Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Yannic Wilkening

Fahrzeuge als Sensorknoten für Big Data Cloud-Anwendungen am Beispiel Parkraumvorhersage

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Yannic Wilkening

Fahrzeuge als Sensorknoten für Big Data Cloud-Anwendungen am Beispiel Parkraumvorhersage

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Franz Korf
Zweitgutachter: Prof. Dr. Kai von Luck

Eingereicht am: 25. April 2017

Yannic Wilkening

Thema der Arbeit

Fahrzeuge als Sensorknoten für Big Data Cloud-Anwendungen am Beispiel Parkraumvorhersage

Stichworte

Parkraumanalyse, Fahrzeugsensorik, Cloud-Plattform, Webanwendung, Big Data, Data-Mining, Prognose

Kurzzusammenfassung

Diese Arbeit zeigt eine mögliche Lösung für das Problem der zeitaufwendigen Parkplatzsuche in Großstädten auf. Anhand von Fahrzeugsensoren als Datenbasis wird eine Big Data Cloud-Plattform zur Erfassung, Aufbereitung sowie Visualisierung von Parkraumbelegungsständen entwickelt und evaluiert. Durch Data-Mining Verfahren werden Modelle, zur Vorhersage der Parkraumauslastung, erzeugt und ausgewertet.

Title of the paper

Cars as sensor nodes for Big Data Cloud Services exemplified by predictive analysis of parking space

Keywords

Parking Space Analysis, Automotive Sensors, Cloud Services, Web Application, Big Data, Data-Mining, Predictive Analytics

Abstract

This paper introduces a possible solution to reduce the time consuming process of finding an unoccupied place to park in big Cities. Vehicular sensor data is used to develop and evaluate a big data cloud platform for sensor data collection, analysis and visualisation of the parking space occupancy. By use of the cloud platform, data mining processes are performed to develop and evaluate models for a parking space prediction.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einführung

Die fortschreitende Digitalisierung der Welt bestimmt zunehmend unser alltägliches Leben. Auch im Bereich der Automobilindustrie spielt die Vernetzung von Fahrzeugen und Infrastruktur eine immer größere Rolle. Laut Aussage des Verbandes der Automobilindustrie ist die Vernetzung von Fahrzeugen sowie die Digitalisierung rund um das Automobil Grundlage für ein leistungsfähiges Verkehrssystem (VDA). Auf der anderen Seite rückt die Gewinnung, Erfassung und Auswertung von großen Datenmengen, in sogenannten Big Data Anwendungen, immer weiter in den Vordergrund. Durch die Erhebung großer Datenmengen lassen sich anhand spezieller Analyseverfahren bessere Vorhersagen treffen als jemals zuvor (McAfee u. a. (2012)). Viele Städte, wie auch Hamburg, streben eine Digitalisierung ihrer Stadt an. Ziele der Digitalisierung sind unter anderem eine Steigerung der Servicequalität und eine Verbesserung des städtischen Lebens, durch Projekte wie eine digitale Verwaltung, intelligente Verkehrssysteme oder digitale Geodaten (von Vogel). Vor diesem Hintergrund werden im Rahmen dieser Bachelorarbeit die Erhebung und Auswertung von Sensor Daten, aus intelligenten Fahrzeugen, am Beispiel einer Big Data Cloud-Anwendung zur Analyse und Vorhersage der aktuellen Parkraumsituation in Hamburg erprobt. Dazu wird eine Cloud-Plattform (im folgenden auch als *SmartParking* bezeichnet) entwickelt, die mit Hilfe von Messfahrzeugen, den ruhenden Verkehr analysiert und den aktuellen Parkraumbelegungsstatus, für Endanwender, verfügbar macht. Abschließend werden Prognoseverfahren zur Vorhersage von freiem Parkraum evaluiert.

Aufbau der Arbeit

In Kapitel 2 wird auf den aktuellen Stand intelligenter Fahrzeuge sowie das Thema Fahrzeugvernetzung eingegangen. Die Schwerpunkte werden dabei auf die aktuelle Fahrzeugsensorik, die Kommunikationsstrukturen zwischen Fahrzeugen und die Kommunikation zwischen Fahrzeugen mit der Infrastruktur gelegt.
Kapitel 3 gibt eine Einführung in die Erhebung großer Datenmengen im Rahmen von Big Data Cloud-Anwendungen. Des Weiteren wird auf den Prozess des Data-Mining, zur Gewinnung von nutzbarem Wissen aus großen Datenmengen, eingegangen und ausgewählte Klassifikationsalgorithmen näher betrachtet.
In Kapitel 4 wird die Entwicklung und Umsetzung einer Cloud-Server Plattform zur Erfassung und Analyse der Parkraumbelegung, mit Hilfe von Messfahrzeugen, beschrieben. Es wird auf die dabei entstandene Architektur eingegangen und der Umgang mit externen Datenquellen veranschaulicht. Die Entwicklung und Umsetzung einer Endnutzeranwendung zur Übersicht über den aktuellen Parkraumbelegungsstatus in Hamburg, auf Basis der entwickelten Cloud-Server Plattform, schließt das Kapitel ab.
Zur Nutzung der durch die Cloud-Server Plattform erhobenen Daten, wird in Kapitel 5 der Prozess zur Analyse und Prognose von Parkraumbelegungsständen, anhand des Data-Mining Prozesses, beschrieben. Es werden entstandene Modelle, ausgewählter Klassifikationsalgorithmen, verglichen und abschließend bewertet.

2 Stand der Technik

Der Trend zur Vernetzung bzw. Car-IT schreitet immer weiter voran und ist eines der Zukunftsthemen der Automobilindustrie. Bereits heute erhält das Internet und die Vernetzung von Fahrzeugen durch Systeme wie VW Car-net[1], Audi Connect[2] oder BMW ConnectedDrive[3] Einzug in die Automobilwelt (Abbildung 2.1).

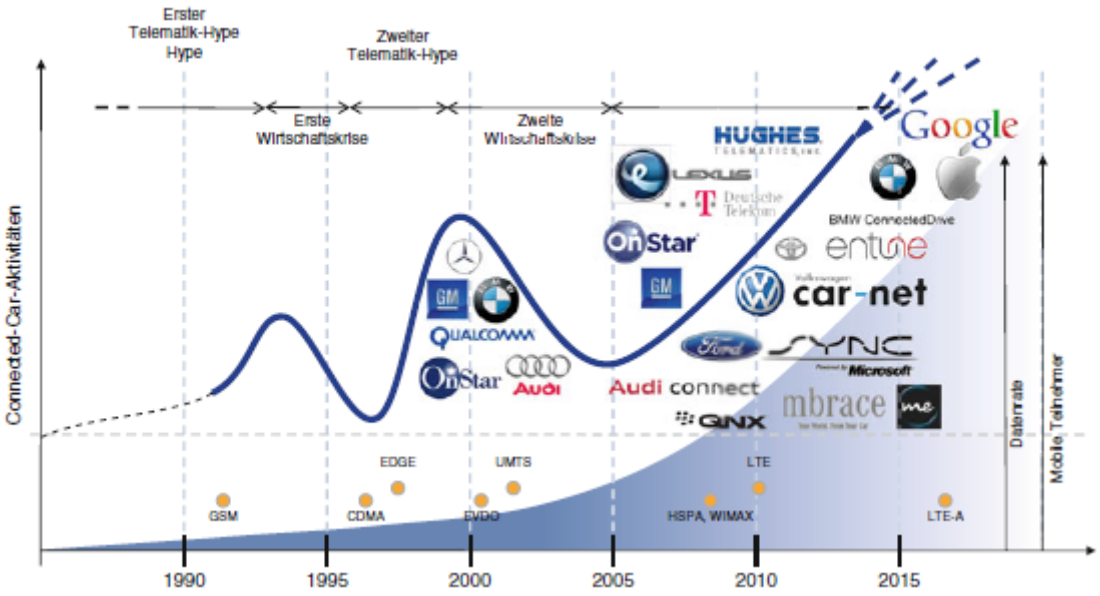


Abbildung 2.1: Trend vernetzter Fahrzeuge (Johanning und Mildner (2015))

Diese Systeme ermöglichen es dem Fahrzeugnutzer beispielsweise Verkehrsinformationen abzurufen, automatische Notrufe abzusetzen, POIs[4] zu finden, online Entertainment zu nutzen

[1] http://volkswagen-carnet.com/de/de/start.html
[2] https://www.audi.de/de/brand/de/kundenbereich/connect.html
[3] https://www.bmw-connecteddrive.de/
[4] Points of interest

oder remote Funktionen des Fahrzeuges per App[5] zu bedienen. Dienste wie BMW ParkNow[6] ermöglichen bereits das Finden von vorhandenen Parkflächen in der Umgebung sowie das online Bezahlen von Parkplatzgebühren (Johanning und Mildner (2015)).

2.1 Sensorik aktueller Fahrzeuge

Aktuelle Fahrzeuge verfügen über ein breites Spektrum verschiedener Sensoren, um Fahrzeug- sowie Komfortfunktionen zu ermöglichen. Dazu zählen Helligkeitssensoren, Temperatursensoren, Ultraschallsensoren, Lasersensoren, Radarsensoren, Kameras, GPS[7] und unzählige weitere Sensoren, die für den Fahrbetrieb erforderlich sind (Abbildung 2.2).

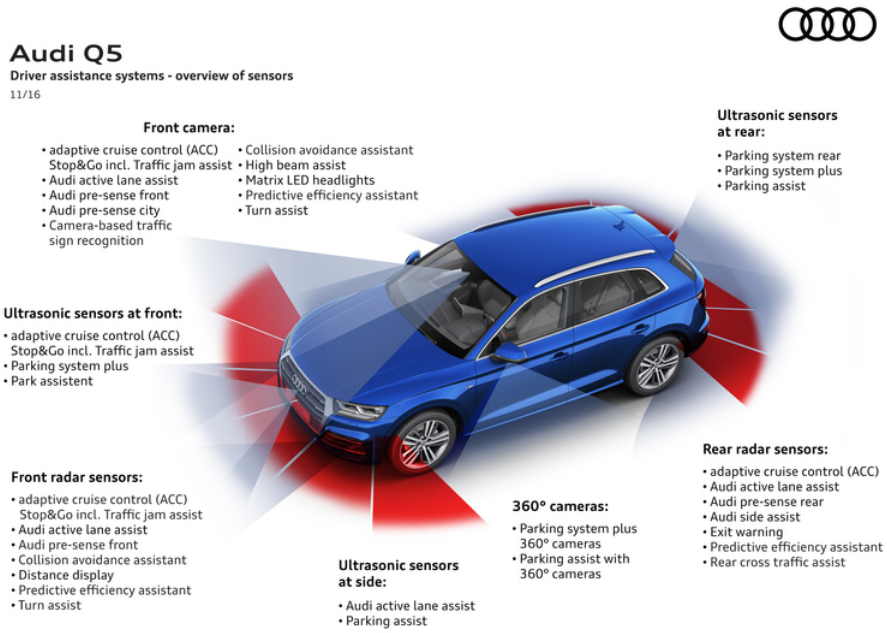


Abbildung 2.2: Übersicht Assistenzsensoren Audi Q5 (Audi)

Automobilfirmen arbeiten bereits an der Anbindung von Fahrzeugen an Cloud-Server. In einer von Audi durchgeführten Erprobung von Car-to-X[8] Testflotten, sammelte Audi von Mitte 2015

[5] Application software
[6] https://de.park-now.com/
[7] Global Positioning System
[8] Fahrzeug zu Infrastruktur

bis Mitte 2016 Daten in ganz Deutschland. Dabei erzeugten die Testfahrzeuge mehr als sechs Milliarden Datensätze, gesammelt aus ca. 850 Signalen und Steuergeräten des Fahrzeuges, welche anonymisiert und verschlüsselt durch ein Cloud basiertes System an ein Rechenzentrum übertragen wurden. Zur live Übertragung der Sensordaten an Rechenzentren kam dabei eine fest verbaute e-SIM[9] mit LTE[10] Zugang zum Einsatz, welche in den Steuergeräten, im Rahmen des Audi-Connect Systems, bereits in Serienfahrzeugen verwendet wird (Audi (2016)).

2.2 Vernetzung von Fahrzeugen

Bei der Vernetzung von Fahrzeugen werden zwei primäre Ansätze verfolgt. Das Car-to-X Verfahren zielt auf die Kommunikation von Fahrzeugen mit der umgebenden Infrastruktur ab, wohingegen sich der Car-to-Car Ansatz auf die Kommunikation zwischen Fahrzeugen, in so genannten Adhoc Netzwerken, konzentriert. Verschiedene branchenführende Automobilhersteller und Zulieferer, darunter Audi, BMW und Volkswagen, haben sich zu einem Car2Car Konsortium zusammengeschlossen, um eine Standardisierung der Kommunikationsinfrastruktur festzulegen. Ziele des Car2Car Konsortiums sind die Einführung von europäischen sowie weltweiten Industriestandards, die Sicherstellung der Kompatibilität zwischen verschiedenen Automobilherstellern, die Etablierung neuer Funkstandards sowie die Entwicklung nötiger Systeme voran zu treiben (C2C-CC, 2007).

Für die Kommunikation zwischen Fahrzeugen und Cloudservern, eignet sich keines der etablierten Verfahren. Der Car-to-Car Ansatz dient der Kommunikation zwischen Fahrzeugen. Auch der Car-to-X Ansatz ist nicht für die Kommunikation mit Cloudservern ausgelegt. Er verfolgt den Ansatz der direkten Kommunikation von Fahrzeugen mit der Infrastruktur, wie beispielsweise Ampelanlagen.
Die Kommunikation zwischen Fahrzeugen und Cloudservern erfordert daher ein weiteres, bisher wenig erforschtes, Kommunikationsverfahren. Das so genannte Car-to-Cloud Verfahren dient der direkten Kommunikation zwischen Fahrzeugen und Cloudservern. Das Unternehmen HERE[11] arbeitet, zusammen mit weiteren branchenführenden Unternehmen, an einer Standardisierung des Car-to-Cloud Verfahrens (HERE, 2016).
Für die spätere Versuchsanwendung (Kapitel 4) wird der Car-to-Cloud Ansatz verfolgt, um Messdaten des Fahrzeuges direkt an eine Cloudplattform übermitteln zu können.

[9] Elektronische SIM-Karte
[10] Long Term Evolution
[11] https://here.com/

3 Big Data

In diesem Kapitel wird speziell auf die Erhebung und Auswertung von Daten, für spätere Vorhersage- und Prognoseverfahren, eingegangen. Der Begriff Big Data wird stark durch Internet Firmen wie beispielsweise Google, Amazon oder Facebook geprägt. Unter Big Data wird die Ansammlung großer, zum Teil unstrukturierter, Datenmengen verstanden, welche mit herkömmlichen Analysemethoden, wie relationalen Datenbankabfragen, nicht mehr verarbeitet werden können. Dabei beschränkt sich Big Data nicht ausschließlich auf selbst erhobene Daten, sondern umfasst auch den Einbezug großer externer Datenmengen (Fasel und Meier (2016)). Die gewonnenen Daten sind die Basis für spätere Prognose- und Vorhersagealgorithmen. Im Folgenden wird auf die, für Vorhersagen notwendige, Verarbeitung und Aufarbeitung gesammelter Daten eingegangen.

3.1 Data-Mining

Bei der Analyse großer Datenmengen geht es primär um das Data-Mining. Ziel des Data Mining ist es, relevantes Wissen aus vorhandenen Datensätzen zu extrahieren. Dabei müssen die Ergebnisse des Mining Prozesses ausgewertet und anschließend bewertet werden, um sicherzustellen, dass ein zufriedenstellendes Ergebnis erreicht wurde. Dabei beinhaltet die Datenanalyse Themen aus verschiedenen wissenschaftlichen Teilbereichen wie der Statistik, der Mustererkennung oder dem maschinellen Lernen (Runkler (2015)).

3.1.1 Knowledge Discovery Prozess

Oft wird für das Data-Mining der sogenannte KDD-Prozess[1] (Abbildung 3.1) verwendet. Der Prozess setzt sich aus der Vorbereitung, Vorverarbeitung, Analyse sowie der Nachbereitung zusammen. Der Prozess kann iterativ, mehrfach komplett oder in Teilschritten durchlaufen und angepasst werden, bis ein zufriedenstellendes Ergebnis erzielt wird.

[1] Knowledge Discovery Prozess

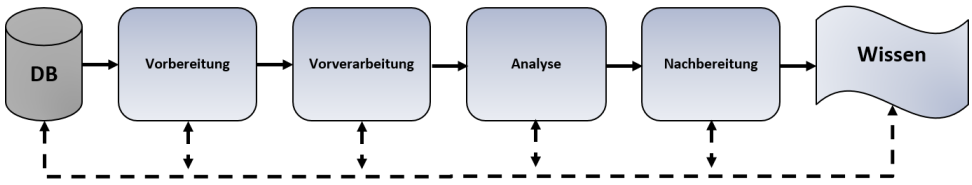


Abbildung 3.1: Knowledge Discovery Prozess

Vorbereitung In der Vorbereitungsphase werden die angesammelten Daten selektiert. Oft sind nicht alle gesammelten Daten für die angestrebte Analyse relevant. Des Weiteren können in diesem Prozessabschnitt Daten aus verschiedenen Quellen zusammengeführt werden, um neue Attribute zu erzeugen oder den vorhandenen Datenbestand zu erweitern.

Vorverarbeitung Gesammelte Daten müssen nicht zwingend fehlerfrei oder vollständig sein. Ziel der Vorverarbeitung ist es, falsche oder unvollständige Datensätze heraus zu filtern, zu bereinigen, zu ergänzen oder zu korrigieren. Diese qualitative Bewertung der Daten ist ausschlaggebend für ein unverfälschtes Analyseergebnis. Abhängig von der Form der vorliegenden Daten sowie der verwendeten Analysesoftware, ist eine Transformation der Daten vorzunehmen (z.B. Einschränkung von Wertebereichen oder die Transformation von textuellen Inhalten auf verarbeitbare, eindeutige Schlüssel).

Analyse Die Analyse beschreibt das eigentliche Data-Mining. Ziel ist es, Muster in den vorliegenden Daten zu finden und zu bewerten. Dazu werden in dieser Phase die für die geplante Analyse passenden Algorithmen und Verfahren identifiziert sowie parametrisiert und angewendet. Um die entstandenen Ergebnisse bewerten oder nachvollziehen zu können, ist eine passende visuelle Darstellung notwendig. Oft trägt die visuelle Darstellung dazu bei eine neue Sicht auf die Daten zu erlangen und so neue, bisher unbekannte, Beziehungen zwischen Attributen zu erkennen.

Nachbereitung Bei der Nachbereitung werden die aus der Analyse gewonnenen Ergebnisse ausgewertet und dokumentiert. Eine anschließende Interpretation der Ergebnisse ist notwendig, um zu ermitteln, ob durch die Analyse neue oder angestrebte Informationen gewonnen werden konnten. Je nach Ausgang dieser Untersuchung ist eine weitere Anpassung des Analysevorgangs notwendig und dieser zu wiederholen, bis ein zufriedenstellendes Ergebnis erzielt wird.

3.1.2 Verfahrensansätze

Beim Data-Mining gibt es fünf primäre Verfahrensansätze für verschieden Anwendungsfälle. Dabei wird zwischen den Verfahren Klassifikation, Clustering, Assoziation sowie dem Text- und Web Mining unterschieden (Cleve und Lämmel, 2016). Im Folgenden wird eine Einführung in die verschiedenen Verfahren gegeben.

Klassifikation Anhand verschiedener Merkmalskombinationen eines Objektes wird versucht dieses einer vorgegebenen Menge an Klassen zuzuordnen, beziehungsweise die Wahrscheinlichkeit der Zugehörigkeit zu einer Klasse zu ermitteln. Zur Klassifikation muss die Menge an Klassen sowie die entsprechende Zugehörigkeit der Trainings- und Testdaten bekannt sein. Daher wird bei der Klassifikation von einem überwachten Lernen gesprochen. Anhand der Testdaten lässt sich die Fehlerrate des Verfahrens bestimmen und somit eine nachträgliche Verfahrensanpassung vornehmen, bis eine ausreichend geringe Fehlerrate erzielt wird. Gängige Algorithmen zur Realisierung der Klassifikation sind das k-Nearest-Neighbour Verfahren, Entscheidungsbäume, Naive-Bayes, neuronale Netze und Support Vector Machines (Cleve und Lämmel, 2016).

Clustering Im Gegensatz zur Klassifikation wird der Ansatz des Clusterings dazu verwendet Objekte, gemessen an ihrer Ähnlichkeit, zu sogenannten Clustern zusammenzufassen. Dabei sollen Objekte innerhalb eines Clusters möglichst homogene Merkmale aufweisen, wohingegen Objekte unterschiedlicher Cluster möglichst heterogen sein sollen. Das Clustering gehört damit zu den unüberwachten Lernverfahren (Cleve und Lämmel, 2016).

Assoziation Die Assoziationsanalyse findet vorrangig bei der Warenkorbanalyse Anwendung. Sie dient dem Erkennen von häufig gleichzeitigem auftreten von Objekten oder Ereignissen anhand von Regeln. Ein typischer Anwendungsfall ist die Bestimmung von regelmäßig zusammen gekauften Artikeln in einem Warenwirtschaftssystem (Petersohn (2005)).

Text und Web Mining Das Text Mining dient der Analyse von unstrukturierten Textdokumenten. Mögliche Einsatzgebiete sind die Kategorisierung anhand von Schlüsselwörtern oder die Bestimmung der Ähnlichkeit von Dokumenten. Beim Web Mining werden Webinhalte analysiert. Dabei wird zwischen dem Web Content Mining und dem Web Usage Mining unterschieden. Das Web Content Mining zielt auf die Analyse von Texten sowie Links zu anderen Websites ab. Beim Web Usage Mining hingegen werden Verhaltensmuster der Internetnutzer analysiert (Cleve und Lämmel, 2016).

3.2 Algorithmen

Im Folgenden werden ausgewählte Algorithmen zur Umsetzung des Klassifikationsverfahrens vorgestellt. Die ausgewählten Algorithmen finden im Rahmen dieser Arbeit Einsatz bei der Evaluierung eines geeigneten Algorithmus zur Parkraumvorhersage sowie bei der Analyse (Unterabschnitt 5.3.5).

3.2.1 Neuronale Netze

Neuronale Netze sind dem menschlichen Gehirn nachempfunden und bestehen aus vielen miteinander verknüpften Neuronen. Wie die Nervenzellen in der Biologie, haben die Neuronen in künstlichen neuronalen Netzen die Aufgabe, bei genügend Anregung, aktiv zu werden und ein Signal an andere Neuronen zu senden. In künstlichen neuronalen Netzen wird dieses Verhalten durch so genannte Perzeptrons nachgebildet, welche beim Überschreiten eines numerischen Schwellwertes, ein Signal an die weiteren verknüpften Perzeptrons ausgeben (Kruse u. a. (2015)). Es existieren verschiedene Varianten neuronaler Netze. Für Klassifikationsaufgaben eignen sich vorwärtsgerichtete neuronale Netze (auch Backpropagation-Netze oder Multilayer Perceptrons genannt).

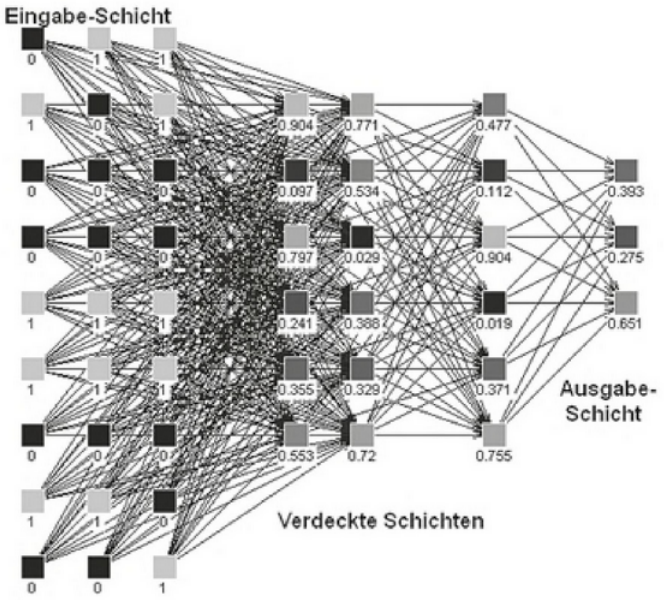


Abbildung 3.2: Vorwärtsgerichtetes neuronales Netz (Cleve und Lämmel (2016))

Das vorwärtsgerichtete neuronale Netz besteht aus einer Eingabe-, Verdeckter- sowie Ausgabeschicht (Abbildung 3.2). Die Anzahl der Neuronen in der Ausgabeschicht ist, für die Klassifikation, vom Klassifikationsergebnis abhängig. Ein einzelnes Neuron kann dabei lediglich zwei Zustände annehmen (aktiv/inaktiv), daher wird für jede Klasse ein Ausgabeneuron benötigt. Die benötigte Anzahl von Neuronen in der Eingabe-Schicht ergibt sich aus den für die Klassifizierung erforderlichen Objektattributen. Für die Größe der verdeckten Schicht existiert, im Gegensatz zu Eingabe- und Ausgabeschicht, keine konkrete Vorgabe, wodurch eine experimentelle Annäherung an ein zufriedenstellendes Ergebnis erforderlich wird (Cleve und Lämmel (2016)). Im Zusammenhang mit neuronalen Netzen wird häufig der Begriff *Deep Learning* verwendet. Ein *Deep Learning* Netz bezeichnet dabei ein künstliches neuronales Netz mit einer großen Anzahl an verdeckten Schichten.

3.2.2 Entscheidungsbäume

Ein effizientes Verfahren zur Klassifikation sind Entscheidungsbäume. Ein Vorteil von Entscheidungsbäumen, beispielsweise gegenüber neuronalen Netzen, ist die Transparenz der Suchbäume. Suchbäume lassen sich gut veranschaulichen und ermöglichen so einen Einblick in den Prozess der Entscheidungsfindung.

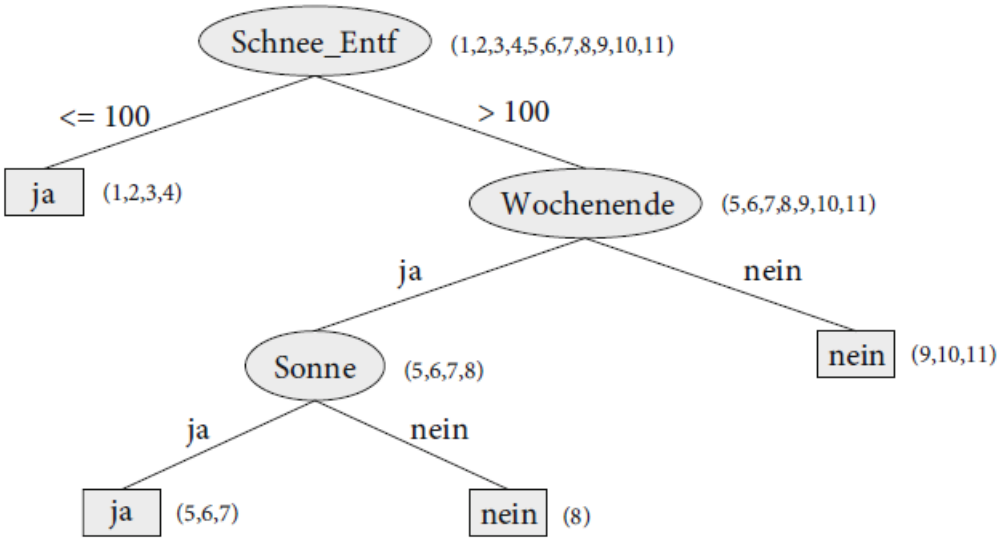


Abbildung 3.3: Entscheidungsbaum Beispiel Skifahren (Ertel (2016))

In einem Entscheidungsbaum repräsentieren Knoten die Objektattribute, Kanten wiederum stehen für die möglichen Attributwerte. Die Blätter des gerichteten Entscheidungsbaumes legen die Zugehörigkeit zu einer Klasse fest (Abbildung 3.3). Risiko der Verwendung von Entscheidungsbäumen ist das Auswendiglernen (auch als Overfitting bezeichnet). Für eine möglichst effiziente Generierung des Baumes existieren neben dem naiven händischen Ansatz entsprechende Verfahren (z.B. ID3, C4.5 oder CART) (Ertel (2016)). Aus der Baumdarstellung lassen sich vollständige Regeln ableiten, welche anschließend für die Entscheidungsunterstützung/Vorhersage verwendet werden können (Cleve und Lämmel (2016)).

3.2.3 Naive Bayes

Im Gegensatz zu neuronalen Netzen oder Entscheidungsbäumen nimmt der Naive-Bayes-Algorithmus keine feste Zuordnung zu einer konkreten Klasse vor, sondern ermittelt die höchste Wahrscheinlichkeit für die Zugehörigkeit zu einer Klasse. Dem Bayes-Algorithmus liegt der Satz des Bayes für die Berechnung bedingter Wahrscheinlichkeiten zugrunde. Die Objektattribute werden für das Verfahren als unabhängig betrachtet. Des Weiteren findet im Vergleich zu anderen Algorithmen kein Training eines Modells statt, sondern eine konkrete Berechnung anhand der Trainingsdaten (Cleve und Lämmel (2016)).

4 Versuchsanwendung zur Detektion von freiem Parkraum

Um das Potenzial fahrzeuginterner Sensordaten für Vorhersagemodelle und Cloud basierte Anwendungen zu erforschen, wurde im Rahmen dieser Arbeit ein Projekt, zur Parkraumanalyse und Vorhersage, geplant und realisiert. In diesem Kapitel geht es um eine Plattform zur Erfassung von Parkplatzbelegungsdaten. Die Behandlung von Vorhersagemodellen erfolgt gesondert im Kapitel 5. Auf die Umsetzung der einzelnen Komponenten wird in Abschnitt 4.3 eingegangen. Eine abschließende Evaluation sowie Auswertungen und Ergebnisse werden in den Kapiteln 4.5 und 4.6 vorgenommen.

4.1 Zielsetzung und Versuchsbeschreibung

Die Parkraumsituation in Großstädten ist ein anhaltendes Thema. Autofahrer verbringen viel Zeit mit der Suche nach einem freien Parkplatz. Die Suche nach einem freien Parkplatz führt zu einem erhöhten Verkehrsaufkommen und somit zu einem schlechteren Verkehrsfluss. In Hamburg existieren nach aktuellem Stand (2017) zwar Parkleitsysteme, allerdings können diese lediglich über die Anzahl freier Parkplätze in öffentlichen Parkhäusern informieren. Eine Analyse des öffentlichen Parkraum ist nach aktuellem Stand nicht möglich, wodurch ein wichtiges Instrument zur effizienten Verkehrsplanung, Steuerung und Überwachung fehlt (Hamburg (2014)).

In Zusammenarbeit mit der Firma IAV Automotive Engineering, wurde das Projekt SmartParking realisiert. Ziel des Projektes ist die Nutzung bereits in Serienfahrzeugen vorhandener Sensordaten, wie GPS und Ultraschall, zur Analyse des Parkraums während der Fahrt. Die mit SmartParking ausgestatteten Fahrzeuge sollen selbstständig den ruhenden Verkehr am Straßenrand überwachen können, um so den aktuellen Belegungsstatus zu erfassen und in einer cloudbasierten Infrastruktur zur Verfügung zu stellen. Anhand der erfassten und gesammelten Daten werden Vorhersagealgorithmen, zur Prognose der zukünftigen Parkraumbelegung,

erprobt. Eine Cloud basierte Web-App ermöglicht es Endanwendern die aktuelle Parkraumsituation einzusehen und den nächstgelegenen, freien Parkplatz zu finden (Abbildung 4.1).

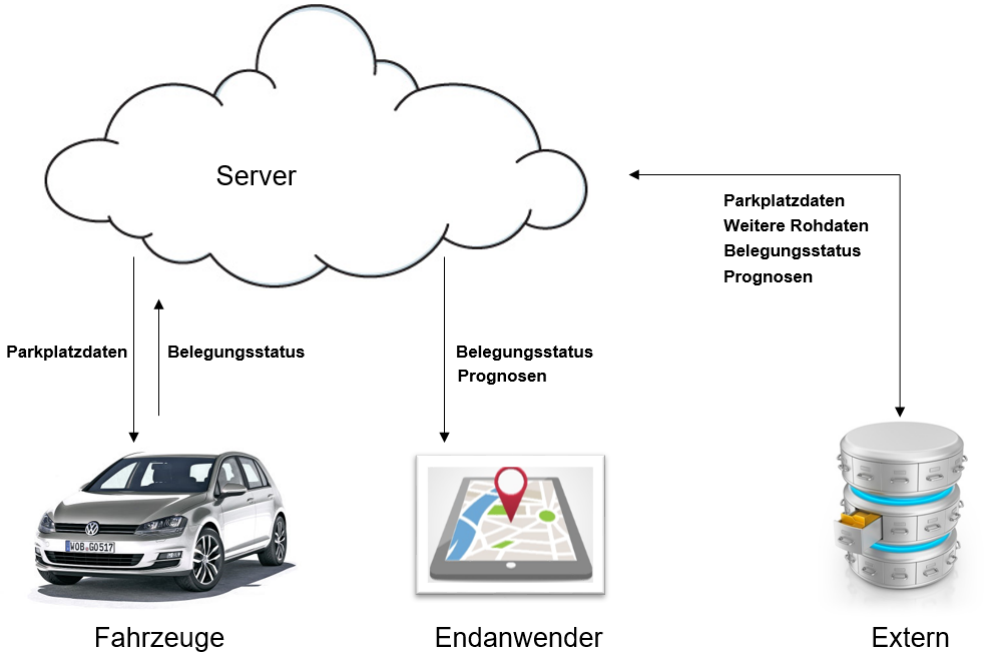


Abbildung 4.1: Funktionale Architektur

Ein Cloudserver bezieht aus externen Datenquellen (Unterabschnitt 4.3.4) die in der städtischen Infrastruktur vorhandenen Parklücken sowie weitere, für Algorithmen und Vorhersagen notwendige, Metadaten. Fahrzeuge rufen über einen Internetzugang die vorhandenen Parkplatzinformationen aus der Cloud ab und nutzen sie zur Lokalisierung von vorhandenen Parklücken. Eine anschließende Ermittelung des Belegungsstatus der entsprechend erfassten Parklücken findet innerhalb des Fahrzeuges, mit Hilfe von Ultraschallsensoren, statt. Das Fahrzeug sendet anschließend die Messergebnisse an den Cloudserver zurück. Endanwender können sich über die SmartParking Web-App, welche die aktuelle Belegungssituation vom Cloudserver bezieht, die Parkraumbelegungsdaten in nahezu Echtzeit darstellen lassen. Bei Bedarf kann die aktuelle Parkraumbelegung von weiteren externen Diensten über geeignete Schnittstellen abgefragt werden (Abbildung 4.1).

4.2 Verwandte Arbeiten und Abgrenzung

Bereits heute gibt es erfolgreich umgesetzte Projekte zur Überwachung des öffentlichen beziehungsweise privaten Parkraums. Die Firma Siemens hat im Jahr 2015 ein Pilotprojekt zur Parkraumüberwachung gestartet. Dabei werden Radarsensoren in bestehende Infrastruktur wie Straßenlaternen integriert (Zwick (2015)). Andere Projekte wurden durch eine Überwachung mittels Bodensensoren realisiert. In München hat die Firma Park-Here[1] Sensoren entwickelt und verbaut, welche ohne zusätzliche Stromversorgung betrieben werden. Der Vorteil in der Nutzung von fest installierten Überwachungssystemen liegt hauptsächlich in der hohen Verfügbarkeit des aktuellen Belegungsstatus.

Die bisher umgesetzten Projekte erfordern einen Umbau beziehungsweise eine Aufrüstung der bestehenden städtischen Infrastruktur. Für eine flächendeckende Umsetzung ist daher mit hohen Kosten zu rechnen. Des Weiteren besteht kein einheitlicher Standard, der Städte übergreifend einsetzbar ist. Bisherige Pilotprojekte wurden lediglich in einzelnen Stadtteilen umgesetzt.

SmartParking ist eine Möglichkeit der Parkraumüberwachung, welche nicht auf fest installierte Sensoren angewiesen ist. Sie erfordert eine Flotte entsprechend ausgestatteter Messfahrzeuge. Es ist allerdings anzunehmen, dass die anfallenden Rüstkosten im Vergleich zu der festen Installation von Sensoren, mit der entsprechenden Anbindung an das Stromnetz sowie Backends für die Kommunikation, vergleichsweise gering ausfallen. Duch die fortschreitende Entwicklung in der Anbindung von Fahrzeugen an Onlinedienste (Kapitel 2), ist für die Zukunft denkbar das SmartParking System fest in Serienfahrzeugen zu etablieren. Die benötigten Sensoren sind bereits heute in einer Vielzahl von Serienfahrzeugen verbaut. Laut einer Studie der Robert Bosch GmbH aus dem Jahr 2014 verfügen rund 52 % aller neu zugelassenen Fahrzeuge über die, für SmartParking benötigten, Parkassistenzsysteme auf Ultraschallbasis (Bosch (2016)). Ein weiterer Vorteil der Nutzung des Fahrzeuges als fahrenden Sensorknoten ist die Möglichkeit, über den Belegungsstatus von Parklücken hinaus, weitere Sensordaten für spätere Auswertungen und Analysen zu erfassen.

[1] http://park-here.eu/

4.3 Softwarearchitektur und Umsetzung

Die finale SmartParking Architektur besteht aus drei primären Komponenten. Die Erste Komponente umfasst die Cloudserver Architektur, die neben der Kommunikation auch für die Verwaltung, Aufarbeitung, Bereitstellung und Analyse von Parkplatzbelegungsdaten zuständig ist. Zudem fungiert eine eigenständige Anwendung im Fahrzeug als Sensor zur Erkennung des Belegungsstatus einer Parklücke. Die dritte Komponente ist eine WebApp zur Darstellung der aktuellen Parkraumsituation (Abbildung 4.2).

Im Nachfolgenden wird auf die einzelnen Komponenten der Architektur eingegangen. In den Unterkapiteln, zu den einzelnen Komponenten, werden jeweils die Anforderungen analysiert, benötigte Komponenten selektiert sowie die konkrete Umsetzung beschrieben. Die Parkraumprognose wird gesondert im späteren Kapitel 5 behandelt.

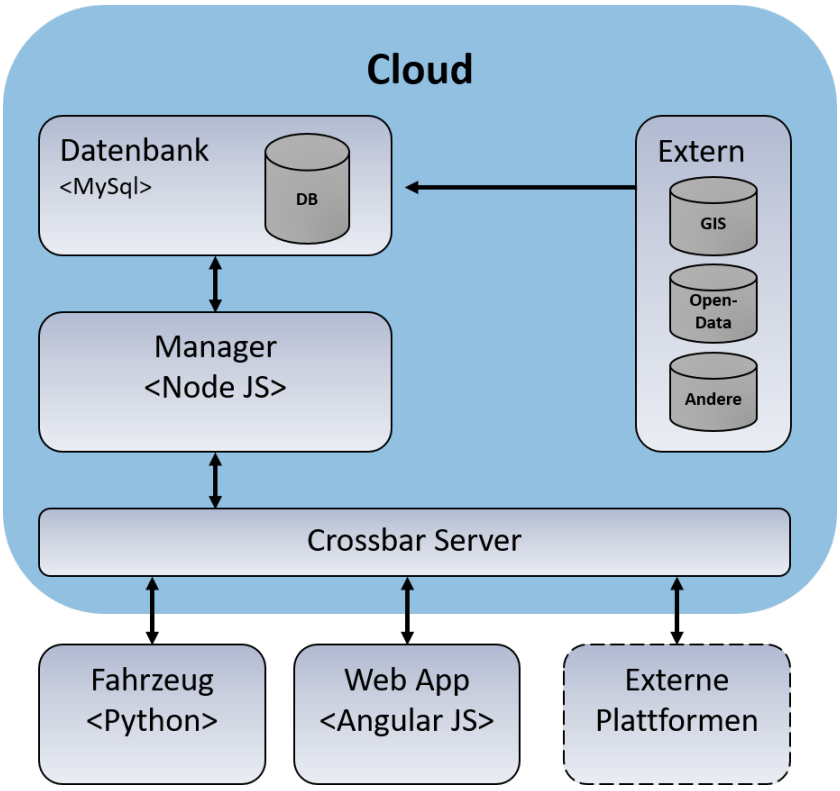
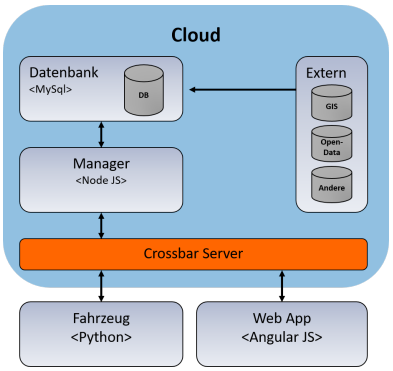


Abbildung 4.2: Softwarearchitektur

4.3.1 Crossbar Server



Für die zentrale Kommunikation zwischen Cloudserver, Fahrzeug, WebApp und externen Diensten wird ein Kommunikationsdienst benötigt, der sowohl *Remote Procedure Calls* beispielsweise zur Abfrage der Parkplatzinformationen als auch das *Publish & Subscribe* Verfahren zum Verteilen von Änderungen des Belegungsstatus, ohne permanentes *pollen*, nativ unterstützt. Aus diesem Grund kommt ein Crossbar Server auf Basis des Web Application Messaging Protocol[2] zum Einsatz. WAMP[3] ist ein freier WebSocket Standard, welcher neben *Remote Procedure Calls* auch das *Publish & Subscribe* Verfahren zur *soft real-time* Kommunikation unterstützt. WAMP Client Bibliotheken sind für eine große Anzahl an Programmiersprachen frei verfügbar, werden regelmäßig aktualisiert und stellen somit eine plattformübergreifende Kommunikation sicher.

Der Crossbar Server (Abbildung 4.2) fungiert dabei als WAMP-Router. Ein WAMP-Router besteht aus einem Broker und einem Dealer (Abbildung 4.3).

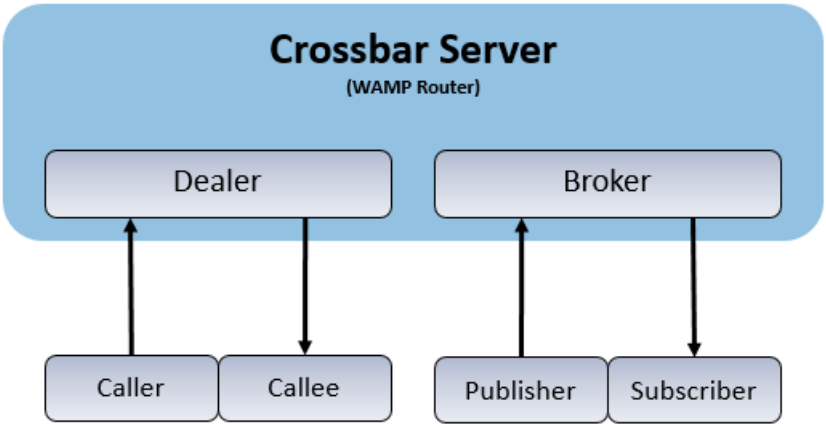


Abbildung 4.3: WAMP Router

Der Dealer übernimmt das Routing der Funktionsaufrufe sowie der Rückgabewerte. Durch den Dealer für *Remote Procedure Calls*, müssen *Caller* und *Callee* ihren gegenseitigen Standort nicht kennen. Es besteht somit keine starke Bindung zwischen *Caller* und *Callee*. Der *Broker*

[2] http://http://wamp-proto.org/
[3] Web Application Messaging Protocol

ist für die Verwaltung der *subscriptions* für das *Publish & Subscribe* Verfahren zuständige und arbeitet ebenfalls nach dem Prinzip der losen Kopplung.

Die Registrierung der *Remote Procedure Calls* sowie *subscriptions* für das *Publish & Subscribe* Verfahren erfolgt über die *Uniform Resource Identifier*. Im Kontext des Crossbar Servers werden diese als *Crossbar Topic* bezeichnet.

Definition der Server Schnittstelle

Anhand der in Kapitel 4.1 beschriebenen Anforderungen wurde die Crossbar Schnittstelle für die Kommunikation zwischen Fahrzeug, WebApp und Cloudserver definiert. Für die Schnittstelle ergaben sich die im folgenden konkret formulierten Anforderungen.

Funktionale Anforderungen

Anhand von GPS Koordinaten und der Angabe eines Suchradius für das gewählte Suchgebiet können die im Cloudserver Datenbankbestand vorhandenen Parklücken abgefragt werden. Die vorhandenen Parklücken sollen mit einer eindeutigen ID zurückgegeben werden. Anhand optionaler Funktionsparameter können, neben der aktuellen Belegungswahrscheinlichkeit des Parkplatzes, die vorhandenen Metadaten (Tabelle 4.1) gezielt abgefragt werden, um so die zu übertragene Datenmenge möglichst gering zu halten.

Neue Parkplatzdaten sollen sich über eine externe Schnittstelle in den internen Datenbestand einpflegen lassen. Dazu werden neu zu erfassende Parkplätze anhand einer eindeutigen ID und der Parkplatzkontur in Form eines Polygons, basierend auf GPS Koordinaten, an den Cloudserver übergeben. Optional sollen die unter 4.1 beschriebenen Metadaten übergeben werden können. Die für das Einpflegen neuer Parkplätze zuständige Schnittstelle soll gleichzeitig genutzt werden um Attribute, bereits bestehender Parkplätze, zu aktualisieren.

Es soll möglich sein nicht mehr vorhandene oder nutzbare Parkplätze aus dem aktiven Datenbestand auszuschließen, ohne die Parkplatzhistorie zwecks Prognoseauswertungen zu verlieren.

Um ein ständiges *pollen* der aktuellen Parkplatzbelegungsstände seitens der WebApp sowie möglichen externen angebundenen Klienten zu vermeiden, soll der Cloudserver bei Änderungen der Belegungswerte die entsprechenden Informationen in Form der Parkplatz ID, des neuen Belegungsstatus sowie der Änderungszeit des Belegungswertes automatisch an die Interessenten verteilen.

Anhand der eindeutigen Parkplatz-ID können Fahrzeuge, unter Angabe einer eindeutigen Fahrzeugidentifikationsnummer und einer zusätzlichen Fahrzeugbeschreibung, den Belegungsstatus des Parkplatzes an den Cloudserver übermitteln. Die Übertragung einer eindeutigen Fahrzeugidentifikationsnummer ist in der Erprobungsphase erforderlich, um Messwerte gezielt dem verursachenden Fahrzeug zuordnen und so eine gezielte Fehleranalyse betreiben zu können. Aus Datenschutzgründen können die Fahrzeugidentifikationsnummern zu einem spätere Zeitpunkt zufällig vom Fahrzeug generiert werden, damit eine nicht fahrzeugbezogene, anonymisierte Übermittlung der Messdaten ermöglicht wird. Der übermittelte Belegungswert soll, zur qualitativen Bewertung der Messung, nicht binär erfolgen, sondern mit einer Wahrscheinlichkeit in einem Wertebereich von 0.0 (sicher belegt) bis 1.0 (sicher frei) übergeben werden. Zusätzlich muss ein Zeitstempel mit dem Zeitpunkt der erfassten Messung vorhanden sein, damit die übermittelten Messwerte in einen zeitlichen Kontext gesetzt werden können.

Schnittstellen Definition

Anhand der konkret formulierten Anforderungen ergaben sich folgende Crossbar Schnittstellen:

Name	**Caller**	**Callee**	**Crossbar Topic**
getLotsInfo	Fahrzeug, webApp	Manager	iot.smp.lots.getLotsInfo

Mit der *getLotsInfo* Funktion können durch Übergabe der Zentrumskoordinaten sowie dem Suchradius alle im Umkreis befindlichen, erfassten Parkplätze mit dazugehörigen Attributen abgefragt werden. Mögliche optional abzufragende Attribute werden im Folgenden aufgeführt.

- ***location*** Polygon aus GPS-Koordinaten der Parkplatzkontur
- ***vehicle*** Für Parkplatz geeigneter Fahrzeugtyp
- ***surface*** Oberflächenbeschaffenheit
- ***street*** Zugehörige Straße
- ***orientation*** Ausrichtung zur Fahrbahn
- ***type*** Art der Parkplatzbewirtschaftung
- ***measure_probability*** Letzter gemessener Belegungsstatus

- ***measure_update_time*** Zeitpunkt der letzten Belegungsmessung
- ***forecast_probability*** Prognosewert des Belegungsstatus

Zusätzliche Informationen zu den möglichen Attributwerten können der Tabelle 4.1 entnommen werden.

Name	**Caller**	**Callee**	**Crossbar Topic**
addLots	Importscript	Manager	iot.smp.lots.addLots

Die Funktion *addLots* dient sowohl der Aufnahme von neuen Parkplätzen in den internen Datenbankbestand als auch der Aktualisierung bereits vorhandener Parkplätze. Sie wird verwendet, um den Datenbestand mit externen Datenquellen zu synchronisieren (Unterabschnitt 4.3.4).

Name	**Caller**	**Callee**	**Crossbar Topic**
delteLots	Importscript	Manager	iot.smp.lots.delteLots

Durch die *delteLots* Funktion ist es möglich, nicht mehr vorhandene oder verfügbare Parkplätze aus dem aktiven Datenbestand zu entfernen.

Name	**Publisher**	**Subscriber**	**Crossbar Topic**
onMeasuement	Fahrzeug	Manager	iot.smp.lots.measurement

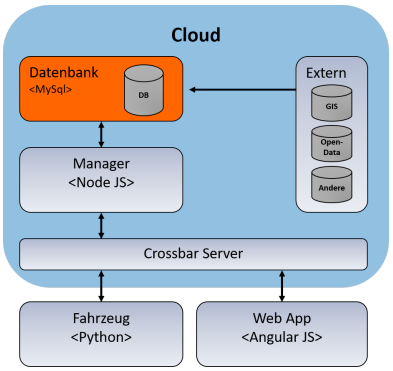
Über das *onMeasurement* Topic übermittelt das Fahrzeug die neu gemessenen Parkplatzbelegungsstände mit der zugehörigen Messsicherheit an den Cloudserver.

Name	**Publisher**	**Subscriber**	**Crossbar Topic**
onOccupancy	Manager	webApp	iot.smp.lots.occupancy

Anwendungen, die über Änderungen der Parkplatzbelegungsstände informiert werden sollen, nutzen das *occupancy* Topic, um ohne *pollen* über die entsprechenden Änderungen informiert zu werden.

4.3.2 Datenbank

Entscheidend für die Auswahl der zu nutzenden Datenbank war im Rahmen dieser Arbeit, dass die Datenbank sowohl unter einer freien Lizenz angeboten als auch die Verarbeitung von Geodaten unterstützt wird. Datenbanken mit Unterstützung für Geodaten besitzen meist spezielle geografische Datentypen (z.B. Polygon, Point, LineString) sowie Funktionen für performante geografische Datenbankabfragen.



In die engere Auswahl kam sowohl die PostgreSQL[4] Datenbank der PostgreSQL Global Development Group, welche mit der PostGIS Erweiterung Geodaten unerstützt, als auch die MySQL[5] Datenbank von Orcale mit nativer GIS[6] Unterstützung ab Version 5.7[7].

Neben der nativen Unterstützung von GIS Funktionen und einer guten Anbindungsmöglichkeit an die verwendeten Programmiersprachen hat die MySQL-Datenbank einen hohen Verbreitungsgrad und wird aktiv weiter entwickelt. Aus diesen Gründen wurde sich im Rahmen des SmartParking Projektes für die Nutzung der MySQL Datenbank entschieden.

Das entstandene Datenbankmodell lässt sich der Abbildung 4.4 entnehmen. Die Aufteilung in die verschiedenen Tabellen dient der Vermeidung von Redundanzen in der Datenhaltung. Im späteren Produktivbetrieb würden Redundanzen, bei vielen aktiven Messfahrzeugen, zu einem erhöhten Speicherbedarf führen. Neben der für die Erfassung der Belegungsmessung erforderlichen Datenbanktabellen *lots* und *probability* wurde die zusätzliche Tabelle *history* angelegt. Die *history* Tabelle dient der Archivierung aller eingehenden Belegungsmeldungen von Parkplätzen. Anhand der *device_id* sowie *config* ist es möglich, Testdaten von Produktivdaten oder Daten aus unzuverlässigen Quellen zu filtern und aus späteren Wahrscheinlichkeitsberechnungen für den Belegungsstatus auszuschließen.

[4] https://www.postgresql.org/
[5] https://www.mysql.de/
[6] Geoinformationssystem
[7] https://dev.mysql.com/doc/refman/5.7/en/spatial-extensions.html

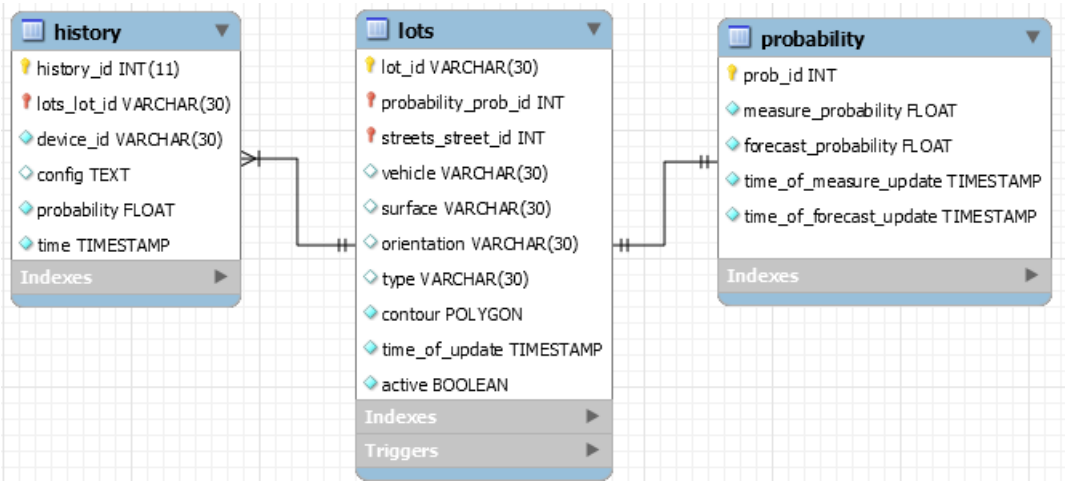


Abbildung 4.4: SmartParking Datenbankmodell

Innerhalb der Datenbank wurde die Tabellen *history*, *lots* und *probability* mit UTC[8]-Zeitstempeln versehen, welche auch über verschiedene Zeitzonen hinweg eine Vergleichbarkeit der Datensätze erlauben. Zusätzlich wurde die Tabelle *lots* mit einem *Trigger* versehen, welcher automatisch, bei Änderungen der Parkplatzbelegungsstände, die jeweils aktuellen Zeitstempel direkt setzt. Im späteren Data-Mining Prozess (Abschnitt 5.3) werden die Zeitstempel genutzt, um die Parkraumbelegungsstände, zu einem bestimmten Zeitpunkt, ermitteln zu können.

Um der gestellten Anforderung an die Umkreissuche von Parklücken gerecht zu werden, wurden die Parkplatzkoordinaten *contour* als *POLYGON* Datentyp angelegt. Durch die Nutzung des *POLYGON* Datentyps können die GIS-Funktionalitäten der MySQL-Datenbank für eine direkte, performante Umkreisabfrage innerhalb der Datenbank verwendet werden.

Eine weitere Anforderung die berücksichtigt werden muss sind Änderungen im Parklückenbestand. Durch bauliche Veränderungen innerhalb der Verkehrsinfrastruktur ist anzunehmen, dass vorhandene Parklücken mit der Zeit nicht mehr nutzbar sind, oder Änderungen in der Bewirtschaftung unterliegen. Damit die nicht mehr aktiven Parklücken dennoch für die Parkraumprognose genutzt werden können, wurde in der Datenbanktabelle *lots* das Attribut *active* eingeführt. Anhand des Attributes lassen sich gezielt Parklücken aus dem aktiven Parklückenbestand ausschließen, ohne sie aus der Datenbank entfernen zu müssen.

[8] Coordinated Universal Time

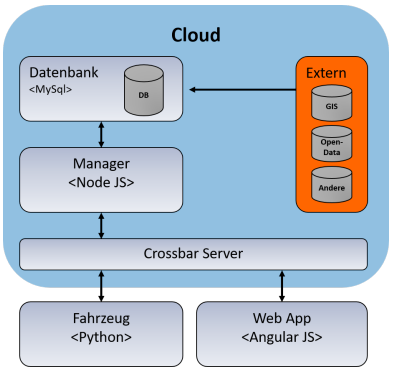
Die Tabelle *probability* dient der direkten Abrufbarkeit von den jeweiligen Belegungsdaten der Parklücken. Sie verknüpft eine Parklücke mit den aktuellen Belegungsdaten zur späteren Nutzbarkeit in Endanwendungen (Unterabschnitt 4.3.7). Dabei enthält das Attribut *measure_probability* den zuletzt erfassten Belegungswert der Parklückenvermessung. Der zugehörige Zeitstempel *time_of_measure_update* gibt an wann die letzte Parklückenvermessung erfolgt ist, um Rückschlüsse auf die Aussagekraft des Messwertes ziehen zu können. Das Attribut *forecast_probability* dient der späteren Aufnahme des aktuell gültigen Prognosewertes für den Belegungsstand einer Parklücke. Der Zeitstempel *time_of_forecast_update* gibt Auskunft über den Aktualisierungszeitpunkt des Vorhersagewertes.

Ausblick

Zukünftig ist vorgesehen das Datenbankmodell um die Erfassung weiterer Sensordaten zu erweitern. Daten aus Regensensoren, Lichtsensoren, Temperatursensoren, ABS[9] Sensoren und andere für Analysezwecke geeigneten Quellen könnten in ein erweitertes Datenbankmodell aufgenommen werden und die Grundlage für weitere Prognosen und Vorhersagemodelle bilden. Für die Aufnahme weiterer Messwerte in die Datenbank, muss eine zusätzliche Datenbanktabelle angelegt werden, welche die neuen Sensorwerte aufnehmen kann. Alternativ lässt sich die Tabelle *history* für die Aufnahme zusätzlicher Sensorwerte erweitern, wenn diese im direkten Kontext einer Parklückenvermessung erfasst wurden.

[9] Anti-Blockier-System

4.3.3 Open-Data Hamburg



Für die Parkplatzbelegungsdetektion ist es erforderlich, die in der zu messenden Umgebung vorhandenen Parkplätze mit den zugehörigen, genauen GPS Koordinaten und Abmessungen zu erfassen. Eine manuelle Erfassung der Parkplätze ist zwar denkbar, allerdings mit einem hohen Arbeitsaufwand verbunden, um eine große Anzahl an Parklücken in einer ausreichenden Qualität zur Verfügung zu stellen. Daher wurde im Rahmen dieser Arbeit eine Quelle für Parkplatzdaten gesucht, welche einen möglichst aktuellen, frei verfügbaren und qualitativ hochwertigen Datenbestand für die benötigten Parkraumdaten liefert.

Die Stadt Hamburg verfügt über ein eigenes Open-Data Portal. Ziel des Open-Data Portals ist es, Informationen in Form von Rohdaten ohne Einschränkungen für alle Bürger, sowohl für private als auch für kommerzielle Zwecke, kostenfrei zur Verfügung zu stellen. Zu den bereitgestellten Rohdaten zählen beispielsweise Wetterdaten, Geodaten, Umweltdaten, Verkehrsinformationen sowie Statistiken. Die Daten können dabei aus unterschiedlichen Quellen stammen. Mögliche Quellen sind die städtische Verwaltung, die Privatwirtschaft, Hochschulen sowie weitere freiwillige Teilnehmer. Um einen uneingeschränkten Zugriff auf die Rohdaten zu ermöglichen, werden diese in freien Datenformaten, welche nicht an kommerzielle Software gebunden sind, veröffentlicht.

Am 6. Oktober 2012 trat in Hamburg das hamburgische Transparenzgesetz[10] in Kraft. Das Transparenzgesetz hat den Zweck, unter Wahrung des Schutzes personenbezogener Daten, behördliche Daten unmittelbar der Allgemeinheit zu Verfügung zu stellen. Im Zuge dessen wurde das Hamburger Transparenzportal[11] geschaffen. Das Transparenzportal vereint die Daten des Open Data Portal mit den, durch das Transparenzgesetz, gesetzlich geforderten Daten und ersetzt damit das ehemalige Open-Data Portal der Stadt Hamburg.

Unter den veröffentlichen Daten befinden sich als Geodaten unter anderem eine Vielzahl kartografierter Parklücken im Bereich der Stadt Hamburg, welche als grundlegende Datenbasis der SmartParking Plattform genutzt werden (Abbildung 4.5).

[10] http://www.hamburg.de/transparenzgesetz/
[11] http://transparenz.hamburg.de/

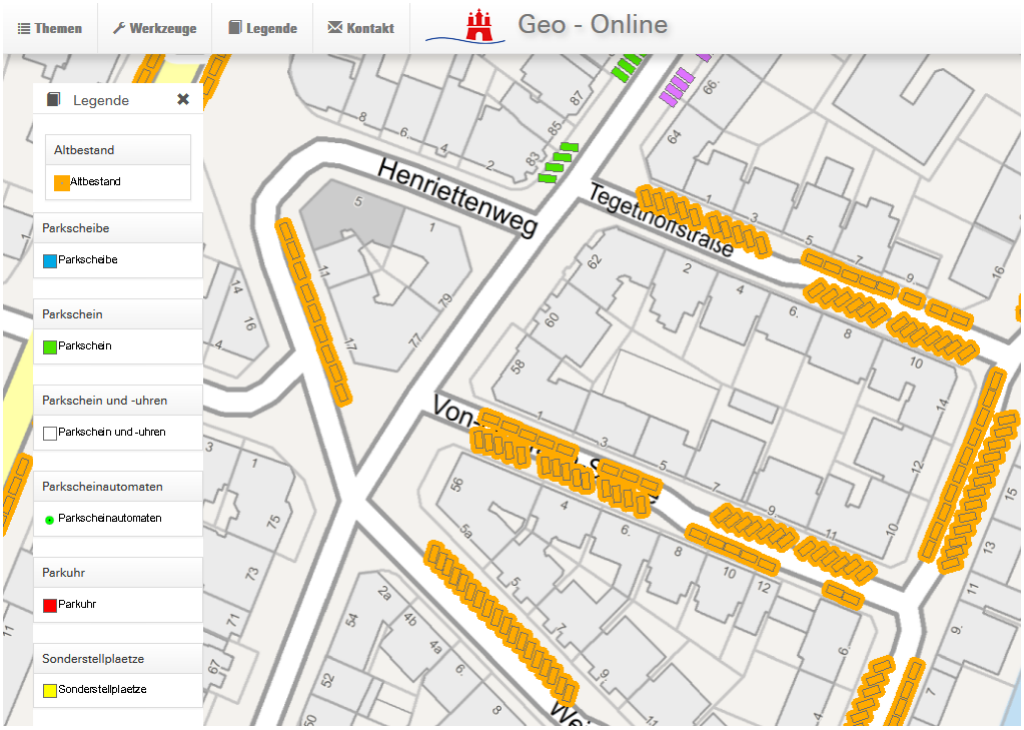


Abbildung 4.5: Kartografierte Parkplätze Transparenzportal

Ein Gespräch mit Dr. André Iost des LBV [12] in Hamburg ergab, dass in absehbarer Zeit (stand 2017) mit einer Erweiterung und Aktualisierung des digital erfassten Parkplatzbestandes der Stadt Hamburg zu rechnen ist. Die aktualisierten Datenbestände sollen ebenfalls frei verfügbar gemacht werden und eine überarbeitete Schnittstelle zur Abfrage der Daten erhalten.

[12] Landesbetrieb Verkehr

4.3.4 Anbindung an externe Parkplatzquellen

Für das SmartParking Projekt wurde entschieden, dass Fahrzeuge keinen internen Parkplatzbestand führen sollen, um die zu speichernde Datenmenge innerhalb des Fahrzeuges möglichst gering zu halten und eine Auswertung, basierend auf dem jeweils aktuellsten Datenbestand, zu ermöglichen. Das Fahrzeug fragt die benötigten Parklückenbestände daher von der Cloud-Server Plattform ab. Um eine möglichst schnelle Antwortzeit seitens der Cloud-Server Plattform sicherzustellen und für weitere Auswertungen einen eigenen Datenbestand zu führen, wird auf eine zusätzliche direkte Abfrage der Parkplatzdaten aus dem Open-Data Portal oder anderen externen Quellen verzichtet und die Parkplatzdaten direkt in den internen Datenbankbestand übernommen.

Durch die Übernahme ist eine permanente lokale Verfügbarkeit der Daten sichergestellt und keine Abhängigkeit, bezüglich der Verfügbarkeit externer Dienste, gegeben.

Um eine möglichst große Kompatibilität zu einer Vielzahl von externen Schnittstellen und verschiedenen Datenformaten für den Parkplatzimport sicher zustellen, wird der Import der externen Parkplatzquellen durch eigenständige Importprogramme vorgenommen. Die Importprogramme lassen sich auf die jeweils genutzte externe Schnittstelle anpassen und nutzen direkt die vom Crossbar Server bereitgestellten Schnittstellen zum einpflegen der Parklücken (Abschnitt 4.3.1).

Die Bereitstellung der kartografierten Parkplätze erfolgt für die Stadt Hamburg durch ein Geoinformationssystem. Bei einem Geoinformationssystem handelt es sich um ein System zur Erfassung, Bearbeitung, Verwaltung, Analyse und der genauen Darstellung von räumlichen Daten (Mennecke und Crossland (1996)). Die benötigten Parkplatzdaten sind im GML[13]-Format verfügbar. Das freie GML-Format wurde vom Open Geospatial Consortium festgelegt und mit der Norm ISO 19136:2007 standardisiert [14]. Die GML nutzt das XML[15]-Format zur Erfassung und Übertragung von sowohl geografischen als auch nicht geografischen Attributen.

[13] Geography Markup Language
[14] https://www.iso.org/standard/32554.html
[15] Extensible Markup Language

```
<gml:featureMember>
    <fme:Parkschein gml:id="id38d3db2c-0234-40bb-9829-1d921af59aef">
        <fme:OBJECTID>1</fme:OBJECTID>
        <fme:Fahrzeug>Pkw</fme:Fahrzeug>
        <fme:Oberflaechenbefestigung>Pflaster</fme:Oberflaechenbefestigung>
        <fme:HNR>20</fme:HNR>
        <fme:Kat>Parkschein</fme:Kat>
        <fme:AzuStr>längs</fme:AzuStr>
        <fme:BwZone>1</fme:BwZone>
        <fme:ParkAuto>418</fme:ParkAuto>
        <fme:HpDauer>60</fme:HpDauer>
        <fme:BrTaste>j</fme:BrTaste>
        <fme:Bparken>n</fme:Bparken>
        <fme:Akt>20110202</fme:Akt>
        <fme:PpGrp_ID>SH13</fme:PpGrp_ID>
        <fme:Strasse>Hopfensack</fme:Strasse>
        <fme:Bezirk>Hamburg-Mitte</fme:Bezirk>
        <fme:Stadtteil>Hamburg-Altstadt</fme:Stadtteil>
        <fme:STRSCHL>H6220</fme:STRSCHL>
        <fme:KML_ID>6</fme:KML_ID>
        <fme:Shape_Length>12.7785591069853</fme:Shape_Length>
        <fme:Shape_Area>8.77668760156111</fme:Shape_Area>
        <gml:surfaceProperty>
            <gml:Surface srsName="EPSG:4647" srsDimension="2">
                <gml:patches>
                    <gml:PolygonPatch>
                        <gml:exterior>
                            <gml:LinearRing>
                                <gml:posList>
                                    5933737.0678 32566203.3003
                                    5933737.4505 32566198.927
                                    5933739.4421 32566199.1013
                                    5933739.0594 32566203.4747
                                    5933737.0678 32566203.3003
                                </gml:posList>
                            </gml:LinearRing>
                        </gml:exterior>
                    </gml:PolygonPatch>
                </gml:patches>
            </gml:Surface>
        </gml:surfaceProperty>
    </fme:Parkschein>
</gml:featureMember>
```

Abbildung 4.6: Auszug GML Datei Parkraum GIS

Das Open-Data Portal Hamburg stellt die Parkplatzdaten als Archivdownload, bestehend aus GML-Dateien, zur Verfügung. Für die Datenübernahme wurde ein Importprogramm auf Basis von Node.js entwickelt, welches die GML-Datei in ein XML Document Object Model einliest, um so die gewünschten Objekte und Attribute extrahieren, vervollständigen, aufarbeiten und anschließend, anhand der Crossbar Schnittstellen (Abschnitt 4.3.1), an den Manager (Unterabschnitt 4.3.5) übermitteln zu können. Ein regelmäßiger Abgleich zwischen interner Datenbank und externen Datenquellen ist ebenfalls vorgesehen. Dadurch wird sichergestellt, dass der lokale Datenbestand auf einem aktuellen Stand bleibt. Der Abgleich des Datenbestandes stellt sicher, dass auch bei baulichen Veränderungen der Verkehrsstruktur, beispielsweise beim verlegen von Parklücken, der alte Datenbestand zu Prognosezwecken weiterhin verfügbar

bleibt. Der Aufbau eines GML Parkplatzobjektes kann der Abbildung 4.6 entnommen werden. Für den Datenimport wurde entschieden, zur Reduzierung des Speicherbedarfs, lediglich die für den SmartParking Betrieb erforderlichen Attribute in die Datenbank zu übernehmen. Durch die vorgenommene Selektion ergibt sich die, der Tabelle 4.1 zu entnehmende, Attributmenge für den Parkplatzimport.

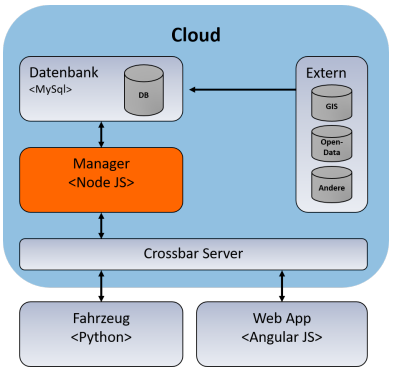
Attribut (*optional*)	**Bedeutung**	**Wertebereich**
KML_ID	Eindeutige ID	[String]
Fahrzeug	Fahrzeugart	Bus, Krad, Lkw, Pkw, Sonstige
Oberflaechenbefestigung	Oberflächenmaterial	Asphalt, Pflaster, Platten, Rasengittersteine, Wabenstein, Unbefestigt, Sonstiges
AzuStr	Ausrichtung	Längs, Quer, Schräg, Sammelparken, Straßenrand
Strasse	Zugehörige Straße	[String]
Kat	Art des Parkplatz	Absolutes Halteverbot, Eingeschränktes Halteverbot, Ohne Bewirtschaftung, Ohne Kategorie, Parkscheibe, Parkschein, Parkuhr, Reines Bewohnerparken, Sonderstellplätze
surfaceProperty	Parkplatz Polygon	[Polygon aus Koordinaten]

Tabelle 4.1: Selektierte Parkplatzattribute der GML Datei

Die Koordinaten des Parkplatz-Polygons sind im UTM[16]-Koordinatensystem als Koordinatenpaare angegeben. Dabei stellt der erste Wert eines Koordinatenpaares den Nordwert (Hochwert) und der zweite Wert den Ostwert (Rechtswert) innerhalb der UTM Zone an. Die UTM-Zone ist als Präfix dem Ostwert vorangestellt. Im Falle der Stadt Hamburg ist die entsprechende UTM-Zone die Zone 32. Für die SmartParking Anwendung wird durch den Einsatz von GPS Sensoren sowie die Beschaffenheit vieler APIs[17] das GPS-Koordinatenformat mit Längen- und Breitenangaben bevorzugt. Daher wird bei der Übernahme der Parkplatzdaten eine vorherige Umrechnung in das GPS-Koordinatenformat, durch das Importprogramm, vorgenommen.

[16] Universal Transverse Mercator
[17] Application Programming Interface

4.3.5 Manager



Der Manager fungiert als zentrale Verwaltungseinheit innerhalb des Cloudservers und ist das Bindeglied zwischen Farzeug, WebApp und Datenbank. An den Manager wurden die nachfolgend beschriebenen funktionalen Anforderungen gestellt. Die vom Crossbar Server weitergeleiteten *remote procedure call* Anfragen werden im Manager verarbeitet. Der Manager sendet Änderungen der Parkplatzbelegungsstände an die Interessenten. Er verwaltet Fahrzeugmesswerte und pflegt sie für spätere Prognosen in die Datenbank ein.

Nichtfunktionale Anforderungen

Der Manager muss vielen Clientanfragen, von Fahrzeugen, Endanwendern und externen Diensten, standhalten und Anfragen mit geringen Latenzen bearbeiten können. Gleichzeitig soll der Manager keine speziellen Hardwareanforderungen an die zugrunde liegende Infrastruktur (Server) stellen und schnell sowie performant arbeiten. Eine Anbindung an den Crossbar Server sowie die MySQL-Datenbank muss über entsprechende Softwarebibliotheken möglich sein.

Aus diesem Grund wurden verschiedene spezielle, serverseitige Softwareplattformen für Webanwendungen wie PHP, Python oder Node.js in Betracht gezogen. Entsprechende WAMP Bibliotheken für die Anbindung an den Crossbar Server und Anbindungsmöglichkeiten an die MySQL-Datenbank sind für alle drei genannten Softwareplattformen frei verfügbar.

Aufgrund der guten Performance bei einer großen Anzahl paralleler Anfragen und großer I/O Last wurde Node.js als geeignete Softwareplattform ausgewählt. Node.js ist eine auf Chrome´s Javascript runtime basierende Softwareplattform mit einem aktuell schnell wachsenden Bekanntheitsgrad und zeichnet sich besonders durch seine ressourcenschonende, performante Arbeitsweise, die ereignisgesteuerte Architektur sowie die große Anzahl möglicher paralleler Verbindungen aus (Lei u. a. (2014)) (siehe Abbildung 4.7).

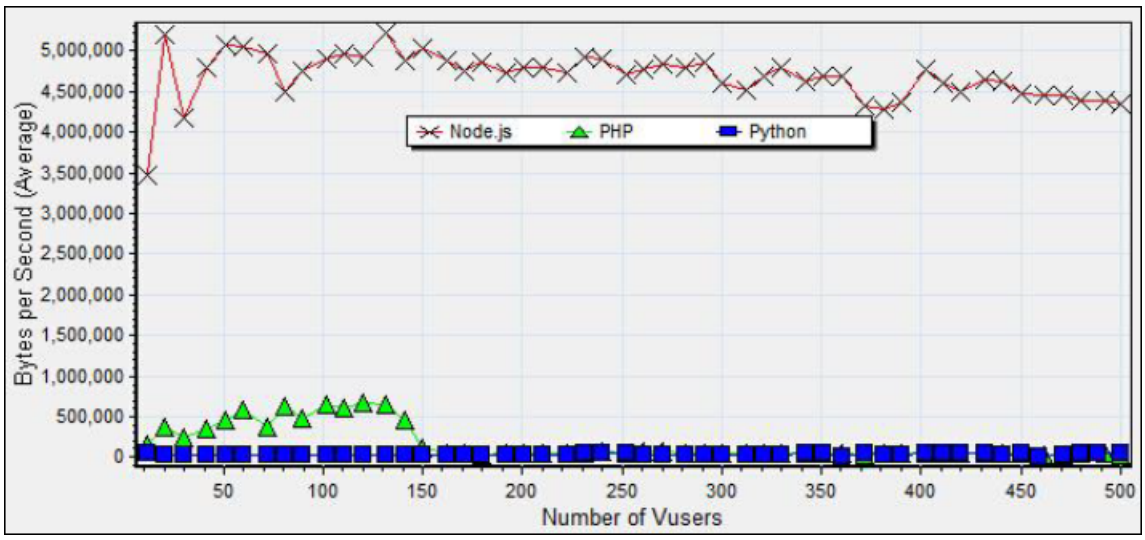


Abbildung 4.7: Durchsatzvergleich Node.js (Lei u. a. (2014))

Ein hoher Durchsatz bei kurzen Anfragen sowie kleinen Datenpaketen ist bei einer großen Anzahl an Messfahrzeugen und Endnutzern im Produktivbetrieb erforderlich. Fahrzeuge fragen in regelmäßigen Abständen Parkplatzinformationen vom Manager ab und veröffentlichen neue Messwerte. Auch durch das Verteilen der aktualisierten Parkraumsituation an Endanwender ist ein hoher Durchsatz notwendig.

Für die Anbindung an die MySQL-Datenbank kommt die Fremdbibliothek mysql[18] zum Einsatz, welche sich neben ihrer großen Verbreitung, durch die Unterstützung von Connection-Pools, auszeichnet. Durch die Nutzung von Connection Pools lassen sich parallele Verbindungen zu der MySQL-Datenbank aufbauen und so die Durchsatzvorteile von Node.js optimal nutzen.

Als WAMP Clientbibliothek wird Autobahn.js[19] verwendet, welche durch das Crossbar Entwicklerteam bereit gestellt wird und somit sowohl eine hohe Zuverlässigkeit als auch eine gute Kompatibilität zu dem verwendeten Crossbar Server gewährleistet.

[18] https://github.com/mysqljs/mysql
[19] http://autobahn.ws/js/

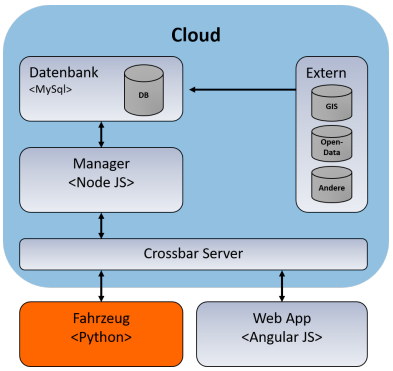
Umsetzung

Der Manager stellt eine Verbindung zur MySQL-Datenbank her und hält einen Connection Pool für die zu tätigenden Datenbankanfragen bereit. Zusätzlich wird eine Verbindung zum Crossbar Server aufgebaut. Die unter Abschnitt 4.3.1 aufgeführten Schnittstellen wurden gemäß der Spezifikation implementiert und bei dem Crossbar Server, unter den entsprechenden Topics, registriert.

Qualitätssicherung

Als zentrales Bindeglied zwischen allen Komponenten muss der Manager fehlerfrei arbeiten und eine hohe Verfügbarkeit gewährleisten. Steht der Manager nicht zur Verfügung, ist kein Datenfluss zwischen Fahrzeug, Web-App und Datenbank möglich. Der Ausfall des Managers würde somit den vollständigen Ausfall der Cloudplattform bewirken. Daher wurde der Manager einem Crashtest unterzogen bei dem gezielt, durch fehlerhafte und nicht der Spezifikation entsprechende Anfragen an den Manager, versucht wurde einen Absturz der Cloudplattform zu erzielen. Die funktionale Richtigkeit des Managers konnte anhand von Black-Box-Tests sicher gestellt werden. Die Testfälle des Black-Box-Tests wurden anhand der Schnittstellenspezifikation (Abschnitt 4.3.1) erarbeitet.

4.3.6 Fahrzeug



Das Fahrzeug dient als Sensorknoten für die Messung der Parkplatzbelegungsstände. Die konkrete Implementierung der Fahrzeugsoftware wurde von der Firma IAV Automotive Engineering[20] vorgenommen. Die Einschränkung auf die Vermessung von Parklücken ist lediglich durch die beschriebene Versuchsanwendung zur Detektion von freiem Parkraum gegeben. Neben der Vermessung von Parklücken ist es nach einer Erweiterung der Cloudserver Plattform angedacht, aus der Sicht auf das Fahrzeug als Sensorknoten, weitere Daten wie Fahrgeschwindigkeit, Regensensoren, Helligkeitssensoren, ABS-Sensoren sowie weitere vorhandene Sensorquellen an die Cloudserver Plattform zu übermitteln. Durch die zusätzliche Erfassung weiterer Sensordaten wird eine Datenbasis für verschieden Analyse- und Prognoseszenarien geschaffen.

An die Fahrzeugsoftware zur Parklückenvermessung wurden definierte Anforderungen gestellt. Diese sind folgend aufgeführt.

Funktionale Anforderungen

Das Fahrzeug muss die Parklücken eigenständig vermessen können. Die erfassten Messwerte der vermessenen Parkplätze werden, mit der zugehörigen Messsicherheit und dem Messzeitpunkt behaftet, an die SmartParking Cloud übermittelt. Die für die Messung des Belegungsstandes notwendigen Parkplatzbestände sowie Parkplatzattribute, im zu vermessenden Gebiet, sollen vom Cloud-Server abgerufen werden.

Hardware Anforderungen

Die Fahrzeugsoftware soll, im Rahmen der Versuchsanwendung, auf einem Entwicklungslaptop als Car-PC, innerhalb des Fahrzeug, betrieben werden. Für die Übermittlung der vermessenen Parklücken sowie die Abfrage der relevanten Parkplatzbestände, werden die Crossbar Schnittstellen (Abschnitt 4.3.1) des Cloudservers verwendet. Für den Zugriff auf den Cloudserver muss ein Internetzugang für den Car-PC bereitgestellt werden. Das Fahrzeug muss seine aktuelle Position für die Vermessung der Parklücken sowie die Abfrage der Parkplatzdaten anhand von GPS-Sensoren eigenständig ermitteln. Für die Vermessung der Parklücken sollen die im

[20] https://www.iav.com/

Serienfahrzeug vorhandenen Ultraschallsensoren (Abbildung 4.8) verwendet und direkt, mit Hilfe eines CAN-Bus Interface, über die CAN-Bus Schnittstelle des Fahrzeugs abgefragt werden.

Verwendete Hardware

In der nachfolgenden Auflistung wird eine Übersicht über die verwendete Hardware, zur Parkplatzvermessung, gegeben.

- PDC[21] Sensoren des Golf 7 Versuchsfahrzeug
- Car-PC auf Basis eines ThinkPad T470 Laptop
- Mobiler LTE-Router
- PEAK CanBus Interface[22]
- u-Blox GPS Modul[23]

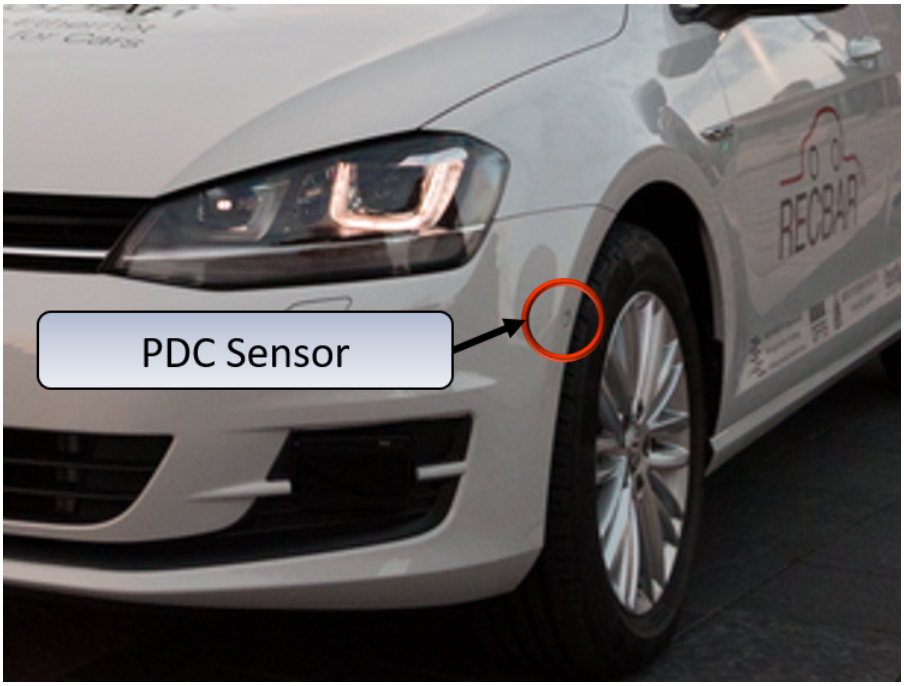
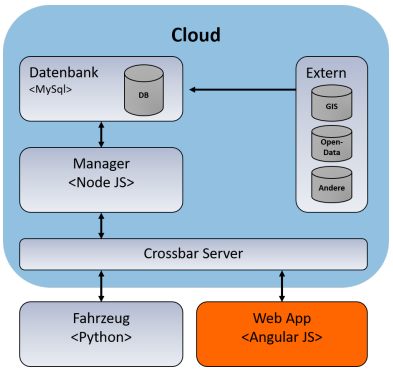


Abbildung 4.8: PDC Sensor des Golf 7 Versuchsfahrzeuges

[21] Park Distance Control
[22] http://www.peak-system.com/
[23] https://www.u-blox.com/de

4.3.7 Web-App



Die Web-App ist eine der möglichen Anwendungen für Endbenutzer, um eine Übersicht über die aktuellen Parkplatzbelegungsstände, in den von SmartParking Fahrzeugen (Unterabschnitt 4.3.6) vermessenen Regionen, zu erhalten. Sie ist dabei als eine von vielen denkbaren Endanwendungen zu sehen. Durch die Schnittstelle zum Cloudserver (Abschnitt 4.3.1) können weitere Anwendungen für Endanwender realisiert werden. Mögliche weitere potenzielle Nutzer der Schnittstelle können Anbieter von Navigationssystemen, Verkehrsportalen oder Kartenanportalen sein.

Funktionale Anforderungen

Endanwender sollen, mithilfe der Web-App, unter Angabe der Zieladresse und einem Suchradius, die im Suchgebiet vorhandenen Parkplätze, mit dem dazugehörigen Belegungsstatus, auf einer Karte angezeigt bekommen. Um den Belegungssatatus visuell erkenntlich zu machen, müssen die Parkplätze farblich markiert werden. Dabei wird zwischen den Farben rot (Parkplatz sicher belegt), grau (Belegungsstatus unbekannt) und grün (Parkplatz sicher frei) unterschieden. Die Messsicherheit ist dabei in linearen Abstufungen der Farbintensität, zwischen rot und grau (unsicher belegt) sowie grün und grau (unsicher frei), kenntlich zu machen. Die aktualisierte Darstellung der Belegungswerte erfolgt automatisch, ohne zusätzliche Interaktion mit dem Anwender. Nach Selektion eines Parkplatzes auf der Karte werden zusätzliche Parkplatzattribute (Tabelle 4.1) angezeigt. Eine alternative Darstellungsmöglichkeit zur Karte ist durch eine Listenansicht umzusetzen. Sie ist nach Entfernung der Parkplätze zum Suchort sowie dem Belegungsstatus zu sortieren, um so eine Übersicht der nächstgelegenen, freien Parklücken zu liefern.

Hardware Anforderungen

Die Web-App soll sowohl auf Smartphones (iOS, Android, Windows Phone) als auch auf Desktop Systemen lauffähig sein, damit eine große Anzahl an Endanwendern zu erreichen ist. Zur Kommunikation mit dem Cloudserver werden die Crossbar-Schnittstellen (Abschnitt 4.3.1) verwendet. Das *onOccupancy* Topic wird dabei genutzt, um den aktuellen Belegungsstatus in Echtzeit, ohne *pollen*, empfangen zu können.

Umsetzung

Zur Erfüllung der Anforderung der Unabhängigkeit von Endgeräten, wurde sich für die Umsetzung in Form einer Web-App entschieden. Native Endanwendungen für beispielsweise iOS oder Android erfordern, durch die Nutzung unterschiedlicher primärer Programmiersprachen, unterschiedlicher APIs und verschiedener Designrichtlinien, eine Mehrfachentwicklung oder Anpassung der Endanwendung an die zugrunde liegende Plattform. Im Gegensatz zu nativen Anwendungen, sind Web-Apps nicht von den plattformabhängigen APIs abhängig. Es werden gängige Web Standards wie HTML5[24], CSS[25], Javascript[26] verwendet, welche unabhängig der Plattform, mit den Webbrowsern des jeweiligen Endgerätes, ausgeführt werden (de Andrade u. a., 2016).

Als Webframework wurde sich für AngularJS[27] entschieden. AngularJS ist ein, auf JavaScript basierendes, von Google entwickeltes, Webframework für Webanwendungen, welches aktiv weiter entwickelt wird und nach dem Model-View-Controller Design Pattern (Abbildung 4.9) arbeitet.

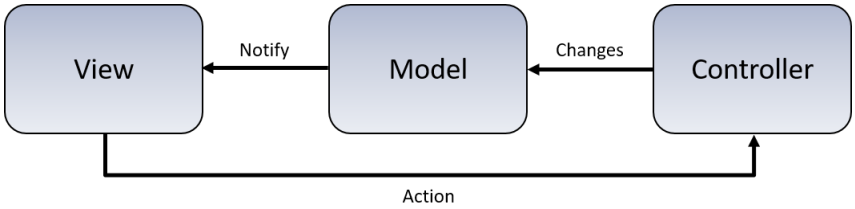


Abbildung 4.9: Model-View-Controller Design Pattern

AngularJS ermöglicht es die statischen HTML Seiten um dynamische Ansichten zu erweitern, ohne dabei auf eine direkte Manipulation des Document Object Model zurückgreifen zu müssen.

Für die Integration der Kartenansicht kommt der weitverbreitete Kartendienst Google Maps[28] zum Einsatz. Google Maps ermöglicht, durch entsprechende APIs, das dynamische Anlegen von Objekten wie Markierungen, Kreisen und Polygonen, welche zur Darstellung von Parkplätzen und Suchgebieten verwendet werden. Eine entsprechende Einbindung in AngularJS ist durch

[24] https://developer.mozilla.org/de/docs/Web/HTML/HTML5
[25] https://www.w3.org/standards/techs/css
[26] https://developer.mozilla.org/de/docs/Web/JavaScript
[27] https://angularjs.org
[28] https://developers.google.com/maps

die Fremdbibliothek ng-map[29] gegeben.
Eine Anbindung an die Crossbar Schnittstelle des Cloudservers wurde mit Hilfe der angular-wamp[30] Bibliothek realisiert.

Um die Web-App auf mobilen Endgeräten wie eine native Anwendung aussehen zu lassen und Endanwendern das gewohnte Nutzungsgefühl einer nativen Anwendung zu bieten, wurde für die Gestaltung der Anwendungsoberfläche das ebenfalls von Google entwickelte UI[31] Framework Angular-Material[32] verwendet. Durch die Einhaltung der Material design Richtlinien[33], welche auch bei nativen Anwendungen auf dem Android Betriebssystem zum Einsatz kommen, vermittelt die Web-App den Eindruck einer nativen Anwendung (Abbildung 4.10).

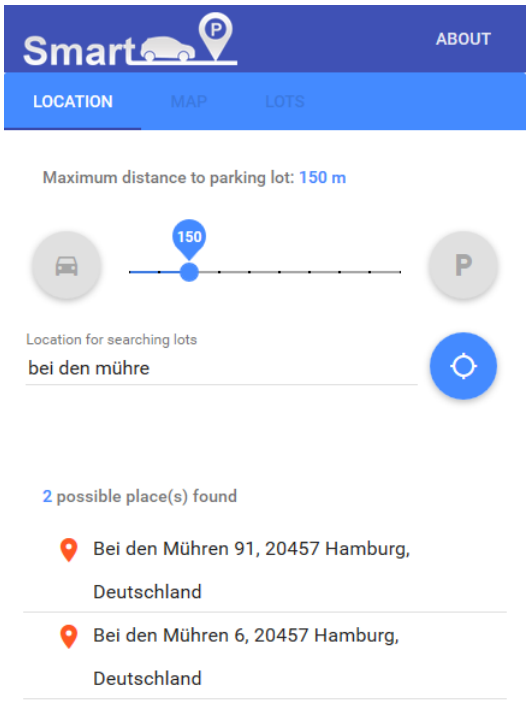


Abbildung 4.10: Web-App UI mit Angular-Material

[29] https://ngmap.github.io/
[30] https://github.com/voryx/angular-wamp
[31] User Interface
[32] https://material.angularjs.org
[33] https://material.io/guidelines/

Damit dem Endanwendern die Suche nach der Adresse des gewünschten Zielortes erleichtert wird, erscheint bei der Eingabe weniger Buchstaben des Zielortes, eine Vorhersage der möglichen Zieladresse, welche anhand einer Liste von Adressvorschlägen visualisiert wird (Abbildung 4.10). Für die Adressvorschläge kommt die Google Maps Geocoding API[34] zum Einsatz. Alternativ lässt sich, durch die Bestimmung des aktuellen Standortes, die Adresse des momentanen Aufenthalts ermitteln und als Zielort setzen.

Die Parkplätze werden in der Kartenansicht als Polygone aus GPS Koordinaten dargestellt. Um die Parkplatzdaten, für das ausgewählte Suchgebiet, unter Berücksichtigung des eingestellten Suchradius zu empfangen, wird die Crossbar Schnittstellenfunktion *getLotsInfo* (Abschnitt 4.3.1) verwendet. Als Rückgabewert wird ein Array, über alle im Suchgebiet gefundenen Parkplätze mit den angeforderten Attributen, zurückgegeben. Anschließend erfolgt die Überführung der Parkplätze in google Maps Polygone. Die ng-map Komponente verwendet die *Simple Polygons*[35] Schnittstelle der google Maps API zur Verwaltung der Polygone. In späteren Tests zeigte sich, dass *Simple Polygons* bei einer großen Anzahl an Parkplätzen nicht performant genug arbeitet und zu einer hohen CPU-Auslastung führt. Daher wurde die ng-map Komponente in der finalen Web-App lediglich für die Darstellung von Markierungen auf der Karte sowie die Verwaltung von *click Handlern* verwendet. Die Verwaltung der Parkplatzpolygone wurde durch einen direkten Zugriff auf die Maps API, unter Verwendung von performanteren *Polygon Data Layern*[36], realisiert.

Eine Veranschaulichung des Suchradius erfolgt durch einen Kreis auf der Karte, welcher den aktiven Suchradius visualisiert und so bei der Orientierung hilft (Abbildung 4.11). Durch einen Klick auf ein Parkplatzpolygon in der Karte, wird eine Detailansicht mit den zusätzlichen Attributen der Parklücke (wie die Zeit der letzten Belegungsmessung oder die Art der Parkplatzbewirtschaftung) angezeigt und das Zoom-Level, zur besseren Übersicht, auf den ausgewählten Parkplatz gesetzt. Eine zusätzliche Listenansicht, sortiert nach Entfernung und Belegungsstatus, hilft dem Anwender den nächstgelegenen, freien Parkplatz zu finden (Abbildung 4.12). Die Auswahl eines Parkplatzes aus der Liste führt zu einer Markierung des Parkplatzes in der Kartenansicht.

[34] https://developers.google.com/maps/documentation/geocoding
[35] https://developers.google.com/maps/documentation/javascript/examples/polygon-simple?hl=de
[36] https://developers.google.com/maps/documentation/javascript/examples/layer-data-polygon

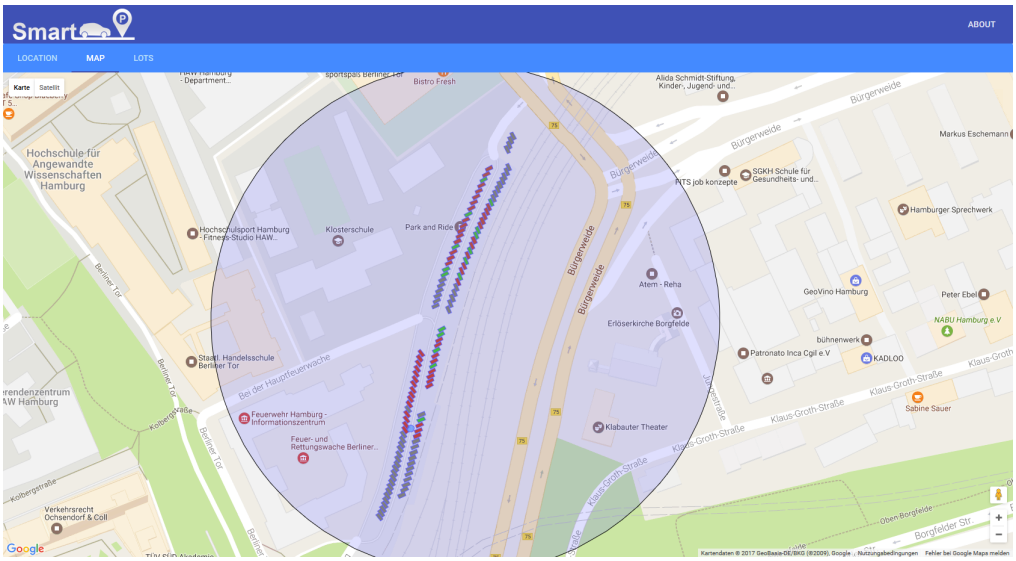


Abbildung 4.11: Web-App Kartenansicht

Abbildung 4.12: Web-App Listenansicht

Dank der, nach dem *Publish & Subscribe* Verfahren, implementierten Schnittstellenfunktion *onOccupancy* (Abschnitt 4.3.1) empfängt die Web-App die vom Manager (Unterabschnitt 4.3.5) bereitgestellten Änderungen des Belegungsstatus in nahezu Echtzeit und aktualisiert den Belegungsstatus der jeweiligen Parkplätze.

Damit auch bei unregelmäßig, in zeitlich großen Abständen, vermessenen Gebieten eine plausible Darstellung der Parkraumsituation möglich ist, wurde eine Verfallsfunktion für die Belegungsmesswerte implementiert (Abbildung 4.13).

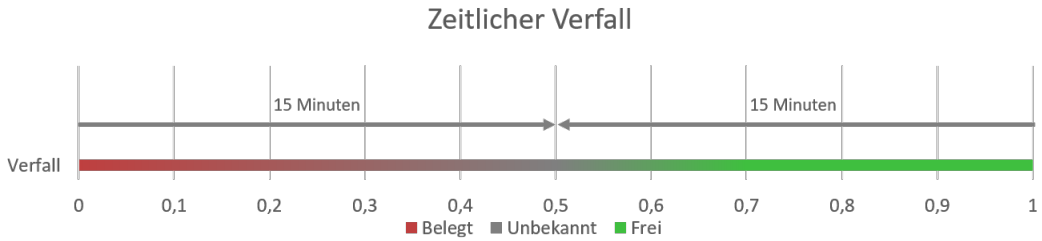


Abbildung 4.13: Verfallsfunktion der Belegungsmessung

Die lineare Verfallsfunktion über 15 Minuten von *belegt* (0.0) und *frei* (1.0) zu *unbekannt* (0.5), orientiert sich an der in Hamburg für Kurzparker vorgesehenen Kurzparkdauer (ARGUS, 2009).

4.4 Serverbetrieb

Um die entwickelte SmartParking Plattform in Betrieb nehmen zu können, ist es erforderlich Manager, Datenbank, Crossbar Server und Web-App auf einem Server aufzusetzen. Der benötigte Linuxserver wurde von der Firma IAV Automotive Engineering bereit gestellt. Für den Testbetrieb während der Entwicklung sowie für spätere Weiterentwicklungen wurde entschieden eine lokal lauffähige, vom Server unabhängige, Testumgebung zu schaffen.

Daher wurde sich für eine virtualisierte Laufzeitumgebung, mit Hilfe von Docker[37], entschieden. Dockercontainer[38] sind ressourcenschonende, virtuelle Maschinen die eine Kapselung sowie Ressourcentrennung, zwischen Container und dem Serverbetriebssystem, ermöglichen. Im Vergleich zu herkömmlichen Virtualisierungsumgebungen, wie VMware[39] oder Virtualbox [40], laufen Dockercontainer im Hostbetriebssystem und nutzen dessen Ressourcen. Dies führt zu kleineren Images und weniger Ressourcenbedarf.

Die Entwickler des Crossbar Server und der MySQL Datenbank stellen bereits fertige Dockerimages bereit. Für den Manager und die Web-App wurden eigene Dockerimages erstellt und entsprechend konfiguriert. Durch diese Form der Virtualisierung ist es möglich sowohl die komplette SmartParking Umgebung, als auch Teilkomponenten, auf einem lokalen System aufzusetzen. Nach abgeschlossener Entwicklung werden die Dockerimages direkt auf den vorgesehenen Server transferiert und dort als Dockercontainer ausgeführt, ohne zusätzliche Konfigurationen oder Installationen auf dem Server vornehmen zu müssen.

[37] https://www.docker.com/
[38] https://www.docker.com/what-container
[39] http://www.vmware.com/de.html
[40] https://www.virtualbox.org/

4.5 Evaluation

Damit die korrekte Kommunikation zwischen der Cloud-Server Plattform, dem Fahrzeug und der Web-App sichergestellt werden kann, wurde eine Reihe von Versuchsfahrten unter realen Bedingungen durchgeführt. Als Teststrecke dient die Park and ride Anlage am Westphalenweg in Hamburg. Diese eignet sich besonders durch den regelmäßigen Wechsel der Parkauslastung sowie einer guten GPS Abdeckung. Die Versuchsfahrten wurden verkehrsbedingt mit einer Geschwindigkeit von 30 km/h durchgeführt. Die Parkplätze im Versuchsgebiet sind quer zur Fahrbahn ausgerichtet.

Vermessen wurde die rechte Fahrbahnseite. Auf der Teststrecke sind 61 zu vermessende Parklücken vorhanden. Für die Auswertung der Messfahrten wurde während der Fahrt die reale Auslastung der Parklücken ermittelt und anschließend mit den im Cloudserver erfassten Belegungswerten verglichen.

Die Vermessung der Parklücken anhand der Ultraschallsensoren erfolgt mit 20 Hz. Die Messfrequenz ist durch die Ausgabefrequenz des Steuergerätes im Serienfahrzeug (Abschnitt 4.3.6) begrenzt. Bei der durchgeführten Messfahrt mit 30 km/h ergeben sich annähernd sechs Messpunkte, bei einer angenommenen Breite der Parklücke von durchschnittlich 2,30 m und einer queren Ausrichtung zur Fahrbahn. Es ist anzunehmen, dass sich bei einer maximalen Fahrtgeschwindigkeit im innerstädtischen Bereich von 60 km/h, mit drei Messpunkten pro Parklücke, vergleichbare Messwerte erzielen lassen. Bei einer Ausrichtung der Parklücken mit einer Ausrichtung längs zur Fahrbahn steigt die Anzahl der Messpunkte pro Parklücke entsprechend an.

Die im Folgenden dargestellte Auswertung der Messergebnisse basiert auf vier aufeinanderfolgenden Messfahrten bei einer gleich bleibenden Parkraumauslastung, um die Zuverlässigkeit der Parklückenvermessung beurteilen zu können (Abbildung 4.14).

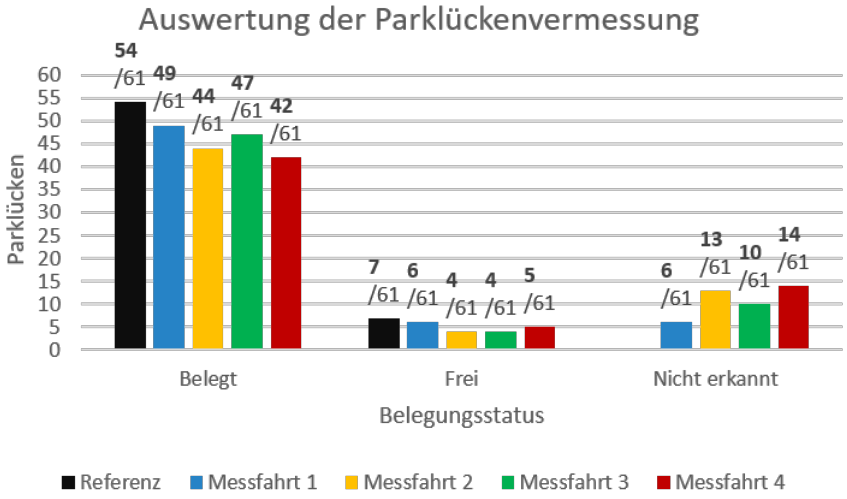


Abbildung 4.14: Auswertung der Parklückenvermessung

Aus den oben gezeigten Messwerten (Abbildung 4.14) ergibt sich die nachfolgend dargestellte Fehlerrate der Parklückenvermessung. (Abbildung 4.15).

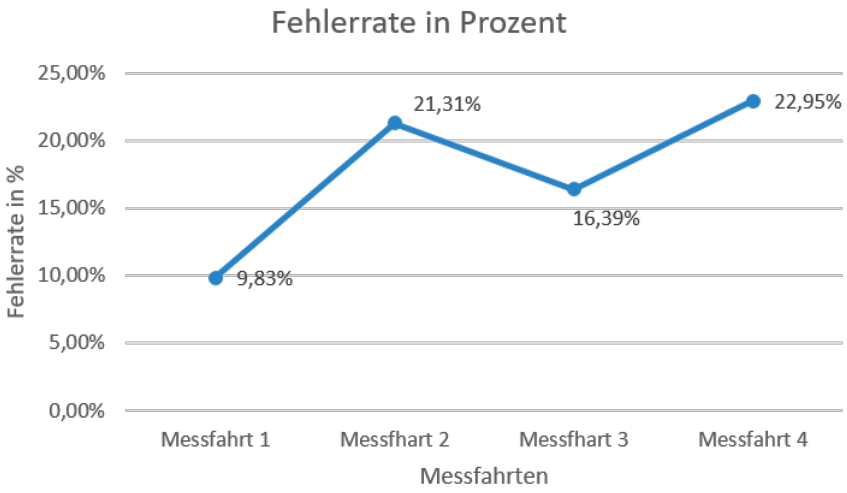


Abbildung 4.15: Fehlerrate der Parklückenvermessung

Die Fehlerrate von durchschnittlich 17,62% ergibt sich vorrangig aus den nicht erkannten, beziehungsweise nicht vermessenen, Parklücken. Dies ist auf die, auch bei gutem GPS-Empfang teilweise starke, Abweichung bei der Lokalisierung zurückzuführen. Der genutzte Algorithmus für die Vermessung der Parklücken wurde von der Firma IAV Automotive Engineering entwickelt und implementiert. Er steht daher nicht im direkten Fokus dieser Arbeit. Die Firma IAV bestätigte, dass sich die Anzahl nicht vermessener Parklücken, die Qualität der Belegungsmessung sowie die Fahrzeuglokalisierung durch entsprechende Optimierungsmaßnahmen der Algorithmen und einen Wechsel der Lokalisierungstechnik weiter verbessern lässt.

Da die Parklückenvermessung nicht im Fokus der Arbeit steht wurde exemplarisch eine Auswertung bezüglich der Wiederholgenauigkeit der Parklückenvermessung, bei einer gleich bleibenden Parkraumsituation, durchgeführt. Für eine umfassendere Bewertung der Parklückenbelegungsmessung sind weitere Testfahrten mit wechselnder Parkraumbeschaffenheit und Auslastung erforderlich.

Durch die Testfahrten, konnte die korrekte Funktionsweise der einzelnen Komponenten der SmartParking Plattform unter realen Bedingungen getestet werden. Dazu wurden die vom Fahrzeug erzeugten Messwerte sowie die reale Parkraumsituation protokolliert und mit den in der Datenbank erfassten Werten abgeglichen. Es konnten keine Abweichungen oder unvollständige Messdaten im Datenbestand der Datenbank ermittelt werden. Auch die vom Fahrzeug, zur Belegungsmessung, angeforderten Parkraumdaten wurden ohne Verbindungsabbrüche oder Fehler korrekt übertragen. Die Änderungen der Parkraumbelegung wurden automatisch an die WebApp übertragen und fehlerfrei dargestellt.

Bei den Testfahrten zeigten sich Fehler im importierten Parklückenbestand (Unterabschnitt 4.3.4). Teilweise sind Parklücken der externen Datenquelle fehlerhaft ausgerichtet. Um eine Qualitätssteigerung zu erreichen, muss der Parklückenbestand zukünftig von fehlerhaften Einträgen bereinigt werden.

4.6 Auswertung und Ergebnisse

Das Zusammenspiel zwischen den einzelnen Komponenten der Cloudserver Plattform sowie Fahrzeug und Web-App hat sich als zuverlässig und praktikabel herausgestellt.

Die Qualität des Parkplatzbestandes muss für einen großflächigen Betrieb verbessert werden. Teilweise fehlende oder falsch ausgerichtete Parklücken im importierten Datenbestand stören den Einsatz in einem produktiven Umfeld.

Die Cloudplattform ermöglicht externen Diensten den Zugriff auf die Parkraumbelegungsdaten. Dadurch wird beispielweise eine Einbindung in Navigationssysteme ermöglicht, die es Nutzern erlaubt direkt, zur nächstgelegenen, freien Parklücke navigiert zu werden.

Die Vermessung der Parklücken mittels fahrzeuginterner Sensorik hat sich als durchaus zuverlässig herausgestellt. Die erhöhte Anzahl nicht vermessener Parklücken kann bereits durch Optimierungen des Algorithmus reduziert werden. Für eine weitere Steigerung in der Zuverlässigkeit der Messergebnisse ist ein präziseres, weniger anfälliges Verfahren zur Fahrzeuglokalisierung nötig.

Durch eine zukünftige Erweiterung der Cloudplattform, um verschiedenartige Fahrzeugsensoren, wie beispielsweise Regensensoren, Helligkeitssensoren oder ABS-Sensoren, kann eine umfangreiche Datenbasis für verschiedene Analysen und Prognosen geschaffen werden.

5 Vorhersagemodell zur Prognose von freiem Parkraum

In diesem Kapitel werden Vorhersagemodelle für die Prognose von freiem Parkraum, anhand der in Kapitel 4 vorgestellten Cloud Architektur, nach dem KDD-Prozess (Unterabschnitt 3.1.1) entwickelt. Ziel ist die selektive Prognose der Parkraumauslastung unter der Berücksichtigung des Zielortes, des Datums und der Tageszeit. Es werden verschiedene Vorhersageverfahren verglichen und abschließend bewertet (Abschnitt 5.3.5).

5.1 Ziele der Parkraumprognose

Neben der Erfassung der aktuellen Belegungsstände ist es sinnvoll, basierend auf den in der Vergangenheit erfassten Daten, Rückschlüsse auf die zukünftige Parksituation zu ziehen. In Gebieten die selten von entsprechenden Messfahrzeugen durchfahren werden kann, anhand vergangener Messungen, die aktuelle Wahrscheinlichkeit, in einem bestimmten Bereich, eine freie Parklücke zu finden ermittelt werden.

5.2 Toolchains

Für die Durchführung des Data Mining gibt es verschiedene Softwarepakete, die den Anwender bei der Durchführung des Data Mining Prozesses unterstützen. In die engere Auswahl kamen drei, im professionellen Umfeld genutzte, Softwareumgebungen (Stefan Lanig, 2010) Orange[1], RapidMiner Studio[2] und KNIME[3]. Nachfolgend werden die verschiedenen Softwareumgebungen evaluiert, um ein geeignetes Werkzeug für die spätere Parkraumprognose zu ermitteln. Alle drei Softwareumgebungen beinhalten einen grafischen Editor mit Funktionsblöcken zur Erstellung der Analyseabläufe, eine grafische Visualisierung der Ergebnisse des Data Mining sowie Schnittstellen zur Ausführung von Scripten.

Orange ist eine von der University of Ljubljana entwickelte Open Source Software, die unter der GNU[4] Lizenz frei verwendbar ist. RapidMiner Studio steht ebenfalls unter der GNU Lizenz zur Verfügung, allerdings nur in einer Version mit reduziertem Funktionsumfang und einer Beschränkung auf 10000 analysierbare Datensätze[5]. KNIME ist an der Universität Konstanz entstanden und steht wie Orange mit einem uneingeschränkten Funktionsumfang unter der GNU Lizenz zur freien Verfügung.

Im Falle der Versuchsanwendung zur Detektion von freiem Parkraum liegen die Datensätze in Form einer MySQL Datenbank vor (Unterabschnitt 4.3.2). Damit der zusätzlichen Schritt des exportierens der Datensätze aus der Datenbank in ein von der Mining Software unterstütztes Datenformat vermieden werden kann ist es wünschenswert, dass die Softwareumgebung einen direkten, lesenden Zugriff auf die MySQL Datenbank vornehmen kann. In Orange wurde die MySQL Anbindung erst in der aktuellen Version 3 eingeführt und weist im Vergleich zu RapidMiner Studio und KNIME einen geringeren Funktionsumfang auf. Die für die Parkraumanalyse durchzuführenden Verfahren Neuronale Netze, Entscheidungsbäume und Naive Bayes (Abschnitt 3.2) werden ebenfalls von allen drei Softwareumgebungen unterstützt.

Alle evaluierten Tools weisen einen vergleichbaren Funktionsumfang auf. Aufgrund der freien Verfügbarkeit, einer aktiven Weiterentwicklung und einer umfangreichen Anbindungsmöglichkeit an MySQL Datenbanken wurde sich für den Einsatz der KNIME Software entschieden.

[1] http://orange.biolab.si/
[2] https://rapidminer.com/products/studio/
[3] https://www.knime.org/
[4] https://www.gnu.org/licenses/licenses.de.html
[5] https://rapidminer.com/pricing/

5.3 Prozess der Parkraumprognose

Nachfolgend wird anhand des KDD Prozesses (Unterabschnitt 3.1.1) ein Modell zur Parkraumvorhersage, auf Basis der durch die SmartParking Cloudplattform (Kapitel 4) erfassten Messdaten, entwickelt.

5.3.1 Vorbereitung und Vorverarbeitung

Gemäß den Prozessschritten Vorbereitung (Abschnitt 3.1.1) und Vorverarbeitung (Abschnitt 3.1.1) des KDD Prozesses, werden im Folgenden die für die Analyse (Abschnitt 3.1.1) benötigten Datensätze selektiert, ergänzt, bereinigt und durch Transformation in eine analysierbare Form gebracht.
Die für die Analyse erforderlichen Messdaten sind, in Form einer Historie, in einer MySQL Datenbank (Unterabschnitt 4.3.2) abgelegt. Die Attribute für die Analyse werden den Datenbank Tabellen *history* und *lots* entnommen. Externe Attribute, in Form der Parkplatzdaten, wurden bereits in den internen Datenbankbestand der Tabelle *lots* überführt. Die Tabelle *propability* enthält keine für die Analyse geeigneten Attribute. Der Tabelle 5.1 zu entnehmenden Attribute aus der Datenbank wurden für die spätere Analyse selektiert.

Attribut	**Bedeutung**	**Wertebereich**
vehicle	Fahrzeugart	[String] Bus, Krad, Lkw, Pkw, Sonstige
street	Zugehörige Straße	[String]
type	Art des Parkplatz	[String] Absolutes Halteverbot, Eingeschränktes Halteverbot, Ohne Bewirtschaftung, Ohne Kategorie, Parkscheibe, Parkschein, Parkuhr, Reines Bewohnerparken, Sonderstellplätze

Tabelle 5.1: Selektierte Datenbankatrribute für den KDD Prozess

Es wurde ermittelt, dass die Parkplatzattribute *lot_id* (eindeutige Parkplatzidentifikationsnummer), *surface* (Oberflächenbeschaffenheit des Parkplatzes), *orientation* (Ausrichtung zur Fahrbahn) und *contour* (Parkplatzkoordinaten) für die Analyse keine geeigneten, relevanten Merkmale beinhalten.

Endnutzern sollen die Belegungswahrscheinlichkeiten im Suchgebiet, anhand von Position und Suchradius, angezeigt bekommen. Da eine Analyse für jeden einzelnen Parkplatz nicht repräsentativ ist, müssen die Parkplätze in Gruppen zusammengefasst werden. Dabei ist es möglich die Parkplätze, in Form von geografischen Zonen, anhand ihrer Koordinaten zu gruppieren. Es muss beachtet werden die Größe und Lage der geografischen Zonen so zu wählen, dass eine gleichmäßige Verteilung aller Parklücken auf die entsprechenden Zonen erfolgt, damit in den Zonen eine vergleichbare, repräsentative Parklückenmenge vorhanden ist. Ein alternativer Ansatz ist es, die Parkplätze anhand der zugehörigen Straße zusammenzufassen. Im Gegensatz zu einer Einteilung in feste, geografische Zonen haben Straßen keine einheitliche Länge, sodass die Aussagekraft der Prognose bei langen Straßen mit vielen Parkplätzen verfälscht werden kann. Um den Fokus der Arbeit auf die Erstellung eines Modells zur Parkraumprognose legen zu können, wurde sich gegen den zeitintensiven Schritt der Erarbeitung von Algorithmen für die Einteilung in Zonen entschieden. Daher wird die Gruppierung der Parklücken anhand der zugehörigen Straße vorgenommen. Das dafür benötigte Attribut *street* ist bereits in der Datenbank vorhanden.

Durch die Gruppierung der Parkplätze ist es notwendig die einzelnen, vom Parkplatz abhängigen, Attribute ebenfalls zusammenzufassen. Dabei werden für die Attribute *vehicle*, *street* und *type* der in der jeweiligen Straße dominierende Wert übernommen. Zusätzlich wird für die Straßenzüge das Attribut *load* eingeführt, welches die Auslastung des Straßenzuges in den Abstufungen *full*, *high*, *low* und *empty*, in Abhängigkeit der Zeit, beinhaltet.

Der Datenbanktabelle *histroy* lassen sich die Messwerte der Belegungsmessungen, einschließlich eines Zeitstempels (*time*) des Messzeitpunktes sowie den aktuellen Belegungswerten (*probability*) der vermessenen Parkplätze, entnehmen.

Anhand der Parkplatzhistorie werden, durch Auswertung der Zeitstempel und Belegungsdaten, für die einzelnen Straßen die jeweiligen Auslastungen zu einem gegebenen Zeitpunkt bestimmt. Vergleichbar mit der Analyse von jeder einzelnen Parklücke sind auch die Zeitstempel in Millisekunden zu hochauflösend für die angestrebte Analyse. Daher findet auch hier eine Einteilung der Messzeit in ein gröberes Raster statt. Es wird eine Transformation des Zeitstempels in Monate, Wochentage und Stunden vorgenommen.

Nach den Anpassungen ergeben sich die der Tabelle 5.2 zu entnehmende, aufbereitete Attributmenge für die spätere Analyse.

Attribut	Bedeutung	Wertebereich
street	Name des Straßenzuges	[Nominal]
main_vehicle	Überwiegende Fahrzeugart	[Nominal] Bus, Krad, Lkw, Pkw, Sonstige
main_type	Überwiegende Parkplatzart	[Nominal] Absolutes Halteverbot, Eingeschränktes Halteverbot, Ohne Bewirtschaftung, Ohne Kategorie, Parkscheibe, Parkschein, Parkuhr, Reines Bewohnerparken, Sonderstellplätze
month	Monat	[Nominal] Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
weekday	Wochentag	[Nominal] Mon, Tue, Wed, Thu, Fri, Sat, Sun
daytime	Tageszeit	[Nominal] morning (5-8 Uhr), late_morning (9-11 Uhr), noon (12-14 Uhr), afternoon (15-17 Uhr), evening (18-20 Uhr), late_evening (21-23 Uhr), night (0-4 Uhr)
load	Auslastung des Straßenzuges	[Nominal] empty (0-20%), low(21-50%), high (51-80%), full (81-100%)

Tabelle 5.2: Aufbereitete Attribute für das Data Mining

Der KDD-Prozess fordert eine Bereinigung und Konsistenzprüfung der Attributwerte. Bereits bei dem Parkplatzimport (Unterabschnitt 4.3.4) wurde durch das Setzen entsprechender Defaultwerte sichergestellt, dass alle in der Datenbank vorhandenen Datensätze konsistent und vollständig sind. Eine weitere Bereinigung der Attributwerte ist somit an dieser Stelle nicht mehr erforderlich.

5.3.2 Verfahrensauswahl

Wie Abschnitt 3.1.1 zu entnehmen, ist es erforderlich ein geeignetes Verfahren (Unterabschnitt 3.1.2) für den Analysevorgang zu selektieren. Es soll bei der Prognose vorrangig die aktuelle Parkraumsituation einer Straße, anhand der Merkmale einer bereits festgelegten Ausgangsklasse *load*, zugeordnet werden. Weitere Abhängigkeiten und Einflüsse der Merkmale untereinander sollen nicht gesucht werden. Im Falle der Parkraumprognose wurde daher das Klassifikationsverfahren (Abschnitt 3.1.2) als geeigneter Ansatz ermittelt.

5.3.3 Transformation und Analyse

Aufgrund einer, für die Prognose, zu geringen Anzahl an Messfahrten sowie dem Fehlen einer ausreichend großen Anzahl an Messfahrzeugen, werden die historischen Parkplatzdaten für die nachfolgenden Untersuchungen der Klassifikationsverfahren simuliert (Abschnitt 5.4). Dabei beschränkt sich die Simulation auf zwei Straßenzüge und unterliegt der Annahme einer späteren Übertragbarkeit auf eine umfangreichere Analyse bei ausreichender Anzahl im Produktivbetrieb erfasster Messdaten.

Als Hilfsmittel kommt wie in Abschnitt 5.2 selektiert die Software KNIME zum Einsatz. Die SmartParking Datenbank enthält zwar alle nötigen Attribute, allerdings liegen diese noch nicht in einer transformierten Form vor. Die notwendigen Transformationen wurden in Tabelle 5.2 beschrieben. Die Gruppierung der Parkplätze nach *street* sowie das Ermitteln der überwiegenden Parkplatz- und Fahrzeugart (*main_vehicle*, *main_type*) lässt sich mit den in KNIME enthaltenen Manipulationsbausteinen[6], durch die große Anzahl benötigter Operationen, nicht angemessen abbilden. Daher wurde das Datenbankschema (Unterabschnitt 4.3.2) der SmartParking Datenbank für die Prognose erweitert (Abbildung 5.1).

[6] https://www.knime.org/nodeguide/etl-data-manipulation

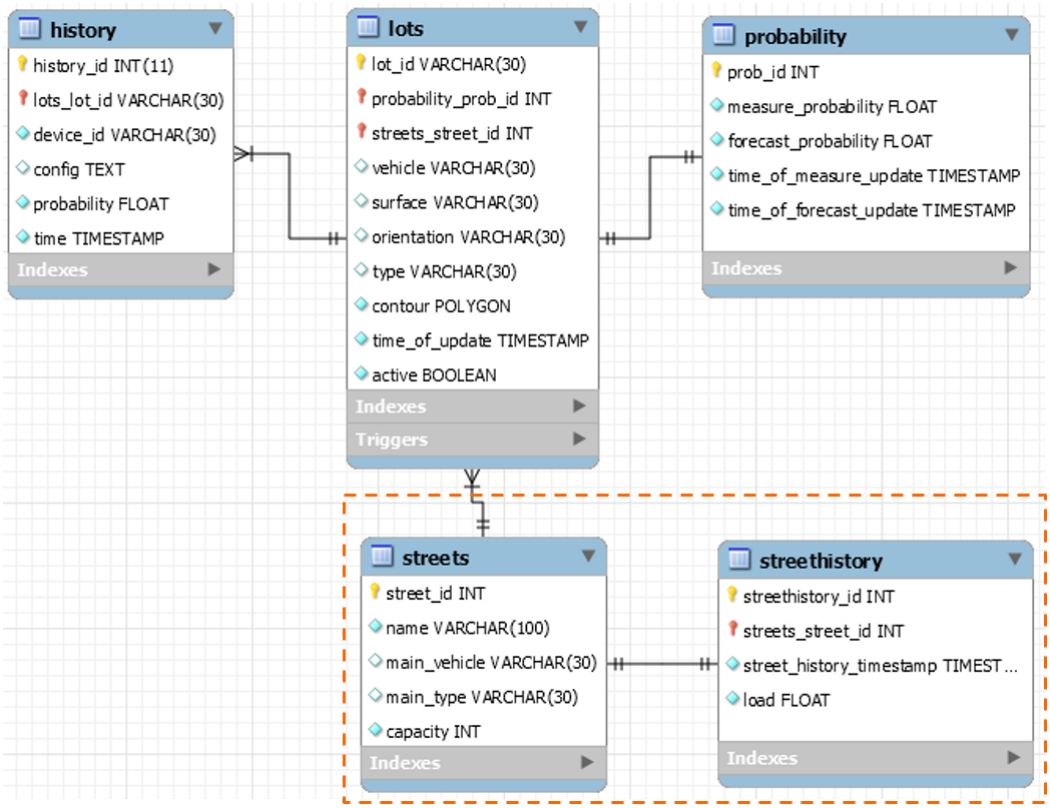


Abbildung 5.1: Erweitertes Datenbankschema für Parkraumprognose

Die ergänzte Datenbanktabelle *streets* enthält alle Straßenzüge mit der Anzahl vorhandener Parklücken (*capacity*) sowie der überwiegenden Parkplatz- und Fahrzeugart (*main_vehicle*, *main_type*). Sie wird automatisch, nach der Erfassung neuer Parklücken, mithilfe der MySQL Triggerfunktion [7] erweitert und aktualisiert. Die Tabelle *streethistory* wird täglich aktualisiert und umfasst die durchschnittlichen Belegungsstände aller Straßenzüge (*load*), unterteilt in einstündige Zeitfenster (Abbildung 5.1). Für die automatische, tägliche Aktualisierung der Datenbanktabelle *streethistroy* wird die Event Funktion[8] der MySQL Datenbank eingesetzt.

[7] https://dev.mysql.com/doc/refman/5.7/en/trigger-syntax.html
[8] https://dev.mysql.com/doc/refman/5.7/en/event-scheduler.html

Der Analyseablauf wurde in KNIME mit Hilfe von Funktionsblöcken modelliert (Abbildung 5.2). Für die Analyse wurde KNIME an die SmartParking Datenbank angebunden und die erforderlichen Tabelle *streets* und *streethistroy* eingelesen. Mit den Funktionsbausteinen *Time Field Extractor* sowie*Date Field Extractor* wurde die Transformation des Zeitstempels der *streethistroy* Tabelle (Abbildung 5.1) in die Attribute *month*, *weekday* und *daytime* gemäß Tabelle 5.2 vorgenommen. Durch die Einteilung der Tageszeiten in Intervalle ist eine weitere Transformation des *daytime* Attributes notwendig. Mithilfe der *Dictionary Binner* Funktion wurde die zusätzliche Transformation von numerischen Stunden auf intervallbasierte, nominale Attribute durchgeführt. Das Attribut *load* für Zielklasse der Auslastung wurde ebenfalls durch einen *Dictionary Binner* nominal transformiert.

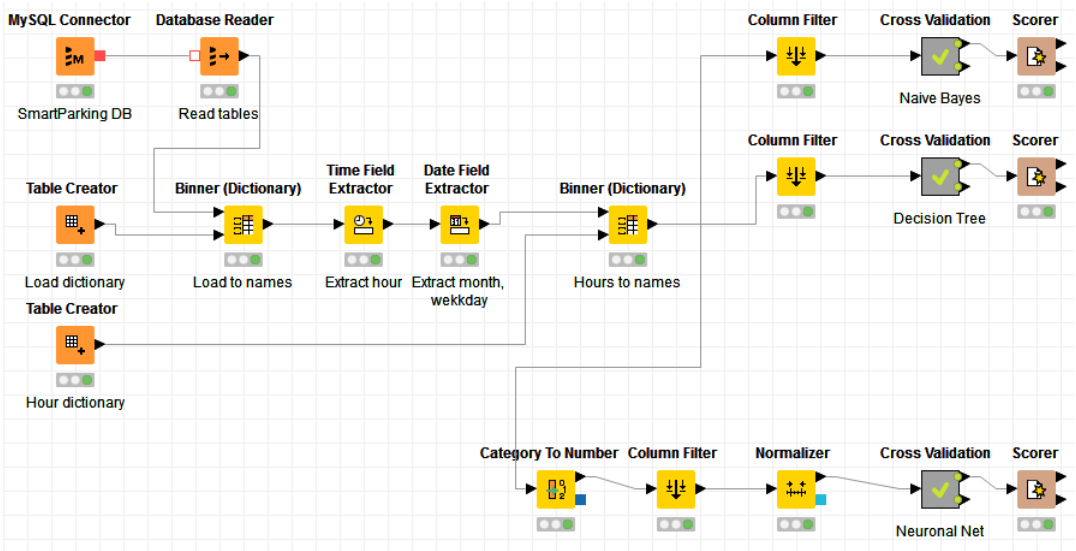


Abbildung 5.2: KNIME Analyseablauf der Parkraumprognose

Im Gegensatz zu den *Entscheidungsbäumen* und *Naive Bayes*, erfordern *neuronale Netzte* Attribute in numerischer, normalisierter Form. Lediglich als Ausgabeklasse kann ein nominales Attribut verwendet werden. Aus diesem Grund wurde für die Analyse mit dem *neuronalen Netz* eine zusätzliche Transformation aller Eingangsattribute auf numerische, normalisierte Attributwerte durchgeführt.

5.3.4 Validierung

Die verschiedenen Klassifikationsverfahren werden anhand der simulierten Parkraumhistorie (Abschnitt 5.4) trainiert. Um abschließend die erzeugten Modelle der untersuchten Verfahren qualitativ bewerten zu können und gleichzeitig eine Überanpassung zu vermeiden, bietet sich das Verfahren der Kreuzvalidierung an. Es ist ein häufig genutztes Verfahren für den qualitativen Vergleich zwischen verschiedenen Klasifikationsalgorithmen (Payman Refaeilzadeh, 2008). Bei der Kreuzvalidierung muss kein separater Bestand an Validierungsdaten erzeugt werden. Die Gewinnung der Validierungsdaten findet auf Basis der Trainingsdaten statt. Dazu wird die vorliegende Datenmenge in die disjunkten Teilmengen der Trainings- und Validierungsdaten aufgeteilt. Die Aufteilung in die entsprechenden Teilmengen kann linear oder zufällig vorgenommen werden. Das Modell wird anschließend mit der Trainingsdatenmenge trainiert und die Fehlerrate anhand der Validierungsdatenmenge bestimmt. Das Kreuzvalidierungsverfahren kann mehrfach wiederholt werden, wobei nach jedem Durchlauf eine erneute Einteilung in Trainings- und Testdaten vorgenommen wird (Runkler, 2015). KNIME bietet für die Kreuzvalidierung den vordefinierten Funktionsbaustein *Cross Validation* an, welcher in den Workflow eingebunden wird (Abbildung 5.3).

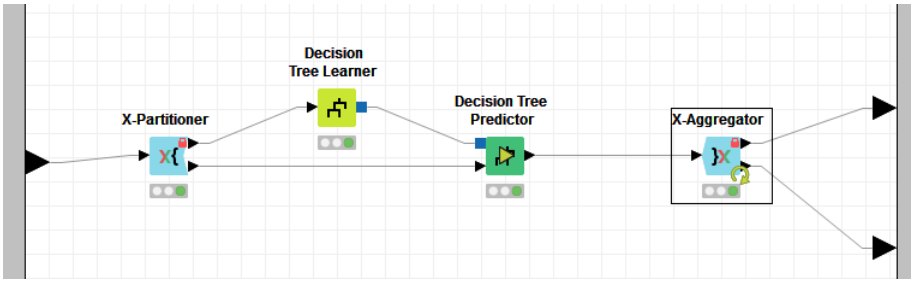


Abbildung 5.3: KNIME Kreuzvalidierungsverfahren

Die historischen Belegungsdaten liegen in der Datenbank chronologisch sortiert in einer zeitlichen Reihenfolge vor. Bei einer linearen Aufteilung der Datensätze werden zeitlich aufeinanderfolgende Datenblöcke ausgewählt und je nach Größe der Datenmenge einzelne Monate aus der Datenmenge der Trainingsdaten ausgeschlossen. Aus diesem Grund wurde für die Analyse ein zufälliges Selektionsverfahren für die Erzeugung der Teilmengen ausgewählt. Der Kreuzvalidierungsvorgang wurde, um ein breiteres Spektrum an Trainings- und Validierungsdaten zu berücksichtigen, mit zehn Iterationen durchgeführt.

5.3.5 Auswertung und Ergebnisse

Die nachfolgende Auswertung und der Vergleich der einzelnen Klassifikationsverfahren erfolgt anhand einer statistischen Auswertung. Erfasst wird dabei die Zuordnung, anhand der Eingangsattribute zu der entsprechenden Ausgangsklasse. Bei der Zuordnung wird zwischen korrekt zugeordnet (*TruePositives*), fälschlicherweise zugeordnet (*FalsePositives*), korrekt abgewiesen (*TrueNegatives*) und fälschlicherweise abgewiesen (*FalseNegatives*) unterschieden. Für den direkten Vergleich zwischen den Klassifikationsverfahren wird der Wert *Accuracy* herangezogen. Er gibt das Verhältnis von korrekt zugeordneten Testdaten zur gesamten Testdatenmenge wieder.

$$Accuracy = \frac{TruePositives + TrueNegatives}{Positives + Negatives}$$

Naive Bayes

Load value	TruePositives	FalsePositives	TrueNegatives	FalseNegatives
empty	235	614	30824	3367
low	10670	8557	13901	1912
high	7747	3581	19105	4607
full	2026	1610	26928	4476

Tabelle 5.3: Auswertung Naive Bayes

Der Naive Bayes Klassifikator (Unterabschnitt 3.2.3) erzielt, nach Auswertung der Statistik (Tabelle 5.3), eine Treffergenauigkeit von $Accuracy = \mathbf{59,0}\%$. Er zählt zu den statistischen Klassifikationsmethoden und berechnet die Wahrscheinlichtkeit der Zugehörigkeit zu der Ausgangsklasse anhand des Satzes von Bayes. Dabei wird angenommen, dass jedes Attribut einen isolierten, direkten Einfluss auf die Klassenzugehörigkeit hat (naiver Ansatz). Die auftretende Kombination der verschiedenen Attribute wird nicht berücksichtigt. Die Zugehörigkeit zur Ausgangsklasse ergibt sich aus den einzelnen, bedingten Wahrscheinlichkeiten der Attribute für die Zugehörigkeit zur Ausgangsklasse *load* (Cleve und Lämmel (2016)).

Entscheidungsbaum

Load value	TruePositives	FalsePositives	TrueNegatives	FalseNegatives
empty	14	123	31315	3588
low	9705	7687	14771	2877
high	8905	4945	17741	3449
full	2004	1657	26881	4498

Tabelle 5.4: Auswertung Entscheidungsbaum

Das Entscheidungsbaumverfahren erreicht gemäß der Statistik (Tabelle 5.4) eine Trefferge-nauigkeit von $Accuracy = \mathbf{58,9}\%$. Der generierte Entscheidungsbau basiert auf dem C4.5 Algorithmus. Durch das so genannte *pruning* wird die Überanpassung des Entscheidungsbaumes vermieden und gleichzeitig, durch das Entfernen überflüssiger Äste, die Größe reduziert (Runkler, 2015, S. 103-104).

Neuronales Netz

Load value	TruePositives	FalsePositives	TrueNegatives	FalseNegatives
empty	3255	17	31421	347
low	12531	381	22077	51
high	12275	84	22602	79
full	6452	45	28493	50

Tabelle 5.5: Auswertung Neuronales Netz

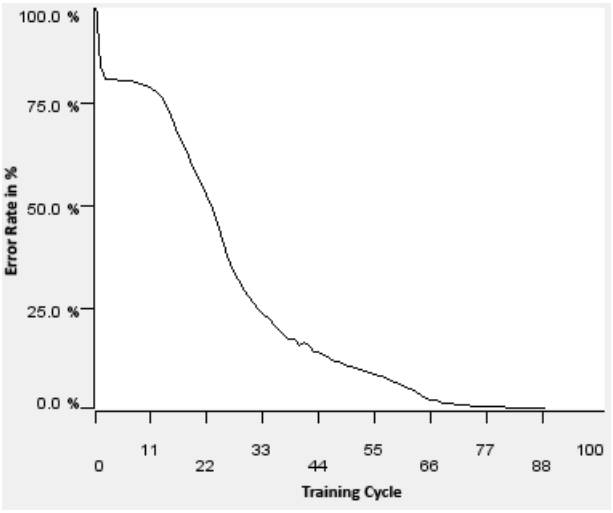


Abbildung 5.4: Fehler nach Lerndurchgängen im neuronalen Netz

Das verwendete neuronale Netz (Unterabschnitt 3.2.1) wurde mit fünf verdeckten Schichten und zehn Neuronen per Schicht konfiguriert. Die Anzahl durchzuführender Iterationen im Lernprozess wurde empirisch ermittelt. Nach 77 Iterationen konnte ein Lernfehler im neuronalen Netz, von nahezu 1%, erreicht werden (Abbildung 5.4). Im Gegensatz zu den Verfahren Naive Bayes und Entscheidungsbaum, kann das Neuronale Netz nicht mit den nominalen Attributwerten trainiert werden. Das Training fand auf Basis numerischer Werte statt (Unterabschnitt 5.3.3). Nach Auswertung der Statistik (Tabelle 5.5) konnte das neuronale Netz eine Treffergenauigkeit von $Accuracy = \mathbf{98,5}\%$ erzielen.

Bewertung

Im direkten Vergleich erzielen das Entscheidungsbaumverfahren mit einer Treffergenauigkeit von **58,9%** sowie der Naive Bayes Ansatz mit **59,0%** vergleichbare Ergebnisse. Die falsch zugeordneten Testdaten wurden hauptsächlich mit der Abweichung um eine Auslastungsstufe eingeordnet. Zukünftig kann geprüft werden, ob die Einteilung der Auslastungsklasse in ein feineres Raster zu einer höheren Treffergenauigkeit führt. Mit einer Treffergenauigkeit von **98,5%**, liegt das neuronale Netz weit vor den anderen untersuchten Verfahren. Es sei anzumerken, dass nicht ermittelt werden konnte ob die hohe Treffergenauigkeit, trotz Nutzung der Kreuzvalidierung, auf eine Überanpassung des neuronalen Netzes zurück zu führen ist. Wenn eine direkte Überführbarkeit in eine mathematische Funktion notwendig ist, sollte dem Naive Bayes Verfahren, für die SmartParking Plattform, der Vorzug geben werden.

5.4 Simulation der Parkraumauslastung

Mangels einer ausreichenden Menge an real erfassten Messdaten, wurden die benötigten Messdaten simuliert. Die Simulation der Parkraumauslastung dient als Datenbasis für die durchgeführten Analysen und die nachfolgenden Bewertungen der Klassifikationsverfahren (Unterabschnitt 5.3.5). Die simulierten Messdaten wurden in die SmartParking Datenbank (Unterabschnitt 4.3.2) übernommen. Durch die Übernahme in die SmartParking Datenbank, lässt sich der im Abschnitt 5.3 beschriebene Prozess zur Parkraumprognose sowohl für die simulierten Messdaten als auch für spätere Produktivdaten verwenden. Simuliert wurde die Parkraumauslastung in zwei Straßenzüge über eine zeitliche Dauer von einem Jahr. Es wurden gezielt zwei Straßenzüge mit unterschiedlich ausgeprägten Parkraumauslastungen gewählt. Ein Straßenzug liegt dabei in einem Wohngebiet, im innerstädtischen Randbereich, während für den zweiten Straßenzug eine direkte innerstädtische Lage gewählt wurde. Es sei anzumerken, dass die simulierte Parkraumauslastung und damit die zugrunde liegenden Muster, auf plausibilisierten Annahmen beruhen und nicht zwingend der Realität entsprechen. Die getroffenen Annahmen werden nachfolgend beschrieben.

Annahme zur Auslastung von Straßen in der Innenstadt

Monate In den kühleren Monaten Januar bis April ist mit dem niedrigsten Besucheraufkommen in der Innenstadt zu rechnen und damit die Parkraumauslastung am geringsten. Die Monate Mai, Juni sowie Oktober und November werden mit einer höheren Auslastung angenommen. Die höchste Auslastung ist in den Monaten Juli, August, September und Dezember gegeben. In den Monaten Juli, August und September ist bedingt durch die Schulferien und die warmen Temperaturen mit einem erhöhten Besucheraufkommen zu rechnen. Der Monat Dezember führt, durch den Weihnachtsmarkt und das Weihnachtsgeschäft im Einzelhandel, zu einer hohen Auslastung.

Wochentage Es wird angenommen, dass alle Werktage einen gleich großen Einfluss auf die Parkraumauslastung haben. Daher wird zwischen den Werktagen Montag bis Freitag nicht weiter differenziert. Der Samstag ist ein für Wochenendbesucher attraktiver Tag und führt zu einer höheren Auslastung, als es an den Werktagen Montag bis Freitag der Fall ist. Der Sonntag ist ein typischer Abreisetag für Wochenendbesucher und die Geschäfte in der Innenstadt sind geschlossen. Einheimische hingegen nutzen den Sonntag für Aktivitäten im innerstädtischen Bereich. Deshalb wird die Auslastung an Sonntagen mit der Auslastung an Wochentagen gleichgesetzt.

Tageszeiten In der späten Nacht sowie am frühen Morgen wird mit der geringsten Parkraumauslastung gerechnet. Vormittags und Mittags steigt die Auslastung. Der Tageshöchstwert wird in der Zeitspanne vom Nachmittag bis zum frühen Abend angenommen. Spät abends beginnt die Auslastung zu sinken.

Annahme zur Auslastung von Straßen im Wohngebiet

Monate Im Gegensatz zu Straßen im direkten Innenstadtgebiet wird angenommen, dass die Monate keine nennenswerte Auswirkung auf die Parkraumauslastung haben. Aus diesem Grund wird keine Unterteilung der Monate vorgenommen.

Wochentage An den Werktagen Montag bis Freitag wird angenommen das, bedingt durch die Fahrt zur Arbeit, mit einer Vergleichsweise geringe Auslastung zu rechnen ist. Am Wochenende (Samstag und Sonntag) verbringen Familien mehr Zeit zu Hause und die Parkraumauslastung steigt an.

Tageszeiten Die geringste zeitliche Auslastung wird vormittags angenommen. Zwischen der Mittagszeit und dem späten Nachmittag steigt die Auslastung an und erreicht ab den Abendstunden ihren Höchststand.

Durch Kombination der getroffenen Annahmen wurden die in Abbildung 5.5 und Abbildung 5.6 dargestellten Auslastungsprofile erstellt. Auf Basis der Auslastungsprofile erfolgte anschließend die Generierung der simulierten Messdaten. Dafür wurde eine Simulationssoftware entwickelt, welche durch Eingabe einer Liste, bestehende aus Auslastungsprofilen, die benötigten Daten erzeugt. Um auch Anomalien in den simulierten Daten zu berücksichtigen, wurden die Messwerte, anhand der Auslastungsprofile, normalverteilt.

Day	Daytime	Month					
		Jan - Apr		May - Jun, Oct - Nov		Jul - Aug, Sep, Dec	
Mon-Fri,Sun	morning	low		low		low	
	late_morning	low		low		high	
	noon	low		high		high	
	afternoon	high		full		full	
	evening	high		high		full	
	late_evening	low		high		full	
	night	empty		empty		low	
Sat	morning	low		low		high	
	late_morning	low		low		full	
	noon	low		high		full	
	afternoon	high		full		full	
	evening	high		full		full	
	late_evening	high		high		full	
	night	empty		low		low	
		100	Auslastung in % 0	100	Auslastung in % 0	100	Auslastung in % 0

Abbildung 5.5: Simulierte Parkraumauslastung in der Innenstadt

Day	Daytime	Month	
		Jan - Dec	
Mon-Sat	morning	empty	
	late_morning	empty	
	noon	low	
	afternoon	low	
	evening	high	
	late_evening	full	
	night	full	
Sat, Sun	morning	full	
	late_morning	full	
	noon	high	
	afternoon	low	
	evening	high	
	late_evening	high	
	night	full	
		100	Auslastung in % 0

Abbildung 5.6: Simulierte Parkraumauslastung im Wohngebiet

6 Zusammenfassung, Fazit und Ausblick

Ziel dieser Bachelorarbeit war die Erhebung und Auswertung von Sensor Daten aus Fahrzeugen, welche als Datenbasis für eine Cloud-Plattform zur Erfassung, Analyse, Prognose sowie Visualisierung von freiem Parkraum dient. Dazu wurde im Rahmen dieser Arbeit, in Zusammenarbeit mit der Firma IAV Automotive Engineering, erfolgreich eine Cloudplattform zur Suche von freiem Parkraum entwickelt und in Betrieb genommen, welche das Finden einer freien Parklücke erleichtert. Im Gegensatz zu anderen Parkraumüberwachungssystemen, nutzt die SmartParking Plattform Fahrzeugen als Sensorknoten und erfordert keine ortsfeste Sensorik zur Überwachung des Parkraumes.

Zusammenfassung

Am Anfang der Arbeit, wurde auf den aktuellen Stand von vernetzten Fahrzeugen und der in Fahrzeugen verbauten Sensorik eingegangen. Es wurden aktuelle Versuchsprojekte und Studien der Automobilbranche zu den Themen Fahrzeugvernetzung und Sensordatenerhebung vorgestellt und analysiert.

Im darauffolgenden Kapitel, wurde eine Einführung in Big Data Anwendungen gegeben. Dazu wurde der Prozess des Data-Mining vorgestellt, Verfahrensansätze beschrieben und einzusetzende Algorithmen vorgestellt.

Im Hauptteil der Arbeit wurde die Umsetzung einer Cloudplattform zur Detektion von freiem Parkraum beschrieben. Dabei wurde auf die entstandene Softwarearchitektur, Kommunikationsverfahren, die Arbeit mit externen Datenquellen, die Erfassung von Parkplatzbelegungsdaten mittels Fahrzeugsensorik, sowie die Erstellung einer Web-App als eine mögliche Endnutzeranwendung eingegangen. Abschließend wurde die Funktionsfähigkeit der Plattform im Versuchsbetrieb evaluiert.

Im nächste Kapitel wurden die durchgeführten Schritte des KDD-Prozesses mit dem Ziel der Entwicklung eines Vorhersagemodells zur Prognose von freiem Parkraum beschrieben. Dazu wurde eine geeignete Toolchain zur Durchführung der späteren Analyse ermittelt und die Durchführung der notwendigen Prozessschritte des KDD-Prozesses beschrieben. Den Abschluss des Kapitels bildete der qualitative Vergleich verschiedener Klassifikationsalgorithmen, anhand von simulierten Parkraumbelegungsdaten.

Fazit

Dank der immer weiter voranschreitenden Entwicklung in der Fahrzeugvernetzung und Sensorik sind Fahrzeuge eine wertvolle Datenquelle für Big Data Analysen. Die Erhebung von Sensordaten aus Fahrzeugen kann zukünftig die Basis für neuartige Prognosen im Straßenverkehr bilden.
In der Versuchsanwendung zur Detektion von freiem Parkraum konnte ein mögliches Anwendungsszenario erprobt werden. Das Zusammenspiel zwischen den einzelnen Komponenten der SmartParking Plattform hat sich als zuverlässig heraus gestellt. Durch eine bedachte Auswahl der jeweiligen Softwarekomponenten ist davon auszugehen, dass die Versuchsplattform auch für den Produktivbetrieb mit einer großen Anzahl an Endnutzern und Messfahrzeugen eingesetzt werden kann. Die Prognosemodelle liefern anhand der simulierten Parkraumbelegungshistorie eine hohe Vorhersagegenauigkeit.
Probleme ergaben sich bei der genauen Zuordnung der vermessenen Parklücken, durch die unzureichende Genauigkeit bei der Lokalisierung.
Wegen des Mangels an einer ausreichend großen Menge historischer Belegungsmessungen, konnten die Klassifikationsmodelle lediglich mit einer simulierten Belegungshistorie trainiert werden. Es kann zu diesem Zeitpunkt keine exakte Aussage über die Repräsentativität der simulierten Belegungsdaten gemacht werden.

Ausblick

Die Erhebung der Parkplatzbelegungsdaten kann als Datenbasis für weitere, detailliertere Vorhersagemodelle genutzt werden. Dazu müssen in einer größeren Versuchsreihe, über einen längeren Zeitraum, Belegungsstände erfasst werden. Durch die Einbeziehung weiterer Merkmale wie Feiertage, Großveranstaltungen oder Wetterdaten kann das Vorhersagemodell zukünftig optimiert werden. Durch die Verwendung von Clusteranalysen können Zusammenhänge zwischen den einzelnen Merkmalen, abseits der Parkraumprognose, ermittelt und so neue, interessante Zusammenhänge aufgezeigt werden.
Auch die Thematik der exakten Fahrzeuglokalisierung bietet Potential für zukünftige Arbeiten. Die genutzten GPS Lokalisierung stellte sich, für eine exakte Lokalisierung, als zu ungenau heraus. Zukünftig können neue Technologien zur Fahrzeuglokalisierung evaluiert werden.
Ebenso lässt sich der Themenbereich *security* in zukünftigen Arbeiten aufgreifen. Es können Sicherheitskonzepte entwickelt werden, um die Kommunikation zwischen Fahrzeugen und Cloudplattformen, durch entsprechende Sicherheitsmaßnahmen, vor einem unbefugten Zugriff zu schützen.

Literaturverzeichnis

[de Andrade u. a. 2016] Andrade, Paulo R. de ; Albuquerque, Adriano B. ; Frota, Otávio F ; Silveira, Robson V. ; Silva, Fátima A da: Cross Platform App. (2016). – Online erhältlich unter https://arxiv.org/pdf/1503.03511; abgerufen am 23. April 2017.

[ARGUS 2009] ARGUS: Parken in Hamburg Abschlussbericht / Straßen- und Verkehrsplanung. 2009. – Forschungsbericht. – 38–39 S. Online erhältlich unter http://daten.transparenz.hamburg.de/Dataport.HmbTG.ZS.Webservice.GetRessource100/GetRessource100.svc/f4b783b7-6a9c-4958-a411-1abffe421d56/Akte_744.5400-001.pdf; abgerufen am 21. März 2017.

[Audi] Audi: Website. – Online erhältlich unter https://audi-illustrated.com/en/q5/Fahrerassistenzsysteme; abgerufen am 04. Dezember 2016.

[Audi 2016] Audi: Schwarmintelligenz/Car-to-X. (2016). – Online erhältlich unter https://www.audi-mediacenter.com/de/techday-connectivity-6597/schwarmintelligenzcar-to-x-6602; abgerufen am 04. Dezember 2016.

[Bosch 2016] Bosch, Robert: *Vernetztes und automatisiertes Parken.* Robert Bosch GmbH. 2016. – Online erhältlich unter http://www.bosch-presse.de/pressportal/de/media/migrated_media/automatisiertes_parken_2016/automatisiertes_parken_2016_7600_de-de.pdf; abgerufen am 22. November 2016.

[C2C-CC 2007] C2C-CC: C2C-CC Manifesto. (2007). – Online erhältlich unter https://www.car-2-car.org/index.php?eID=tx_nawsecuredl&u=0&g=0&t=1483799468&hash=3cbec6a23bab7665e84843979787c1a853af3082&file=fileadmin/downloads/C2C-CC_manifesto_v1.1.pdf; abgerufen am 06. Januar 2016.

[Cleve und Lämmel 2016] Cleve, Jürgen ; Lämmel, Uwe: *Data Mining.* In: *Data Mining.* Berlin : De Gruyter Oldenbourg, 2016. – ISBN 978-3-11-045675-2

[Ertel 2016] Ertel, Wolfgang: *Maschinelles Lernen und Data Mining.* S. 191–264. In: *Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung.* Wiesbaden : Springer Fachmedien Wiesbaden, 2016. – URL http://dx.doi.org/10.1007/978-3-658-13549-2_8. – ISBN 978-3-658-13549-2

[Fasel und Meier 2016] Fasel, Daniel ; Meier, Andreas: *Was versteht man unter Big Data und NoSQL?* S. 3–16. In: Fasel, Daniel (Hrsg.) ; Meier, Andreas (Hrsg.): *Big Data: Grundlagen, Systeme und Nutzungspotenziale.* Wiesbaden : Springer Fachmedien Wiesbaden, 2016. – URL http://dx.doi.org/10.1007/978-3-658-11589-0_1. – ISBN 978-3-658-11589-0

[Hamburg 2014] Hamburg, Handelskammer: Stadtmobilität in Hamburg 2030. (2014). – Online erhältlich unter https://www.hk24.de/blob/hhihk24/produktmarken/interessenvertretung/wirtschaft-politik/wirtschaftspolitik/downloads/1153060/150421f8a1dc1bb8bfc368d3197cc2e1/Standpunkt_Stadtmobilitaet_in_Hamburg_2030-data.pdf; abgerufen am 06. Januar 2016.

[HERE 2016] HERE: HERE, automotive companies move forward on car-to-cloud data standard. (2016). – Online erhältlich unter https://lts.cms.here.com/static-cloud-content/Newsroom/290616_HERE_automotive_companies_move_forward_on_car_to_cloud_data_standard.pdf; abgerufen am 18. April 2016.

[Johanning und Mildner 2015] Johanning, Volker ; Mildner, Roman: *Die Ausgangssituation: Von der Mechanik über die Elektrik/Elektronik zur IT im Auto.* S. 1–16. In: *Car IT kompakt: Das Auto der Zukunft – Vernetzt und autonom fahren.* Wiesbaden : Springer Fachmedien Wiesbaden, 2015. – URL http://dx.doi.org/10.1007/978-3-658-09968-8_1. – ISBN 978-3-658-09968-8

[Kruse u. a. 2015] Kruse, Rudolf ; Borgelt, Christian ; Braune, Christian ; Klawonn, Frank ; Moewes, Christian ; Steinbrecher, Matthias: *Einleitung.* S. 7–15. In: *Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze.* Wiesbaden : Springer Fachmedien Wiesbaden, 2015. – URL http://dx.doi.org/10.1007/978-3-658-10904-2_2. – ISBN 978-3-658-10904-2

[Lei u. a. 2014] Lei, K. ; Ma, Y. ; Tan, Z.: Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js. In: *2014 IEEE 17th International Conference on Computational Science and Engineering*, Dec 2014, S. 661–668

[McAfee u. a. 2012] McAfee, Andrew ; Brynjolfsson, Erik ; Davenport, Thomas H. ; Patil, DJ ; Barton, Dominic: Big data. In: *The management revolution. Harvard Bus Rev* 90 (2012), Nr. 10, S. 61–67

[Mennecke und Crossland 1996] Mennecke, B. E. ; Crossland, M. D.: Geographic information systems: applications and research opportunities for information systems researchers. In: *Proceedings of HICSS-29: 29th Hawaii International Conference on System Sciences* Bd. 3, Jan 1996, S. 537–546 vol.3

[Payman Refaeilzadeh 2008] Payman Refaeilzadeh, Huan L.: Cross-Validation / Arizona State University. 2008. – Forschungsbericht. – 1–3 S. Online erhältlich unter https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=10&sqi=2&ved=0ahUKEwj5grjboJ3TAhWrLsAKHQ5hD_sQFghwMAk&url=http%3A%2F%2Fleitang.net%2Fpapers%2Fency-cross-validation.pdf&usg=AFQjCNFPfZE_4_bOf_JMd-xDcW265NHFNQ&bvm=bv.152180690,d.bGs&cad=rja; abgerufen am 11. April 2017.

[Petersohn 2005] Petersohn, Helge: *Data Mining: Verfahren, Prozesse, Anwendungsarchitektur.* S. 28. In: *Data Mining: Verfahren, Prozesse, Anwendungsarchitektur*, Oldenbourg Verlag, 2005

[Runkler 2015] Runkler, Thomas A.: *Einführung.* S. 1–3, 80. In: *Data Mining: Modelle und Algorithmen intelligenter Datenanalyse.* Wiesbaden : Springer Fachmedien Wiesbaden, 2015. – URL http://dx.doi.org/10.1007/978-3-8348-2171-3_1. – ISBN 978-3-8348-2171-3

[Stefan Lanig 2010] Stefan Lanig, Philipp M.: Evaluation von Data Mining Werkzeugen / Institut fÃ¼r Visualisierung und Interaktive Systeme Universität Stuttgart. 2010. – Forschungsbericht. – 11–16 S. Online erhältlich unter https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwi2zcHuop3TAhXGDpoKHeZCCpgQFgglMAA&url=http%3A%2F%2Fwww.wi.hs-wismar.de%2F~cleve%2Fvorl%2Fprojects%2Fdm%2Fss13%2FHierarClustern%2FLiteratur%2FFACH_0108.pdf&usg=AFQjCNGMGPZsKMc8laDXF_PFRHFyY7Fyjw&bvm=bv.152180690,d.bGs; abgerufen am 11. April 2017.

[VDA] VDA: *Vernetzte Mobilität.* Website. – Online erhältlich unter https://www.vda.de/de/themen/innovation-und-technik/vernetzung/vernetzte-mobilitaet.html; abgerufen am 22. November 2016.

[von Vogel] Vogel, Dr. A. von: *Die Digitalisierung der großen Stadt.* Website. – Online erhältlich unter http://www.hamburg.de/pressearchiv-fhh/4435132/2015-01-13-bwf-digitalisierung-der-grossen-stadt/; abgerufen am 26. November 2016.

[Zwick 2015] Zwick, Marcus: *Parkplatzsuche ade.* Siemens AG. 2015. – Online erhältlich unter http://www.siemens.com/innovation/de/home/pictures-of-the-future/infrastruktur-und-finanzierung/lebensqualitaet-in-staedten-intelligente-parkraumueberwachung.html; abgerufen am 22. November 2016.

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 25. April 2017 Yannic Wilkening