with-pandas-e3690ad8504b>

Hierbei ist zu beachten, dass die gelisteten Merkmale weiterhin hochgradig miteinander korreliert sind und daher ähnlich hohe Scores erreicht wurden. Die SelectKBest Methode soll hier die „besten“ stellvertretenen Merkmale aufzeigen, welche eine hohe Abhängigkeit zur Zielgröße aufweisen. Einige dieser Merkmale wurden daher ins Feature Set aufgenommen, welches nach den beschriebenen Verfahren wie folgt aussieht:

Tabelle 11 Feature Set für die Analyse

Merkmalsbezeichnung	Bedeutung
DEM_3_29	Anzahl Personen älter als 80 Jahre in einer Region
GEB_6_4	Anzahl Gebäude mit 3 bis 6 Wohnungen in einer Region
DEM_2_4	Anzahl ledige Personen in einer Region
HH_4_4	Anzahl Haushalte ohne Senioren in einer Region
WHG_4_2	Anzahl Wohnungsgemeinschaften in einer Region
osm_nodes_counter	Anzahl infrastruktureller Punkte (Bildungseinrichtung, Einkaufsmöglichkeiten, öffentliche Gebäude, etc.) innerhalb einer Region
BEV_INSGESAMT	Gesamtanzahl der Bevölkerung in einer Region
BEV_JE_QM	Bevölkerung je Quadratkilometer in einer Region
FLAECHE_QM	Gesamtfläche in Quadratkilometer in einer Region
ANTEIL_ARBEITSLOSE	Anteil der Arbeitslosigkeit insgesamt in einer Region
ANTEIL_ARBEITSLOSE_ ALTER_15_24	Anteil der Arbeitslosigkeit der 15 bis 24-jährigen in einer Region
ANTEIL_SIEDLUNG_VERKEHR	Anteil der Siedlungsfläche in einer Region
EINK_JE_EINW	Durchschnittliches Jahreseinkommen pro Kopf in einer Region
WHG_6_5	Anzahl der leerstehenden Wohnungen in einer Region
BESIEDLUNGSSCHLUESSEL	Verstädterungsgrad einer Region (städtisch, halbstädtisch, ländlich)
REGIONSTYP ⁵⁹	Regionstyp (Gemeinde oder Stadt)

⁵⁹ In der Datenexplorationsphase konnte festgestellt werden, dass sich Städte und Gemeinden hinsichtlich ihrer Merkmalsausprägung unterscheiden. Daher wird probeweise das Merkmal in ein Feature überführt und in das Feature Set aufgenommen.

4.2.3. Zusammenfassung und Auswahl der Verfahren

Die Untersuchung der Daten und die Feature Selection hat Erkenntnisse geliefert, die im Folgenden kurz zusammengefasst werden.

Die Untersuchung der Merkmale hat gezeigt, dass diese keine gleichmäßige Verteilung besitzen. Hierfür sind unter anderem statistische Ausreißer verantwortlich. Grund hierfür ist die Vermischung von Städten und Gemeinden. Dabei sind Städte tendenziell bevölkerungsreicher und Gemeinden bevölkerungsärmer, wodurch auch infrastrukturelle Eigenschaften beeinflusst werden. Eine Verdichtung der Datenmenge auf Regionen, die für den Anwendungsfall relevant sind, konnte die Ausreißer Problematik etwas mindern. Dabei wurden beispielsweise Städte wie Berlin entfernt, da diese nicht mit Regionen vergleichbar sind, für die eine Umsatzprognose erfolgen soll.

Da sich Gemeinden und Städte in ihren Eigenschaften voneinander abgrenzen lassen, könnten regionstypabhängige Modelle zu besseren Vorhersagen führen. Die zu Verfügung stehende Datenmenge für das jeweilige Modell ist dabei sehr unterschiedlich. Daher könnten unterschiedliche Prognosequalitäten erwartet werden. Dies soll jedoch überprüft werden, sodass drei Prognosemodelle getestet werden:

M_Gesamt: Modell mit Städten und Gemeinden

M_Stadt: Modell für Städte

M_Gemeinde: Modell für Gemeinden

Der Datenbestand umfasst viele Merkmale zu einer Region. Der Großteil der Merkmale besitzen Werte mit absoluten Angaben (z.B. Einkommen, Bevölkerungsanzahl, etc.). Vereinzelt sind Merkmale mit relativen Angaben vertreten (alle Angaben zur Arbeitslosigkeit). Neben den kontinuierlichen existieren auch kategoriale Merkmale, wie der Regionstyp oder Besiedlungsschlüssel. Außerdem liegen für bestimmte Merkmale fehlende Werte, die auch nach der Behandlung in Kapitel 3.8 bestehen. Die hohe Abhängigkeit der eingehenden Features konnten nur teilweise über die Feature Selektion aufgelöst werden. Es existieren weiterhin Korrelationen der Merkmale untereinander.

Nach (Géron, 2018, S. 177) ist der **RandomForest** eines der mächtigsten Machine Learning Verfahren. Die Stärke des Verfahrens liegt vor allem in der Verständlichkeit des zugrundeliegenden Konzepts. Dabei besitzt das Verfahren weitere Eigenschaften, vgl. (Géron, 2018, S. 169):

- Robust gegen Ausreißer ist

- Geringe Vorbereitung der Daten, da das Verfahren mit Merkmalen verschiedener Skalenniveaus umgehen kann

- Auch Merkmale mit fehlenden Werten können für das Verfahren verwendet werden, da diese über ein spezielles Vorgehen ersetzt werden

Da die Eigenschaften der vorliegenden Daten beinahe deckungsgleich mit denen des Random Forests sind, wird dieses für die Umsatzprognose verwendet.

Ein weiteres prominentes Verfahren ist das **neuronale Netz**, welches nach (Géron, 2018, S. 253) ideal für große und sehr komplexe Probleme einsetzbar ist. Neben der Muster- und Spracherkennung können diese auch für Klassifikations- und Regressionsprobleme verwendet werden. Die ungleichmäßige Verteilung der Merkmale und die nicht-linearen Zusammenhänge können damit bei der Umsatzprognose berücksichtigt werden. Im Gegensatz zum Random Forest, erfordert die Verwendung des neuronalen Netzes eine umfangreichere Vorverarbeitung der Daten. Dabei müssen folgende Schritte durchgeführt werden, vgl. (Domingos, 2017).

- Dekorrelierung der Inputvariablen (wurde teilweise durch die Feature Selection erreicht).

- Normalisierung der Inputvariablen auf einen Wertebereich von 0 bis 1.

- Datensätze mit fehlenden Werten entfernen.

Die gewählten Verfahren werden in den nachfolgenden Kapiteln näher betrachtet und daraufhin für die Prognose des Umsatzes für Regionen ohne Filialstandort verwendet.

4.3. *Random Forest mit Spark ML*

Bevor das Verfahren angewendet wird, soll zunächst das grundlegende Konzept erläutert werden. Die Implementierung des Verfahrens wird dann mit Spark ML erfolgen und beschrieben. Danach erfolgt die Modellbewertung über die vorgestellten Fehlermetriken.

4.3.1. *Konzept*

Der Random Forest basiert auf dem Entscheidungsbaumverfahren, bei dem mehrere Merkmalen über sog. Knoten in einer Baumstruktur angeordnet werden, vgl. (VanderPlas, 2018, S. 448). Die Struktur des Entscheidungsbaumes kann entlang der Knoten durchlaufen werden. Für jeden Datensatz wird anhand der Merkmalsausprägungen entschieden, welcher Pfad im Entscheidungsbaum genommen wird. Je nach Problemstellung wird am Ende des Pfades eine Entscheidung über ein Klassifikation- oder Regressionswert getroffen.

Im RandomForest Verfahren erfolgt eine Entscheidung über mehrere unterschiedliche Entscheidungsbäume, bei denen die einzelnen Entscheidungen der Bäume aggregiert werden⁶⁰. Bei einem Klassifikationsproblem handelt es sich dabei um eine Mehrheitsentscheidung. Im Rahmen eines Regressionsproblems werden die Prognosewerte aller Bäume gemittelt. Die Aggregation der Entscheidungen wird auf Abbildung 65 schematisch dargestellt.

⁶⁰ Dieses Vorgehen wird auch *Ensembling* genannt.

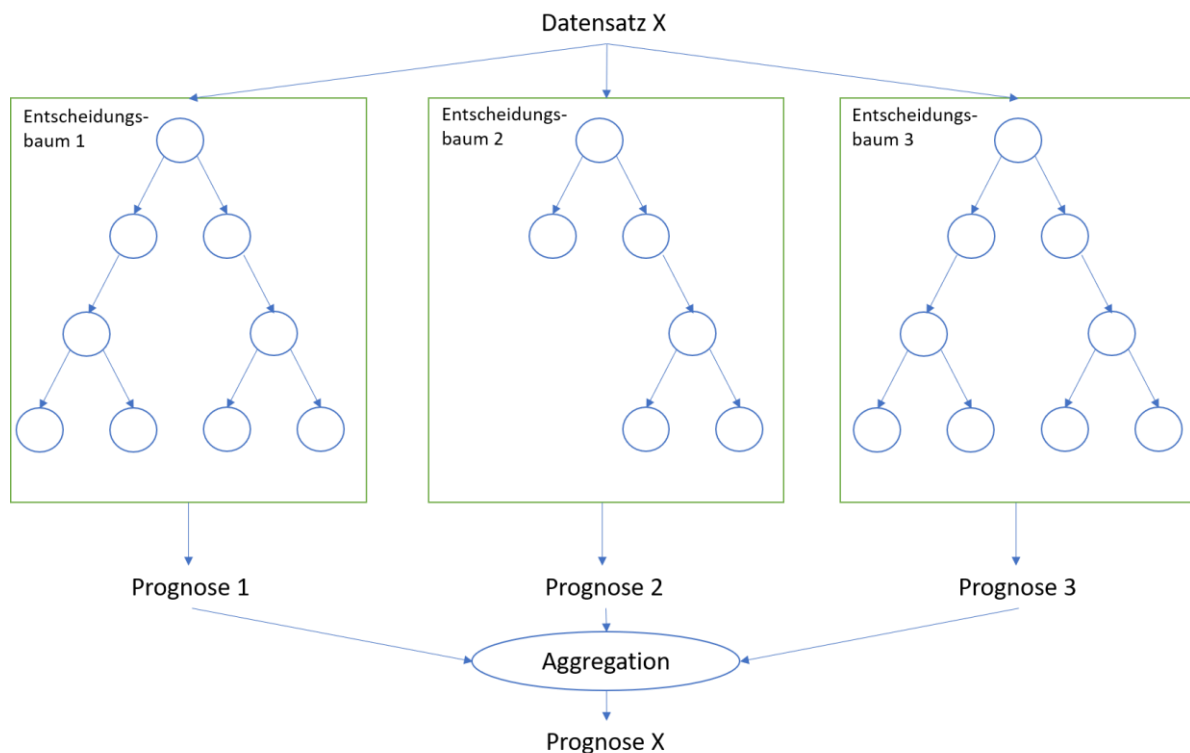


Abbildung 65 Entscheidungsverfahren mit mehreren Entscheidungsbäumen

Um die Baummodelle möglichst unterschiedlich zu gestalten, werden zwei Zufallsmechanismen verwendet. Zum einen wird beim Erstellen eines Baummodells eine zufällige Stichprobe (Ziehen mit Zurücklegen) aus den vorhandenen Datensätzen ausgewählt (sog. Bootstrapping), sodass jeder Entscheidungsbaum mit einer unterschiedlichen Datenbasis trainiert wurde. Zum anderen wird der Entscheidungsbaum mit einer zufällig gewählten Teilmenge an Merkmalen aufgebaut, vgl. (Géron, 2018, S. 182).

Das Verfahren besitzt mehrere Konfigurationsmöglichkeiten, um die Vorhersage des Modells zu optimieren:

- Die Anzahl der erzeugten Entscheidungsbäume die trainiert werden.

- Die maximale Tiefe der Baumstruktur (Anzahl an maximal aufeinanderfolgenden Entscheidungsknoten).

- Die Anzahl an Merkmalen mit denen ein Entscheidungsbaum aufgebaut wird.

- Die Metrik, die dafür eingesetzt wird, die Reihenfolge der Merkmalsknoten in einem Entscheidungsbaum zu bestimmen. Für eine Erläuterung der Metriken siehe (Géron, 2018, S. 178).

Zusätzlich kann das Verfahren die Wichtigkeit der eingehenden Merkmale ausgeben, die den Einfluss auf die Entscheidung anzeigen soll.

4.3.2. Durchführung

Die Umsetzung des Verfahrens erfolgt mit der Spark ML Bibliothek, da hier Bausteine für die Datenvorbereitung (sog. *Transformer*) und Modellvalidierung (sog. *Evaluator*) bereitgestellt werden. Die Implementierung des Verfahrens wird in Spark über einen *Estimator* repräsentiert.

Wie zuvor angedeutet, hängt die Prognosequalität eines Modells nicht nur von der geeigneten Wahl der Merkmale im Features Sets ab. Ein weiterer Bestandteil ist die Abstimmung der verfahrensinternen Hyperparameter, wie z.B. die Anzahl der gebildeten Bäume, oder maximale Baumtiefe. Eine optimale Kombination an Parametern führt dazu, dass der Fehler der Prognose minimiert wird. Das Finden der optimalen Wahl der Hyperparameter kann sehr zeitaufwändig sein, da viele Kombinationen ausprobiert werden müssen. Daher bieten viele Bibliotheken eine sog. *Rastersuche* an, in der unterschiedliche Kombinationen automatisiert getestet werden. Am Ende des Vorgangs wird die Kombination an Parametern zurückgegeben, die die betrachtete Fehlermetrik minimiert. In Spark wird dies über ein Parameter-Raster, sog. ParamGrid umgesetzt, welches eine Auswahl an Parametereinstellungen enthält:

```
paramGrid = ParamGridBuilder()\n    .addGrid(rf.subsamplingRate, [0.7, 0.8, 0.9, 1.0])\n    .addGrid(rf.numTrees, [7, 15, 40, 70, 120, 150])\n    .addGrid(rf.maxDepth, [3, 5, 6])\n    .addGrid(rf.featureSubsetStrategy, ["all", "onethird"])\n    .build()
```

Abbildung 66 Spark ParamGrid mit Kombinationen von Hyperparametern

Um zu verhindern, dass die Wahl des besten Modells nicht abhängig von der Aufteilung der Trainings- und Testmenge ist, werden alle Modellparameter Kombinationen auf verschiedene Teilmengen durchgeführt. Dieser Ansatz nennt sich *Kreuzvalidierung* und wird in Zusammenhang mit der Rastersuche benutzt. Angenommen über die Kreuzvalidierung werden drei unterschiedliche Teilmengen an Trainings- und Testdaten gebildet, kann eine Parameterkombination auf all diesen Mengen getestet werden. Die Fehlerwerte, die bei einer Parameterkombination auf den drei Teilmengen erzielt werden, werden anschließend gemittelt. Das beste Modell beinhaltet dann die Parametereinstellung, die anhand der drei Teilmengen insgesamt den geringsten

durchschnittlichen Fehlerwert aufweist, vgl. (VanderPlas, 2018, S. 388). In Spark kann das beschriebene Vorgehen mit einem **CrossValidator** automatisiert durchgeführt werden. Dazu wird dem CrossValidator neben der Anzahl an Teilmengen (*folds*) der Estimator (hier RandomForest), das Parameter-Raster und ein Evaluator übergeben:

```
rf = RandomForestRegressor(featuresCol="features")

crossval = CrossValidator(estimator=rf,
                          estimatorParamMaps=paramGrid,
                          evaluator=RegressionEvaluator(metricName="r2"),
                          numFolds=3)
```

Abbildung 67 Spark CrossValidator für die automatische Suche der besten Parameterkombinationen für eine Schätzung

Da die Modellparameter über das ParamGrid festgelegt wird, erfolgt die Instanziierung des RandomForestRegressors, bis auf die Angabe der Spalte des Features Sets, parameterlos. Das RegressionEvaluator Objekt beinhaltet Funktionen zur Berechnung von allen gängigen Fehlermetriken, die in dieser Arbeit vorgestellt wurden. Über die Angabe des „metricName“ kann definiert werden, anhand welcher Fehlermetrik das beste Modell bestimmt werden soll. Warum an dieser Stelle das R^2 verwendet wurde, wird in der Diskussion der Ergebnisse im nächsten Kapitel ersichtlich. Der Vorgang der Kreuzvalidierung kann dabei sehr rechenintensiv sein, da die Kombinationen an Modellparametern und Anzahl der Teilmengen, sich schnell auf über 100 Möglichkeiten multiplizieren.

Da eine aufbereitete Datenbasis in der Curated Zone vorliegt, können nun verfahrensspezifische Vorverarbeitungen durchgeführt werden. Da in Spark die Verfahren mit numerischen Vektoren durchgeführt werden, müssen die eingehenden Features über einen **VectorAssembler** transformiert werden. Zuvor müssen jedoch kategoriale Features in Einheitsvektoren transformiert werden, wie in Kapitel 4.1.2. beschrieben wurde. In Spark erfolgt dies über die Transformer **OneHotEncoder** und **StringIndexer**. Das Vorgehen wird anhand des Merkmals „Regionstyp“ mit den Ausprägungen „stadt“ und „gemeinde“ erläutert.

1. Überführung des kategorialen Features in eine numerische Darstellung mit dem StringIndexer. Nach der Transformation wird die Bezeichnung „stadt“ mit 1 und „gemeinde“ mit 0 repräsentiert.
2. Das nun numerische Features wird mit dem OneHotEncoder in einen binären Vektor transformiert.
3. Wenn alle kategorialen Features transformiert wurden, können diese mit den restlichen Features über den VectorAssembler in einen Vektor überführt werden (sog. Feature-Vektor).

Am Ende dieses Prozesses liegt für jede Region ein Feature-Vektor vor, in denen die Merkmale aus dem Feature Set enthalten sind. Im nachfolgenden Programmcode ist wurde das Vorgehen umgesetzt. Dabei wird für jeden Transformer die Bezeichnung des eingehenden und des transformierten Features angegeben. Die Transformation erfolgt dabei über die Methoden *fit()* und *transform()*.

Auch wenn das RandomForest Verfahren mit fehlenden Werten umgehen kann, können die Transformer von Spark nicht mit diesen arbeiten. Daher fallen an dieser Stelle alle Regionen weg, die fehlende Merkmalswerte besitzen.

```
## Umformung des Merkmals "Regionstyp" in eine numerische Darstellung
indexer = StringIndexer(inputCol="REGIONSTYP", outputCol="REGIONSTYP_INDEX")
data_index = indexer.fit(data).transform(data)

## Transformation der nun numerischen Merkmale "Besiedlungsschlüssel" und "Regionstyp" in einen binären Vektor
encoder = OneHotEncoder(inputCols=["BESIEDLUNGSSCHLUESSEL", "REGIONSTYP_INDEX"],
                        outputCols=["BESIEDLUNGSSCHLUESSEL_encoded", "REGIONSTYP_encoded"])

data_encoded = encoder.fit(data_indexed).transform(data_index)

## Transformation aller Features in einen Vektor
vector_assembler = VectorAssembler(inputCols=input_feat, outputCol="features")
data_vec = vector_assembler.transform(data_encoded)
```

Abbildung 68 Feature Transformation mit Spark

Die daraus resultierenden Feature Vektoren werden auf nachfolgender Abbildung für einige Regionen ausgegeben.

```
+-----+
|features|
+-----+
|[217.0,376.0,4613.0,4088.0,608.0,254.0,13120.0,201.0,65.24,8.4,13.6,10.9,18912.0,317.0,0.0,0.0,0.0,0.0]|
|[384.0,707.0,8764.0,7090.0,1342.0,408.0,20886.0,653.0,31.97,8.4,13.6,10.9,18912.0,323.0,0.0,0.0,0.0,0.0]|
|[6.0,4.0,221.0,191.0,20.0,17.0,599.0,66.0,9.07,8.4,13.6,10.9,18912.0,17.0,0.0,0.0,0.0,1.0]|
|[21.0,9.0,435.0,283.0,33.0,13.0,1076.0,74.0,14.55,8.4,13.6,10.9,18912.0,12.0,0.0,0.0,0.0,1.0]|
|[72.0,127.0,1493.0,1227.0,198.0,65.0,4237.0,377.0,11.23,8.4,13.6,10.9,18912.0,96.0,0.0,0.0,0.0,1.0]|
|[21.0,19.0,529.0,397.0,31.0,22.0,1367.0,148.0,9.21,8.4,13.6,10.9,18912.0,28.0,0.0,0.0,0.0,1.0]|
|[11.0,12.0,295.0,199.0,18.0,12.0,781.0,61.0,12.79,8.4,13.6,10.9,18912.0,15.0,0.0,0.0,0.0,1.0]|
```

Abbildung 69 Feature Vektoren Spark

Nun kann die Aufteilung der Trainings- und Testmenge erfolgen. Zuvor wird die gesamte Datenmenge in Vorhersage- und Lerndaten unterteilt. Die Vorhersagedaten enthalten die Regionen, für die kein Umsatz existiert. Die Regionen, für die ein Umsatz gemessen werden kann, liegen in den Lerndaten vor. Die Regionen in den Lerndaten werden dann zufällig zu den Trainings- oder Testdaten im Verhältnis 70% und 30% zugeordnet. Daraufhin kann das beste Modell durch den CrossValidator selektiert werden, welches den Umsatz am besten schätzt. Der entsprechende Programmcode ist nachstehender Abbildung zu entnehmen.

```
vorhersagedaten = data_vec.select("*").where(col(pred_col).isNull())
lerndaten = data_vec.select("*").where(col(pred_col).isNotNull()).withColumnRenamed(pred_col, "label")

training, test = lerndaten.randomSplit([0.7, 0.3])

cvModel = crossval.fit(training)
```

Abbildung 70 Aufteilung Trainings- und Testdaten sowie Durchführung der Trainingsphase in Spark

Die Prognosequalität des besten Modells kann jetzt über die Testdaten validiert werden. An dieser Stelle könnte mit dem CrossValidator weitergearbeitet werden, was jedoch nicht zu empfehlen ist. Der CrossValidator fungiert als Wrapper, sodass umständlich auf den Evaluator zugegriffen werden muss, der zudem nicht den vollen Funktionsumfang bietet. Dabei kann beispielsweise nicht die Wichtigkeit der Features (*Importance*) ausgelesen werden. Aus diesem Grund wird ein neues RandomForest Modell mit den Modellparametern des besten Modells erzeugt und trainiert. Danach kann die Validierung anhand der Testdaten erfolgen. Die daraus resultierenden Ergebnisse werden nun diskutiert.

4.3.3. Diskussion der Ergebnisse

Die Umsatzprognose wurde nach dem beschriebenen Vorgehen mit drei Analysemodellen durchgeführt – dem regionsunabhängigen Modell (M_Gesamt) und den regionsabhängigen für Städte und Gemeinden (M_Stadt und M_Gemeinde).

Bevor ein Vergleich der Modelle vorgenommen wird, soll zunächst die Validierung von M_Gesamt erfolgen. In der nachfolgenden Abbildung sind die Fehlerwerte festgehalten.

mae	mse	rmse	r2
1061522.17	2067673400040.05	1437940.68	0.91

Abbildung 71 Random Forest - regionsunabhängiges Modell: Fehlerkennzahlen und R^2

Da die Fehlerwerte zu (R)MSE und MAE extrem groß sind, vermittelt dies zunächst den Eindruck, dass die Prognose keine guten Ergebnisse liefert. Dabei muss der Wertebereich der Zielgröße berücksichtigt werden. Der durchschnittliche Jahresumsatz in einer Region liegt zwischen ca. 400.000 und 2.8 Milliarden. Der Vergleich der statistischen Profile von vorhergesagten (*prediction*) und wahren Umsatz (*label*) zeigt, dass die Verteilungen sehr ähnlich sind, wie nachfolgende Abbildung zeigt.

	label	prediction
count	5.590000e+02	5.590000e+02
mean	7.667166e+06	7.672591e+06
std	4.698143e+06	4.005639e+06
min	4.263784e+05	2.172087e+06
25%	4.390467e+06	4.630541e+06
50%	6.593357e+06	6.924262e+06
75%	9.590947e+06	9.530365e+06
max	2.873314e+07	2.160806e+07

Abbildung 72 Random Forest - regionsunabhängiges Modell. Statistisches Profil von Real und Schätzwert

Der Mittelwert (*mean*), die Standardabweichung (*std*) und die Quantile weichen kaum voneinander ab. Lediglich der Wertebereich des prognostizierten Umsatzes von 2.1 Millionen (Minimum) bis 2,1 Milliarden (Maximum) ist kleiner. Bei diesem Wertebereich

sind Fehlerwerte im Millionenbereich, wie bei dem RMSE und MSE, relativ klein. Auch das R^2 von 0,91 zeigt, dass der Umsatz der Regionen in der Testmenge recht gut prognostiziert werden kann. Da die rein numerische Betrachtung bei Werten dieser Größenordnung schwierig ist, wird eine visuelle Darstellung zur Bewertung des Modells hinzugezogen. In der nachfolgenden Abbildung ist der tatsächliche und prognostizierte Umsatz der Regionen im Testdatensatz abgebildet. Durch die aufsteigende Sortierung lassen sich Abweichungen besser darstellen.

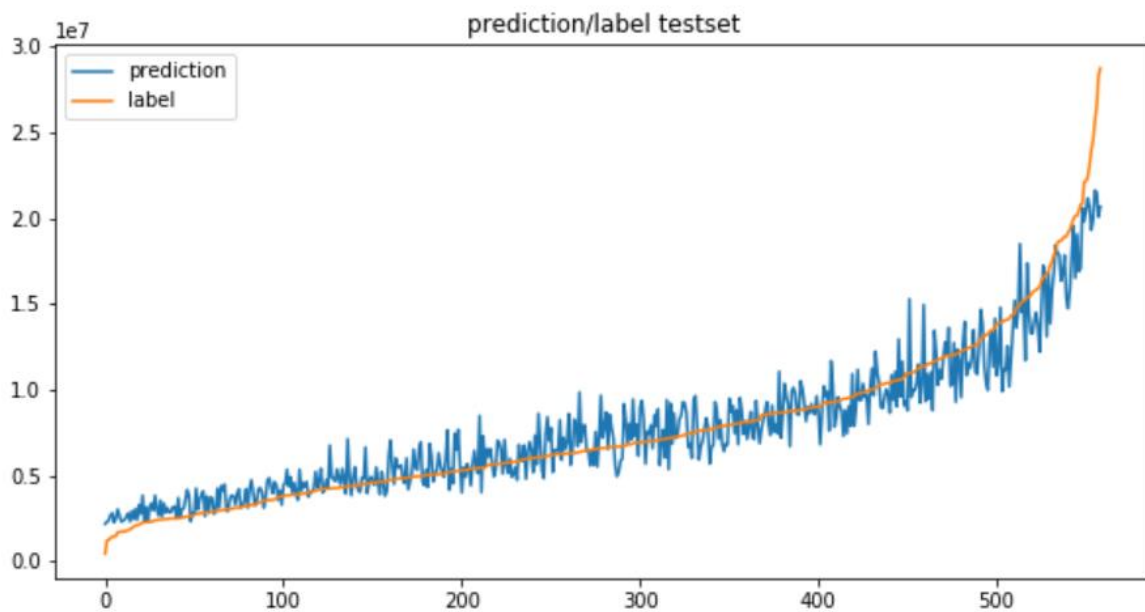


Abbildung 73 Random Forest - regionsunabhängiges Modell: Vergleich Real- und Schätzwerte

Allgemein ist zu erkennen, dass das Modell (blaue Linie) den nicht-linearen Verlauf des wahren Umsatzes (orange Linie) gut approximiert. Es kann behauptet werden, dass das Modell im oberen Umsatzbereich eine pessimistische, im unteren Bereich eine optimistische Schätzung vornimmt. Gerade im umsatzschwächeren Segment (was in den nicht besetzten Regionen zu erwarten ist), ist die Schätzung genauer als in den anderen Umsatzbereichen. Diese Annäherung sollte dabei nicht zu genau sein, um ein Overfitting zu vermeiden. Es ist anzunehmen, dass Regionen im umsatzschwächeren Bereich ab einer gewissen Grenze zuverlässiger geschätzt werden können, als Regionen im mittleren und oberen Umsatzsegment. Bei der Untersuchung der Daten wurde festgestellt, dass die Vorhersagedaten fast vollständig aus Gemeinden bestehen, für die ein Umsatz im unteren Wertebereich wahrscheinlich ist.

Als nächstes werden die Importance Kennzahlen der eingehenden Features interpretiert. Auf Abbildung 74 sind die Werte der einzelnen Features in einem Diagramm abgetragen.

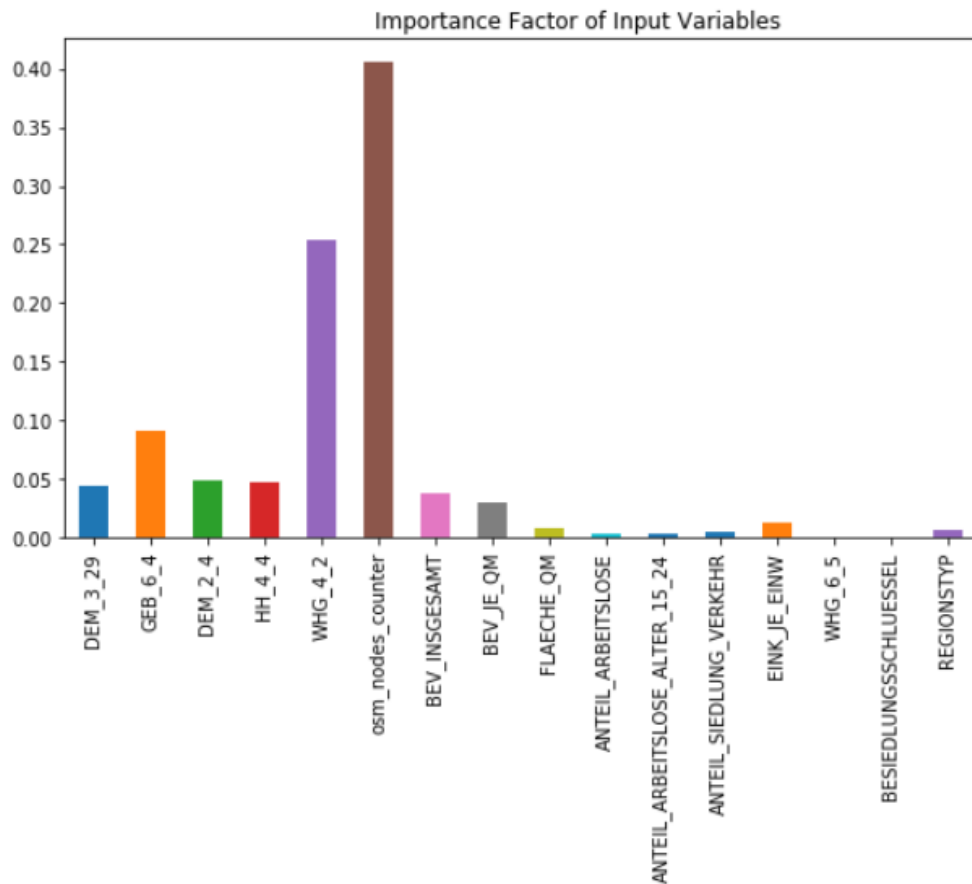


Abbildung 74 Visualisierung der Importance Faktoren des Random Forest

Infrastrukturmerkmale wie GEB_6_4, WHG_4_2 und osm_nodes_counter haben den höchsten Importance Faktor. Dagegen werden Bevölkerungs- und Wirtschaftsdaten wie Bevölkerungsanzahl und Einkommen eine geringere Bedeutung zugesprochen. Bei einer Modelloptimierung könnten Features, die in der Grafik rechts von „BEV_JE_QM“ liegen, aus dem Feature Set entfernt werden.

Der Importance Faktor des Regionstyp zeigt sich ebenfalls unauffällig. Dadurch kann abgeleitet werden, dass für die Umsatzschätzung nicht ausschlagend ist, ob die Region eine Gemeinde oder Stadt ist. In der Analyse konnte dagegen gezeigt werden, dass die Höhe des erzielten Umsatzes in einer Region sehr wohl vom Regionstyp abhängig ist.

Um herauszufinden, ob eine regionsabhängige Umsatzschätzung zu besseren Ergebnissen führt, sollen die Modelle M_Gesamt, M_Stadt und M_Gemeinden verglichen werden. Die einzelnen Modelleigenschaften können aus Tabelle 12 entnommen werden.

Tabelle 12 Parameterausprägungen der einzelnen Modelle für das Random Forest Verfahren

Modelle	M_Gesamt	M_Stadt	M_Gemeinde
Anzahl gelabelte Daten	1825	1221	604
Anzahl Trainingsdaten	1266 (70%)	846 (70%)	431 (70%)
Anzahl Testdaten	559 (30%)	375 (30%)	173 (30%)
Anzahl Vorhersagedaten	7293	667	6626
Anzahl Bäume	150	120	120
Maximale Baumtiefe	7	5	5
Feature Set Strategie	all	all	all
Subsampling Rate	0.9	1.0	1.0

Zunächst ist festzustellen, dass bei M_Stadt und M_Gemeinde die optimale Wahl von Hyperparametern gleich ist. Bei einem Umfang von 18 Merkmalen im Feature Set, wurden maximal 5 Features je Baum für die Umsatzprognose verwendet. Dabei wurden die einzelnen Prognosewerte über 120 Bäume gemittelt. Es kann gemutmaßt werden, dass durch die geringe Komplexität der einzelnen Bäume, viele Bäume benötigt wurden, um für eine optimale Umsatzprognose zu erzielen, bzw. den Fehlerwert zu minimieren.

Das M_Gesamt benötigte für eine optimale Schätzung 150 Bäume mit einer maximalen Tiefe von 7. Auch hier kann vermutet werden, dass durch die vermischten Trainingsdaten (mit Städten und Gemeinden) eine komplexere Struktur für eine Umsatzprognose zu angelernt werden musste, um den Fehlerwert zu minimieren. Die Fehlerwerte zu den Modellen sind in der nachfolgenden Tabelle aufgelistet.

Tabelle 13 Fehlerwerte der Modelle für das Random Forest Verfahren

Metrik	M_Gesamt	M_Stadt	M_Gemeinde
RMSE	1.437.940	1.544.873	1.046.136
MSE	2.067.673.400.040	2.386.634.630.438	1.094.401.459.828
MAE	1.061.522	1.214.986	725.650
R ²	0,91	0,89	0,84

Die absoluten Fehlerwerte von M_{Stadt} übersteigt die von M_{Gemeinde} in allen betrachteten Metriken. Insbesondere ist der MSE des Stadtmodells doppelt so hoch wie das des Gemeindemodells. Auf den ersten Blick scheint es, als würde die Umsatzprognose durch M_{Gemeinde} geringere Abweichungen vom wahren Wert aufweisen. Jedoch muss berücksichtigt werden, dass die absoluten Fehlerwerte der Modelle nicht direkt vergleichbar sind. Das liegt daran, dass die Wertbereiche der Umsätze für Gemeinden und Städte unterschiedlich sind. Daher sind für Städte höhere absolute Abweichungen zu erwarten. Interessanter ist der Vergleich der R^2 -Werte. Das Gemeindemodell erzielt mit 0,84 einen schlechteren Wert als das Stadtmodell mit 0,89. Es kann also behauptet werden, dass das Stadtmodell vergleichsweise zu realitätsnäheren Ergebnissen führt. Den höchsten Wert für R^2 erreicht vorliegend das regionsunabhängige Modell M_{Gesamt} . Dabei sei darauf hingewiesen, dass die Prognosequalität zur Vorhersage der Testdatenmenge handelt. Es wäre falsch zu behaupten, dass eines der Modelle generell besser ist als das andere. Ausschlaggebend sind die Regionen, für die einen Umsatz vorhergesagt werden soll. Sind in den Vorhersagedaten Regionen vertreten, die den Regionen aus den Trainingsdaten sehr ähnlich sind, wird das Gemeindemodell M_{Gemeinde} die beste Umsatzschätzung erzielen, obwohl der R^2 -Wert im Vergleich am geringsten ausfällt.

Da es sich im Machine Learning empfiehlt, mehrere Verfahren zu verwenden, wird als nächstes eine Umsatzprognose mit einem neuronalen Netz durchgeführt. Abschließend werden die Verfahren und die Prognosequalitäten miteinander verglichen.

4.4. *Neuronales Netz mit Tensorflow und Keras*

Das grundlegende Konzept künstlicher neuronaler Netze (NN) beruht auf Hypothesen und Modellen über die Funktionsweise des menschlichen Gehirns, welches zum Lösen komplexer Aufgabenstellungen fähig ist, vgl. (Raschka, 2017, S. 41). Das künstliche neuronale Netz kann je nach Konstruktion für viele Problemstellungen verwendet werden, welche aus dem Bereich des supervised (Klassifizierung oder Regression) sowie unsupervised learning, stammen können.

In den nächsten Kapiteln soll das Konzept beschrieben und mit Tensorflow und Keras umgesetzt werden. Das neuronale Netz kann daraufhin mit dem RandomForest verglichen werden.

4.4.1. Konzept

Die Bestandteile eines neuronalen Netzes sind sog. *Neuronen*, welche in Schichten angeordnet und verbunden werden. Dabei orientiert sich die Anzahl der Neuronen in der ersten und letzten Schicht an das Analysemodell. Die Menge der Neuronen in der Eingangsschicht entspricht der Anzahl an eingehenden Merkmalen⁶¹. Die Ausgabeschicht definiert sich durch die Zielgröße des betrachteten Problems, z.B. n Neuronen für n Ausgaben. Da vorliegend kein Klassifizierungsproblem mit vielen Ausgabemöglichkeiten vorliegt, wird nur ein Neuron in Ausgabeschicht benötigt, welches den geschätzten Umsatz repräsentieren soll (siehe Abbildung 75).

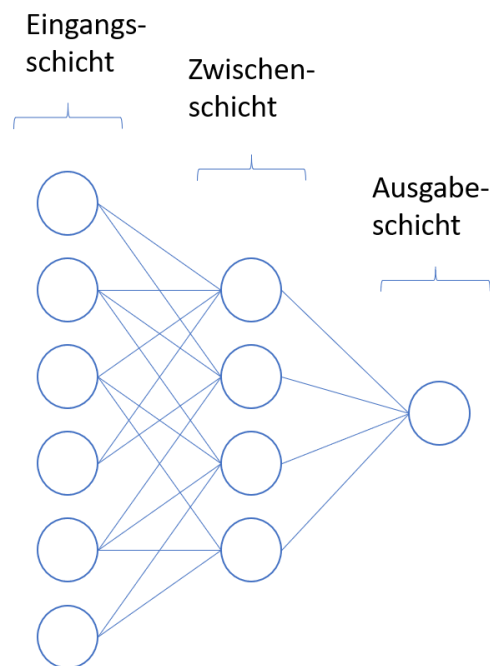


Abbildung 75 Schichten eines künstlichen neuronalen Netzes

Das einzelne Neuron besitzt eine **Aktivierungsfunktion**, die nach Summation eingehender Werte, eine Ausgabe erzeugt. Der Ausgabewert ist von der Wahl der Aktivierungsfunktion abhängig und kann je nach Problemstellung gewählt werden. Die erzeugte Ausgabe wird daraufhin gewichtet und geht als Wert in die Neuronen der nächsten Schicht ein. Der Aufbau eines Neurons ist nachstehender Abbildung verdeutlicht.

⁶¹ Unter gewissen Umständen ist es effizienter, wenn einige Merkmale einem Eingangsneuron zusammengefasst werden

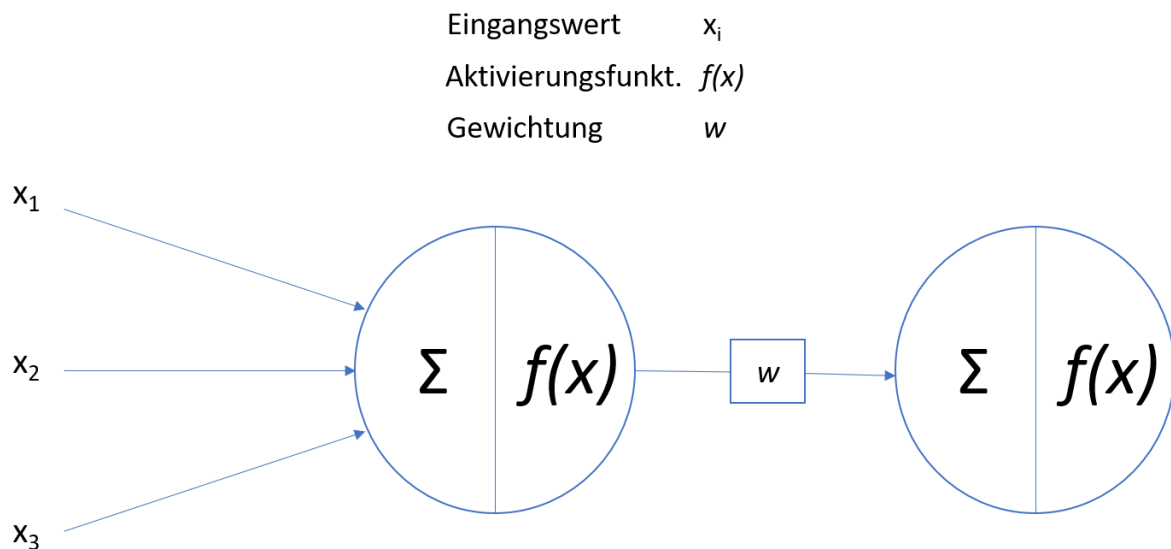


Abbildung 76 Schematischer Aufbau eines künstlichen Neurons

Das neuronale Netz „lernt“, indem es die Gewichte in mehreren Schritten anpasst. Mit jeder Anpassung soll der Fehlerwert minimiert werden, der im neuronalen Netz häufig als Kostenfunktion bezeichnet wird. Ein Schritt, bzw. eine **Lerniteration**, ist dann abgeschlossen, wenn für alle vorhandenen Datensätze eine Schätzung durchgeführt und der Fehlerwert ermittelt wurde. Bevor die nächste Lerniteration startet, werden die Gewichte so angepasst, dass am Ende ein geringerer Fehlerwert erzielt wird. Die Anpassung der Gewichte kann über unterschiedliche Optimierungsverfahren erfolgen, z.B. dem **Gradientenabstiegsverfahren** vgl. (Géron, 2018, S. 262). Über dieses Vorgehen können die Gewichtungen immer weiter auf die Trainingsdaten abgestimmt werden.

Nach (Raschka, 2017, S. 383) können mit dem neuronalen Netz sehr komplexe Zusammenhänge zwischen Eingabemerkmale und einer Zielgröße abgebildet werden.

4.4.2. Durchführung

Neben dem RandomForest ist auch das neuronale Netz in der Spark ML enthalten. Dieses ist jedoch so konstruiert, dass lediglich Klassifikationsprobleme adressiert werden können. Stattdessen wird in dieser Arbeit auf Tensorflow zurückgegriffen, ein Machine Learning Framework von Google, welches seit 2017 als Open Source Projekt unter Apache läuft. Da im vorliegenden Projekt mit Python gearbeitet wird, wird Keras als Python Schnittstelle

für Tensorflow benutzt. Im Folgenden wird ein sog. Feed Forward Netz definiert, bei denen alle Neuronen einer Schicht mit den Neuronen der nachfolgenden Schicht vollständig verknüpft sind. Für den Aufbau eines neuronalen Netzes wird in Keras zunächst ein Sequential model angelegt, welches als Rohform des Netzwerkes dient. Danach wird das Model von der Eingangsschicht bis zur Ausgabeschicht erweitert, wie im nachfolgenden Programmcode zu entnehmen ist.

```
## Rohform des Modells
NN = Sequential()

## Eingabeschicht
NN.add(Dense(5, input_dim=len(input_features),
             kernel_initializer="normal", activation="relu"))

## Zwischenschichten
NN.add(Dense(10, kernel_initializer="normal", activation="relu"))
NN.add(Dense(30, kernel_initializer="normal", activation="relu"))
NN.add(Dense(9, kernel_initializer="normal", activation="relu"))
NN.add(Dense(3, kernel_initializer="normal", activation="relu"))

## Ausgabeschicht
NN.add(Dense(1, kernel_initializer="normal", activation="linear"))

## Modell bauen
NN.compile(loss="mean_squared_error", optimizer="adam")
```

Abbildung 77 Erzeugung eines neuronalen Netzes in Keras

Je nach Art des Netzwerkes können die Neuronen der einzelnen Schichten verknüpft werden. Keras bietet verschiedene Varianten an, um je nach Bedarf eine Verbindung der einzelnen Schichten herzustellen. Um ein Feed Forward Netz zu definieren, wird das Neuron über *Dense()* erzeugt. Die Eigenschaften der Neuronen jeder Schicht können über verschiedene Parameter angepasst werden. Hierzu zählen neben der Anzahl der Neuronen in der Schicht, die verwendete Aktivierungsfunktion (*activation*) sowie die Methode zur Bestimmung der Anfangsgewichte (*kernel_initializer*). Die Anfangsgewichte können über verschiedene zufallsbasierte Verfahren umgesetzt werden. Für alle Schichten werden Zufallswerte generiert, die einer Standardnormalverteilung folgen (hier „normal“). Als Aktivierungsfunktion wurde in der Vergangenheit häufig die sog. Sigmoidalfunktion verwendet, da diese der Reizübertragung des biologischen Neurons am nächsten kommt. Es stellte sich jedoch heraus, dass in dem künstlichen Neuron eine

Aktivierungsfunktion wie Rectifier neural networks (ReLU) zu besseren Ergebnissen führt vgl. (Géron, 2018, S. 263). Daher wird in diesem neuronalen Netz für die Eingangsschicht sowie Zwischenschichten ReLU verwendet. Für eine visuelle Darstellung sowie Vor- und Nachteile der Funktionen soll auf (Géron, 2018, S. 262 ff.) verwiesen werden. Da im neuronalen Netz mit normalisierten Werten gerechnet wird, müssen diese in der Ausgabeschicht auf den ursprünglichen Wertebereich skaliert werden. Daher besitzt hier das Ausgabeneuron eine lineare Aktivierungsfunktion, sodass der prognostizierte Wert mit dem wahren Umsatzwert vergleichbar ist.

Sind alle Schichten definiert, wird das Netz mit `compile()` unter Angabe der Kostenfunktion und des Optimierungsverfahren erstellt.

```
## Modell bauen
NN.compile(loss="mean_absolute_error", optimizer="adam")
```

Abbildung 78 `Compile()` Funktion um das Netzwerk endgültig zu definieren

Als Kostenfunktion (hier `loss`) wird MAE verwendet, da es unempfindlicher gegen Ausreißer ist und sich daher besser eignet als die Metrik des (R)MSE. Für die Anpassung der Gewichte wird ein stochastisches Optimierungsverfahren, welches mit „Adam“ bezeichnet wird. Die Vorbereitung der Daten wurde mit Scikit-learn durchgeführt. Ähnlich wie mit Spark können darüber kategorialen Variablen indexiert und in einen Einheitsvektor überführt werden. Danach erfolgt die Aufteilung der Trainings- und Testdatenmenge in einem Verhältnis 80% zu 20%⁶². Da neuronale Netz mit skalierten Werten arbeiten, werden die Werte vorliegend standardisiert⁶³. Das Standardisieren wird mit dem `StandardScaler` aus der Sklearn Bibliothek durchgeführt.

```
scaler = StandardScaler()

x_train_scaled = scaler.fit_transform(x_train_unscaled)
x_test_scaled = scaler.fit_transform(x_test_unscaled)
```

Abbildung 79 Standardisierung der Features mit Scikit-learn

⁶² Derweil die Aufteilung beim Random Forest 70% zu 30% war, hat beim neuronalen Netz ein Verhältnis 80%/20% zu besseren Ergebnissen geführt.

⁶³ Die Standardisierung bietet sich bei einer Ausreißer Problematik an.

Trainingsdaten sind hier in Features (mit x gekennzeichnet) und den Umsatzwerten (mit y gekennzeichnet) unterteilt. Im letzten Schritt wird das neuronale Netz unter der Angabe der Lerniterationen (*epochs*) mit *fit()* angelernt. Die Minimierung der Fehlerwerte während des Trainings, kann dabei durch eine Zuweisung zwischengespeichert werden, wie auf der nächsten Abbildung gezeigt wird.

```
trainingshistorie = NN.fit(x_training, y_training, epochs=2000, verbose=True)
```

Abbildung 80 Anlernen des neuronalen Netzes Verfahrens und abspeichern der Trainingsschritte über Keras

Aus der Trainingshistorie kann die Minimierung des MAE entnommen und visualisiert werden. Auf Abbildung 81 ist zu erkennen, dass das Minimum des Fehlerwerts nach ca. 175 erreicht wurde. Häufig wird das stationäre Verhalten des Fehlerwertes als Konvergenz bezeichnet. Dabei konvergiert das neuronale Netz zu einem Optimum der Prognosequalität⁶⁴.

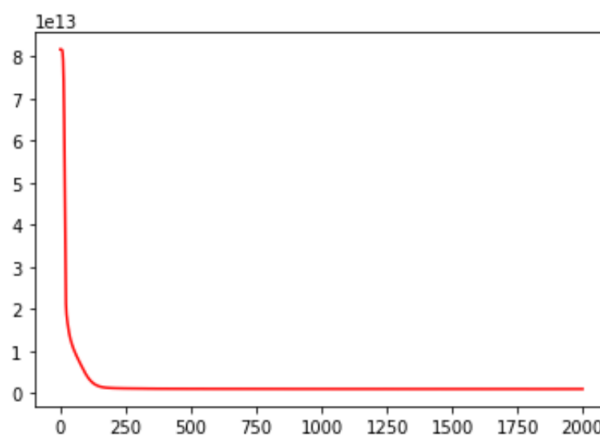


Abbildung 81 Verlauf des Fehlerwertes über die Trainingseinheiten

⁶⁴ Genaugenommen muss von einem lokalen Optimum ausgegangen werden, da in Abhängigkeit der gewählten Anfangsgewichte nicht zwingend das globale Optimum erreicht werden kann.

4.4.3. Diskussion der Ergebnisse

Die Umsatzprognose wurde auch hier mit den Modellen M_Gesamt, M_Stadt und M_Gemeinde durchgeführt. Zunächst wird das regionsunabhängige Modell (M_Gesamt) betrachtet. Die Abweichungen vom wahren Umsatz und dem Prognosewert sind in der nachfolgenden Grafik abgebildet.

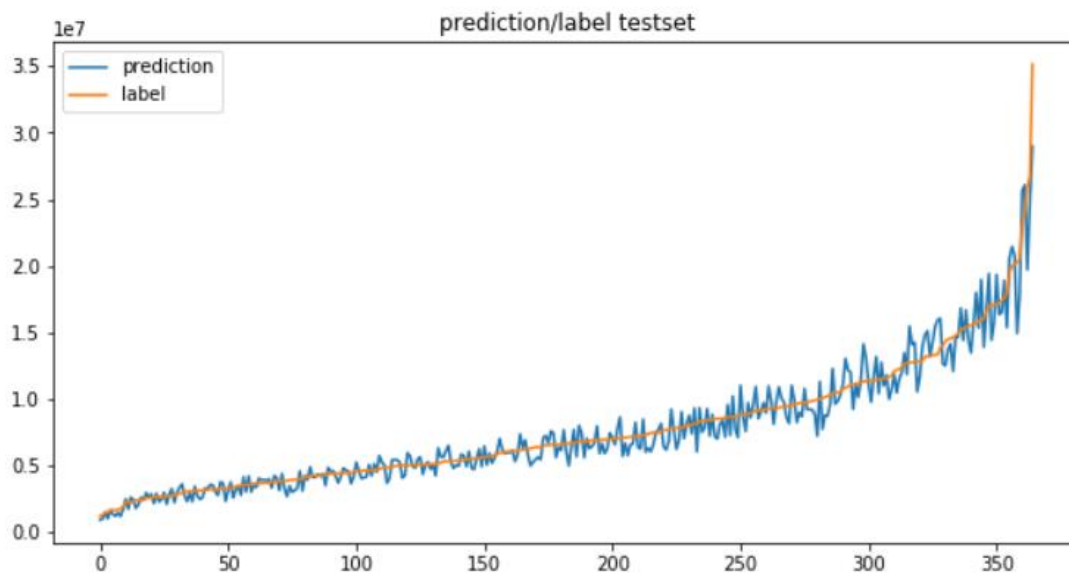


Abbildung 82 Neuronale Netz - regionsunabhängiges Modell: Vergleich Real- und Schätzwerte

Es ist zu erkennen, dass das neuronale Netz den nicht-linearen Verlauf des wahren Umsatzes abbilden kann. Gerade im umsatzschwachen Segment sind dabei geringere Abweichungen zu erkennen. Je höher der Umsatz ist, desto größere Abweichungen sind zu erwarten. Derweil das Minimum des wahren Umsatzes und der Prognose ähnlich sind, unterschreitet das Modell das tatsächliche Umsatzmaximum. Im betrachteten Anwendungsfall könnte eine pessimistische Vorhersage vorgezogen werden, da so Umsätze nicht überschätzt werden.

Auch für M_Stadt und M_Gemeinde konnten die präzise Approximation an den wahren Umsatz grafisch festgestellt werden. Die Konfiguration der unterschiedlichen Modelle ist Tabelle 14 zu entnehmen.

Tabelle 14 Parameterausprägungen der einzelnen Modelle für das neuronale Netz

Modelle	M_Gesamt	M_Stadt	M_Gemeinde
Anzahl gelabelte Daten	9571 (80%)	1221 (80%)	604 (80%)
Anzahl Trainingsdaten	1266 (20%)	846 (20%)	431 (20%)
Anzahl Testdaten	559	375	173
Anzahl Vorhersagedaten	7293	667	6626
Aufteilung Schichten	18-10-30-9-3-1	17-10-30-9-3-1	17-10-30-9-3-1
Anzahl Lerniterationen	2000	2000	2000

Der Aufbau des neuronalen Netzes der einzelnen Modelle ist dabei gleich, da dadurch eine bessere Vergleichbarkeit entsteht. Die Anzahl der Neuronen in der Eingangsschicht ist im M_Gesamt Modell um das Feature „Regionstyp“ erweitert.

Die Fehlermetriken der einzelnen Modelle werden in nachfolgender Tabelle aufgelistet.

Tabelle 15 Fehlerwerte der Modelle über das Random Forest Verfahren

Metrik	M_Gesamt	M_Stadt	M_Gemeinde
RMSE	1.124.155	1.242.490	1.551.399
MSE	1.263.725.349.050	1.543.781.835.578	2.406.841.192.131
MAE	815.594	1.002.448	1.064.702
R ²	0,94	0,94	0,83

Hier schneidet das Gemeindemodell verhältnismäßig schlecht ab. Es übersteigt jegliche Fehlerwerte der Modelle M_Gesamt und M_Stadt bei einem geringeren R² Wert. Interessanterweise wurde mit der Prognose über M_Gesamt die geringsten Fehlerwerte erzielt. Demnach konnte mit dem regionsunabhängigen Modell, die Umsätze der Regionen aus den Testdaten, am besten vorhergesagt werden. Die R² Werte von M_Gesamt und M_Stadt betragen 0,94 und erzielen damit ein sehr gutes Ergebnis. Insgesamt würde M_Gesamt für eine Umsatzschätzung bevorzugt werden.

4.5. Vergleich der Verfahren

Der Vergleich des RandomForests und neuronalen Netzes soll hinsichtlich der Anwendbarkeit und Prognosequalität vorgenommen werden. Dafür werden die Eigenschaften beider Verfahren zunächst zusammengefasst.

RandomForest:

Geringe Vorverarbeitung. Robust gegenüber Ausreißer, keine Skalierung notwendig und unterstützt (theoretisch) fehlende Werte. Jedoch mussten in der Durchführung über Spark ML die Datensätze mit fehlenden Werten entfernt werden, da hier nur mit voll besetzten Vektoren Berechnungen erfolgen können.

Verständlichkeit. Das Konzept des Verfahrens ist für Anwender leicht nachzuvollziehen. Eine Entscheidung ist jedoch nur bei einer geringen Menge an Entscheidungsbäumen transparent, sofern diese nicht tief verschachtelt sind.

Indikatoren für Wichtigkeit von Merkmalen. Der Random Forest gibt Auskunft darüber, wie wichtig ein Merkmal für die Prognose war. Darüber können Zusammenhänge zwischen eingehenden Merkmalen und der Zielgröße aufgedeckt werden, die wiederum in der Feature Selection Phase einfließen können.

Geringes Risiko des overfittings. Durch Bagging (Zufallsstichprobe der Lerndaten) und Ensembling (mehrere Entscheidungsbäume) wird eine zu präzise Anpassung an die Trainingsdaten vermieden.

Neuronales Netz:

Viele Konfigurationsmöglichkeiten. Beim Aufbau des Netzes müssen nicht nur die Schichten und Anzahl Neuronen bestimmt werden, sondern auch Aktivierungsfunktionen und Optimierungsverfahren. Eine automatisierte Rastersuche nach optimalen Parametern ist durch die Anzahl an Freiheitsgraden zu komplex. Das neuronale Netz muss manuell definiert werden.

Lösung für komplexe Probleme. Gerade durch die vielen Konfigurationsmöglichkeiten kann das neuronale Netz auch für sehr komplexe Aufgabenstellungen verwendet werden. Die Gefahr besteht jedoch, dass das Modell zu genau auf die Trainingsdaten abstimmt wird und so zum overfitting führt.

Höherer Verarbeitungsaufwand. Die Features müssen zunächst skaliert und Datensätze mit fehlenden Werten entfernt werden.

Geringe Transparenz. Das neuronale Netz häufig als Blackbox bezeichnet, da die Entscheidung für einen Anwender nicht nachvollziehbar ist. Dies liegt unter anderem daran, dass sich ein Anwender sehr viel Kenntnisse aneignen muss, um die Funktionsweise eines neuronalen Netzes zu verstehen. Indikatoren, wie relevant einzelne Features für die Prognose waren, sind ebenfalls nicht ersichtlich.

Die Prognosequalität des jeweiligen Verfahrens, die auf den Testdaten erzielt wurden, werden nachfolgend wiederholt dargestellt.

Random Forest:

Metrik	M_Gesamt	M_Stadt	M_Gemeinde
RMSE	1.437.940	1.544.873	1.046.136
MSE	2.067.673.400.040	2.386.634.630.438	1.094.401.459.828
MAE	1.061.522	1.214.986	725.650
R ²	0,91	0,89	0,84

Tabelle 16 Fehlerwerte der Modelle über RandomForest Verfahren

Neuronales Netz:

Metrik	M_Gesamt	M_Stadt	M_Gemeinde
RMSE	1.124.155	1.242.490	1.551.399
MSE	1.263.725.349.050	1.543.781.835.578	2.406.841.192.131
MAE	815.594	1.002.448	1.064.702
R ²	0,94	0,94	0,83

Tabelle 17 Fehlerwerte der Modelle über das neuronale Netz

Allgemein ist zu erkennen, dass die Prognose der Modelle M_Gesamt und M_Stadt für das neuronalen Netzes kleinere Fehlerwerte aufweisen und dabei leicht höhere R² Werte erzielen konnten, als die Modelle des RandomForests. Die Ausnahme stellt dabei das Gemeindemodell dar. Die Fehlerwerte des M_Gemeinde für das neuronalen Netzes sind weitaus schlechter, als die des RandomForests – bei einer ähnlichen Ausprägung des R². Dabei ist zu berücksichtigen, dass der RandomForest über die Kreuzvalidierung und Rastersuche optimiert wurden. Die Optimierung des neuronalen Netzes ist in dieser Arbeit nicht erfolgt. Trotzdem können sehr hohe R² Werte mit dem neuronalen Netz erreicht werden, die sogar die des optimierten RandomForest übersteigen.

Da Ausreißer in den Daten vorhanden sind, ist das MAE bei der Bewertung der beiden Verfahren vorzuziehen. Ein Vergleich der Verfahren kann neben dem R² auch über den MAE erfolgen. Die Fehlerwerte sind die für jeweiligen Verfahren und Modelle auf Abbildung 83 dargestellt.

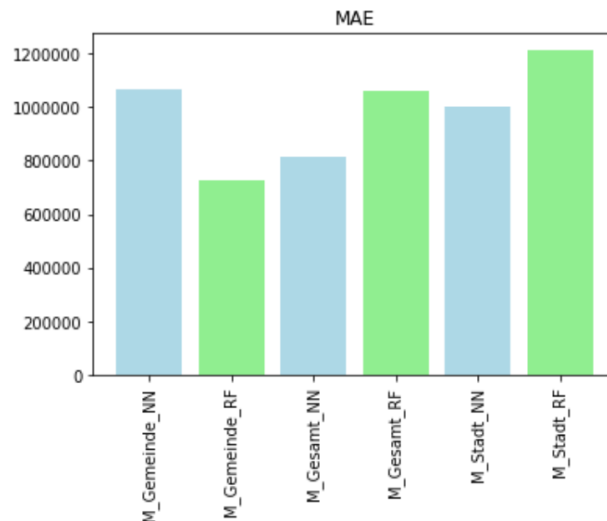


Abbildung 83 Visuelle Darstellungen des MAE der Modelle beider Verfahren

Das neuronale Netz (blaue Balken) kann bei einer regionsunabhängigen Betrachtung (M_Gesamt) bessere Ergebnisse erzielen. Wird der Umsatz dagegen für Städte separat (M_Stadt) geschätzt werden soll, erzielt das neuronale Netz (blaue Balken) zu genauere Vorhersagen. Hinsichtlich des Gemeindemodells kann das RandomForest Verfahren besser abschneiden.

Wie angekungen, besteht bei der Verwendung des neuronalen Netzes die Gefahr der Überanpassung. Die visuelle Gegenüberstellung der Approximtion der wahren Umsatzverteilung aus den Testdaten, ist dabei sehr aufschlussreich. Stellvertretend soll dies für M_Gesamt für beide Verfahren gezeigt werden. Das neuronale Netz (Abbildung 84) kann den Umsatz im umsatzschwachen sowie umsatzstarken Bereich viel genauer Vorhersagen als der RandomForest (Abbildung 85).

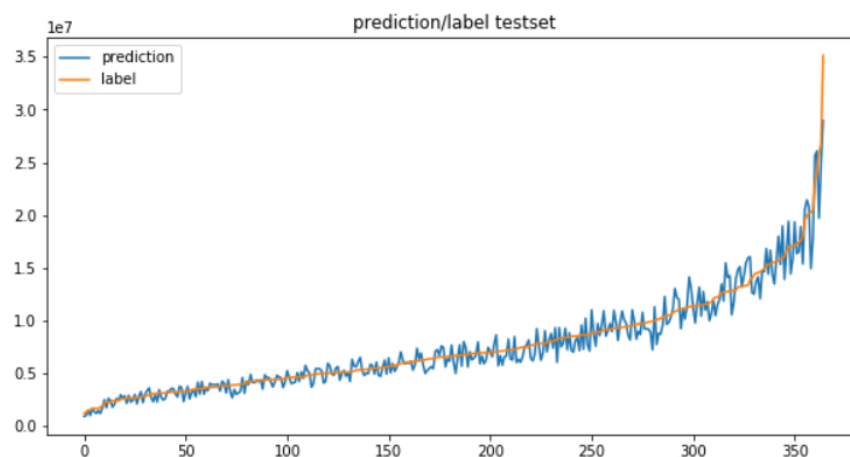


Abbildung 84 Darstellung der Abweichung von wahren und prognostizierten Umsatzwert über das Neuronales Netz.

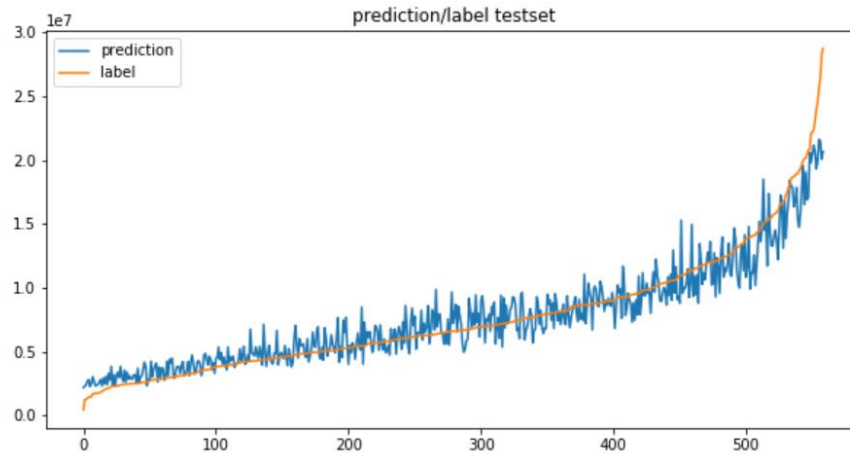


Abbildung 85 Darstellung der Abweichung von wahren und prognostizierten Umsatzwert über das RandomForest Verfahren.

Zusammenfassend kann behauptet werden, dass die Vorhersage des RandomForest „gröber“ ist als das des neuronalen Netzes. Eine Bewertung, ob ein Verfahren dem anderen vorzuziehen ist, kann nicht vorgenommen, da nicht festgestellt werden kann, welches Verfahren zu realitätsnäheren Umsatzschätzungen führt.

Aufbauend auf den vorliegenden Ergebnissen, können beide Verfahren iterativ verfeinert werden. Dabei könnten weitere Untersuchungen der Daten vorgenommen oder das Feature Set angepasst bzw. erweitert werden.

5. Fazit

Mit Hilfe des Anwendungsfalles der Filialnetzoptimierung eines Einzelhändlers wurde gezeigt, dass die Verknüpfung von Open Data mit unternehmensinternen Daten zu besseren Entscheidungen führen kann. Open Data sind frei verfügbare Daten aus verschiedensten Themenbereichen, wie Bevölkerung, Infrastruktur und Wirtschaft, die für die Umsatzschätzung verwendet wurden.

Schlussbetrachtung

Es konnte gezeigt werden, dass viele Eigenschaften von Big Data auch auf Open Data zutreffen. Darunter fällt die Verwendung von heterogenen Formaten unterschiedlicher Strukturierung, uneinheitliche Datenqualität sowie eine stetig wachsende Datenmenge.

Es musste festgestellt werden, dass Data Warehouses für die Speicherung und Verarbeitung von Open Data nur bedingt geeignet sind, da diese traditionellen Systeme nicht flexibel genug sind und eine Erweiterung oft mit hohen Kosten verbunden ist.

Data Lakes stellen eine alternative Form der Datenspeicherung dar, die in dieser Arbeit untersucht wurden. In der Literatur finden sich unterschiedliche Definitionen zum Begriff Data Lake, die sich teilweise überschneiden. In dieser Arbeit wurden die Definitionen zusammengeführt und ein einheitliches Konzept für die Speicherung und Verarbeitung von Open Data vorgestellt. In dieser Architektur wurden Schichten definiert, um Daten und Verarbeitungsschritte zu organisieren.

Die Rohdaten werden zunächst in der Landing Zone abgelegt und mit Metadaten beschrieben. Nach einer Überführung in ein strukturiertes Datenformat werden die Daten in der Raw Schicht gespeichert, um darauf aufbauend Bereinigungs- und Anreicherungsprozesse in der Curated Zone durchzuführen.

Die aufbereitete Datenbasis kann für anwendungsspezifische Zwecke verwendet werden, die in Unterbereichen dieser Zone organisiert werden. In diesen Unterbereichen können beispielsweise fehlende Werte behandelt oder Datenbestände verknüpft werden, da diese Verarbeitungsschritte auf die genauen Anforderungen der Anwendungen ausgerichtet sind. Zusätzlich wird eine gesonderte Working Zone für Machine Learning definiert, um verfahrensspezifische Vorverarbeitungen durchzuführen, z.B. die Auswahl und Umformung der Merkmale, die Definition und Anpassung des Modells und die Durchführung der Prognose.

Für explorative Analysen gibt es einen schichtunabhängigen Sandbox Bereich. Hier können Verknüpfungen von Datenbeständen geprüft werden, um so beispielweise neue Anwendungsszenarien zu erzeugen. Außerdem kann hier getestet werden inwieweit weitere Datenquellen eine bestehende Anwendung ergänzen.

Der Data Lake konnte mit einem Linux System realisiert werden, auf dem Hadoop, Spark, Jupyter und verschiedene Python Bibliotheken installiert wurden. Für die Speicherung der Daten wurde eine Kombination aus lokalem Dateisystem und dem verteilten Dateisystem von Hadoop (HDFS) gewählt. Die Erstablage der Rohdaten in der Landing Zone wurde auf dem lokalen Dateisystem durchgeführt, da ein Teil der offenen Daten mit zusätzlichen Programmen wie Excel verarbeitet werden mussten. Die Daten aus allen anderen Schichten wurden auf dem HDFS verteilt abgelegt, damit die Speicherkapazität bei Bedarf skaliert werden kann. Alle definierten Verarbeitungsprozesse konnten mit Python Programmbibliotheken und Apache Spark umgesetzt werden. Da die Entwicklungsumgebung Jupyter sich mit den verwendeten Komponenten integrieren lässt, konnten jegliche Aufgaben hierüber erledigt werden. Dazu zählt das Schreiben von Python Code und die Steuerung des Clusters.

Für die Analyse von Open Data und unternehmensinternen Daten mussten diese zunächst verknüpft werden. Das Vereinen von Daten von öffentlichen Institutionen wie dem Zensus, Bundesamt für Statistik oder Geodäsie, konnte weitestgehend über den Regionalschlüssel durchgeführt werden. Generell ist das Verknüpfen von Datenbeständen ohne einheitlichen Schlüssel schwierig oder gar nicht möglich. Sofern Daten von unterschiedlichen Bereitstellern verwendet werden, ist dies in der Regel der Fall, weswegen viele Verknüpfungswege getestet werden müssen. Für diesen Zweck bietet sich die Sandbox Zone im Data Lake an. Eine Zusammenführung von Daten aus anderen Quellen, wie von OpenStreetMap oder unternehmensinternen Daten, wurde über eine geografische Verknüpfung gelöst. Es stellte sich heraus, dass die geografische Verknüpfung sehr rechenintensiv ist und über eine Python Funktion und Pandas etwa 50 Tage dauern würde. Dies stellte einen geeigneten Fall dar, um das verteilte System von Spark zu verwenden. Mit Spark konnte die Verarbeitungsdauer auf 7 Stunden reduziert werden. Es war jedoch notwendig, die Funktionen und Methoden (Shared Variables, Partitionierung und Datenformat) verstanden zu haben, um diese auch anwenden zu können. An dieser Stelle ist zu erwähnen, dass Spark weitere Optimierungsmöglichkeiten bietet, die mit der Architektur des Clusters zu tun haben. Da dies ein tiefergehendes Verständnis über die

technischen Eigenschaften des Clusters voraussetzt, wurde diese Optimierung nicht durchgeführt.

Außerdem musste festgestellt werden, dass komplexe Operationen mit Spark nicht effizient umzusetzen sind. Dies liegt daran, dass bei einer Transformation in Spark nur auf Spalten eines Datentyps zugegriffen werden kann und das Ergebnis der Verarbeitung lediglich in eine neue Spalte überführt wird. Außerdem werden die Transformationen erst bei Verwendung von lesenden Operationen ausgeführt. Somit werden Fehler erst am Ende der Verarbeitungskette ausgegeben und müssen zunächst nachvollzogen werden. Fehlermeldung somit erst am Ende der Verarbeitungskette ausgegeben. Daher die Verarbeitung mit mehreren Spalten und Datentypen über Pandas durchgeführt.

Die Analyse der offenen Daten hat sich als sehr zeitintensiv herausgestellt, da die Eigenschaften einzelner Merkmale und generelle Zusammenhänge untersucht werden mussten. Ein kleiner Ausschnitt aus der Untersuchung wurde in dieser Arbeit präsentiert. Dabei wurde gezeigt, dass die Daten unausgewogen verteilt und untereinander hochgradig korreliert sind. Durch die Ausreißer Problematik konnten die Eigenschaften einzelner Merkmale nur schwer erfasst werden. Dabei konnten die Ausreißer nicht mit üblichen Methoden zur Ausreißererkennung behandelt werden, da unter den Ausreißern Datensätze waren, die für den Anwendungsfall relevant sind. Es wurde eine Lösung gefunden, die sich an den betrachteten Anwendungsfall orientiert. Somit konnte die Datenmenge eingegrenzt werden.

Bei der großen Anzahl an Merkmalen war die Suche nach geeigneten Merkmalen für eine Analyse aufwendig, da herkömmliche Verfahren zur Feature Selection nicht angewendet werden konnten. Grund dafür ist die hohe Abhängigkeit, die unter den Merkmalen besteht, sodass zu viele Merkmale als relevant beurteilt wurden. Daher wurden viele Merkmale einzeln analysiert und eine anwendungsspezifische Lösung gefunden werden.

Dagegen finden sich viele Beispiele wie die Vorverarbeitung und Anwendung von Machine Learning Verfahren stattfinden sollte. Daher konnte die Durchführung der Verfahren schnell umgesetzt werden. Die Ergebnisse zu interpretieren und die Verfahren zu optimieren ist dagegen eine schwere Aufgabe, da eine genaue Auseinandersetzung mit den Konzepten der Verfahren notwendig ist. Eine nicht triviale Frage ist beispielsweise, ab wann ein Modell gute Ergebnisse liefert. Die Betrachtung der numerischen Fehlermaße war nicht ausreichend, da diese im Anwendungsfall zu große Wertebereiche aufweisen.

Daher mussten eigene Visualisierungen erstellt werden, in denen die Verteilung von Prognosewert und wahren Wert verglichen wurde. Erst danach konnte die Prognosequalität des Random Forests und des neuronalen Netzes miteinander verglichen werden.

Kritische Auseinandersetzung

Ein Data Lake erfordert eine gute Organisationsstruktur der Daten, die über viele Standards definiert werden muss. In dieser Arbeit wurden nur einige Probleme vorgestellt. Das hier vorgestellte Architektur ist dafür das organisatorische Grundgerüst. Zusätzlich sind Werkzeuge erforderlich für Metadaten Management, Suche von Datenbeständen, und anderen Data Governance Themen, die in dieser Arbeit nicht vertieft wurden. Diese sind bei einem Betrieb eines Data Lakes mit größeren Datenmengen und vielen Anwendern essentiell, da ansonsten eine Data Swamp Problematik entsteht.

Der betrachtete Anwendungsfall der Filialnetzoptimierung müsste erweitert werden, um in der Praxis verwendet zu werden. Der Prozess zur Begründung eines neuen Filialstandorts hängt von weiteren Faktoren ab. Es werden Verkaufsflächen von Gemeinden und Städte ausgeschrieben, worauf sich ein Einzelhandler bewerben kann. Das in dieser Arbeit vorgestellte Vorgehen kann in der Praxis dafür benutzt werden, um bereits ausgeschriebene Standorte zu bewerten. Es stellt bei einer Entscheidung für einen neuen Standort einen wichtigen Indikator dar.

Ausblick

Es wurde gezeigt, dass der Data Lake eine alternatives zum DWH Konzept aufweist. Dabei musste sowohl für DWH als auch für den Data Lake Stärken und Schwächen erkannt werden. Eine Fortsetzung der Thesis könnte über eine Betrachtung eines parallelen Betriebes von DWH und Data Lake erfolgen. Dabei würden so die Schwächen des einen Konzeptes durch die Stärken des anderen kompensiert. Ob und inwieweit dies Sinn macht, müsste genauer analysiert und eine technische Umsetzung übergeprüft werden.

Das Open Data Thema wird durch verschiedene Initiativen und Gesetzesentwürfe auf nationaler und internationaler Ebene vorangetrieben. Es wurden in dieser Arbeit nur wenige Datenquellen identifiziert, die aus dem öffentlichen Sektor stammen. Es wäre daher eine systematische Auseinandersetzung mit den im Open Data Bereich vorhandenen Datenquellen interessant, damit das volle Potenzial erkannt werden kann.

Literaturverzeichnis

- Alserafi, A., Abello, A., Romero, O., & Calders, T. (12. 12 2016). Towards Information Profiling: Data Lake Content Metadata Management. *2016 IEEE 16th International Conference*, S. 178-185. doi:10.1109/ICDMW.2016.0033
- Baack, S. (2013). *Die Open-Data-Bewegung : Das Verhältnis von Praktiken, Zielen und Selbstbild der Open Knowledge Foundation Deutschland*. Abgerufen am 14. 12 2018 von <http://nbn-resolving.de/urn:nbn:de:0168-ssoar-363745>
- Brell, C. (2017). *Statistik von Null auf Hundert*. Berlin: Springer. doi:10.1007/978-3-662-53632-2
- Coiera, E. (Oktober 2014). Embedding AI and Crowdsourcing in the Big Data Lake. *Journal of medical Internet research*, S. 70-73. doi:1541-1672/14/
- Domingos, P. (2017). A Few Useful Things to Know about Machine Learning. *University of Washington*, 9.
- Fasel, D., & Meier, A. (2016). *Big Data Grundlagen, Systeme und Nutzungspotenziale*. n Wiesbaden: Springer Fachmedien. doi:10.1007/978-3-658-11589-0
- Freytag, C. (2016). Problems, Methods, and Challenges in Comprehensive Data Cleansing. *Humboldt-Universität zu Berlin*, 23.
- Gadatsch, A., & Landrock, H. (2017). *Big Data für Entscheider*. Wiesbaden: Springer Fachmedien Wiesbaden. doi:10.1007/978-3-658-17340-1
- Géron, A. (2018). *Machine Learning mit Scikit-Learn & Tensorflow*. Heidelberg: O'Reilly.
- Gluchowski, P. (2016). Externe Daten ohne Zusatzkosten nutzen. *BI Spektrum*.
- Grover, M. (2015). *Hadoop Application Architectures*. Sebastopol: O'Reilly.
- Grulich, P. (2016). *Skalierbare Echtzeitverarbeitung mit Spark Streaming: Realisierung und Konzeption eines Car In-formation Systems*. Hamburg: Hochschule für angewandte Wissenschaften Hamburg.
- Guyon, IBM Research,
24.
- Janssen, M., Charalabidis, Y., & Zuiderwijk, A. (2012). Benefits, Adoption Barriers and Myths of Open Data and Open Government. *Information Systems Management (ISM)*, S. 258-268.
- Kimball, R., & Ross, M. (2013). *Data Warehouse Toolkit, The Definitive Guide to Dimensional Modeling*. Indiana: Wiley. doi:978-1-118-53080-1
- King, S. (2014). *Big Data Potential und Barrieren der Nutzung im Unternehmenskontext*. Wiesbaden: Springer Fachmedien. doi:10.1007/978-3-658-06586-7
- LaPlante, A., & Sharma, B. (2016). *Architecting Data Lakes, Data Management Architectures for Advanced Business Use Cases*. -1-491-95257-3
- LeCun, Y., & Bottou, L. (1998). Efficient BackProp. *Springer*, 44.
- M., W., & S., S. (2017). *Machine Learning in Data Lake for Combining Data Silos*. Cham: Springer. doi:<https://doi.org/10.1007>
- Maletic, J., & Marcus, A. (2000). Data Cleansing: Beyond Integrity Analysis. *Division of Computer Science Memphis*, 10.
- Marz, N. (2016). *Entwicklung und Programmierung von Systemen für große Datenmengen und Einsatz der Lambda-Architektur* (Bd. mitp Professional). Frechen: MITP. doi:9783958451759
- Meinert, A. (2018). Open Data – Herausforderungen und Chancen für die Wirtschaftsinformatik. *Seminararbeit Westfälische Hochschule*, 50.

- Miloslavskaya, N., & Tolstoy, A. (2016). Big Data, Fast Data and Data Lake Concepts. *Procedia Computer Science*, 300-3006.
- Raschka, S. (2017). *Machine Learning mit Python und Scikit-learn und TensorFlow*. mitp.
- Termer, F. (2017). *Open Data Neue Konzepte erfolgreich umsetzen*. Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V. Bitkom.
- Terrizzano, I. (2015). Data Wrangling: The Challenging Journey from the Wild to the Lake. *IBM Research*.
- VanderPlas, J. (2018). *Data Science mit Python*. Frechen: mitp.

Anhang

A. 1 Beigelegte CD

A. 2 Programmcode zur geografischen Verknüpfung der Filialstandorte mit Regionsumrissen

```
for region in regionen.itertuples():

    region_umriss = konvertiere_string_zu_geo_objekt(region.valid_geojson)

    filialen_liste = []

    for filiale in filialen.itertuples():
        #filiale --> tuple: (index, (ID, lat, lon))
        filiale_koordinaten = Point(filiale.longitude, filiale.latitude)

        if(region_umriss.contains(filiale_koordinaten)):
            #print("match: " + region.RS + " und " + str(branch_location[1]["ID"]))
            filialen_liste.append(int(filiale.ID))
            filiale_region_matchingliste.append({"filiale_id" : filiale.ID, "region_rs" : region.RS})

    if(len(filialen_liste)>0):
        print(len(filialen_liste))
        region_filiale_matchingliste.append({"AGS" : region.AGS, "RS" : region.RS, "RS_ALT" : region.RS_ALT,
                                             "filialen_ids" : filialen_liste, "anzahl_filialen" : len(filialen_liste)})

def summiere_filalumsaetze(filialen_ids):

    summierter_umsatz = 0

    if (type(filialen_ids) == list):

        for filiale_id in filialen_ids:

            filialumsatz = filialen[filialen["ID"] == filiale_id]["Jahresumsatz"].values[0]
            summierter_umsatz = summierter_umsatz + filialumsatz

    else:
        summierter_umsatz = None

    return summierter_umsatz

join["gesamtumsatz"] = join["filialen_ids"].apply(lambda x: summiere_filalumsaetze(x))

join["durchschnittlicher_umsatz"] = join["gesamtumsatz"]/join["anzahl_filialen"]
```

```

from shapely.geometry import Point, Polygon, MultiPolygon, shape
def konvertiere_string_zu_geo_objekt(geo_objekt_str):
    """ Konvertiert einen String mit einem Geo-Objekt in eine geometrische Repräsentation in Python (Polygon, Point, Multipolygon) """
    # Hier konvertiert eval() eine String-Respräsentation einer List in eine Liste

    # eval() ist eine builtin-Funktion und wertet einen String aus und interpretiert diesen in bekannte Typen
    # Hier wird der String in die komplexe Listenstruktur des GeoJSON überführt
    geo_objekt = eval(geo_objekt_str)

    if(geo_objekt["type"] == "Polygon"):

        geo_objekt_koordinaten = geo_objekt["coordinates"][0]

        neues_polygon = []

        for punkt in geo_objekt_koordinaten:

            neues_polygon.append(tuple(punkt))

        validates_geo_objekt = Polygon(neues_polygon)

    elif(geo_objekt["type"] == "MultiPolygon"):

        multipolygon_koordinaten = geo_objekt["coordinates"]

        neues_polygon = []

        for polygon in multipolygon_koordinaten:

            neues_polygon_punkte = []

            for point in polygon[0]:

                neues_polygon_punkte.append(tuple(point))

            neues_polygon.append(Polygon(neues_polygon_punkte))

        validates_geo_objekt = MultiPolygon(neues_polygon)
    else:
        print("feature type nicht erkannt: " + geo_objekt)

    return validates_geo_objekt

```


A. 3 Programmcode zur geografischen Verknüpfung der Filialstandorte mit Regionsumrissen

```
## Regionsumrisse in Hauptspeicher laden
regionen.cache()

def nodes_zaeher(valid_geojson, nodes_liste):
    """Funktion zum Zählen von Infrastrukturpunkten (nodes) in einer Region"""

    regionsumriss = konvertiere_string_zu_geo_objekt(valid_geojson)
    zaehler = 0

    for node in nodes_liste:
        if(regionsumriss.contains(node)):
            zaehler = zaehler + 1

    return zaehler

## udf Funktion bilden
nodes_zaeher_udf = udf(nodes_zaeher, IntegerType())

## Liste mit Punkten broadcasten sodass diese jedem Worker zur Verfügung stehen
spark.broadcast(osm_punkte_liste)

from shapely.geometry import Point, Polygon, MultiPolygon, shape
def konvertiere_string_zu_geo_objekt(geo_objekt_str):
    """ Konvertiert einen String mit einem Geo-Objekt in eine geometrische Repräsentation in Python (Polygon, Point, Multipolygon) """
    # Hier konvertiert eval() eine String-Respräsentation einer List in eine Liste

    # eval() ist eine builtin-Funktion und wertet einen String aus und interpretiert diesen in bekannte Typen
    # Hier wird der String in die komplexe Listenstruktur des GeoJSON überführt
    geo_objekt = eval(geo_objekt_str)

    if(geo_objekt["type"] == "Polygon"):

        geo_objekt_koordinaten = geo_objekt["coordinates"][0]

        neues_polygon = []

        for punkt in geo_objekt_koordinaten:

            neues_polygon.append(tuple(punkt))

        valides_geo_objekt = Polygon(neues_polygon)

    elif(geo_objekt["type"] == "MultiPolygon"):

        multipolygon_koordinaten = geo_objekt["coordinates"]

        neues_polygon = []

        for polygon in multipolygon_koordinaten:

            neues_polygon_punkte = []

            for point in polygon[0]:

                neues_polygon_punkte.append(tuple(point))

            neues_polygon.append(Polygon(neues_polygon_punkte))

        valides_geo_objekt = MultiPolygon(neues_polygon)
    else:
        print("feature type nicht erkannt: " + geo_objekt)

    return valides_geo_objekt
```

A. 4 Programmcode zur hierarchischen Schätzung von Regionsmerkmalen

```
## AGS_12 = RS
look_up_tabelle = daten_ohne_fehlenden_werten["AGS_12"].unique()
##über RS iterieren die null-Werte einhalten
for RS in daten_mit_fehlenden_werten["AGS_12"].unique():

    #Für jedes Objekt, fülle spaltenweise die missing-values
    for spalte in spalten_mit_fehlenden_werten:
        #print(RS, col)
        #Falls eine Spalte missing values enthält, dann schau im Look-up-table nach Referenzwerten
        if (daten_mit_fehlenden_werten.loc[daten_mit_fehlenden_werten["AGS_12"] == RS, spalte].isnull().iloc[0]):
            #Befülle die Spalte im Dataframe mit den Werten aus dem hierarchisch Höheren Gebiet
            vorhandene_werte_hoherer_hierarchie = [wert for wert in daten_ohne_fehlenden_werten.loc[daten_ohne_fehlenden_werten["AGS_12"] == RS, spalte].values]
            daten_mit_fehlenden_werten.loc[daten_mit_fehlenden_werten["AGS_12"] == RS, spalte] = vorhandene_werte_hoherer_hierarchie

# Fort
```



```
def finde_naechsthoeheren_RS(RS, look_up_tabelle):
    """ Suche und Rückgabe nach einer nächsthöheren Region in einem Look up Datensatz """
    if (RS in look_up_tabelle):
        naechsthoeherer_RS = RS
    else:
        naechsthoeherer_RS = finde_naechsthoeheren_RS(erzeuge_naechsthoeheren_RS(RS), look_up_tabelle)

    return naechsthoeherer_RS

def erzeuge_naechsthoeheren_RS(RS):
    """ Erzeugt den Regionschlüssel der nächsthöheren Hierarchieebene """

    ## String des RS in seine Bestandteile aufteilen
    RS_Land, RS_Reg, RS_Kreis, RS_VG, RS_Gem = RS[:2], RS[2:3], RS[3:5], RS[5:9], RS[9:]

    if (RS_Gem > "000"):
        RS_Gem = "000"

    elif (RS_VG > "0000"):
        RS_VG = "0000"

    elif (RS_Kreis > "00"):
        RS_Kreis = "00"

    elif (RS_Reg > "0"):
        RS_Reg = "0"

    elif (RS_Land > "00"):
        RS_Land = "00"

    else:
        print("Die höchste Hierarchieebene ist erreicht")

    return RS_Land + RS_Reg + RS_Kreis + RS_VG + RS_Gem
```

A. 5 Programmcode zur Zeitreihenschätzung von Regionsmerkmalen

```
## Damit über die Interpolation die Zeitreihe in der richtigen Folge ist (von Jahr 2010 bis 2015),
## müssen die Daten zunächst nach Jahr sortiert werden
daten_sortiert = daten.sort_values(by=["RS", "JAHR"], ascending=[True, True]).reset_index(drop=True)
daten_sortiert[["JAHR", "RS"]].head(20)

def schaeetze_werte(zeitreihe):
    """ Schätzt die Werte einer Zeitreihe """

    try: #pchip besseres monotonie-verhalten als spline
        zeitreihe = zeitreihe.interpolate(method="spline", order=2, limit_direction="both")
    except: #Falls zu wenig Inputvariablen sind, dann linear schätzen
        zeitreihe = zeitreihe.interpolate(method="linear", limit_direction="both")

    return zeitreihe

## Die Zeitreihe kann nach der Sortierung mit einer Gruppierung nach Region erfolgen. Auf den Gruppen kann
## kann das die Schätzfunktion angewendet werden

daten_fehlende_werte_geschaezt = daten_sortiert.groupby("RS").apply(lambda zeitreihe: schaeetze_werte(zeitreihe))
```

A. 6 Programmcode zur Schätzung des Umsatzes, Verkaufsfläche und Mitarbeiteranzahl der Filialen

```
# Umsatzschätzung
#Beispiel: https://de.statista.com/statistik/daten/studie/365021/umfrage/jahresumsatz-pro-filiale-der-drogeriemaerkte-rossmann-in-deutschland/

"""
# Infrastruktur-Merkmale
region["osm_nodes_counter"], region["ANTEIL_SIEDLUNG_VERKEHR"], region["BEV_JE_QM"], region["BESIEDLUNGSSCHLUESSEL"]
region["GEB_6_4"] # Gebäude mit Anzahl Wohnungen 3-6

# Bevölkerungs-Merkmale
region["HH_1_4"] # Haushalte mit Paare und Kindern , region["DEM_2_7"] # Anzahl verheiratete Personen insgesamt, region["EINK_JE_EINW"] region["ANTEIL_ARBEITSLOSE"]
"""

def schaeetze_umsatz(region):

    infrastruktur = (int(region["GEB_6_4"]) + int(region["osm_nodes_counter"])*4 + float(region["ANTEIL_SIEDLUNG_VERKEHR"])*2 + float(region["BEV_JE_QM"])*5)
    bevoelkerung = int(region["HH_1_4"])*5 + int(region["DEM_2_7"])*2 + float(region["EINK_JE_EINW"])*10 - (float(region["ANTEIL_ARBEITSLOSE"])* int(region["BEV_INSGE"]
    wert = infrastruktur * bevoelkerung * np.random.randint(9, 15)

    ## Umsatz auf einen Wertebereich von 400.000 bis 3 Milliarden skalieren
    umsatz = (wert / np.log2(value)**5 ) * np.random.randint(900000, 1500000)

    return round(revenue, 2)

def schaeetze_verkaufsflaeche(umsatz):

    ## Verkaufsfläche auf 200 bis 1000 skalieren
    verkaufsflaeche = ((float(umsatz) / np.random.randint(5000, 7000))**0.5 ) + np.random.randint(50, 120)

    return int(verkaufsflaeche)

def schaeetze_anzahl_mitarbeiter(umsatz):

    ## Anzahl Mitarbeiter auf 15 bis 300 skalieren
    anzahl_mitarbeiter = (float(umsatz) / np.random.randint(50000, 70000))**0.4 + np.random.randint(12, 21)

    return int(anzahl_mitarbeiter)
```

```

for filiale in df_comp.itertuples():

    filialen_angereichert = {}

    filialstandort = df_full_data[df_full_data["AGS_12"] == filiale.region_fuer_schaetzung]

    if(len(filialstandort) == 0):
        unmatched_filiale.append(filiale)
        continue

    filiale_umsatz = schaeetze_umsatz(branch_region)
    verkaufsflaeche, anzahl_mitarbeiter = schaeetze_verkaufsflaeche(filiale_umsatz), schaeetze_anzahl_mitarbeiter(filiale_umsatz)

    filiale_neu = {"ID": filiale.branch_id,
                   "Strassenname": filiale.Strassenname,
                   "PLZ": filiale.PLZ,
                   "Region": filiale.Region,
                   "revenue": filiale,
                   "sales_area": verkaufsflaeche,
                   "number_of_employees": anzahl_mitarbeiter,
                   "region_counter": filiale.region_counter,
                   "region_list": filiale.region_list,
                   "gemeinde_rs": filiale.gemeinde_rs,
                   "kreis_stadt_rs": filiale.kreis_stadt_rs,
                   "krfr_stadt_rs": filiale.krfr_stadt_rs,
                   "region_rs_for_estimation": filiale.region_fuer_schaetzung
                  }

    filialen_angereichert.append(filiale_neu)

```