HOCHSCHULE COBURG

Hochschule für angewandte Wissenschaften Coburg

Fakultät Elektrotechnik und Informatik

Studiengang: Informatik

Bachelorarbeit

Klassifizierung der Bewegungsmuster von Mobilfunkteilnehmern zur erweiterten Umfeldwahrnehmung autonomer Fahrzeuge

Maximilian Sohl

Abgabe der Arbeit: 25. März 2022

Betreut durch:

Prof. Dr. Thomas Wieland, Hochschule Coburg

Zusammenfassung

Maschinelle Lernverfahren gewannen in den vergangenen Jahren zunehmend an Relevanz in den entsprechenden Fachbereichen. Die vorliegende Bachelorarbeit beschäftigt sich mit der Anwendung dieser maschinellen Lernverfahren zur Vorhersage eines Klassifikationsproblems. Es soll im Rahmen der erweiterten Umfeldwahrnehmung autonomer Fahrzeuge erforscht werden, inwieweit sich Positionsdaten von bewegten Mobilfunkteilnehmern im Straßenverkehr dazu eignen, Vorhersagen über die Art des Verkehrsteilnehmers zu treffen. Dafür werden zunächst verschiedene Verkehrsszenarien mit der Simulationssoftware CARLA simuliert, aus denen die erzeugten Positionsdaten entnommen und im Nachgang zu relevanten Informationen (Geschwindigkeit, Beschleunigung etc.) umgewandelt werden. Diese werden in verschiedenen Datensätzen aufgezeichnet und später für das Training/die Evaluation der Algorithmen verwendet. Um eine möglichst umfassende Perspektive in Bezug auf den Forschungsgegenstand zu generieren, werden drei verschiedene Klassifikationsmethoden (K-Nearest-Neighbor, Decision Tree, Support Vector Machine) in Betracht gezogen, deren Ergebnisse darauffolgend analysiert und für die Bewertung der einzelnen Verfahren verglichen werden. Zudem werden je Lernverfahren verschiedene Aspekte (Klassenanzahl, Umgebungskomplexität, Genauigkeit der GNSS-Daten) betrachtet und deren Auswirkung auf die Genauigkeit der jeweiligen Erkennungsraten untersucht. Die Resultate der Untersuchungen ergeben, dass sich die Informationen, welche sich durch die GNSS-Daten gewinnen lassen - je nach Szenario - gut dazu eignen, unbekannte Objekte durch Klassifikationsverfahren korrekt vorherzusagen.

Abstract

In recent years, machine learning methods have become increasingly relevant in several disciplines. This bachelor thesis thematizes the application of machine learning methods for the prediction of a classification problem. Within the context of the extended environment perception of autonomous vehicles, it needs to be investigated in which way position data of moving mobile phone users in road traffic are suitable for making predictions about the type of road user. Firstly, various traffic scenarios will be simulated by using the CARLA simulation software, from which the position data is generated. This data will be extracted and subsequently converted into relevant information (speed, acceleration, etc.). These will be recorded in different data sets and later used for training/evaluation of the algorithms. In order to generate a comprehensive perspective with respect to the concerning research subject, three different classification methods (K-Nearest-Neighbor, Decision Tree, Support Vector Machine) will be considered. After that the results will be subsequently analyzed and compared for the evaluation of each method. In addition, for each learning method, different aspects (number of classes, environment complexity, accuracy of GNSS data) will be considered and their impact on the accuracy of the respective recognition rates will be investigated. The results of the investigations demonstrate that the information that can be obtained from the GNSS data - depending on the scenario - is well suited to correctly predict unknown objects by classification methods.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

DT	Decision Tree
etc.	et cetera
km/h	Kilometer pro Stunde
KNN	K-Nearest-Neighbor
SVM	Support Vector Machine

1 Einleitung

“Predicting the future isn’t magic, it’s artificial intelligence” Dave Waters zitiert nach [Terré 2019, S. 5]. Die Thematik der Künstlichen Intelligenz gewann in den vergangenen Jahren zunehmend an Bedeutung. Maschinelle Lernverfahren werden mit steigender Tendenz dazu verwendet, Problemstellungen aus dem Alltag zu lösen.

In dieser Bachelorthesis sollen diese maschinellen Lernverfahren dazu verwendet werden, detaillierte Vorhersagen hinsichtlich eines Klassifikationsproblems aus dem Fachbereich der autonomen Fahrzeuge zu generieren. Um die Problemstellung korrekt einordnen zu können, wird zunächst ihr spezifischer Kontext erörtert.

1.1 Vorstellung des Kontexts der Arbeit

Dem Begriff des autonomen Fahrens kann im heutigen Zeitalter der Digitalisierung eine wachsende Relevanz zugeordnet werden [Andelfinger+2015, S. 5]. Zahlreiche Arten von Sensorsystemen sowie Assistenzfunktionen, welche „[…] in den höheren Ausbaustufen des Gesamtsystems zusammenwirken“, bilden die grundlegenden Komponenten des autonomen Fahrens [Scheffels+2022]. Hinsichtlich dieser kann eine Differenzierung von aktiven und passiven Systemen vorgenommen werden, wobei in Hinblick auf die weiterführende Bearbeitung lediglich die passiven Systemelemente betrachtet werden. Passive Systeme beziehen sich vorwiegend auf die detaillierte Wahrnehmung des Fahrzeugumfelds unter Verwendung sensibler Sensorik. Zu primären Elementen zählen hierbei vorwiegend Kamera-, Laser-, Ultraschall- und Radarsensoren [Johanning+2015, S.63]. Um eine zuverlässige, lückenlose Wahrnehmung der umgebenden Peripherie zu erlangen, erfordert das autonome Fahren eine steigende Quantität von leistungsfähigen Sensoren [Tille 2016, S. 281f]. Je nach Modell und Ausstattung beläuft sich die Anzahl der Sensoren in modernen Fahrzeugen bereits auf über 150 integrierte Elemente. Der Sensor-Markt der Automobilindustrie wächst zudem deutlich, um jährlich fünf Prozent [Tille 2018, S. 18].

Abgesehen von der Lokalisierung und der exakten Abstands- und Geschwindigkeitsmessung spielt im Straßenverkehr die Klassifikation von Objekten und Gegenständen eine fundamentale Rolle für die Verwirklichung autonomer Fahrzeuge. Um den angestellten Anforderungen zu entsprechen, werden optische Messverfahren eingesetzt, welche Objekte in der näheren Umgebung identifizieren und die Fähigkeit besitzen, Geschwindigkeit als auch Bewegungsrichtung und Entfernung von Verkehrsteilnehmern zu bestimmen. Stellvertretend

hierfür sind LiDAR-Sensoren, welche in periodischen Intervallen Lichtimpulse aussenden, die an den Objektoberflächen reflektiert werden. Diese Reflexionen werden unter Einsatz von Fotodioden registriert, elektronisch ausgewertet und ermöglichen somit Rückschlüsse auf die Position sowie die Beschaffenheit des Objektes [Tille 2018, S. 29f]. Der Vorteil der optischen Sensoren liegt vor allem in der hohen Genauigkeit der gelieferten Messwerte. Dennoch gehen mit der Verwendung optischer Messsysteme Nachteile einher, da die Sensorik zumeist räumlich durch die umliegende Umgebung beschränkt wird und die Elemente fehleranfällig gegenüber Feuchtigkeitsbeschlag und Verschmutzungen sind [Tille 2016, S. 237].

1.2 Zielsetzung und Motivation

Die in Kapitel 1.1 erwähnten Umstände, bilden die Grundlage für die Motivation und Zielstellung dieser Arbeit. Bislang wird sich bei der Klassifizierung unbekannter Objekte auf Sensoren gestützt, welche in ihrer Reichweite beschränkt sind. Aufgrund dessen gründet sich ein Primärziel dieser Arbeit im Willen eine Möglichkeit zu erforschen, welche die Klassifikation der Objekte über diese Reichweite hinaus realisieren kann. Deshalb soll im Kontext der erweiterten Umfeldwahrnehmung autonomer Fahrzeuge untersucht werden, inwiefern sich GNSS-Daten von bewegten Mobilfunkteilnehmern im Straßenverkehr dazu eignen, Prognosen über den Typ des Verkehrsteilnehmers zu treffen. Der Vorteil der GNSS-Informationen liegt hierbei in der globalen Verfügbarkeit. Dies könnte in Kombination mit den optischen Sensoren dazu führen, eine flächendeckende Klassifikation der unbekannten Objekte im Straßenverkehr zu ermöglichen, um das autonome Fahren sicherer zu gestalten. Damit die Potentiale dieses Ansatzes ermittelt werden können, sollen maschinelle Lernverfahren dazu genutzt werden, auftretende Muster in den Positionsdaten zu erkennen, um gegebenenfalls Rückschlüsse auf den Typ des Objekts abzuleiten. Hinsichtlich der Mustererkennung kommt der Klassifizierung eine elementare Rolle zu. Um eine möglichst ausführliche Perspektive in Hinblick auf den Forschungsgegenstand erschaffen zu können, werden die drei unterschiedlichen Klassifikationsmethoden K-Nearest-Neighbor, Decision Tree und Support Vector Machine in Betracht gezogen. Um die Datensätze für die Umsetzung der Verfahren zu generieren, wird auf die Simulationsumgebung CARLA zurückgegriffen. Zudem sollen die GNSS-Daten der Objekte hinsichtlich verschiedener Verkehrsszenarios simuliert und aufgezeichnet werden. Das Ziel besteht darin, herauszufinden, ob eine Kategorisierung der Instanzen nach der Art allein auf Grundlage von Positionsdaten, möglich ist und mit welcher Genauigkeit die Klassifizierungen vonstattengehen.

1.3 Aufgabenstellung

Aus der bereits im Abschnitt 1.2 formulierten Zielstellung lassen sich konkrete Aufgaben ableiten. Zu Beginn werden die Anforderungen an das Forschungsvorhaben spezifiziert und darauf aufbauend die Bestandteile in Hinsicht auf die Realisierung der Lernverfahren erarbeitet. Nachdem die wesentlichen Spezifikationen getroffen wurden, erfolgt eine Simulation verschiedenartiger Verkehrsszenarien mit der Simulationssoftware CARLA simuliert, aus welchen die generierten GNSS-Daten extrahiert und zu bedeutsamen Informationen transformiert werden. Es schließt sich die Aufzeichnung der Werte in unterschiedlichen Datensätzen an. Diese werden für das spätere Training der Algorithmen vorbereitet, indem eine Aufteilung in Trainings- und Testdaten erfolgt. In Bezug auf die Klassifikationen werden die Daten eines Objektes einer bestimmten Kategorie zugeordnet und infolgedessen für den Algorithmus unterscheidbar gemacht. Darüber hinaus soll das Kennzeichnen der Einträge die Grundlage dafür bieten, die Datenfelder für das Modell interpretierbar zu machen. Nachdem die Datenvorverarbeitung abgeschlossen wurde, werden die maschinellen Lernverfahren implementiert, um diese auf die Problemstellung anwenden zu können. Um einen möglichst umfassenden Standpunkt hinsichtlich der Problemstellung entwickeln zu können, sollen verschiedene Techniken bei der Realisierung der Klassifikation die Grundlage bilden. Einleitend wird zunächst die K-Nearest-Neighbor-Klassifikation betrachtet, welche zu den Methoden der Lazy Learner gehört. Im Anschluss dessen erfolgt zum Vergleich der beiden Eager-Learner-Methoden die Implementierung von Decision Tree und Support Vector Machine. Ferner werden die Verfahren hinsichtlich verschiedener Aspekte beleuchtet und deren Auswirkungen auf die Resultate miteinander verglichen beziehungsweise ausgewertet.

1.4 Aufbau der Arbeit

Nachdem in Kapitel 1 dargestellt wurde, unter welchem Aktionsspektrum angesetzte Ziele und Erwartungshaltungen realisiert werden können, thematisiert Kapitel 2 die dementsprechenden, theoretischen Inhalte und Definitionen. Dabei erfolgt eine Betrachtung jener Begriffe, die benötigt werden, um den Kontext bestimmter Aspekte korrekt einzuordnen. Kapitel 3 beinhaltet die Beschreibung spezifischer Werkzeuge, welche in Bezug auf die Umsetzung Anwendung finden. Das vierte Kapitel beschäftigt sich mit den Anforderungen, welche an die zu implementierenden Algorithmen gestellt werden, als auch mit den Eigenschaften des Gesamtkonzepts. Um ausgehend von diesen Implementierungsaufgaben einen Softwareentwurf

zu gestalten, beinhaltet dieses Kapitel zudem eine Designbeschreibung dieser Implementierungen. Darauf aufbauend wird im Kapitel 5 Bezug auf die tatsächliche Umsetzung dieser Überlegungen und den groben funktionalen Ablauf innerhalb dieser Implementierungen genommen. Am Ende dieser Bachelorarbeit werden in Kapitel 6 die Ergebnisse der Algorithmen evaluiert und anschließend im siebten Kapitel zu einem Fazit zusammengefasst. Abschließend werden die Resultate im letzten Kapitel in Relation zu möglichen Chancen und Weiterentwicklungen gesetzt.

2 Allgemeine Grundlagen und Methodiken

Dieses Kapitel beinhaltet die Erläuterung theoretischer Grundlagen, welche dem Leser zum Verständnis der nachfolgenden Thematiken dienen soll.

2.1 Maschinelles Lernen

Der Begriff „maschinelles Lernen“ stellt eine Teildisziplin der künstlichen Intelligenz dar und bezeichnet dabei die Techniken, welche im Fachbereich der KI verwendet werden [Frochte 2021, S. 16]. Grundsätzlich beschäftigt sich maschinelles Lernen mit dem Entwurf von Algorithmen, welche das Nachahmen des menschlichen Lernens zum Ziel haben und deren Adaption an Gegebenheiten durch komplexe Analysen weitgehend selbstständig erfolgt ([Maurer+2015, S. 466], [Raschka+2021, S. 29f]). Die Informationen, welche dabei für die Realisierung benötigt werden, gewinnt das Verfahren selbstständig aus den jeweiligen Datensätzen. Aus diesem Grund wird das Gebiet des maschinellen Lernens auch als die Wissenschaft oder Anwendung von Algorithmen bezeichnet, welche Daten analysieren und deren Sinn erfassen können. Das erhaltene Wissen kann zu verschiedensten Rückschlüssen führen. Zumeist werden leistungsstarke Algorithmen dafür eingesetzt, Prognosen über bevorstehende Ereignisse zu generieren, indem auftretende Muster in den Datensätzen erkannt und anschließend interpretiert werden. Maschinelles Lernen findet deshalb besonders dann Verwendung, wenn Datensätze vorliegen, die potenziell Informationen beinhalten, welche maschinell aufgeschlüsselt und entnommen werden können. Zudem finden maschinelle Lernverfahren in Bereichen Verwendung, in denen sich die Bedingungen ändern und häufig der Bedarf nach dynamischen Konfigurationen besteht [Maurer+2015, S. 468]. Das Resultat ist zumeist ein Modell, dass im Nachgang auf die konkrete Problemstellung angewendet werden kann. Der Einsatz der Verfahren sorgt dafür, dass alltägliche Applikationen wie Spamfilter, Sprach- und Texterkennungssoftware oder Suchmaschinen unterstützt beziehungsweise verbessert werden. Auch im medizinischen Bereich konnten die Algorithmen bereits Anwendung finden und Ergebnisse erzielen, welche mit menschlichen Leistungen vergleichbar sind. Ein Vorteil der maschinellen Lernverfahren liegt darin, dass sich das manuelle Herleiten von Modellen und Regelungen durch die Untersuchung immenser Datenbestände und selbstlernender Prozesse zunehmend erübrigt. Aufgrund dessen lassen sich die Verfahren für eine einfache und effiziente Detektion von Mustern in Datensätzen einsetzen [Raschka+2021,

S. 29f]. In Bezug auf Training und Evaluierung müssen die Maße und Regeln je nach Aufgabe im Vorfeld vom Softwareentwickler definiert werden [Maurer+2015, S. 468].

Grundsätzlich wird das Fachgebiet des maschinellen Lernens in die Kategorien bestärkendes Lernen, unüberwachtes Lernen und überwachtes Lernen eingeteilt. Beim bestärkenden Lernen handelt es sich um Entscheidungsvorgänge, deren Aktionen durch ein Belohnungssystem erlernt werden. Das unüberwachte Lernen hingegen beschäftigt sich damit, versteckte Strukturen in den Datensätzen aufzudecken, ohne dabei auf Feedback, Kennzeichnungen der Daten oder definierte Ziele zurückzugreifen [Raschka+2021, S. 30]. Da bei der folgenden Bearbeitung vorwiegend überwachte Lernalgorithmen im Fokus stehen, werden deren Charakteristiken, im Gegensatz zu den vorangegangenen Arten, im nächsten Abschnitt detaillierter betrachtet.

2.2 Überwachtes Lernen

Beim Überwachten Lernen steht im Wesentlichen das Ziel im Vordergrund, ein Modell mittels gekennzeichneter Trainingsdaten zu entwerfen, welches für zukünftige oder unbekannte Datensätze eine Vorhersage treffen kann. Der Terminus „überwacht“ bezieht sich hierbei auf die Eingabedaten, welche bereits mit gewünschten Ausgabewerten, auch Labels genannt, versehen wurden [Raschka+2021, S. 31]. Die überwachten Lernverfahren werden in Lazy Learner und Eager Learner differenziert. Bei den Lazy Learnern werden die gelabelten Trainingsdaten abgespeichert, woraufhin auf Basis dessen ein Ähnlichkeitsmaß festgelegt wird. Dieses Maß dient dem Verfahren als Ausgangspunkt, die Werte der unbekannten Objekte zu vergleichen, um im selben Zug eine Klasse abzuleiten. Im Gegensatz dazu wird beim Training der Eager-Learner-Methoden auf Grundlage der gelabelten Trainingsdaten ein generalisierbares Modell generiert, welches mittels Deduktion auf die unbekannten Objekte angewendet wird [Maurer+2015, S. 469].

Den überwachten Lernverfahren wird allgemein eine große Anzahl an Ein- und Ausgabeinformationen bereitgestellt, welche im Vorfeld bereits in die richtigen Kategorien eingeordnet wurden. Diese gelabelten Datensätze sind deshalb elementar für die Funktionsweise der überwachten Methoden. Grundsätzlich handelt es sich im Anwendungsbereich der überwachten Lernverfahren um Regressions- oder Klassifikationsprobleme [Frochte 2021, S. 21f]. Regressionsanalysen werden dazu verwendet, Zielvariablen für stetige Werte vorauszusagen. Klassifikationsmodelle hingegen werden dazu

eingesetzt, unbekannte Objekte auf Grundlage von Merkmalen in Klassen einzuordnen [Raschka+2021, S. 44]. Da der Fokus dieser Bachelorthesis jedoch auf einer Klassifikationsproblematik liegt, wird diese Technik im folgenden Abschnitt nochmal genauer betrachtet.

2.3 Klassifikation

Der Begriff Klassifikation findet sich in vielen Fachbereichen wieder und wird durch unterschiedliche Definitionen beschrieben. In Hinblick auf künstliche Intelligenz oder maschinelles Lernen hat die Klassifikation im Wesentlichen die korrekte Vorhersage von Objektklassen zum Ziel. Klassifikationen werden häufig bei Daten angewendet, welche einen hohen Informationsgehalt beinhalten [Maurer+2015, S. 431].

Wie bereits im vorherigen Kapitel festgestellt wurde, findet die Klassifizierung im Fachgebiet des überwachten Lernens Anwendung. Um deren Zielstellung zu erreichen, werden zunächst die Eigenschaften der Klassen mittels vorausgehender Analysen bestimmt und darauf aufbauend in konkreten Intervallen die Merkmale definiert [Raschka+2021, S. 31].

2.4 Support Vector Machine

Support Vector Machines gehören zu den am häufigsten verwendeten Klassifizierern und werden im Wesentlichen dazu eingesetzt, die Dimensionen eines Problems zu erhöhen, mit dem Hintergrund, die Daten separierbar zu machen und zwischen den Klassengrenzen einen möglichst großen Abstand ohne auftretende Objekte zu erzeugen ([Raschka+2021, S. 104], [Frochte 2021, S. 405]). Dieser Abstand wird als Bereich zwischen der aufteilenden Hyperebene und den Support-Vektoren definiert. Die Support-Vektoren sind die Objekte der Trainingsdaten, welche der Hyperebene am nächsten sind [Raschka+2021, S. 104]. Für diesen Zweck werden sogenannte Kernel-Funktionen eingesetzt, welche sich je nach Anwendungszweck unterscheiden, aber grundsätzlich auf quadratischen Optimierungen basieren und individuelle Nebenbedingungen erfüllen. Die Resultate der SVM können durch das Optimieren der folgenden Hyperparameter verbessert werden [Frochte 2021, S. 405]. Durch den Strafparameter C kann der Algorithmus für eine Fehlklassifizierung bestraft werden. Wenn der Wert für C groß gewählt wird, bedeutet das eine strengere Bestrafung und führt tendenziell zu komplexeren Modellen, welche jedoch die Gefahr der Überanpassung begünstigen. Wird der Wert für C dagegen klein gewählt, erfolgt eine geringere Bestrafung, weshalb Ausreißer

zumeist akzeptiert und in den meisten Fällen falsch klassifiziert werden [Raschka+2021, S. 107]. Der Hyperparameter Gamma wird für nichtlineare Support Vector Machines benutzt, um den Einfluss der Trainingsdaten auf die Hyperebene zu bestimmen. Große Werte für Gamma bewirken dabei, dass die Entscheidungsgrenze weniger glatt verläuft. Kleinere Werte für Gamma sorgen dagegen für einen glatten Verlauf der Entscheidungsgrenze [Raschka+2021, S. 113]. Durch die Optimierungen können detailliertere Schlussfolgerungen über die ideale Trennung der unterschiedlichen Klassen gezogen werden [Frochte 2021, S. 405].

Das Verfahren der SVM hat seit den siebziger Jahren in Hinblick auf Regressions- und Klassifizierungsprobleme stark an Bedeutung gewonnen, weil es sich unter anderem gut für die Lösung von nichtlinearen Klassifikationsproblemen eignet [Schönbrodt 2019, S. 10f]. Die Anwendung des Verfahrens kommt nur bedingt für das Verarbeiten unskalierter Daten in Frage und erzielt deshalb bei der Verwendung von großen Datenbeständen tendenziell schlechtere Ergebnisse [Frochte 2021, S. 405].

2.5 Decision Tree

Decision Trees werden auf Basis der Datenstruktur eines Baumes erzeugt. Aufgrund dessen gehört zu den Bestandteilen des Entscheidungsbaums eine Wurzel und eine variable Anzahl an Knoten und Blättern. Die Auswertung des Algorithmus fängt bei der Wurzel an, welche darauf durch die Knoten zu den Blättern gelangt. Ein Blatt ist, hinsichtlich eines Klassifikationsproblems, ein Knoten, an welchem ein Objekt klassifiziert wird und welcher zudem keine weitere Differenzierung der Daten beinhaltet. Zu den Funktionalitäten des Knotens gehört im Wesentlichen die Entscheidung, welchen Teil des Baumes ein Objekt aufgrund seiner Merkmale, durchlaufen soll [Frochte 2021, S. 129]. Entscheidungsbäume finden prinzipiell dann Verwendung, wenn die Interpretierbarkeit des Klassifikationsprozesses von Relevanz ist. Bei diesem Modell werden die Datensätze durch eine Folge von Fragen analysiert, um darauf aufbauend Entscheidungen an Knoten sowie deren Chronologie im Modell festzulegen. Die Knoten, welche durchlaufen werden, stellen dabei den Klassifizierungsprozess dar. Jeder Durchlauf eines Decision-Tree-Modells endet dabei in einem Blatt, welches dem Objekt eine Kategorie zuordnet [Raschka+2021, S. 115].

2.6 K-Nearest-Neighbor

Das K-Nearest-Neighbor-Verfahren unterscheidet sich von den beiden vorangegangenen Klassifikationsmethoden, da es in Bezug auf die Funktionsweise zu der Klasse der Lazy Learner gehört. Bei dieser Kategorie findet das Training erst im Moment der Anfrage statt. Die Klassifizierung eines Objekts basiert auf seinem Parameter k, welcher für die Anzahl der nächsten Nachbarn steht [Frochte 2021, S. 122]. Bei Lazy Learnern werden die Trainingsdaten lediglich gespeichert, was auch als instanzbasiertes Lernen bezeichnet wird. Vorteil ist, dass mit dem Lernvorgang kein Aufwand verbunden ist. Der grundsätzliche Ablauf kann durch folgende Schritte beschrieben werden. Eingangs erfolgt die Festlegung der Werte für k und der Abstandsmetriken. Danach werden die k nächsten Nachbarn der Instanz ausgewählt und anschließend durch Mehrheitsentscheidung dazu verwendet, dem Objekt eine Klassenbezeichnung zuzuweisen [Raschka+2021, S. 127f]. Ein Vorteil dieses Ansatzes liegt darin, dass das KNN-Modell um die Abfragestelle herum erzeugt wird. Folglich schafft das Verfahren die Opportunität, genauere Ergebnisse im Vergleich zu globalen Modellen zu erreichen. Das KNN-Verfahren ist zudem mit nur wenigen Parametern behaftet, was bedeutet, dass lediglich wenige Parameter eingestellt werden müssen [Frochte 2021, S. 122]. Ein weiterer Vorteil des speicherbasierten Ansatzes liegt darin, dass der KNN-Klassifikator sich bei neu hinzukommenden Trainingsdaten, unmittelbar an die Gegebenheiten anpasst [Raschka+2021, S. 127f].

2.7 Simulation

Simulation ist eine assoziationsreiche Begrifflichkeit und stammt ursprünglich aus dem Bereich des Militärs. Simulationen dienen grundsätzlich dazu, eine konkrete Vorstellung darüber zu gewinnen, welche Abläufe und Geschehnisse in einem definierten Weltausschnitt eintreten können. Der Weltausschnitt basiert auf Szenarien und Gegebenheiten, welche der realen Welt ähnlich sind und somit realistische Schlussfolgerungen auf die Realität begünstigen sollen. Der Ausschnitt einer Simulation stellt jedoch nur das Ergebnis einer Abstraktion dar, weil unter anderem nicht alle Eigenschaften eines Szenarios für die zu erforschende Problemstellung von Relevanz sind [Liebl 1995, S. 3f].

2.8 GNSS

Der Begriff „GNSS“ ist eine Abkürzung für „Global Navigation Satellite System“ und steht übersetzt für ein globales Satellitennavigationssystem. Die Idee hinter dem Satellitennavigationssystem ist, die Position eines Empfängers an jedem Punkt der Erde zu jeder Zeit und ohne zusätzliche Kommunikation exakt bestimmen zu können. Zudem soll es in der Lage sein, präzise Zeitinformationen und die genaue Geschwindigkeit des Nutzers übermitteln zu können [Schüttler 2014, S. 43]. Bis zum Anfang des 21. Jahrhunderts konnten private Nutzer mit GNSS lediglich eine Genauigkeit von etwa 50 bis 100 Meter erreichen. Ohne Hilfsmittel kann gegenwärtig bei der Anwendung des GNSS eine durchschnittliche Genauigkeit von 10 Meter erreicht werden [Schüttler 2014, S. 87].

3 Spezifische Grundlagen

In diesem Abschnitt geht es primär um die verwendeten Werkzeuge, welche speziell zur Umsetzung der folgenden Versuche beigetragen haben.

3.1 CARLA

CARLA bedeutet „Car Learning to Act“ und ist eine Open Source Software, mit der autonomes Fahren realitätsnah simuliert und erforscht werden kann. Die Simulation bietet eine Vielfalt an Sensoren und Szenarien, welche flexibel an gewünschte Umweltbedingungen angepasst werden können. CARLA besitzt ein verständliches Interface, welches die Schnittstelle zwischen der dynamisch simulierten Welt und dem interagierenden User anbietet, um somit eine funktionale Kontrolle und Wahrnehmung der einzelnen Objekte zu realisieren. Der Agent erhält somit Zugriff auf nahezu alle Informationen der dynamischen Objekte. Der sogenannte Hero bezeichnet dabei den Akteur, welcher vom Agenten ausgewählt beziehungsweise gesteuert wird. Überdies wird der Ausgangsagent der Simulation definiert, von welchem die Simulation betrachtet wird. Die Simulation von CARLA zeichnet sich durch realistische Renderings und zahlreiche Modifikationsmöglichkeiten aus [Dosovitskiy+2017].

3.2 Scikit-learn

Scikit-learn ist eine Open Source Software-Bibliothek, mit der vorwiegend überwachte und unüberwachte maschinelle Lernverfahren in Python realisiert werden können. Die Bibliothek deckt eine große Anzahl der populärsten Lernverfahren ab und bietet Einsteigern ein leicht verständliches Interface [Pedregosa+2011].

3.3 pandas

Die Software-Bibliothek pandas bietet ein weitreichendes Angebot an Funktionen und Datenstrukturen, um Daten schnell und unkompliziert bearbeiten/analysieren zu können. In Kombination mit Python wird daraus ein leistungsfähiges Analysewerkzeug, wobei das wichtigste Mittel für die folgenden Ausarbeitungen der Datentyp DataFrame und seine Methoden darstellt. DataFrame ist eine tabellarische zweidimensionale Datenstruktur, in welche die Datensätze gespeichert werden können. Durch die zahlreichen pandas-Methoden können die Daten analysiert, manipuliert und kalkuliert werden [McKinney 2012, S. 4f].

4 Anforderungen und Gesamtkonzept

In diesem Kapitel werden die Anforderungen beschrieben, welche das Gesamtkonzept erfüllen sollen, um eine bestmögliche Realisierung des Forschungsgegenstands zu garantieren. Zur optimalen Erörterung der Forschungsfrage, werden zunächst die benötigten Bestandteile und deren Eigenschaften beleuchtet. Die grundlegende Methode, welche dem Versuch zugrunde gelegt wird, ist die Klassifikation. Im Bereich des überwachten Lernens ordnet sie Objekte mittels vorheriger Beobachtungen in kategoriale Klassen ein, weil sich die meisten Klassifikationsmethoden in Bezug auf Datenanalyse/Mustererkennung als praktikabel erweisen [Raschka+2021, S. 31]. Bevor jedoch auf die Auswahl der Klassifizierungsmethoden eingegangen wird, werden die Vorgänge hinsichtlich der Datenbeschaffung thematisiert. Die Positionsdaten sollen mithilfe einer Simulation generiert werden. Im Folgenden müssen die Daten aus der Simulation extrahiert, auf fehlende Daten kontrolliert und gegebenenfalls umgeformt werden, bevor sie für die maschinellen Lernverfahren aufbereitet sind. Für diese Schritte wird die Softwarebibliothek „pandas“ genutzt, welche das Laden des Datensatzes ermöglicht und Methoden zur Manipulation/Analyse der Daten anbietet.

4.1 Anforderungen der Simulation

Um herauszufinden, ob sich GNSS-Daten zur Klassifikation eignen, müssen diese zunächst in einer Simulation erzeugt und erfasst werden. Abhilfe für die Realisierung der Simulation schafft hierbei die die Open Source Software CARLA, welche eine Vielfalt an Methoden und Spezifikationsmöglichkeiten zur Simulation von sich autonom bewegenden Verkehrsteilnehmern anbietet. CARLA ist als Client-Server-Modell konzipiert worden, wobei der Client das Interface zur Verfügung stellt. Der Server ist für den Betrieb der simulierten Welt und für das Rendern der grafischen Ausgabe verantwortlich [Dosovitskiy+2017]. Zunächst muss die Simulation durch das angebotene Interface konfiguriert werden. Die Konfiguration umfasst in diesem Szenario zum einen die vorherige Auswahl der Karten, um möglichst verschiedene Gegebenheiten zu schaffen, und zum anderen das Bestimmen der Art der Akteure, welche sich in der simulierten Welt fortbewegen, sowie deren Quantität. Durch die zahlreichen Kontroll- und Wahrnehmungsmöglichkeiten, welche CARLA anbietet, werden die Positionsdaten durch eine direkte Anfrage abgefragt und hinsichtlich verschiedener Zeitpunkte gespeichert. Nachdem die Daten empfangen wurden, finden diese prinzipiell in der Berechnung von Ausrichtung, Geschwindigkeit, Beschleunigung, Winkelgeschwindigkeit, Winkel zum

Helden und Distanz zum Helden Verwendung. Falls zu einem benötigten Zeitpunkt keine Aufzeichnung der Daten stattfand, werden diese im Nachgang durch die bereits vorhandenen Werte berechnet. Um die zeitliche Entwicklung der einzelnen Merkmale in ausreichender Form abzubilden, bedarf es einer Anzahl von zehn Zeitpunkten. Die breite Auswahl an Informationen soll dabei helfen, die Klassifikationen so exakt wie möglich zu gestalten.

Um die Chancen des Ansatzes umfassend aufzeigen zu können, beinhaltet jeder der folgenden Datensätze fünf verschiedene Typen an Verkehrsteilnehmern (siehe Tabelle 1), welche manuell separiert wurden. Dabei kann im folgenden Verlauf grundsätzlich zwischen motorisierten und nichtmotorisierten Objekten differenziert werden.

Klassenname	**Typbeschreibung**
Car	Personenkraftwagen (Ford Mustang)
Pedestrian	Fußgänger
Motorcycle	Motorrad (Kawasaki Ninja)
Truck	Nutzfahrzeug (Feuerwehrauto)
Bike	Fahrrad (Mountainbike)

Tabelle 1: Beschreibung der Klassenarten

Quelle: [Eigene Darstellung]

4.2 Genauigkeit der Positionsdaten

Die Positionen, welche der Simulation entnommen werden, garantieren eine hundertprozentige Genauigkeit und stehen deshalb in Konflikt zur Realität, da die GNSS-Daten in der realen Welt zumeist mit Unsicherheiten behaftet sind. Um diesen Konflikt entgegenzuwirken, werden die Positionsdaten mit randomisierten Unsicherheiten verrechnet und in separaten Datensätzen aufgezeichnet. Insgesamt wurden acht verschiedene Datensätze angefertigt, welche sich in Hinblick auf Karte und Genauigkeit der GNSS-Daten unterscheiden (siehe Tabelle 2).

Bezeichnung	Karte	Abweichung
dataset02_0m.csv	Town02	0m
dataset02_3m.csv	Town02	3m
dataset02_5m.csv	Town02	5m
dataset02_10m.csv	Town02	10m
dataset03_0m.csv	Town03	0m
dataset03_3m.csv	Town03	3m
dataset03_5m.csv	Town03	5m
dataset03_10m.csv	Town03	10m

Tabelle 2: Datensätze und ihre Eigenschaften

Quelle: [Eigene Darstellung]

Die voneinander abweichenden Daten sollen dabei helfen, verschiedene Szenarien zu gestalten, um weitere Möglichkeiten oder Restriktionen aufzuzeigen.

4.3 Karten der Simulation

CARLA stellt eine Auswahl von differenten Karten der Simulation zur Verfügung. Dabei lassen sich die einzelnen Umgebungen in Hinblick auf Größe und Komplexität wesentlich voneinander differenzieren. Für die Simulation wurden die Szenarien „Town02“ und „Town03“ gewählt, auf deren Eigenschaften und Unterschiede nachfolgend eingegangen wird. Die Differenzen der Karteneigenschaften sollen dazu dienen, die Merkmale der Klassen deutlicher voneinander unterscheiden zu können, um somit die Genauigkeit der Lernverfahren zu erhöhen.

„Town02“ verkörpert ein kleines Stadtszenario mit simplen Strukturen. Die wesentlichen Bestandteile des Straßennetzes umfassen lediglich kurze Geraden, 90-Grad-Kurven und Straßenmündungen, inklusive Ampeln (siehe Abbildung 1). Die erlaubten Höchstgeschwindigkeiten innerhalb des städtischen Raumes werden auf jeweils 30 oder 50 km/h begrenzt. An einer bewaldeten Geraden sind Geschwindigkeiten von bis zu 90 km/h

zulässig. Da die entsprechenden Geraden jedoch kurz gestaltet wurden, sind die simulierten Verkehrsteilnehmer nicht in der Lage, die zugelassenen Geschwindigkeiten zu erreichen. Bei einem Versuch haben vereinzelte Objekte deshalb eine Höchstgeschwindigkeit von 67 km/h erzielen können.

Abbildung 1: Simulationsmodell von "Town02"

Quelle: [Eigener Screenshot]

„Town03“ ist ein komplexeres Umfeld der Simulation, welches mehrere neue Merkmale in die Funktion der Simulation mit einbezieht. Zudem umfasst die Umgebung von „Town03“ mehr Bestandteile, welche in Abbildung 2 visualisiert werden. Das Verkehrsnetz ist größer und komplexer gestaltet, um den Unterschied der beiden Karten so deutlich wie möglich abzubilden. Zu den Besonderheiten des Straßennetzes zählen Verkehrskreisel, fünfspurige Kreuzungen, Unebenheiten im Straßenbild und Kurven, welche nicht im 90-Grad-Winkel verlaufen. Zudem sind Schnellstraßen mit längeren Geraden eingefügt worden, mit welchen die einzelnen Akteure je nach Klasse Geschwindigkeiten von bis zu 90 km/h erreichen dürfen. Der sonstige Stadtverkehr ist wie in der vorherigen Umgebung auf 30 oder 50 km/h begrenzt.

Abbildung 2: Simulationsmodell von "Town03"
Quelle: [Eigener Screenshot]

Eine detailliertere Ansicht der Karten aus der Vogelperspektive ist im Anhang A1 ersichtlich. Aus diesen Layouts lassen sich die Komplexitäts- und Größenunterschiede nochmal aus einer anderen Perspektive betrachten.

4.4 Fehlende Daten

Die Datensätze, mit welchen die Lernverfahren trainiert werden, werden im Dateiformat CSV gespeichert. Innerhalb der Datei erfolgt die Separation der Werte durch Kommas. Im Datensatz selbst treten wiederholt fehlende Daten auf. Diese entstehen, wenn sich der Held beispielsweise zum Simulationsstart an einer roten Ampel befindet. Aufgrund dessen kann man dieses Phänomen in den Spalten beobachten, in denen die Distanz (distance_to_hero) und der Winkel (angle_to_hero) von der Instanz zum Helden gespeichert werden. Das passiert, weil CARLA zum Beginn noch keine Werte für die Position des Helden zurückgibt. Die fehlenden Einträge werden während der Verarbeitung mit einem Standardwert „0“ belegt, weshalb beim späteren Training keine fehlenden Daten mehr existieren und diese deshalb in den nachstehenden Kapiteln nicht näher thematisiert werden.

4.5 Klassifikationsmethoden

Um herauszufinden, welches maschinelle Lernverfahren sich am besten für die Klassifikation der Positionsdaten eignet, sollen mehrere überwachte Lernverfahren implementiert und getestet werden, um final eine Gegenüberstellung der Ergebnisse zu erzielen. Im Allgemeinen sollen die trainierten Lernverfahren eine gute Vorhersage hinsichtlich des Typs der zukünftigen oder unbekannten Instanzen treffen können.

Bei der Programmierung der Lernverfahren soll auf bereits zur Verfügung stehende Methoden zurückgegriffen werden. Die Implementierungen werden in der Programmiersprache Python realisiert, weil dieser eine Reihe von praktikablen Softwarebibliotheken zur Verfügung steht. Konkret wird bei der Bearbeitung auf die Open Source Software-Bibliothek Scitkit-learn zurückgegriffen, welche viele Funktionalitäten anbietet, die im Kapitel der spezifischen Grundlagen näher erläutert wurden. Bei der Auswahl der Klassifikationsmethodiken wurde auf bereits etablierte und häufig verwendete Klassifikationsverfahren Bezug genommen.

Im ersten Versuch kam der Einsatz des Lazy Learners K-Nearest-Neighbor zustande. Dieser speichert die gelabelten Trainingsdaten ab und erzeugt auf Basis dessen ein Ähnlichkeitsmaß, welches für die schlussendliche Klassifizierung der Objekte genutzt wird. Eine zentrale Aufgabe soll hierbei der Optimierung des Parameters k obliegen, der die Anzahl der nächsten Nachbarn vorgibt, welche für die Einordnung relevant sind. Um bei der Erarbeitung einen optimalen Parameter auszumachen zu können, erfolgt im Zuge der Validierung eine Berufung auf die Anwendung der Scikit-learn-Methode GridSearchCV(). Hierbei wird der Methode eine Liste von Werten mitgegeben, welche parallel zu einer Kreuzvalidierung durchlaufen werden und am Ende den bestmöglichen Wert garantieren. Der Vorteil des KNN-Verfahrens liegt darin, dass mit dem Lernvorgang prinzipiell kein Aufwand verbunden ist und das KNN-Modell um die Abfragestelle herum erzeugt wird. Dies ergibt die Möglichkeit, exaktere Resultate zu generieren. In Relation zu globalen Modellen entsteht deshalb ein geringerer Optimierungsaufwand als Essenz minderer Parameteranzahlen.

Das Decision-Tree-Verfahren, welches als zweite Variante dienen soll, gehört zu den Eager-Learner-Methoden und erzeugt, im Gegensatz zum KNN-Verfahren ein generalisierbares Modell, welches auf gelabelten Trainingsdaten basiert und mittels Deduktion auf die unbestimmten Objekte angewendet werden soll. Das Verfahren wurde aufgrund seiner praktikablen Interpretierbarkeit gewählt. Dessen Darstellung kann dazu genutzt werden, einzelne Schritte des Klassifikationsprozesses zu visualisieren. Für eine ideale Realisierung des

Decision Trees müssen auch hier die Hyperparameter optimiert werden. Hierbei liegt der Fokus auf der Tiefe des Baums, dessen optimaler Wert wiederum durch die Scikit-Learn-Methode GridSearchCV() ermittels wird.

Das dritte Klassifikationsverfahren, welches in den folgenden Ausarbeitungen behandelt wird, ist die Support Vector Machine, welche ebenfalls den Verfahren der Eager Learnern zugehörig ist. Die SVM gehört zu den am häufigsten eingesetzten Klassifizierern und eignet sich gut für die Verarbeitung von skalaren Daten. Zudem eignen sie sich besonders bei der Lösung von nichtlinearen Klassifikationsproblemen. Da die Methode in Hinblick auf unskalierte Daten Schwierigkeiten hat, werden die Daten, welche für den Klassifikationsprozess benötigt werden (X_train, X_test), mit der Scikit-learn-Methode scale() entsprechend angepasst. Um die Entscheidungsgrenze der SVM optimieren zu können, werden verschiedene Werte der Hyperparameter C und Gamma in einer Liste gespeichert und anschließend durch die Scikit-learn-Methode GridSearchCV() kreuzvalidiert. In Hinblick auf die Kernel-Funktionen hat sich im Vorfeld die radiale Basisfunktion als beste Variante herausgestellt, weshalb in den weiterführenden Bearbeitungen lediglich von dieser ausgegangen wird.

Hinsichtlich des Trainings der Modelle werden die Datensätze durch die Scikit-learn-Methode train_test_split() in 70% Trainingsdaten und 30% Testdaten aufgeteilt. Die Architektur der Lernverfahren ähnelt sich im Aufbau stark, unterscheidet sich jedoch an einigen Eckpunkten. Bei den nachstehenden Versuchen kann nur ein Teil der Kombinationsmöglichkeiten hinsichtlich Klassenanzahl, Kartenkomplexität und Genauigkeit der Positionsdaten betrachtet werden, weil der gesamte Umfang an Zusammenstellungen den Rahmen dieser Bachelorarbeit übersteigen würde.

5 Implementierung

Nachdem grundlegende Entscheidungen bezüglich des Designs festgelegt und allgemeine Anforderungen definiert wurden, wird folgend die entsprechende Realisierung thematisiert. Zu Beginn erfolgt die Betrachtung der Komponenten der Lernverfahren sowie deren Funktionalitäten und Abläufe. Anschließend werden die Besonderheiten der gewählten Szenarien in Hinblick auf Umgebung, Genauigkeit der Positionsdaten und Klassenanzahl spezifiziert, um diese auf den Algorithmus anzuwenden. Die Resultate der trainierten Szenarien werden unmittelbar in den jeweiligen Unterkapiteln ausgewertet. Die Intention der praktischen Realisierung ist es, ein detailliertes Bild darüber zu erhalten, wann und in welchen Situationen die einzelnen Lernverfahren am besten mit den gegebenen Informationen performen.

5.1 K-Nearest-Neighbor

Der Implementierungsablauf des K-Nearest-Neighbor Algorithmus ähnelt den bereits thematisierten Verfahren stark. Zu Beginn müssen die Daten gelesen und einer Variablen df zugewiesen werden. Dafür wird die pandas-Methode read_csv()eingesetzt, um die Datei dataset02_0m.csv zu laden und df zuzuweisen. Nachfolgend werden die Spalten des Datensatzes mit der pandas-Methode drop() separiert, welche keine relevanten Informationen in Bezug auf die Klassifikation beinhalten. Analog zu anderen Verfahren trifft dies nur auf Spalte ID zu, welche mit zufällig zugeordneten Werten belegt worden ist. Da K-Nearest-Neighbor tendenziell weniger gelabelte Trainingsdaten benötigt, werden die Daten von df nach dem jeweiligen Klassentyp gefiltert und anschließend mit der Scikit-learn-Methode resample() auf die gewünschte Anzahl an Einträgen heruntergeprobt. Um die weiteren Schritte jedoch mit allen Klassen fortzuführen zu können, müssen die einzelnen Variablen (df_car_downsampled etc.) wieder zu einem Datensatz (df_downsampled) zusammengefügt werden. Damit die vorhandenen Ressourcen für das Training verwendet werden können, werden diese nach der Datenvorverarbeitung in Klassifikationsdaten (X) und Vorhersagedaten (y) aufgeteilt, um sie nachfolgend mit der Scikit-learn-Methode train_test_split() in Trainingsdatensätze (X_train, y_train) und Testdatensätze (X_test, y_test) zu fraktionieren. Nachdem die Werte der Merkmale von X_train und X_test mit der Scikit-learn-Methode scale() normiert wurden, muss zunächst ein untrainierter Klassifizierer knn_clf mit der Sciki-learn-Methode KNeighborsClassifier()erzeugt werden. Daraufhin finden Optimierungsverfahren Anwendung, welche im nächsten Kapitel näher beschrieben werden. Der bereits erzeugte Klassifizierer wird

mit den Scikit-learn-Methoden fit(), predict() und metrics.accuracy_score() zunächst mit den Trainingsdaten X_train_scaled und y_train trainiert, dann für das Vorhersagen unbekannter Daten verwendet und zum Schluss für die Berechnung sowie die damit einhergehende Ausgabe der Genauigkeit genutzt.

5.1.1 Klassenanzahl

Zu Beginn der Untersuchung zum K-Nearest-Neighbor-Verfahren erfolgt zunächst die Überprüfung der Konsequenzen der Klassenanzahl auf die Erkennungsrate des Algorithmus überprüft. Die nachfolgenden Versuche werden aufeinander aufbauend um eine Kategorie erweitert, um die jeweiligen Ergebnisse im Nachhinein zu analysieren. In diesem Unterkapitel findet der Datensatz dataset02_0m.csv Verwendung. Dieser bezieht sich auf die Karte „Town02“ mit idealen GNSS-Daten. Die Auswirkung der Genauigkeit der GNSS-Informationen sowie der Komplexität der Karte werden in den nachfolgenden Kapiteln 5.3.2 und 5.3.3 näher beleuchtet. Einleitend soll erforscht werden, welche Resultate das K-Nearest-Neighbor-Verfahren erreichen kann, wenn zunächst die beiden Kategorien „Car“ und „Pedestrian“ klassifiziert werden sollen und die optimalen Hyperparameter bereits bestimmt wurden.

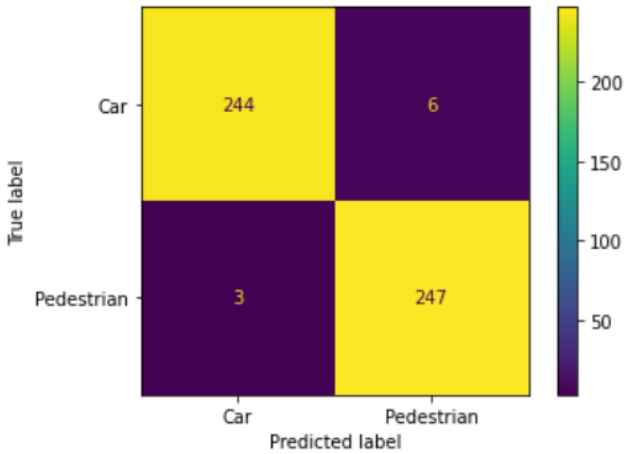

Abbildung 3: KNN - 2-klassig - Town02 - 0m
Quelle: [Eigene Darstellung]

Die in Abbildung 3 enthaltene Wahrheitsmatrix visualisiert die Tatsache, dass die unbestimmten Instanzen beider Kategorien mit einer großen Exaktheit klassifiziert werden konnten. Bei den „Car“-Objekten wurden 244 von 250 korrekt bestimmt. Aus dieser Quote lässt sich eine Genauigkeit von 97,60 % ableiten. Im gleichen Zuge werden 247 von 250 „Pedestrians“ richtig erkannt. Folglich liegt die Erkennungsrate dieser Objekte bei 98,80 %. Infolgedessen beträgt der Durchschnitt der Akkuranz beider Klassen 98,20 %. Das K-Nearest-Neighbor-Verfahren ist demzufolge im Stande, die idealen Positionsdaten der Objekte, welche in der Umgebung „Town02“ erfasst wurden, für die KNN-Klassifikation von unbekannten Objekten einzusetzen. In der Konsequenz dessen wird im weiteren Verlauf die Klassenanzahl um die Klasse „Bike“ ergänzt.

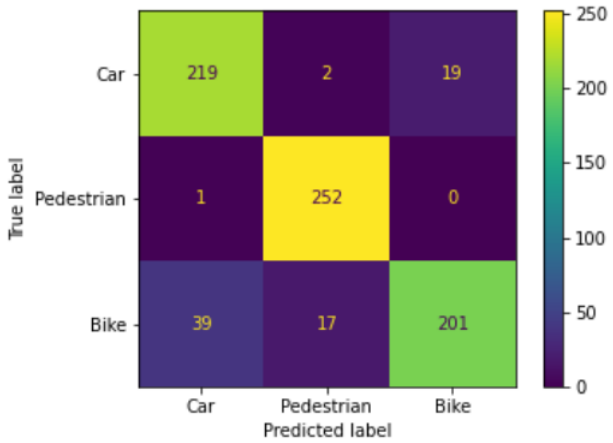

Abbildung 4: KNN - 3-klassig - Town02 - 0m
Quelle: [Eigene Darstellung]

Die Ergebnisse von Abbildung 4 visualisieren, dass weiterhin alle Klassen mit einer großen Exaktheit vorhergesagt werden können. Bei der Klassifikation der „Cars“ wurden 219 von 240 Instanzen korrekt erkannt. Dies resultiert in einer Genauigkeit von 91,25 %. Die „Pedestrians“ erreichten bei diesem Versuch eine Erkennungsrate von 99,60 %, was sich durch den Umstand veräußert, dass 252 von 253 Objekten richtig prognostiziert wurden. Die Exaktheit der ergänzten Klasse „Bikes“ liegt mit 201 von 257 richtig erkannten bei 78,21 %. Infolgedessen werden die Kategorien mit einer durchschnittlichen Genauigkeit von 89,69 % vorausgesagt. Daraus lässt sich schlussfolgern, dass die Klassen des Versuchs in Gegenüberstellung zur 2-klassigen KNN-Klassifikation, um 8,51 % weniger gut erkannt wurden.

Angesichts der Tatsache, dass sich die Ergebnisse des K-Nearest-Neighbor-Verfahren trotz gesteigerter Klassenanzahl weiterhin auf einem hohen Niveau befinden, wird die KNN-Klassifikation um eine weitere Klasse namens „Truck“ erweitert. Im Anschluss werden die Effekte der nochmals erhöhten Komplexität auf die Akkuranz des Verfahrens ermittelt.

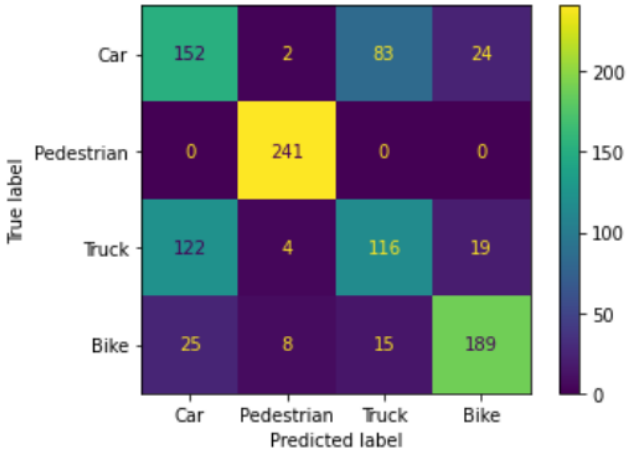

Abbildung 5: KNN - 4-klassig - Town02 - 0m
Quelle: [Eigene Darstellung]

Die Wahrheitsmatrix in Abbildung 5 verdeutlicht, dass sich die Resultate der 4-klassigen KNN-Klassifikation im Vergleich zur 2-klassigen Einordnung, teilweise verschlechtert haben. Die Instanzen der „Cars“ wurden mit einer Genauigkeit von 58,24 % bestimmt, was sich aus dem Umstand ergibt, dass von 261 Testobjekten 152 korrekt prognostiziert wurden. Dies ergibt einen Verlust von 33,01 % im Gegensatz zur 3-klassigen KNN-Klassifikation. Von den Objekten der „Pedestrians“ wurden 251 von 251 korrekt vorhergesagt, was somit eine Erkennungsrate von 100,00 % ergibt. Im Hinblick auf die Resultate der „Bikes“ wird ein Genauigkeitsgewinn von 1,54 % in Gegenüberstellung zur 3-klassigen KNN-Klassifikation deutlich. Dabei wurden von 237 „Bikes“ 189 richtig bestimmt, was in einer Exaktheit von 79,75 % mündet. Die erweiterte Kategorie der „Trucks“ wird mit einer Genauigkeit von 44,44 % erfasst. In diesem Zusammenhang wurden 116 von 261 „Truck“-Instanzen korrekt zugeordnet. In der Konsequenz dessen liegt die Erkennungsrate der Kategorien im Durchschnitt bei 70,61 %.

Um die Grenzen des K-Nearest-Neighbor-Verfahren hinsichtlich der idealen GNSS-Informationen herauszufinden, wird die Komplexität ein letztes Mal durch das Hinzufügen der Klasse „Motorcycle“ erhöht.

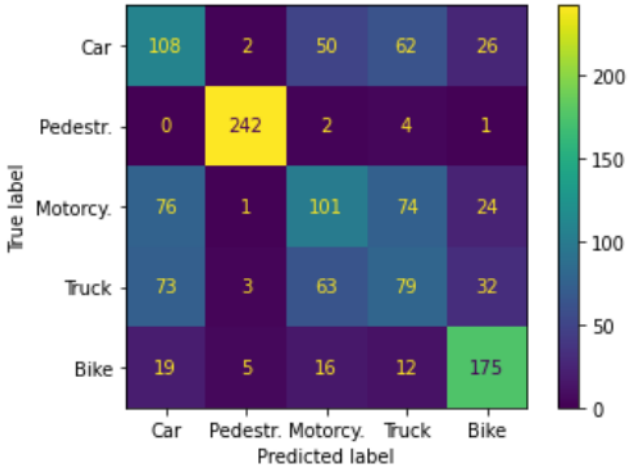

Abbildung 6: KNN - 5-klassig - Town02 - 0m
Quelle: [Eigene Darstellung]

Die grafische Darstellung der Ergebnisse der 5-klassigen KNN-Klassifikation zeigt auf, dass sich lediglich die Erkennungsraten der Kategorien „Pedestrian“ und „Bike“ weiterhin auf einem gehobenen Niveau befinden. Im Gegensatz zum vorherigen Szenario wurden von 248 „Cars“ 108 korrekt erkannt, was aus einer Exaktheit von 43,55 % hervorgeht und parallel einem Defizit von 14,69 % entspricht. Die Instanzen der „Pedestrians“ wurden mit einer Genauigkeit von 97,19 % vorausgesagt, was sich in dem Umstand äußert, dass 242 von 249 Objekten fehlerfrei bestimmt wurden. Im selben Zuge wurden 175 von 227 „Bikes“ richtig vorhergesagt, was einer Erkennungsrate von 77,09 % entspricht. Bei der Kategorisierung der „Trucks“ kann sich im Vergleich zu den Ergebnissen der 4-klassigen KNN-Klassifikation ein Genauigkeitsdefizit von 12,84 % beobachten lassen, was sich durch die Tatsache veräußert, dass lediglich 79 von 250 Testobjekten korrekt prognostiziert wurden. Konkludierend ergibt sich eine Genauigkeit von 31,60 %. Von den Objekten der ergänzten Kategorie „Motorcycle“ wurden 101 von 276 richtig klassifiziert, was einer Erkennungsrate von 36,59 % entspricht. Die durchschnittliche Bestimmungsrate aller Klassen beläuft sich auf 57,20 % und resultiert somit in einem Genauigkeitsverlust von 13,41 % in Gegenüberstellung zur 4-klassigen KNN-Klassifikation.

5.1.2 Auswirkung der Kartenkomplexität

Bisher wurden im ersten Unterkapitel 5.1.1 die grundlegenden Möglichkeiten des K-Nearest-Neighbor-Verfahren in Hinblick auf die Verwendung von idealen Positionsdaten, welche innerhalb einer trivialen Karte aufgezeichnet wurden, untersucht. Anschließend werden die Auswirkungen einer höheren Kartenkomplexität auf die Genauigkeit der KNN-Klassifikationen ermittelt. Daher wird im folgenden Kapitel der Datensatz dataset03_0m.csv verwendet, welcher auf die vielschichtigere Umgebung „Town03“ zurückzuführen ist. Angesichts der Tatsache, dass die durchschnittliche Exaktheit der 5-klassigen KNN-Klassifikation mit 57,20 % relativ niedrig ist, wird diese in den folgenden Unterkapiteln nicht mehr behandelt. Im ersten Schritt wird eingangs die 2-klassige KNN-Klassifikation hinsichtlich der Karte „Town03“ betrachtet.

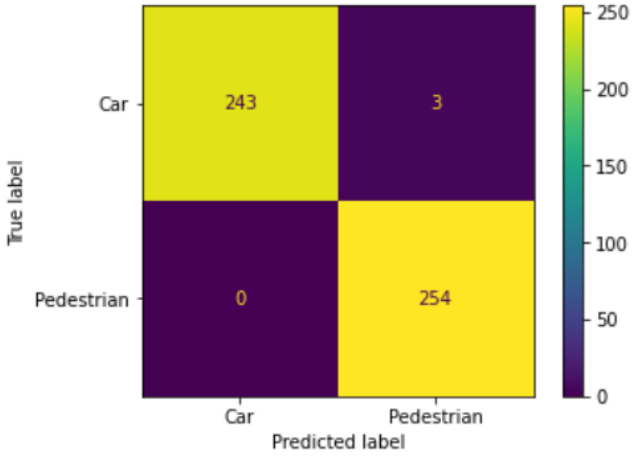

Abbildung 7: KNN - 2-klassig - Town03 - 0m
Quelle: [Eigene Darstellung]

Die Resultate der 2-klassigen KNN-Klassifikation sind, wie in Abbildung 7 zu sehen, annähernd analog zu den Ergebnissen der 2-klassigen Einordnung des vorangegangenen Unterkapitels. Bei den „Car“-Instanzen wurden 243 von 246 Testobjekten korrekt zugeordnet. Dies verkörpert eine Erkennungsrate von 98,78 %. Die „Pedestrian“-Objekte wurden mit einer Genauigkeit von 100,00 % richtig vorausgesagt. Zusammengenommen erzielt die 2-klassige KNN-Klassifikation im Mittel eine Akkuranz von 99,39 % und bedeutet ein Genauigkeitsgewinn von 1,19 % in Gegenüberstellung zu den Ergebnissen der vorherigen 2-klassigen Kategorisierung, welche hinsichtlich des Datensatzes dataset02_0m.csv verzeichnet

wurden. Um die Auswirkungen der Umgebung „Town03“ deutlicher abbilden zu können, wird die Anzahl der Klassen im Folgenden um die Kategorie „Bike“ erweitert und deren Resultate im Anschluss ausgewertet.

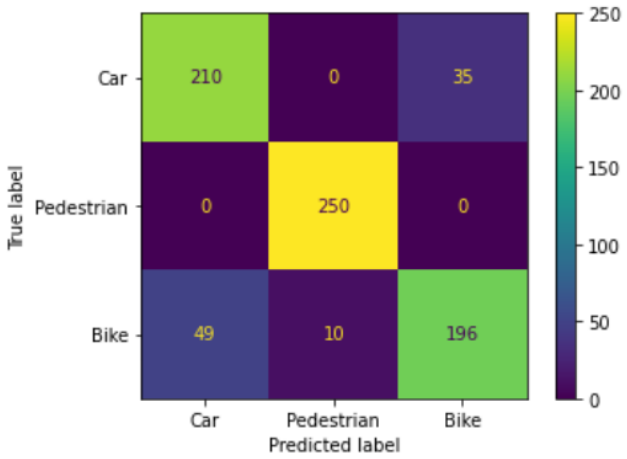

Abbildung 8: KNN - 3-klassig - Town03 - 0m
Quelle: [Eigene Darstellung]

Die Resultate der in Abbildung 8 enthaltenen Wahrheitsmatrix verbildlichen die Tatsache, dass die 3-klassige KNN-Einordnung im Vergleich zum vorherigen Unterkapitel hinsichtlich der komplexeren Karte eine minimal schlechtere Klassifizierungsleistung vollbringt. Die „Car“-Objekte wurden mit einer Erkennungsrate von 85,71 % bestimmt, was sich aus dem Umstand ergibt, dass von 245 Testinstanzen 210 richtig erkannt wurden. „Pedestrians“ wurden in diesem Versuch mit einer Exaktheit von 100,00 % korrekt vorausgesagt. Die Objekte der Kategorie „Bike“ wurden mit 196 von 255 Testobjekten korrekt klassifiziert, was sich in einer Genauigkeit von 76,86 % äußert. Wenn man alle Kategorien zusammen betrachtet, werden diese im Mittel mit einer Erkennungsrate von 87,52 % richtig prognostiziert, was einem Genauigkeitsverlust von 2,17 %, in Gegenüberstellung zu den Ergebnissen des vorherigen 3-klassigen K-Nearest-Neighbor-Verfahrens, repräsentiert.

Um eine Tendenz bezüglich der Auswirkung der Kartenkomplexität auf die Genauigkeit der Klassifikation aufstellen zu können, werden im letzten Versuch dieses Unterkapitels die Effekte der Umgebung „Town03“ auf die 4-klassige KNN-Kategorisierung überprüft.

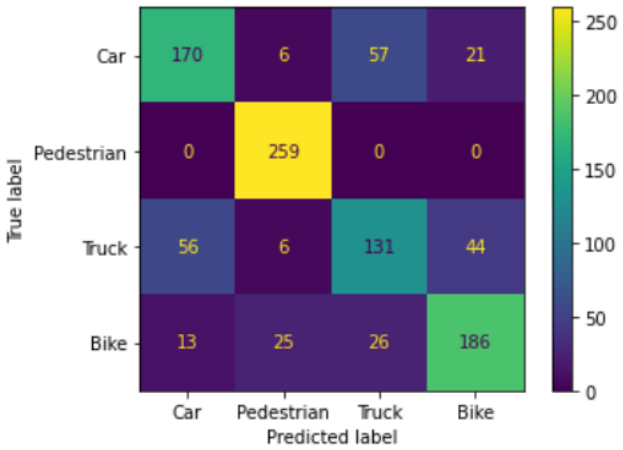

Abbildung 9: KNN - 4-klassig - Town03 - 0m
Quelle: [Eigene Darstellung]

In Hinblick auf die Objekte der Klasse „Car“ kann sich aus Abbildung 9 die Information entnehmen lassen, dass 170 von 254 Testobjekten korrekt klassifiziert wurden und resultierend daraus eine Genauigkeit von 66,93 % abgeleitet werden kann. Zudem konnten die Instanzen der Klasse „Pedestrian“ mit einer Erkennungsrate von 100,00 % richtig zugeordnet werden. Die „Bikes“ konnten mit 186 von 250 richtig erkannten Testobjekten eine Exaktheit von 74,40 % erreichen. Die Kategorie der „Trucks“ wurde mit einer Akkuranz von 55,27 % am wenigsten korrekt vorhergesagt. Dies veräußert sich durch die Tatsache, dass lediglich 131 von 237 „Trucks“ richtig erkannt wurden. In Konsequenz dessen liegt die Genauigkeit aller Klassen im Mittel bei 74,15 % und mündet in der Erkenntnis, dass das 4-klassige K-Nearest-Neighbor-Verfahren mit den Datensatz dataset03_0m.csv eine um 3,54 % höhere Erkennungsrate erzielen kann. Aus dieser Erkenntnis kann sich ableiten lassen, dass die Steigerung der Umgebungskomplexität auf die einzelnen Kategorisierungen erst bei der 4-klassigen KNN-Klassifikation eine sichtliche Verbesserung hervorrufen kann.

5.1.3 Genauigkeit der Positionsdaten

Im finalen Unterkapitel zum K-Nearest-Neighbor-Verfahren werden abschließend die Auswirkungen der Genauigkeit der Positionsdaten auf die Ergebnisse der einzelnen Versuche beleuchtet. Bislang wurden für die Bestimmung der Merkmalswerte ideale GNSS-Daten aus der Simulation verwendet. In der Realität sind die Positionsdaten jedoch mit Unsicherheiten von bis zu zehn Meter behaftet. Aufgrund dessen werden in den folgenden Versuchen

verschiedene Ungenauigkeiten hinsichtlich der Karte „Town03“ untersucht. Dafür kommen die Datensätze dataset03_3m.csv, dataset03_5m.csv und dataset03_10m.csv zum Einsatz. Im Hinblick auf die Klassenanzahl wird in diesem Unterkapitel primär die 3-klassige KNN-Klassifikation verwendet, da aufgrund der vorherigen Untersuchungen die Differenzierung von motorisierten und nichtmotorisierten Klassen am aussagekräftigsten war. Eingangs wird die Kategorisierung in Bezug auf dataset03_3m.csv analysiert.

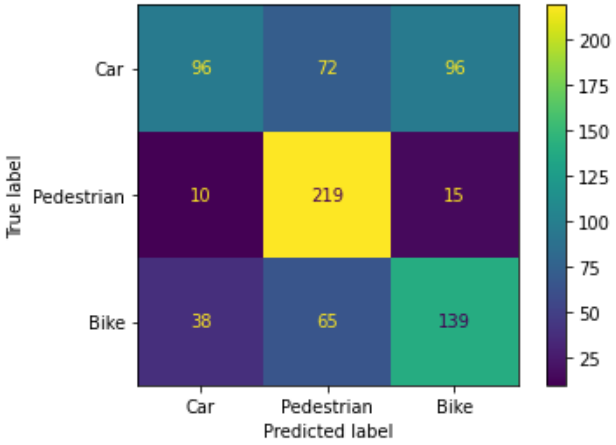

Abbildung 10: KNN - 3-klassig - Town03 - 3m
Quelle: [Eigene Darstellung]

Abbildung 10 veranschaulicht, dass die Ergebnisse des 3-klassigen K-Nearest-Neighbor-Verfahrens signifikant durch die Unsicherheit der Positionsdaten beeinflusst werden. Von den „Car“-Instanzen wurden 96 von 264 korrekt erkannt. Dies resultiert in einer Exaktheit 36,36 % und ergibt, verglichen mit den Resultaten des vorausgehenden Kapitels, einen Verlust von 49,35 %. Die „Pedestrians“ verloren parallel dazu 10,25 %, was sich durch den Umstand verdeutlicht, dass 219 von 244 Testinstanzen richtig vorhergesagt wurden. Die Objekte der Klasse „Bike“ wurden mit einer Genauigkeit von 57,44 % korrekt prognostiziert. Im Mittel liegt die Erkennungsrate aller Klassen bei 61,18 %, was im Vergleich zur 3-klassigen KNN-Klassifikation ohne Verfälschungen ein Defizit von 26,34 % zur Folge hat. Im folgenden Schritt wird die maximale Ungenauigkeit auf fünf Meter angehoben und deren Auswirkung analysiert.

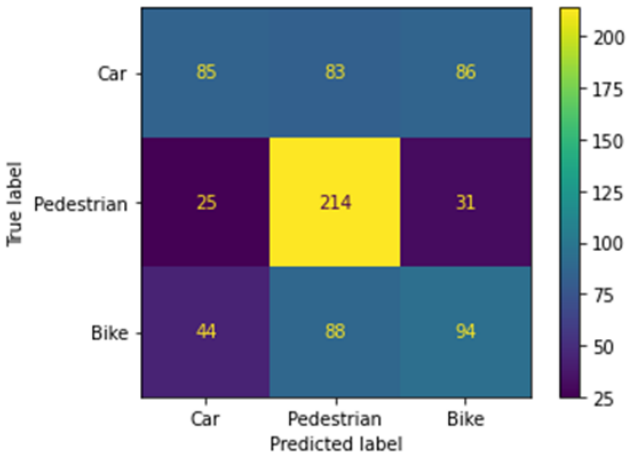

Abbildung 11: KNN - 3-klassig - Town03 - 5m
Quelle: [Eigene Darstellung]

Die Ergebnisse der in Abbildung 11 enthaltenen Wahrheitsmatrix visualisieren den Umstand, dass die erhöhten Verfälschungen von bis zu fünf Metern erneut zu Genauigkeitsverlusten führen. Die „Car“-Objekte wurden mit einer Akkuranz von 33,46 % richtig vorhergesagt, was sich durch die Tatsache äußert, dass von 254 Testobjekten 85 korrekt bestimmt wurden. „Pedestrians“ wurden in diesem Durchlauf mit einer Erkennungsrate von 79,26 % richtig prognostiziert. Die Instanzen der Klasse „Bike“ wurden mit 94 von 226 Testinstanzen richtig erkannt, was sich in einer Exaktheit von 41,59 % veräußert. Wenn man alle Klassen in Summe beleuchtet, werden diese im Durchschnitt mit einer Genauigkeit von 51,44 % korrekt vorhergesagt, was in einem Defizit von 9,74 % im Vergleich zu den vorherigen Ergebnissen resultiert. Im letzten Schritt dieses Unterkapitels werden folgend die Auswirkungen einer maximalen Abweichung von zehn Metern untersucht.

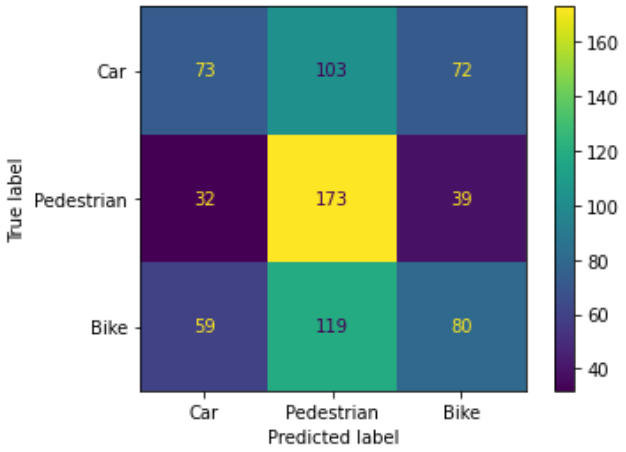

Abbildung 12: KNN - 3-klassig - Town03 - 10m
Quelle: [Eigene Darstellung]

Abbildung 12 visualisiert, dass die Resultate der 3-klassigen KNN-Klassifikation durch die Verfälschung von bis zu zehn Metern sichtlich beeinträchtigt werden. Von den Objekten des Typs „Car“ wurden 73 von 248 richtig zugeordnet. Dies ergibt eine Genauigkeit von 29,44 % und bedeutet, verglichen mit den Ergebnissen der vorherigen KNN-Klassifikation, eine erneute Einbuße von 4,02 %. Die Exaktheit der „Pedestrians“ verlor parallel dazu 8,36 %, was sich durch den Umstand äußert, dass von 244 „Pedestrians“ nur 173 richtig erkannt wurden. Die „Bikes“ wurden mit einer Akkuranz von 31,01 % korrekt prognostiziert. Folglich liegt die durchschnittliche Exaktheit aller Klassen bei 43,78 %. In direkter Gegenüberstellung zur vorangegangenen Klassifikation mit Abweichung der GNSS-Daten von maximal fünf Metern liegt ein Genauigkeitsverlust von 7,66 % vor.

5.2 Decision Tree

Die Implementation des zweiten Verfahrens ist hinsichtlich des Aufbaus ähnlich gestaltet, weshalb lediglich auf die Unterschiede zum ersten Verfahren genauer eingegangen wird. Zuerst wird der Datensatz, mit welchem der Decision Tree erzeugt werden soll, mit der pandas-Methode read_csv() gelesen und in einer Variablen df gespeichert. Nachdem die unrelevanten Spalten von der Tabelle des Datensatzes df entfernt (drop()) und mit den nachfolgenden Spalten ersetzt worden sind, werden die Daten nach ihrem Typ gefiltert und in getrennten Datensätzen (df_car etc.) gespeichert. Da für Entscheidungsbäume nur wenig gelabelte Trainingsdaten benötigt werden, werden die einzelnen Datensätze auf eine bestimmte Anzahl an Einträgen

heruntergeprobt (resample()) und danach erneut zu einer Variablen df_downsampled zusammengefügt (concat()).

Nachdem die Datenvorverarbeitung abgeschlossen wurde, kann nun die eigentliche Vorbereitung für das Training beginnen. Dafür wird zunächst der verkleinerte Datensatz in X (Klassfikationsdaten) und y (Vorhersagedaten) aufgeteilt und danach nochmals mit der Scikit-learn-Methode train_test_split() in Trainingsdaten (X_train, X_test) und Testdaten (y_train, y_test) separiert. Darauffolgend wird mit der Scikit-learn-Methode tree.DecisionTreeClassifier() ein untrainierter Klassifizierer erzeugt und in der Variable dt_clf gespeichert. Anschließend wird dt_clf mit der Scikit-learn-Methode fit() und den Trainingsdatensätzen y_train und X_train trainiert. Daraufhin wird der trainierte Decision Tree mit der Matplotlib-Methode tree.plot_tree() definiert und mit plt.show() ausgegeben. Da die dargestellten Bäume je nach Szenario sehr groß werden können, muss zuvor die Darstellungsgröße mit der Matplotlib-Methode figure() eingestellt werden. Danach sind weitere Analyse- und Evaluierungmöglichkeiten implementiert, welche im Kapitel 6 Fokus finden. Die Methoden sollen dazu dienen, den besten Wert für Alpha beziehungsweise für die Beschneidung des Baumes zu finden und auszugeben. Nachdem der Klassifizierer optimiert wurde, kann df_clf dazu genutzt werden, unbekannte Objekte in die jeweiligen Typenklassen einzuordnen (predict()). Die Resultate werden in pred gespeichert und am Ende dazu genutzt, die Erkennungsrate auszurechnen (metrics.accuracy_score()).

5.2.1 Klassenanzahl

Im Auftakt der Versuchsreihe zum Decision Tree wird die Auswirkung der Klassenanzahl auf die Genauigkeit untersucht. Die folgenden Schritte werden jeweils aufeinander aufbauend um eine Klasse ergänzt, wobei die auftretenden Unterschiede oder Gemeinsamkeiten im Anschluss hervorgehoben werden. Da die Auswirkungen aller verfügbaren Klassen bereits im Kapitel 5.1.1 detailliert untersucht worden sind und die Schlussfolgerung dessen darin besteht, dass dem Algorithmus besonders die Unterscheidung von motorisierten und nichtmotorisierten Objekten gelungen ist, wird in den folgenden Versuchen lediglich der Unterschied zwischen der 3- und 4-klassigen DT-Klassifikation betrachtet. Das soll zur Erkenntnis beitragen, ob ein Klassifikationsverfahren bessere Resultate hinsichtlich der Unterscheidung von motorisierten Verkehrsteilnehmern liefern kann. Bei den folgenden Versuchen dieses Unterkapitels wird zunächst die Umgebung von „Town02“ ohne Abweichung der GNSS-Informationen betrachtet.

Die Auswirkungen der Exaktheit der GNSS-Daten sowie die Komplexität der Umgebung werden in den folgenden Kapiteln 5.2.2 und 5.2.3 detaillierter beleuchtet. Eingangs soll untersucht werden, welche Ergebnisse der Decision Tree erzielen kann, wenn lediglich drei Klassen kategorisiert werden müssen und die optimale Tiefe des Entscheidungsbaums bereits bestimmt wurde. Die Klassen, welche dafür auserwählt wurden, sind „Car“, „Pedestrian“ und „Bike“.

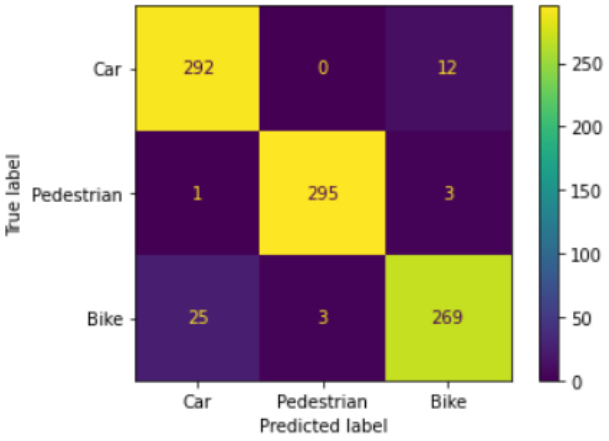

Abbildung 13: DT - 3-klassig - Town02 - 0m
Quelle: [Eigene Darstellung]

Abbildung 13 verdeutlicht den Umstand, dass alle Kategorien mit einer hohen Genauigkeit klassifiziert werden können. Bei der Einordung von „Cars“ werden von 304 Objekten 292 richtig prognostiziert. Dies ergibt eine Exaktheit von 96,05 % und bedeutet ein Defizit von 3,55 % verglichen mit der 2-klassigen DT-Klassifikation. Die Objekte der „Pedestrians“ wurden mit einer Erkennungsrate von 98,66 % korrekt vorausgesagt, was sich durch die Tatsache äußert, dass von 299 „Pedestrian“-Instanzen 295 richtig erkannt wurden. Die Genauigkeit der erweiterten Kategorie „Bikes“ beläuft sich mit 269 von 297 richtig zugeordneten Instanzen auf 90,57 %. Die Klassen werden im Durchschnitt mit einer Korrektheit von 95,09 % erkannt und schneiden somit im direkten Vergleich mit der 2-klassigen Klassifikation um 3,70 % schlechter ab. Da sich die Resultate des Decision Trees hinsichtlich der idealen GNSS-Daten mit über 95 % auf einem hohen Niveau befinden, wird die Klassenanzahl erneut um eine Kategorie „Truck“ ergänzt, um die Komplexität zu erhöhen. Im Folgenden werden dessen Auswirkungen auf die Genauigkeit näher betrachtet.

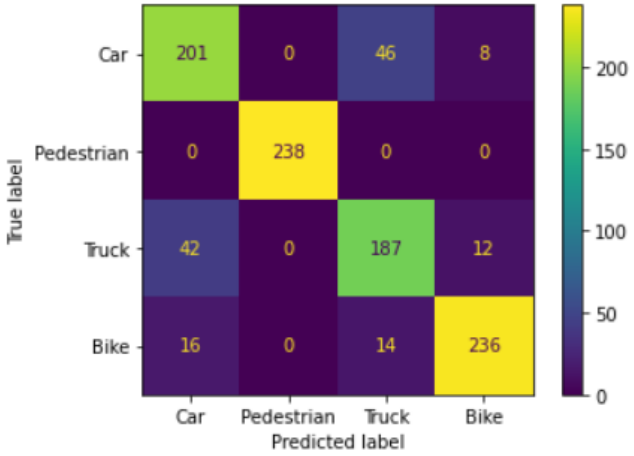

Abbildung 14: DT - 4-klassig - Town02 - 0m

Quelle: [Eigene Darstellung]

Abbildung 14 visualisiert, dass sich in den Ergebnissen der 4-klassigen DT-Klassifikation ein hohes Level an Genauigkeit registrieren ließ. Nachstehend werden die einzelnen Resultate detaillierter betrachtet. Die Objekte der „Cars“ wurden mit einer Erkennungsrate von 78,82 % vorhergesagt, was sich aus der Tatsache ergibt, dass 201 von 255 richtig erkannt wurden. Dies resultiert in einem Defizit von 17,23 % im Vergleich zur 3-klassigen DT-Klassifikation. Die Kategorie der „Pedestrians“ wird mit einer Exaktheit von 100,00 % korrekt vorhergesagt. Bei der Interpretation der „Bike“-Resultate wird ein geringer Verlust der Genauigkeit von 1,85 % im Vergleich zur 3-klassigen DT-Klassifikation ersichtlich. Von 236 „Bikes“ wurden 266 korrekt prognostiziert, was in einer Erkennungsrate von 88,72 % resultiert. Die ergänzte Klasse der „Trucks“ wird mit einer Exaktheit von 77,59 % verzeichnet. Dabei wurden von 241 „Truck“-Objekten 187 richtig erkannt. Infolgedessen liegt die durchschnittliche Erkennungsrate der Klassen bei 86,28 %.

5.2.2 Auswirkung der Kartenkomplexität

Bislang wurden im ersten Schritt die fundamentalen Möglichkeiten des Decision Tree hinsichtlich idealer GNSS-Informationen und einer simplen Umgebung inspiziert. Nachfolgend sollen die Auswirkungen einer gesteigerten Kartenkomplexität auf die Erkennungsraten der Klassen herausgefunden werden. Aufgrund dessen wird in den folgenden Versuchen die komplexere Karte „Town03“ eingesetzt. Die 2- und 5-klassige DT-Klassifikation wird in den

anschließenden Ausführungen nicht betrachtet, weil primär 3- und 4-klassige Modelle die gehaltvollsten Aussagen lieferten. Im Auftakt wird zunächst die 3-klassige DT-Klassifikation hinsichtlich der „Town03“-Umgebung analysiert.

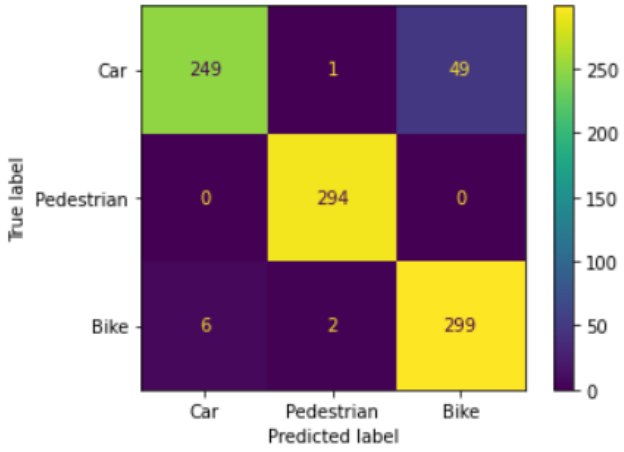

Abbildung 15: DT - 3-klassig - Town03 - 0m
Quelle: [Eigene Darstellung]

Die Wahrheitsmatrix in Abbildung 15 visualisiert den Umstand, dass die Resultate der 3-klassigen DT-Klassifikation in Hinsicht auf die diffizilere Umgebung ähnlich zu den Ergebnissen der vorherigen 3-klassigen DT-Klassifikation sind. „Pedestrians“ wurden hierbei mit einer Genauigkeit von 100,00 % richtig vorhergesagt. Bei den „Car“-Objekten kam es bei 249 von 299 zu einer korrekten Prognose, was einer Erkennungsrate von 83,28 % entspricht. Die Instanzen der Klasse „Bike“ wurden mit 299 von 307 richtig erkannt und resultieren in einer Exaktheit von 97,39 %. Alle Klassen erzielen zusammen in Bezug auf die komplexere Umgebung von „Town03“ im Mittel eine Akkuranz von 93,56 % und stellen infolgedessen eine Verschlechterung von 1,53 % im Vergleich zur Karte „Town02“ dar. Um die These zu ergründen, dass mit zunehmender Kartenkomplexität weitere Genauigkeitsverluste zu verzeichnen sind, wird im letzten Schritt dieses Unterkapitels die Auswirkung auf eine 4-klassige DT-Klassifikation untersucht.

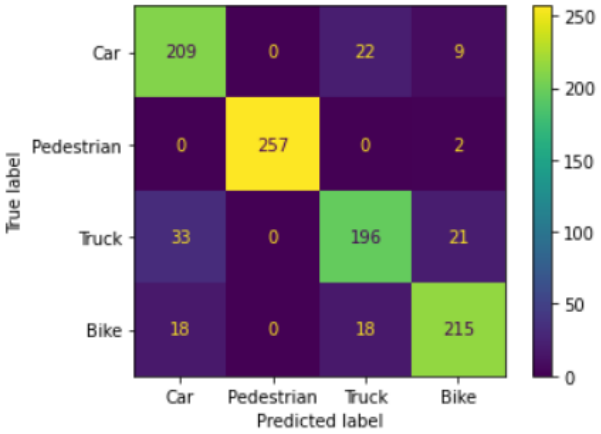

Abbildung 16: DT - 4-klassig - Town03 - 0m
Quelle: [Eigene Darstellung]

In Hinblick auf die Instanzen der Klasse „Car“ lässt sich feststellen, dass 209 von 240 Objekte richtig als solche erkannt wurden, was sich in einer Exaktheit von 87,08 % widerspiegelt. Aus Abbildung 16 lässt sich außerdem entnehmen, dass die Kategorie der „Pedestrians“ mit 257 von 259 korrekt klassifiziert wurde. Dies ergibt eine Erkennungsrate von 99,23 %. Zugleich wurden von den „Bikes“ 215 von 251 richtig vorhergesagt. Die Genauigkeit entspricht demnach einem Wert von 85,66 %. Die Klasse der „Trucks“ wurde mit 78,40 % am geringsten vorausbestimmt. Dies zeichnet sich durch den Umstand ab, dass lediglich 196 von 250 Instanzen richtig bestimmt wurden. Infolgedessen liegt die durchschnittliche Exaktheit aller Kategorien bei 87,59 % und resultiert in der Beobachtung, dass die 4-klassige DT-Klassifikation der Umgebung „Town03“ um 1,31 % bessere Ergebnisse liefert. Daraus kann sich schlussfolgern lassen, dass die Erhöhung der Kartenkomplexität auf die einzelnen Klassifikationen keinen signifikanten Einfluss hat.

5.2.3 Genauigkeit der Positionsdaten

Am Ende der Versuchsreihe des Decision-Tree-Verfahrens wird als letzter Aspekt die Genauigkeit der GNSS-Informationen näher beleuchtet. Bisher wurden für die Berechnung der einzelnen Merkmale ideale Positionsdaten verwendet. In der Praxis sind die GNSS-Daten jedoch mit Abweichungen von bis zu zehn Meter behaftet. Aufgrund dessen werden im weiteren Verlauf des Kapitels die Effekte verschiedener Abweichungen auf die Resultate der DT-Klassifikation untersucht. Da sich die Kartenkomplexität tendenziell nicht auf die

Ergebnisse des Decision-Tree-Verfahrens ausgewirkt hat, wird sich in diesem Abschnitt erneut auf die Daten der komplexeren Karte „Town03“ bezogen. Zu Beginn wird die 3-klassige DT-Klassifikation mit den Kategorien „Car“, „Pedestrian“ und „Bike“ sowie einer maximalen Abweichung von drei Metern analysiert.

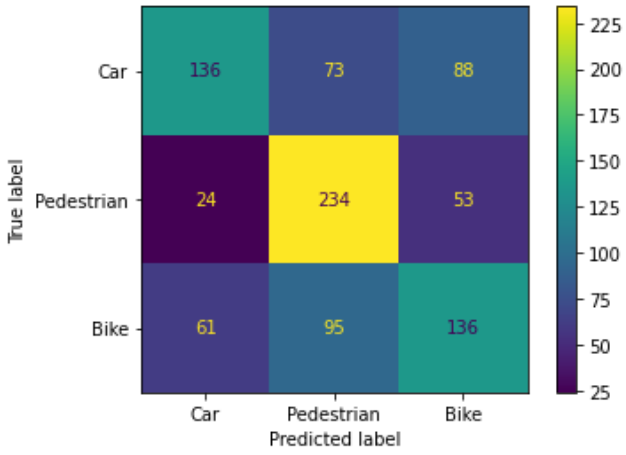

Abbildung 17: DT, 3-klassig - Town03 - 3m
Quelle: [Eigene Darstellung]

Abbildung 17 visualisiert, dass die Resultate der 3-klassigen DT-Klassifikation durch die Verfälschung der GNSS-Informationen sichtlich beeinträchtigt wurden. Von den Objekten des Typs „Car“ wurden 136 von 297 richtig zugeordnet. Dies ergibt eine Genauigkeit von 45,79 % und bedeutet, verglichen mit den Ergebnissen der vorherigen 3-klassigen DT-Klassifikation, eine Einbuße von 37,49 %. Die Exaktheit der „Pedestrians“ verlor parallel dazu 24,76 %, was sich durch den Umstand äußert, dass von 311 „Pedestrians“ nur 234 richtig erkannt wurden. „Bikes“ wurden mit 136 von 292 Testobjekten zu 46,58 % korrekt vorhergesagt. Folglich liegt die durchschnittliche Exaktheit aller Klassen bei 55,87 %. In direkter Gegenüberstellung zur 3-klassigen DT-Klassifikation ohne Abweichung der GNSS-Daten, liegt ein Genauigkeitsverlust von 37,69 % vor. Nachfolgend werden die Effekte der bis zu fünf Meter verfälschten Positionsdaten hinsichtlich einer 3-klassigen DT-Klassifikation untersucht.

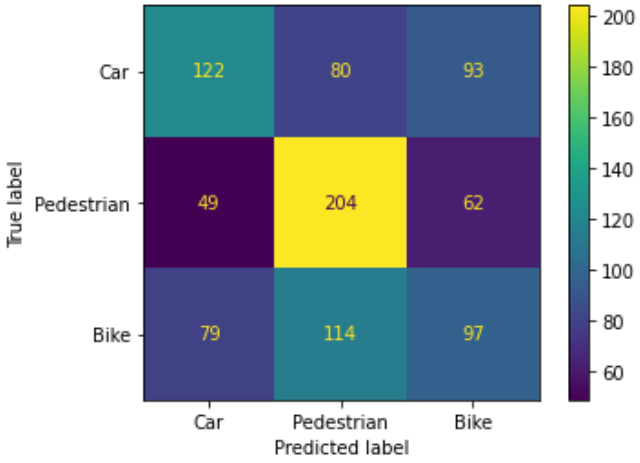

Abbildung 18: DT - 3-klassig - Town03 - 5m

Quelle: [Eigene Darstellung]

Die Ergebnisse des Versuchs zeigen, dass sich die erhöhten Verfälschungen negativ auf die Erkennungsrate auswirken (siehe Abbildung 18). Von den Objekten der „Cars“ wurden 122 von 295 Instanzen richtig erkannt, was somit in einer Genauigkeit von 41,36 % mündet. Die „Pedestrians“ wurden mit einer Erkennungsrate von 64,76 % korrekt zugeordnet, was sich in der Tatsache veräußert, dass 204 von 315 Objekten korrekt vorausgesagt wurden. Die „Bike“-Instanzen wurden mit 97 von 290 Testobjekten zu 33,45 % korrekt erkannt. Zusammengefasst erreicht die 3-klassige DT-Klassifikation eine durchschnittliche Exaktheit von 46,52 % und repräsentiert somit einen Genauigkeitsverlust von 9,35 % im Vergleich zum vorausgegangenen Versuch mit maximal drei Metern Abweichung. Um die Auswirkungen der Ungenauigkeiten besser erforschen zu können, wird im Anschluss der Datensatz dataset03_10m.csv betrachtet und dessen Ergebnisse analysiert.

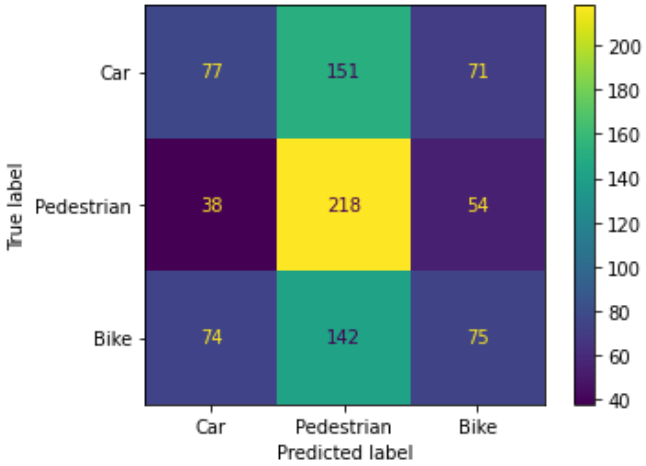

Abbildung 19: DT - 3-klassig - Town03 - 10m
Quelle: [Eigene Darstellung]

Abbildung 19 veräußert die Tatsache, dass die Kategorie „Cars“ mit 77 von 299 korrekt vorhergesagten Instanzen eine Genauigkeit von 25,75 % erzielen kann. Im selben Zuge werden von 310 „Pedestrians“ 218 richtig klassifiziert. Dies entspricht einer Erkennungsrate von 70,32 %. Die „Bikes“ erreichen mit 75 von 291 korrekt erkannten Objekten lediglich eine Exaktheit von 25,77 %. Im Durchschnitt erreichen die Klassen eine Genauigkeit von 40,61 %. Dies resultiert im Gegensatz zur 3-klassigen DT-Klassifikation mit maximal fünf Metern Abweichung in einem Defizit von 5,91 %.

5.3 Support Vector Machine

Das letzte Lernverfahren, auf dessen Implementierung detailliert eingegangen wird, ist das Verfahren der Support Vector Machine. Bei der Umsetzung des Algorithmus wird auf verschiedene, frei verfügbare Phyton-Bibliotheken zurückgegriffen, deren Funktionalitäten in Betracht genutzt werden. Um einen Algorithmus trainieren zu können, muss zunächst der Datensatz geladen werden. Für das Laden und das Manipulieren des Datensatzes wird die Python-Bibliothek pandas verwendet. Hierbei wird der Datensatz, der in der Datei Map_02_0m_error.csv enthalten ist, mit der pandas-Methode read_csv() gelesen und in einer Variablen namens df mit dem Datentyp DataFrame gespeichert. Ausgehend vom Datensatz werden zunächst die Spalten der Tabelle entfernt, die keine Aussagekraft besitzen. In diesem Fall ist es lediglich die erste Spalte ID, die mit der pandas-Methode drop() entfernt wird, da die

Identifikationsnummer bei der Generierung der Objekte zufällig den Objekten zugeordnet wurde und deshalb keine relevanten Informationen verbirgt.

Da die Support Vector Machine nur wenig gelabelte Einträge zum Training benötigt und der Datensatz je nach Szenario mehr als die erforderliche Anzahl an Klassen besitzt, werden die Daten zunächst nach dem Merkmal type gefiltert sowie in seperaten Datensätzen (df_car, df_pedestrian etc.) gespeichert und danach mit der Scitkit-learn-Methode resample() auf die gewünschte Anzahl an Einträgen heruntergeprobt. Danach erfolgt die Zusammenfügung der einzelnen Ergebnisse von resample() mit der pandas-Methode concat() erneut zu einer Variablen namens df_downsampled zusammengefügt.

Nachdem die Datenvorverarbeitung abgeschlossen wurde, können nun die Schritte zum eigentlichen Training beginnen. Für das Training werden zum einen die Daten gesammelt, welche für die Klassifikation benötigt werden (X), und zum anderen die Einträge, welche vorhergesagt werden sollen (y). Dazu wird der vorbereitete Datensatz in Variable X (Klassifikationsdaten) und y (Vorhersagedaten) aufgeteilt. Darauffolgend werden X und y mit der Scitkit-learn-Methode train_test_split() in die Trainingsdatensätze X_train und y_train und die Testdatendätze X_test und y_test aufgeteilt, um anschließend die Werte von X_train und X_test zudem mit der Scikit-learn-Methode scale() zu skalieren. Im weiteren Verlauf wird mit der Scikit-learn-Methode SVC() ein untrainierter Klassifizierer namens svm_clf erzeugt, um im Nachgang durch die Scikit-learn-Methode fit() mit den Trainingsdatensätzen X_train_scaled und y_train trainiert zu werden. Danach kommen die Evaluierungs- und Analysemöglichkeiten, welche im Kapitel der Evaluation nochmals genauer betrachtet werden, zum Einsatz. Diese dienen vor allem dazu, die optimalen Hyperparameter für die Support Vector Machine zu finden und auszugeben. Am Ende der Implementierung kann der optimierte Klassifizierer mit der Scikit-learn-Methode predict() dazu genutzt werden, unbekannte Instanzen nach ihrem Typ einzuordnen. Die dazugehörigen Ergebnisse werden in der Variable pred gespeichert und in der letzten Codezeile dazu genutzt, die Erkennungsrate auszugeben. Diese ergibt sich aus dem Vergleich der Werte aus y_test und pred und wird mit der Scikit-learn-Methode metrics.accuracy_score() berechnet.

5.3.1 Klassenanzahl

Um die Möglichkeiten der Support Vector Machine in Bezug auf unsere Problemstellung abbilden zu können, werden bei der Klassifikation unterschiedliche Klassenanzahlen in

Betracht gezogen und deren Ergebnisse bezüglich der Genauigkeit im Anschluss miteinander verglichen. Die folgenden Versuche und deren Darstellungen beziehen sich auf „Town02“ ohne Abweichungen der GNSS-Daten. Zu den Thematiken der Karte sowie der Genauigkeit der GNSS-Informationen werden in den nachnachfolgenden Kapiteln 5.3.2 und 5.3.3 weitere Betrachtungen angestellt. Im ersten Schritt soll untersucht werden, wie das Lernverfahren performt, wenn lediglich zwei Klassen involviert sind. Hierbei werden einleitend die Straßenverkehrsteilnehmer „Car“, „Pedestrian“ und „Bike“ mit einer hinsichtlich „C“ und „Gamma“ optimierten Support Vector Machine klassifiziert.

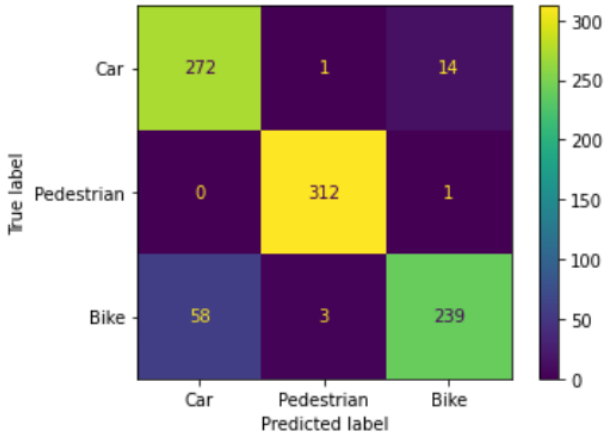

Abbildung 20: SVM - 3-klassig - Town02 - 0m
Quelle: [Eigene Darstellung]

Wie in Abbildung 20 ersichtlich, sind die Erkennungswerte mit drei Typen auf einem hohen Niveau. Bei der Bestimmung der „Cars“ entsprachen 287 Instanzen 272 korrekt Vorhersage, was einer Akkuratesse von 94,77 % entspricht. Von den „Pedestrians“ wurden 312 von insgesamt 313 Objekten richtig vorausgesagt, was bedeutet, dass die Erkennungsrate bei 99,68 % liegt. Die ergänzte Klasse der „Bikes“ kommt auf eine Genauigkeit von 79,66 %, 239 von 300 „Bike“-Instanzen wurden hierbei korrekt zugeordnet. Bemerkenswert ist zudem, dass 19,33 % der „Bike“-Objekte als „Car“ identifiziert wurden. Die Klassifikation der Objekte in „Car“, „Pedestrian“ und „Bike“ wird mit einer Korrektheit von durchschnittlich 91,37 % durchgeführt.

Da die Ergebnisse der Support Vector Machine in Bezug auf unverfälschte Daten immer noch über 91 % liegen, wird die Komplexität nochmals um eine Klasse „Truck“ erweitert. Dies hat

zur Folge, dass die GNSS-Informationen bei der folgenden Klassifikation in Bezug auf vier Kategorien abgebildet werden.

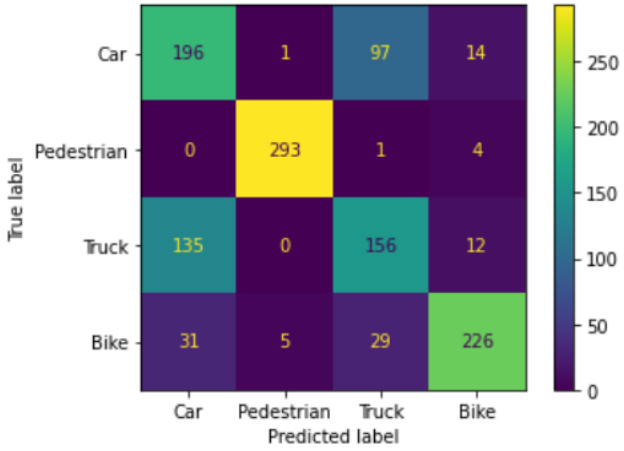

Abbildung 21: SVM - 4-klassig - Town02 - 0m
Quelle: [Eigene Darstellung]

Aus Abbildung 21 geht hervor, dass das Resultat der Testdateneinordnung andauernd auf einem konstant hohen Niveau zu verzeichnen ist. Die Ergebnisse werden im Folgenden chronologisch analysiert. In Bezug auf die „Car“-Objekte wurden 196 von 308 „Cars“ richtig als solche erkannt, was einer Erkennungsrate von 63,63 % entspricht und somit einen Verlust von 31,14 % zur Folge hat. Die Klasse „Pedestrians“ bleibt mit 293 von 298 korrekt vorausgesagten Instanzen relativ konstant und erreicht eine Exaktheit von 98,32 %. Bei der Betrachtung der „Bikes“ kann ein leichtes Abfallen registriert werden. Dies äußert sich in der Tatsache, dass lediglich 226 von 291 richtig erkannt wurden. Dies ergibt eine Genauigkeit von 77,66 % und dementsprechend eine Einbuße von 2,00 % im Vergleich zum vorangegangenen Resultat. Die erweiterte Kategorie der „Trucks“ wird mit einer Korrektheit von 51,49 % prognostiziert. Die durchschnittliche Genauigkeit des Versuchs beläuft sich auf 72,78 %.

5.3.2 Auswirkung der Kartenkomplexität

Im vorherigen Kapitel wurden zunächst die grundlegenden Möglichkeiten der Support Vector Machine in Bezug auf einen unverfälschten Datensatz begutachtet. Dazu kommt, dass die aufgezeichneten Daten beziehungsweise deren Berechnungen von der simplen Karte „Town02“

stammen. Um herauszufinden, welche Auswirkungen die Komplexität der Karte auf die Klassifikationsergebnisse nimmt, werden weiterführend die vielversprechendsten Varianten mit der komplexeren Karte „Town03“ beleuchtet. Die 2- und 5-klassige SVM-Klassifikation wird in die folgenden Ausführungen nicht miteinbezogen. Einleitend wird zunächst die 3-klassige SVM-Klassifikation bezüglich „Town03“ betrachtet.

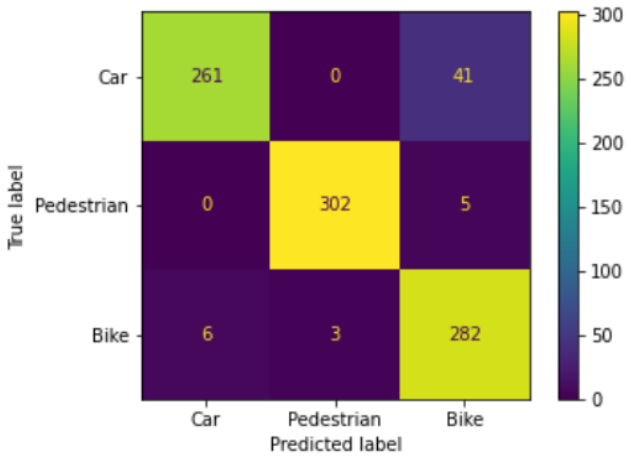

Abbildung 22: SVM - 3-klassig - Town03 - 0m
Quelle: [Eigene Darstellung]

Wie in der Wahrheitsmatrix (siehe Abbildung 22) ersichtlich, sind die Ergebnisse mit der vorherigen 3-klassigen SVM-Klassifizierung vergleichbar. Von den „Pedestrians“ wurden 302 von 307 richtig vorhergesagt, was in einer Genauigkeit von 98,37 % resultiert. Auffallend unter diesen Gegebenheiten ist, dass der Prozentsatz der im vorherigen Versuch nicht richtig klassifizierten „Bikes“ nun aufseiten der „Cars“ zu beobachten ist. „Cars“ werden mit einer Erkennungsrate von 86,42 % korrekt prognostiziert, wobei die „Bikes“-Instanzen mit einer Exaktheit von 96,91 % fehlerfrei erkannt werden. Hinsichtlich der erhöhten Komplexität durch „Town03“ erreicht die Klassifikation durchschnittlich 93,90 %, was folglich eine 2,53-prozentige Verbesserung der Prognose darstellt.

Da ein minimaler Anstieg der Genauigkeit mit zunehmender Klassenanzahl registriert werden konnte, sollen im letzten Teil dieses Unterkapitels die Entwicklungsfolgen der Kartenkomplexität in Hinblick auf eine 4-klassige SVM-Klassifikation beleuchtet werden.

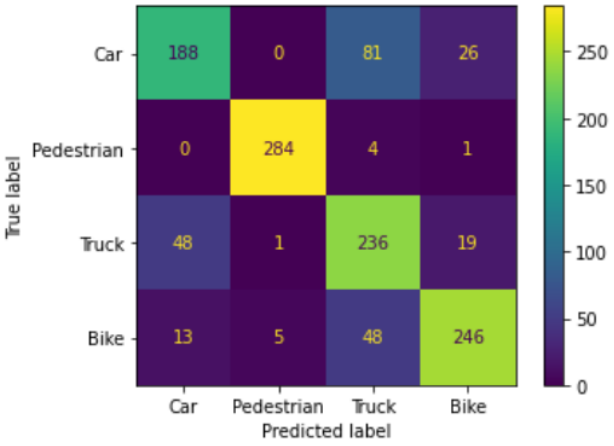

Abbildung 23: SVM - 4-klassig - Town03 - 0m
Quelle: [Eigene Darstellung]

In Bezug auf die „Car“-Instanzen, lässt sich feststellen, dass 188 von 295 „Cars“ korrekt als solche erkannt wurden, was sich in einer Genauigkeit von 63,73 % äußert. Weiterhin lässt sich aus Abbildung 23 erkennen, dass „Pedestrians“ mit 98,27 % und „Bikes“ mit 78,85 % richtig prognostiziert wurden. Die Klasse der „Trucks“ wurde mit 77,63 % am schlechtesten vorhergesagt. Dies veräußert sich in der Tatsache, dass lediglich 236 von 304 Objekten korrekt bestimmt wurden. Die durchschnittliche Erkennungsrate des Versuchs liegt bei 79,62 %. Dies resultiert in der Erkenntnis, dass die 4-klassige SVM-Klassifikation der Karte „Town03“ um 6,84 % bessere Vorhersagen treffen kann. Daraus lässt sich ableiten, dass sich im Verlauf der zunehmenden Klassenanzahl hinsichtlich der erhöhten Kartenkomplexität eine anwachsende Steigerung der Exaktheit von bis zu 6,84 % verzeichnen lässt.

5.3.3 Genauigkeit der Positionsdaten

Der letzte Aspekt, welcher bei der Analyse in Betracht gezogen wird, ist die Genauigkeit der GNSS-Informationen. Bislang wurden die Werte der Merkmale mit ideal simulierten Positionsdaten berechnet. Da die GNSS-Informationen in der Realität jedoch mit einer Unsicherheit von bis zu zehn Meter behaftet sind, werden im Folgenden die Auswirkungen von zufällig verfälschten Positionsdaten hinsichtlich der Ergebnisse der SVM-Klassifikationen beleuchtet. Da im vorherigen Kapitel Karte „Town03“ im Vergleich zu „Town02“ bessere Resultate erzielen konnte, wird sich in diesem Kapitel nur auf erstere bezogen. Einleitend wird

die 3-klassige Support Vector Machine mit den Klassen „Car“, „Pedestrian“ und „Bike“ sowie einer maximalen Abweichung von drei Metern betrachtet.

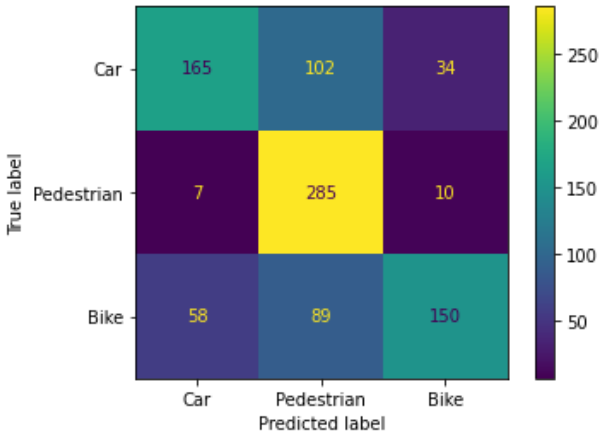

Abbildung 24: SVM - 3-klassig - Town03 - 3m
Quelle: [Eigene Darstellung]

Wie in Abbildung 24 zu sehen, hat die Klassifikation durch die verfälschten Positionsdaten, stark an Genauigkeit verloren. Die Klasse der „Cars“ wurde hierbei lediglich mit einer Exaktheit von 54,82 % richtig zugeordnet. Dies resultiert, verglichen mit den Ergebnissen des vorherigen Kapitels, in einem Defizit von 31,60 %. „Pedestrians“ verloren im gleichen Zuge 4,00 %, was sich in der Tatsache veräußert, dass 285 von 302 Testobjekte korrekt prognostiziert wurden. Parallel dazu, wurden 150 von 297 „Bikes“ korrekt bestimmt, was einer Genauigkeit von 50,50 % entspricht. Im Durchschnitt liegt die Erkennungsrate deshalb bei 66,56 % und bedeutet eine Einbuße von 27,34 % im Vergleich zur 3-klassigen SVM-Klassifikation ohne Abweichungen der Positionsdaten. Im nächsten Schritt werden die Auswirkungen der GNSS-Informationen bezüglich maximal fünf Meter verfälschten Positionsdaten eruiert.

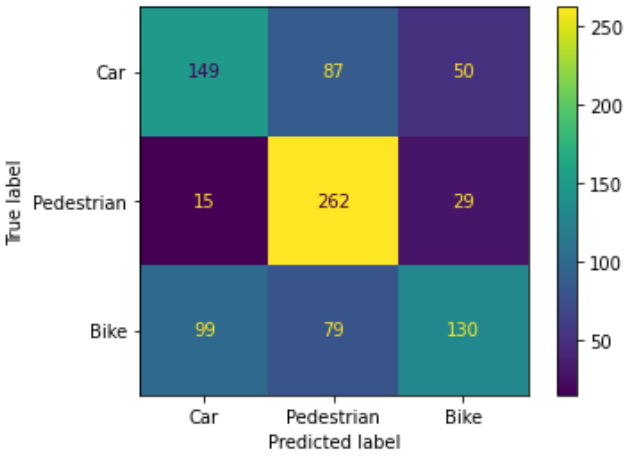

Abbildung 25: SVM - 3-klassig - Town03 - 5m
Quelle: [Eigene Darstellung]

Wie in der Wahrheitsmatrix (siehe Abbildung 25) der 3-klassigen SVM-Klassifikation zu erkennen ist, gab es analog zum vorherigen Szenario einen erneuten Genauigkeitsverlust. Von den „Car“-Instanzen wurden lediglich 149 von 286 richtig vorhergesagt, was einer Erkennungsrate von 52,10 % und parallel einem Defizit von 2,72 % in Gegenüberstellung zum vorangegangenen Ergebnis entspricht. Die „Pedestrians“ wurden mit einer Exaktheit von 85,62 % fehlerfrei bestimmt. Das bedeutet, dass die positive Erkennung um 8,75 % gesunken ist. Gleichzeitig werden 130 von 308 „Bikes“ richtig prognostiziert, was eine Genauigkeit von 42,21 % ergibt. Die mittlere Erkennungsrate aller Klassen liegt bei 59,98 % und repräsentiert eine erneute Einbuße von 6,58 % im Gegensatz zur 3-klassigen SVM-Klassifikaton mit einer Verfälschung von bis zu drei Metern.

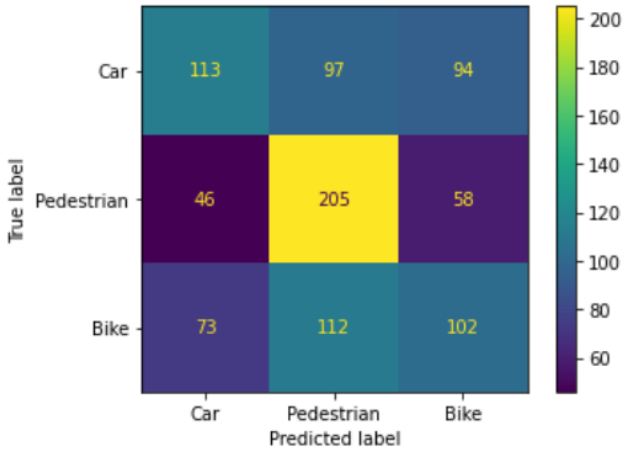

Abbildung 26: SVM - 3-klassig - Town03 - 10m
Quelle: [Eigene Darstellung]

Aus Abbildung 26 ergibt sich, dass „Cars“ mit 113 von 304 richtig vorhergesagten Objekten eine Erkennungsrate von 37,17 % erreichen kann. Parallel dazu werden 205 von 309 „Pedestrian“-Objekte korrekt klassifiziert, was einer Genauigkeit von 66,34 % entspricht. Die Klasse der „Bikes“ erreicht mit 102 von 287 eine Exaktheit von 35,54 %. Zusammen erreichen die Erkennungsraten der Klassen einen durchschnittlichen Wert von 46,35 %. Das entspricht einem Defizit von 47,55 % im Gegensatz zur vorherigen 3-klassigen Einordnung ohne Verfälschungen.

6 Evaluierung

In diesem Abschnitt der Bachelorarbeit wird ein weiterer Schritt zur Untersuchung des Forschungsgegenstands thematisiert. Die Evaluierung ist ein zentraler Bestandteil, um die Güte eines Verfahrens zu bewerten und ein erforderliches Maß an Aussagekraft zu garantieren. Gegenstand der Betrachtungen ist hierbei zunächst die Auswirkung der Klassenquantität auf die Resultate der Klassifikationsmethoden. Da man die grundlegende Information, welche aus den Ergebnissen der unterschiedlichen Methoden gewonnen werden konnte, wiederholt bei den Analysen der einzelnen Verfahren feststellen konnte, werden im Anschluss lediglich Resultate des KNN-Verfahrens berücksichtigt. Zudem konnten sich im Vergleich der Kapitel 5.1.1, 5.2.1 und 5.3.1 ähnliche Muster in den Erkennungsraten verzeichnen lassen.

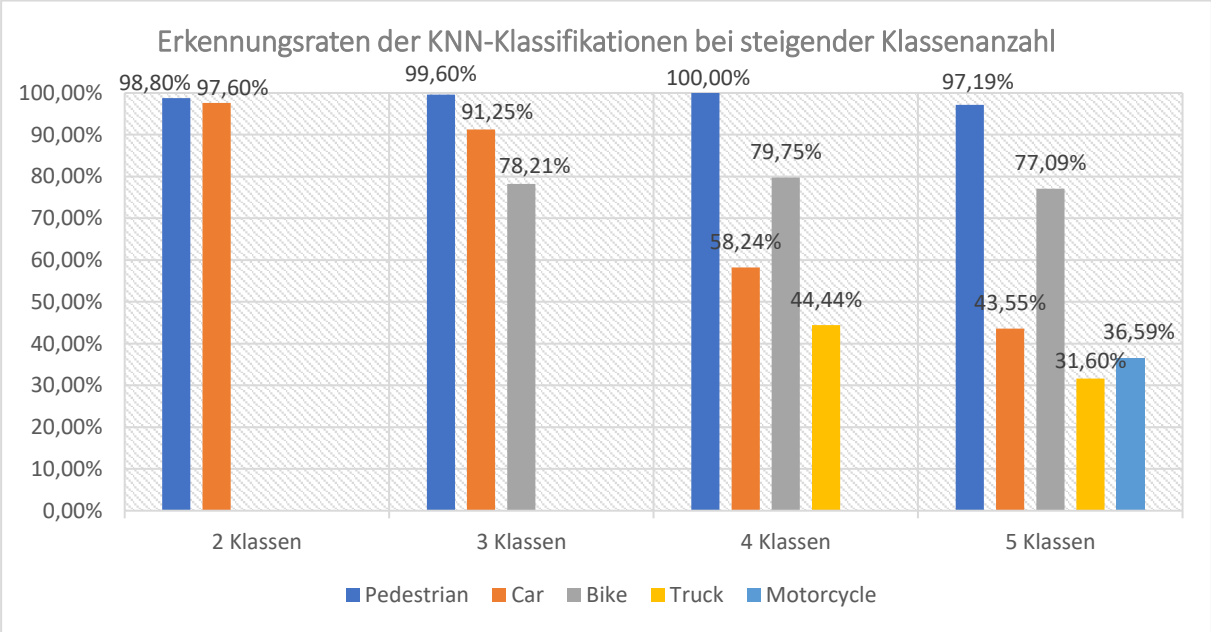

Abbildung 27: Erkennungsraten der KNN bei steigender Klassenanzahl

Quelle: [Eigene Darstellung]

In den Versuchen des Kapitels 5.1.1 wurden die Auswirkungen der Klassenanzahl auf die positiven Erkennungsraten des K-Nearest-Neighbor-Verfahrens analysiert und anschließend in Abbildung 27 visualisiert. Während der Analyse stellte sich heraus, dass die Klassen „Car“ und „Pedestrian“ mit einer durchschnittlichen Genauigkeit von 98,20 % richtig erkannt wurden. Daraus kann der Schluss gezogen werden, dass sich die nichtmotorisierten Objekte gut voneinander unterscheiden lassen. Die Kategorisierung von „Car“, „Pedestrian“ und „Bike“

erzielte im Mittel eine Exaktheit von 89,69 %. Daraus kann sich schlussfolgern lassen, dass die Objekte der 3-klassigen Variante weiterhin auf einem hohen Niveau eingeordnet werden konnten und sich die Instanzen der Kategorie „Bike“ gut von den bisherigen Klassen differenzieren lassen. Im 4-klassigen Versuch zeigte sich, dass die Klassifikation von zwei motorisierten Verkehrsteilnehmern zu vermehrt auftretenden Falschklassifikation geführt hat. Dies kann man dem Umstand entnehmen, dass die Vorhersage der „Car“-Objekte mit 58,24 % Exaktheit einen Genauigkeitsverlust von 33,01 % bedeuten, wobei die Differenzierung von „Pedestrians“ und „Bikes“ konstant geblieben ist. Diese Schlussfolgerung kann sich erneut durch die 5-klassige Klassifizierung bestätigen lassen. Die Resultate in Bezug auf die motorisierten Verkehrsteilnehmer nahmen erneut ab, wobei die nichtmotorisierten Typen nahezu mit denselben Werten vorhergesagt werden konnten. Durch den Unterschied zwischen der 3- und 4-klassigen KNN-Klassifikation die Konsequenz gewonnen werden, dass die Differenzierung von motorisierten und nichtmotorisierten Typen am besten funktioniert, wenn dabei maximal ein motorisch angetriebener Verkehrsteilnehmer eingesetzt wird. Die Kombinationen mit mehreren motorisierten Klassen führt tendenziell zu schlechteren Klassifikationsergebnissen. Daraus gewinnt man die Information, dass sich die Merkmale motorisierter Klassen ähnlich sind.

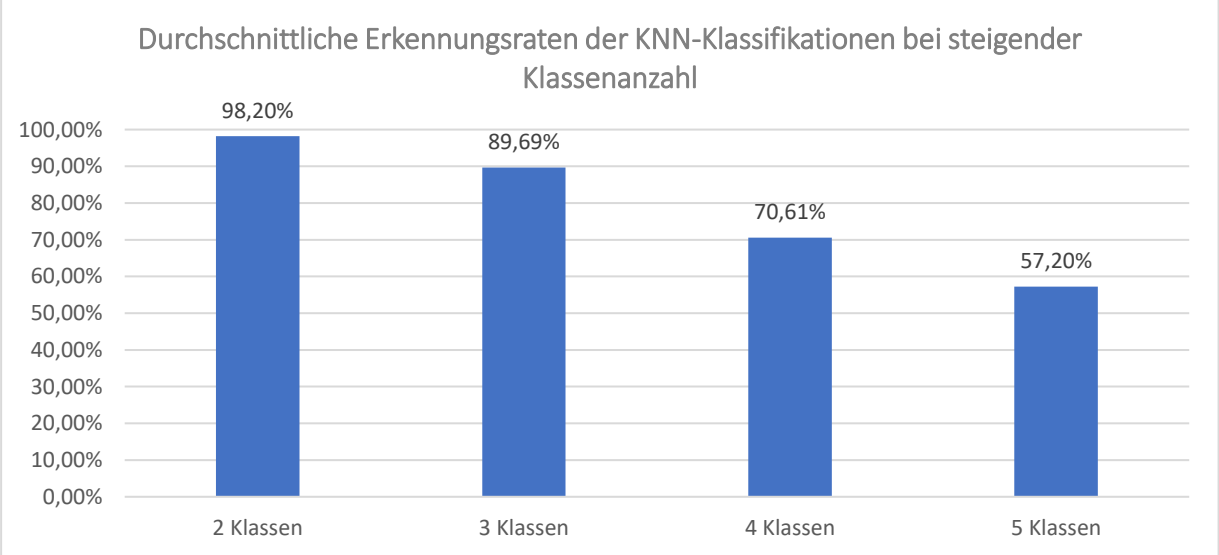

Abbildung 28: Durchschnittliche Erkennungsraten der KNN bei steigender Klassenanzahl
Quelle: [Eigene Darstellung]

Die Versuche des Kapitels 5.1.1 beschreiben den Fakt, dass die durchschnittliche Erkennungsrate der Klassifizierung mit zunehmender Klassenanzahl abnimmt (siehe Abbildung 28). Nachdem die Auswirkungen bezüglich der Klassenquantität eruiert wurden, wird anschließend der Aspekt der Kartenkomplexität untersucht.

Die folgenden Ausführungen basieren auf den Resultaten der Kapitel 5.1.2, 5.2.2 und 5.3.2, und thematisieren die Effekte einer komplexeren Karte auf die Genauigkeit der Vorhersagen. In Anbetracht der in Kapitel 5.1.2 ersichtlichen Ergebnisse, hat sich herausgestellt, dass sich die erhöhte Komplexität minimal auf positive Erkennungsraten des K-Nearest-Neighbor-Verfahren auswirkt. Dieser Schluss kann sich aus der Gegenüberstellung der unterschiedlichen Ergebnisse gewinnen lassen, da das Intervall des Verlusts/des Gewinns zwischen -2,17 % und +3,54 % liegt. Die Auswirkungen auf die Resultate des Decision Trees (Kapitel 5.2.2) sind mit ähnlichen Tendenzen behaftet. Das Intervall der Veränderung beträgt hierbei lediglich -1,52 % und +1,31 %. In Bezug auf die Support Vector Machine, konnten in Kapitel 5.3.2 jedoch signifikante Einflüsse der komplexeren Umgebung registriert werden. Hierbei führte die Verwendung von „Town03“ im Vergleich zu den 3-klassigen Verfahren zu einem Genauigkeitsgewinn von 2,53 %. Eine erneute Steigerung konnte in der Gegenüberstellung der 4-klassigen SVM-Klassifikationen verzeichnet werden. Dabei bewirkte der Einsatz der Umgebung „Town03“ einen Anstieg von 6,84 % bewirken. Daraus kann geschlussfolgert werden, dass sich die erhöhte Kartenkomplexität positiv auf die Erkennungsraten der Support Vector Machine auswirkt.

Der nächste Faktor, welcher folgend behandelt wird, ist die Auswirkung der Genauigkeit der Positionsdaten in Hinblick auf die Erkennungsrate der verschiedenen Verkehrstypen.

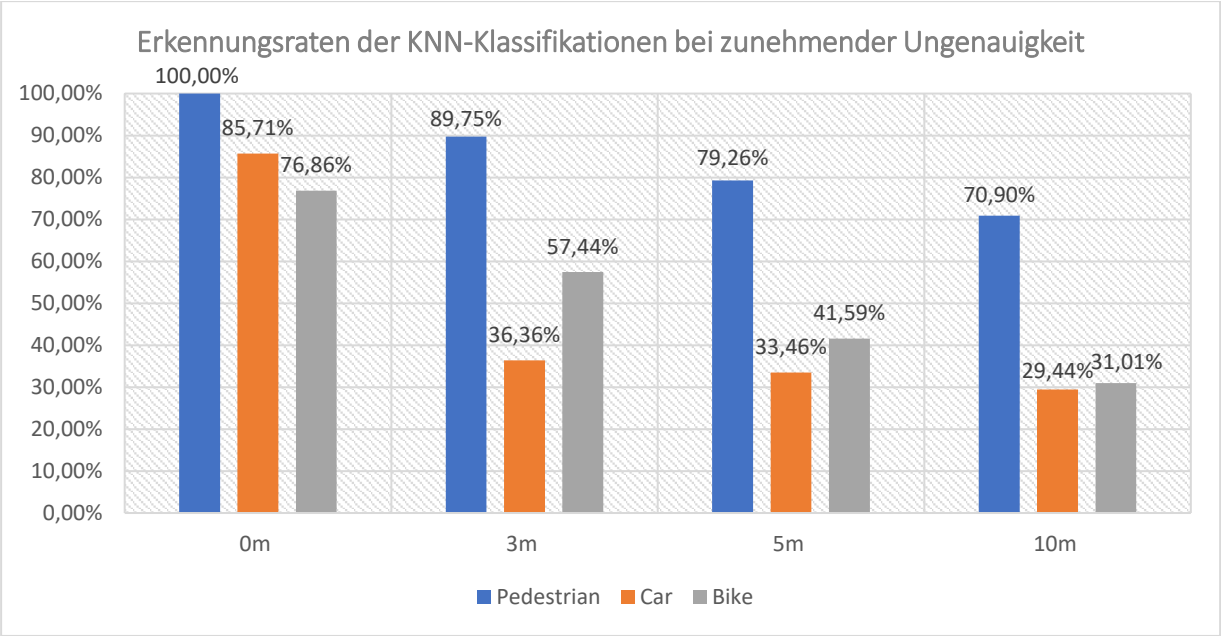

Abbildung 29: Erkennungsraten der KNN bei zunehmender Ungenauigkeit
Quelle: [Eigene Darstellung]

Aus den Ergebnissen der Kapitel 5.1.3, 5.2.3 und 5.3.3 ist ersichtlich, dass die Objekte „Car“, „Pedestrian“ und „Bike“ bei zunehmender Verfälschung am besten mit der Support Vector Machine bestimmt werden konnten. Das SVM-Verfahren konnte im Gegensatz zur KNN- und DT-Klassifikation, alle Verkehrsteilnehmer mit durchschnittlich 6,64 % besser korrekt kategorisieren. Das veräußert sich aus dem Umstand, der aus den Abbildungen 29, 33 und 34 gewonnen werden kann. Im direkten Vergleich des K-Nearest-Neighbor- zum Decision-Tree-Verfahren ergaben die Resultate, dass die KNN-Versuche des Kapitels 5.1.3 im Durchschnitt 1,84 % bessere Erkennungsraten erzielten, wobei lediglich die Klasse „Car“ vom Decision Tree genauer vorhergesagt werden konnte. Im Mittel wurden die Testobjekte der Kategorie „Car“ zu 2,81 % exakter klassifiziert. Eine weitere Besonderheit, welche durch die Versuchsreihen ersichtlich wird, ist die Tatsache, dass „Pedestrians“ selbst bei einer maximalen Abweichung von zehn Metern mit 70,90 % richtig prognostiziert werden konnten. Daraus kann wiederum geschlussfolgert werden, dass sich die Merkmale der „Pedestrians“ gut von Merkmalen der anderen Klassen differenzieren lassen.

In Bezug auf die direkte Gegenüberstellung der Verfahren lässt sich infolgedessen der Schluss ableiten, dass die Support Vector Machine tendenziell besser mit komplexeren Bedingungen umgehen kann, jedoch selbst bei geringfügigen Abweichungen stark an Genauigkeit verliert.

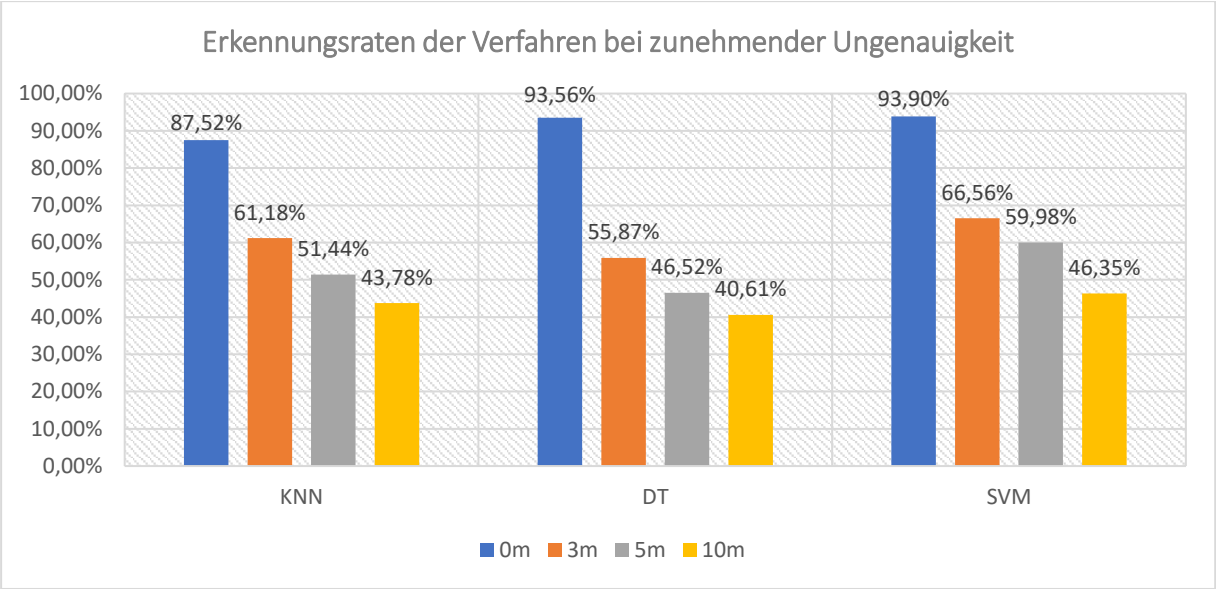

Abbildung 30: Erkennungsraten der Verfahren bei zunehmender Ungenauigkeit
Quelle: [Eigene Darstellung]

Eine weitere Schlussfolgerung, welche aus den Kapiteln 5.1.3, 5.2.3 und 5.3.3 und der Abbildung 30 entnommen werden kann ist, dass der Lazy Learner K-Nearest-Neighbor, welcher konzeptionell simpler gestaltet ist, in der Klassifizierung der Kategorien praktikablere Resultate erzielen konnte.

Nachdem die wichtigsten Schlüsse aufgezeigt wurden, werden abschließend die Methodiken beleuchtet, welche zur Bewertung beziehungsweise Optimierung der Ergebnisse geführt haben. Um die idealen Parameter der einzelnen Klassifikationsarten herauszufinden, wurden die zur Verfügung stehenden Hilfsmittel der Software-Bibliothek Scikit-learn verwendet. Primär lag der Fokus auf der Verwendung der Scikit-learn-Methode GridSearchCV() konzentriert, welche eine Kreuzvalidierung mit unterschiedlichen Kombinationen ausführt. Um mit dieser Technik zu den optimalen Parametern gelangen zu können, musste zunächst, je Verfahren, eine Liste mit den Werten angefertigt werden, welche während der Ausführung der Methode erprobt werden sollten. In Bezug auf das KNN-Verfahren wurden die Werte für k in einem Intervall von eins bis 30 in der Liste gespeichert. In Hinblick auf die Optimierung des Decision Trees, wurden die Werte von 1 bis 25 für die Tiefe des Baums festgelegt, wohingegen in der Optimierungsliste der Support Vector Machine verschiedene Werte für die Hyperparameter C und Gamma festgehalten wurden. Nach Definition der Listen, konnten diese auf die Scikit-learn-Methode GridSearchCV() angewendet werden. Dabei wurden die verschiedenen Kombinationen der Parameter nacheinander kreuzvalidiert und deren Resultate abgespeichert,

damit sie im Nachhinein abgerufen werden konnten. Zusätzlich trug die durchgeführte Kreuzvalidierung dazu bei, die Gefahr einer möglichen Überanpassung zu relativieren. Die idealen Werte werden im Nachgang dazu verwendet den Klassifizierer entsprechend zu trainieren.

Um das Potenzial der Klassifikationsmethoden auswerten/darstellen zu können, wird die Scikit-learn-Methode plot_confusion_matrix() zum Einsatz gebracht. Diese generiert mit dem jeweiligen Klassifizierer und unter Verwendung der Testdaten X_test und y_test eine grafische Darstellung der Ergebnisse.

7 Zusammenfassung

Die Bachelorarbeit startete mit der detaillierten Erfassung der Problemstellung und deren kontextuellen Einordnung. Der dabei gewonnene Ansatz äußerte sich durch die Frage, ob der Typ eines bewegten Mobilfunkteilnehmers allein aufgrund seiner Positionsdaten klassifiziert werden kann. Um den Forschungsgegenstand der Frage zu ergründen sowie die damit einhergehenden Anforderungen zu erfüllen, wurden im Rahmen der Bearbeitung zunächst verschiedene Datensätze mit unterschiedlichen Eigenschaften angefertigt. Zur Generierung der Positionsdaten, kam die Simulationssoftware CARLA zum Einsatz. Es folgte die Implementierung von den Klassifikationsmethoden K-Nearest-Neighbor, Decision Tree und Support Vector Machine und Anwendung auf die jeweiligen Datensätze. Die einzelnen Methoden wurden während der Versuchsreihe hinsichtlich differenter Aspekte betrachtet und im Nachgang ausgewertet. Die Analyse der Ergebnisse erbrachte die Erkenntnis, dass die Klassifizierung von motorisierten und nichtmotorisierten Klassen bei allen Verfahren mit einer hohen Genauigkeit durchgeführt werden konnte. Fehlklassifikationen sind vermehrt dann aufgetreten, wenn mehrere motorisierte Kategorien differenziert werden sollten. In Hinblick auf die Auswirkungen der Umgebungskomplexität der Simulation konnte in den K-Nearest-Neighbor- und Decision-Tree-Versuchen keine aussagekräftige Tendenz erkannt werden. Lediglich bei der Auswertung der SVM-Resultate ließ sich während der Verwendung einer komplexeren Karte ein positiver Trend in Bezug auf Genauigkeiten verzeichnen. Bei der Untersuchung der Auswirkungen von zunehmend verfälschten Positionsdaten kam es zur Beobachtung eines ähnlichen Ergebnisses. Das SVM-Verfahren konnte erneut die besten Resultate erzielen und zeichnete sich bei zunehmender Komplexität minimal von den anderen Klassifikationsmethoden ab. Ein weiterer Schluss, der sich während der Analyse feststellen ließ, ist der Umstand, dass „Pedestrians“ von allen Kategorien am genausten klassifiziert werden konnte. Das bedeutet, dass sich die Merkmale der „Pedestrians“ auch bei zunehmenden Verfälschungen am besten von den Merkmalen der anderen Klassen differenzieren lassen können.

Aufgrund dieser Ergebnisse kann eine abschließende Antwort auf die zentrale Fragestellung abgeleitet werden. Die Klassifizierungsmethoden waren, je nach Szenario, in der Lage die bewegten Mobilfunkteilnehmer allein durch die Verwendung von Positionsdaten vorherzusagen.

8 Ausblick

Die Auswertung der vorherigen Kapitel resultiert in dem Sachverhalt, dass die Klassifikation bewegter Mobilfunkteilnehmer zur erweiterten Umfeldwahrnehmung autonomer Fahrzeuge prinzipiell möglich ist. Allerdings haben die Versuche auch gezeigt, dass motorisierte Kategorien weniger voneinander differenziert werden können und die Ergebnisse der Klassifizierung mit zunehmender Ungenauigkeit der GNSS-Informationen enorm an Exaktheit verlieren. Das hat zur Folge, dass ein gewinnbringender Einsatz der Klassifikationen in der Realität lediglich unter dem Aspekt von nahezu idealen Positionsdaten in Betracht gezogen werden sollte. In Hinblick auf den aktuellen Forschungsstand könnte durch die Kombination der behandelten Klassifikationen und den bereits vorhandenen Wahrnehmungs- und Klassifizierungsmöglichkeiten eines autonomen Fahrzeugs eine unterstützende Wirkung erzielt werden. Im Bezug auf Restriktionen und Grenzen ergeben sich allerdings Sachverhalte, die mit dem Umfeld der Realität in Konflikt stehen. In der Praxis kann man nach aktuellem Stand nicht davon ausgehen, dass jeder Verkehrsteilnehmer ein Mobilfunknutzer ist. Ein Fußgänger ohne elektronische Geräte könnte somit nicht von den Algorithmen erkannt und müsste infolgedessen durch andere Sensoren wahrgenommen und klassifiziert werden. Zudem kann sich der Typ des Objektes in der Realität ändern. Sobald ein klassifizierter Autofahrer aus seinem Vehikel aussteigen würde, müsste eine erneute Kategorisierung und Einordnung erfolgen. In Anbetracht eines weiterführenden Ansatzes, könnten weitere Klassifikationsmethoden untersucht werden, welche den Bereichen der unüberwachten und bestärkenden Lernverfahren zugehörig sind. Außerdem könnte man den grundlegenden Forschungsansatz um die Fragestellung „Ist das klassifizierte Objekt für meinen Fahrweg relevant?“ erweitern, wobei der Ansatz dadurch wesentlich komplexer wird und mit überwachten Lernverfahren nicht mehr zu realisieren wäre. Da sich der Bereich der künstlichen Intelligenz jedoch rasant entwickelt und zunehmend erforscht wird, könnten zukünftige Methoden bald in der Lage sein, solche Problemstellungen automatisiert zu lösen.

„Es ist wahrscheinlich, dass die Maschinen vor dem Ende des Jahrhunderts klüger sein werden als wir – nicht nur bei Schach- oder Quizfragen, sondern bei fast allem, von Mathematik und Technik bis hin zu den Naturwissenschaften und der Medizin.“ Gary Marcus zitiert nach [Katzlberger 2020].

Quellenverzeichnis

[Andelfinger+2015] Andelfinger, V.; Hänisch, T.: Internet der Dinge – Technik, Trends und Geschäftsmodelle, Springer Gabler, Wiesbaden, 2015.

[CARLA a] CARLA, https://carla.readthedocs.io/en/latest/img/Town02.jpg (Zugriff: 15.03.2022).

[CARLA b] CARLA, https://carla.readthedocs.io/en/latest/img/Town03.jpg (Zugriff: 15.03.2022).

[Dosovitskiy+2017] Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V.: CARLA: An Open Urban Driving Simulator, in: Proceedings of Machine Learning Research, Jg. 78, Nr. 1, S. 1-16, 2017.

[Frochte 2021] Frochte, J.: Maschinelles Lernen – Grundlagen und Algorithmen in Phyton, 3. Aufl., Carl Hanser Verlag, München, 2021.

[Johanning+2015] Johanning, V.; Mildner, R.: Car IT kompakt – Das Auto der Zukunft – Vernetzt und autonom Fahren, Springer Vieweg, Wiesbaden, 2015.

[Katzlberger 2020] Katzlberger, M.: Die besten Zitate über Künstliche Intelligenz, https://katzlberger.ai/2020/03/13/die-besten-zitate-ueber-kuenstliche-intelligenz-in-deutscher-sprache/ (Zugriff: 15.03.2022).

[Liebl 1995] Liebl, F.: Simulation – Problemorientierte Einführung, 2. Aufl., Oldenbourg Verlag, München, 1995.

[Maurer+2015] Maurer, M.; Gerdes, C.; Lenz, B.; Winner, H.: Autonomes Fahren – Technische, rechtliche und gesellschaftliche Aspekte, Springer Vieweg, Wiesbaden, 2015.

[McKinney 2012] McKinney, W.: Python for Data Analysis, O'Reilly Media, Sebastopol, 2012.

[Pedregosa+2011] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E.: Scikit-learn: Machine Learning in Python, in: Journal of Machine Learning Research, Jg. 12, Nr. 85, S. 2825-2830, 2011.

[Raschka+2021] Raschka, S.; Mirjalili, V.: Machine Learning mit Phyton und Keras, TensorFlow 2 und Scikit-learn – Das umfassende Praxis-Handbuch für Data Science, Deep Learning und Predictive Analytics, 3. Aufl., mitp Verlag, Frechen, 2021.

[Scheffels+2022] Scheffels, G.; Gelowicz, S.: Autonomes Fahren – Definition, Level & Grundlagen, https://www.automobil-industrie.vogel.de/autonomes-fahren-definition-leve%20grundlagen-a-786184/ (Zugriff: 15.03.2022).

[Schönbrodt 2019] Schönbrodt, S.: Maschinelle Lernmethoden für Klassifizierungsprobleme – Perspektiven für die mathematische Modellierung mit Schülerinnen und Schülern, Springer Fachmedien Wiesbaden GmbH, Wiesbaden, 2019.

[Schüttler 2014] Schüttler, T.: Satellitennavigation – Wie sie funktioniert und wie sie unseren Alltag beeinflusst, Springer Verlag, Berlin, 2014.

[Terré 2019] Terré, H.: Interpreting Deep Learning for cell differentiation – Supervised and Unsupervised models viewed through the lens of information and perturbation theory, Cambridge, 2019.

[Tille 2016] Tille, T.: Automobil-Sensorik – Ausgewählte Sensorprinzipien und deren automobile Anwendung, Springer, Vieweg, Wiesbaden, 2016.

[Tille 2018] Tille, T.: Automobil-Sensorik 2 – Systeme, Technologien und Applikationen, Springer Vieweg, Wiesbaden, 2018.

Anhang A1. Straßenlayout der Karten aus der Vogelperspektive

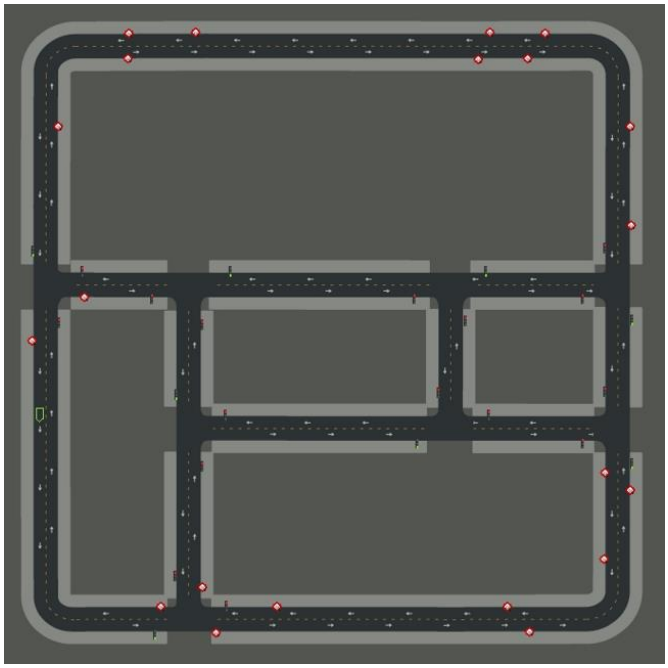

Abbildung 31: Straßenlayout von "Town02"
Quelle: [CARLA a]

Abbildung 32: Straßenlayout von "Town03"
Quelle: [CARLA b]

Anhang A2. Resultate der Klassifikationen hinsichtlich zunehmender Ungenauigkeiten

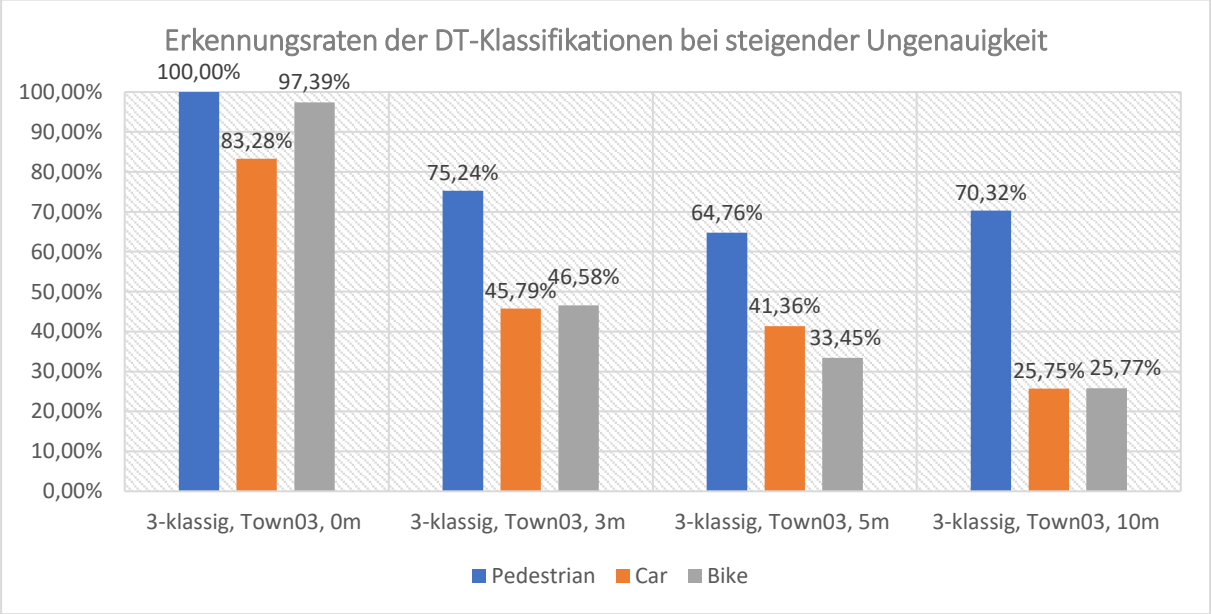

Abbildung 33: Erkennungsraten der DT-Klassifikationen bei steigender Ungenauigkeit

Quelle: [Eigene Darstellung]

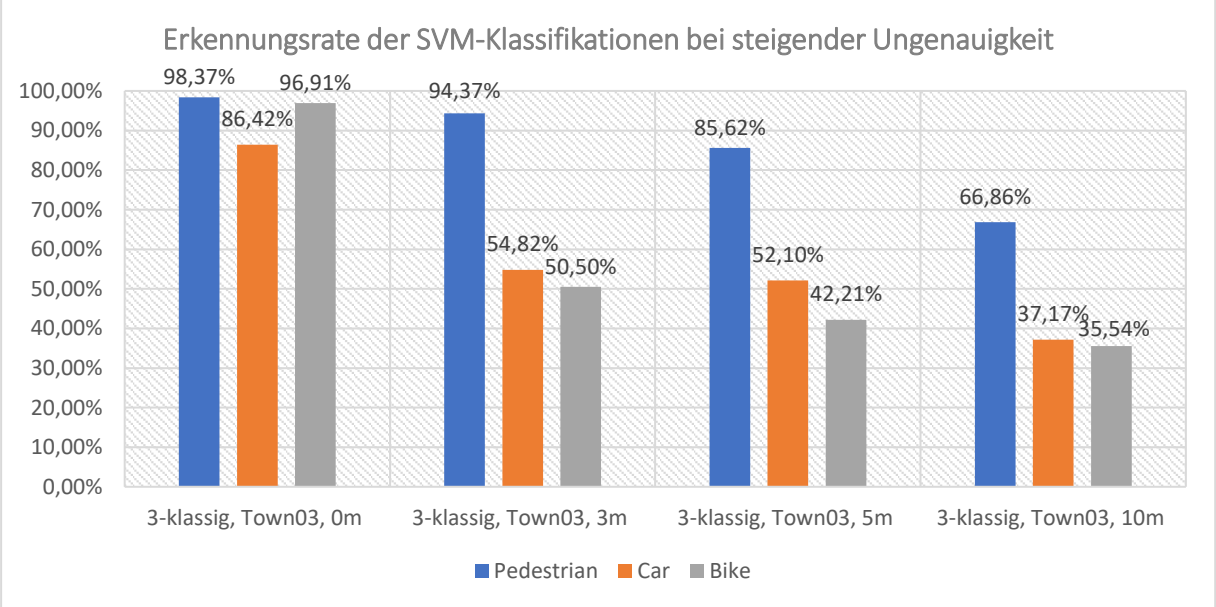

Abbildung 34: Erkennungsrate der SVM-Klassifikationen bei steigender Ungenauigkeit

Quelle: [Eigene Darstellung]

Ehrenwörtliche Erklärung

Ich versichere hiermit, dass ich meine **Bachelorarbeit** mit dem Titel

_____

_____

selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie nicht an anderer Stelle als Prüfungsarbeit vorgelegt habe.

_____
Ort

_____ _____
Datum Unterschrift