technische universität
dortmund

Bachlor Arbeit

Machine Learning basierte dezentrale Low-Power Indoor Positionierung für Warenlager

Sabrina Schawohl
Dezember 2018

Gutachter:
Prof. Dr. Katharina Morik
Dr. Nico Piatkowski

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Künstliche Intelligenz (LS-8)
https://www-ai.cs.uni-dortmund.de/index.html

Inhaltsverzeichnis

Kapitel 1

Einleitung

Eine der zentralen Aufgaben in der Logistik ist die Positionsbestimmung einer Lagereinheit innerhalb eines Lagers. Dies ist wichtig, da wenn man eine Lagereinheit zum Beispiel auslagern möchte, man erst einmal wissen muss wo sie steht.

Durch die Entwicklung zur Industrie 4.0 wird die Verwendung von Internet of Things (IoT) Geräten in der Logistik immer bedeutender. Ein Grund dafür ist, dass man IoT Geräte dezentral organisieren kann, was für eine bessere Skalierung von Rechenleistung sorgt [6]. Auch kann bei einzelnen Ausfällen von Rechnern der Betrieb in einem Lager aufrechterhalten werden.

Zu wissen wo eine Lagereinheit in einem Lager ist, ist extrem wichtig in der Logistik, vor allem da es in der Logistik darum geht Objekte zu bewegen. Daher müssen die Systeme, zum Verwalten von Positionen von Objekten, extrem zuverlässig sein. Durch das IoT sollen kostenarme, automatisierte Verfahren, zum auffinden von Lagereinheiten entwickelt werden.

Eine Möglichkeit zum Nachhalten von Lagerpositionen sind Datenbanken, in denen bei jeder Bewegen einer Lagereinheit, ihre Position neu abgespeichert wird. Da das Abspeichern meist durch Scanner, die von Menschen bedient werde, funktioniert, ist dieses System nicht immer zuverlässig.

Zu beachten ist dabei, dass die Positionierung unter Ressourcenbeschränkung erfolgt. Die benutze Hardware in den PhyNodes, sowie die zur Verfügung stehende Energie pro Aktion ist beschränkt[6].

Diese Arbeit versucht daher Lagereinheiten mithilfe von Sensordaten und maschinellem Lernen zu lokalisieren. Dafür werden die Umgebungsdaten wie Licht, Temperatur, Ausrichtung und die empfangene Signalstärke von drei Empfängern von verschiedenen Modellen gelernt, und die Ergebnisse miteinander verglichen.

Im folgenden Kapitel wird auf bisherige Ansätze für Indoor-Positionierung eingegangen. Anschließend wird eine Einführung in maschinelles Lernen, vor allem in die verwendeten Methoden, gegeben. In Kapitel 4 wird auf die verwendete Hardware zur Datensammlung, die Daten und die Vorverarbeitung eingegangen. Danach wird der Ansatz der Arbeit erklärt. Kapitel 6 behandelt dann die durchgeführten Experimente, die verwendeten Parameter und ihre Ergebnisse. Zum Schluss wird die Arbeit zusammengefasst und ein Ausblick gestellt.

Kapitel 2

Verwandte Arbeiten

In der Arbeit von Masoudinejad et al. [6] wird mit Decision Trees, Random Forest Classifiers, k-Nearest-Neighbors, Support Vector Machines und Gaussian Naive Bayes verschiedene Modelle (siehe Kapitel 3) erstellt und verglichen, welches die besten Ergebnisse liefert. Die Daten für diese Modelle wurden in einem Versuchsaufbau des Logistik Labors im Fraunhofer IML erzeugt und dann auf verschiedene Arten vorverarbeitet. Die Modelle wurden mithilfe der Scikit-Learn Bibliothek, einer Phython Bibliothek für maschinelles Lernen, erstellt.

In der Vorverarbeitung wurden zunächst alle Beispiele, in denen eines der Attribute um mehr als die dreifache Standartabweichung abweicht, aussortiert. Für einen zweiten Datensatz wurden zusätzlich die Beschleunigungen in x, y und z-Richtung entfernt. Als drittes wurde die Datensatz erstellt wo zusätzlich zu den direkten Daten, ohne Beschleunigung, das Verhältnis zu den zwei Referenzboard, das sind zwei PhyNodes (siehe Kaptiel 4), die immer an der selben stelle stehen, gelernt wurde. Dafür wurden für die einzelnen Attribute, die Daten von den Referenzboard genommen, die zeitlich am nächsten zu der Messung der PhyNode waren und ein Quotient der PhyNode Daten zu den Referenzboardaten gebildet.

Bei den Daten ist eine Zuordnung auf Tag und Lichtpegel der Testreihe möglich. Die Daten eines ganzen Tages wurden zum überprüfen der Modelle verwendet. Die Daten vom anderen Tag wurden anhand des Lichtsegels so aufgeteilt, dass in den Test und Trainingsdaten jeder Lichtpegel in beiden oder keinem der Sets vorhanden war.

Um die besten Parameter für die einzelnen Modelle zu finden wurde Grid Search benutzt und die Ergebnisse mit Kreuzvalidierung (Kapitel 3) gesichert. Es wurde eine zehnfache Kreuzvalidierung verwendet.

Das Decision Tree Modell wurden mit einer optimierten Version des CART Algorithmus [1] erstellt. Jeder Knoten im Baum entspricht einer Regel zum Trennen der Daten. Es wurden solange Regeln zusammengestellt, bis sich die Daten ausreichend voneinander trennen ließen. Die Größe des Modells hängt davon ab, wie viele Knoten, Regeln, der Baum hat.

Das Random Forest Modell besteht aus mehreren Entscheidungsbäumen. Die Position einer PhyNode wird über den Majority Vote (Kapitel 6.1) der Entscheidungsbäume bestimmt, das heißt das die Position, die am meisten von den verschiedenen Bäumen vorhergesagt wird, als Postion genommen wird.

Die Modellgröße bei k-NN ist $n * m$ für n Beispiele und m Attribute, da zur Berechnung der nächsten Nachbarn, alle Daten vorhanden sein müssen. Das optimale Modell funktionierte mit $k = 5$ und der Cityblock-Metrik (Manhattan Distanz) als Distanzfunktion.

Bei der SVM hat der radial basis function (RBF) Kernel mit $c = 1E2$ und $\gamma = 1E\text{-}2$ das beste Ergebnis geliefert (Kapitel 6.1).

Für naive Bayes-Klassifizierer muss der Mittelwert μ und die Varianz σ^2 für jedes Attribut x_i, für alle Klassen g und die a-priori Wahrscheinlichkeiten für alle Klassen gespeichert werden. Wenn m die Anzahl der Attribute und $|g|$ die Anzahl an Klassen ist, ergibt sich eine gesamte Modellgröße von $m + 2 * (m * |g|)$ zum Speichern von Naive Bayes [6].

Die Arbeit von Liu et al. [4] gibt einen Überblick über verschiedene kabellose Indoor Positionierungslösungen. Es werden drei grundlegende Methoden beschrieben, auf denen Indoor Positionierungssysteme aufbauen können. Die Methoden sind Triangulation, Szenen Analyse und Nachbarschaftsbestimmung (Proximity). Auch werden unterschiedlichen Funktechnologien vorgestellt, mit der man kabellose Indoor-Positionsbestimmung umsetzen kann, sowie verschiedene Bewertungsparameter angegeben, an denen verschiedene existierende Lösungen für die Indoor-Positionsbestimmung untersucht werden.

Bei der Triangulation werden die geometrischen Eigenschaften von Dreiecken ausgenutzt. Es wird zwischen Lateration und Angulation unterschieden. Bei der Lateration wird die Position über die Distanz Messung zu mehreren Punkten bestimmt. Dabei wird jedoch nicht die empfangene Signalstärke, sondern die Zeit, die das Signal zum zurücklegen braucht, gemessen.

Bei der Angulation wird die Position eines Objektes über die Winkel der Sendestationen bestimmt.

In 2-D senden zwei und in 3-D mindestens drei Sender an das zu ortende Objekt. Die Position kann dann relativ zu den Sendern bestimmt werden. Zwischen den Messstationen wird keine Zeitsynchronisation benötigt, allerdings muss die Winkelbestimmung genau erfolgen.

Szenen Analyse wird häufig für maschinelle Lernverfahren wie Wahrscheinlichkeitsmodelle, k-Nearest-Neighbours, Neuronale Netze und Support Vector Machines benutzt.

In der Szenen Analyse werden Algorithmen benutzt, wo zuerst die Attribute einer Szene (Fingerabdruck) erstellt wird und dann der Standort mit aktuellen Umgebungsdaten über die a-priori Wahrscheinlichkeiten der Fingerabdrücke abgeschätzt wird. Häufig werden die Fingerabdrücken über received signal strength (RSS) Werte gebildet.

Es gibt zwei Phasen wenn Standorte mit Hilfe von Fingerabdrücken bestimmt werden sollen. Zuerst die offline Phase, in der die Umgebungsdaten und Signalstärken zu einzelnen Positionen gesammelt und abgespeichert wird. In der online Phase wird dann eine Position über die gesammelten Fingerabdrücke bestimmt.

Bei der Nachbarschaftsbestimmung gibt es ein dichtes Gitter von Antennen, dessen Position man kennt. Wenn ein Gerät von einer Antenne erkannt wird, ist es mit ihr Verbunden. Wenn das Gerät von mehreren Antennen erkannt wird, ist es mit der Antenne verbunden zu der die Signalstärke am größten ist. Dieses Verfahren basiert darauf, dass man über Mobielfunknetze eine ungefähre Position von Mobiltelefonen ermitteln kann, indem geschaut wird, welcher Standort zu welchem Zeitpunkt verwendet wird.

Die Bewertung von verschiedenen Positionierungstechniken kann nicht nur anhand der erzielten Genauigkeit erfolgen. Als Vergleichspunkte werden Genauigkeit, Präzision, Komplexität, Skalierbarkeit, Robustheit und Kosten vorgeschlagen.

Des weiteren gibt es verschiedene Funktechnologien wie GPS, RFID, WLAN und Bluetooth, die je nach Bedarf genutzt werden.

Untersucht wurden Lösungen die drahtlose Technologien und Positionierungsalgorithmen verwenden, auf ihre Anwendungssituationen in geschlossenen Räumen.

Das Ergebnis ist, das man je nach Anwendungsfall genau abwägen muss, was ein geeignetes System ist.

In Zou et al. [11] wird ein Indoor-Positionsbestimmung Algorithmus auf Basis von online sequential extreme learning machine (OS-ELM) vorgestellt. Die Ergebnisse werden mit denen einer batch ELM verglichen.

Jedes Positionsbestimmungsverfahren, das mit Fingerabdrücke arbeitet, braucht eine offline Phase um die Daten zu sammeln. Dabei ergeben sich zwei Probleme. Erstens dauert es lange, die Daten zu sammeln. zweitens können sich die offline gesammelten Fingerabdrücke von denen im Live-Betrieb unterscheiden.

Eine batch extreme learning machine (ELM) ist eine Art maschineller Lernalgorithmus der auf Single-hidden Layer Feedforward Neural Network (SLFN) basiert. Die OS-ELM kann online sequentiell lernen, muss also nicht komplett neu trainiert werden, wenn neue Daten hinzukommen.

Bei der vorgestellten Lösung wird die Lokalisierung als Regressionsproblem betrachtet. Zunächst wird in einer offline Phase, die erste OS-ELM trainiert. Da aufgrund der schnellen Lerngeschwindigkeit der OS-ELM nicht so viele Daten gesammelt werden müssen, um ein Modell zu erstellen, kann die offline Phase kürzer sein. In der Online Phase werden neue Fingerabdrücke in das Modell integriert, wodurch das Modell aktuell gehalten wird.

In dem Experiment soll nicht die Position aus einer Menge von festen Plätzen bestimmt werden, sondern der Platz innerhalb eines Gebäudes in 2-D, die Höhe wird nicht betrachtet. Die Ergebnisse der OS-ELM sind in dem Versuchsaufbau genauer, als die der batch ELM.

In Madigan et al. [5] werden mehrere Clients gleichzeitig lokalisiert. Clients sind zum Beispiel Laptops, Handys und ähnliche mit kabelloser Hardware versehenen Boxen. Dazu werden hierarchical Bayesian graphical models und RSS benutzt.

Ein Graph Modell, ist ein multivariates statistisches Modell, das bedingte Unabhänigkeitsbeziehungen darstellt. Diese Beziehungen werden in einem Graphen dargestellt. Dabei entsprechen die Knoten Zufallsvariablen und die Kanten den Beziehungen [5].

Es wird ein zwei Dimensionales Lokalisationsproblem behandelt, welches vier Zugangspunkte für WLAN Signale hat. Untersucht werden ein Non-Hierarchical Bayesian und ein Hierarchical Bayesian Graphical Model. In der ersten Ebene sind die X und Y Koordinaten um die Position anzugeben. Die zweite Ebene gibt die Euklidische Distanz zwischen dem Objekt und dem jeweiligen Zugangspunkt an. In der dritten Ebene ist die empfangene Signalstärke von den einzelnen Zugangspunkten dargestellt.

Für das Hierarchical Bayesian Graphical Model wird noch betrachtet, das in den Zugangspunkten immer der selbe physikalische Prozess abläuft.

Die Arbeit von Zou et al. [10] behandelt die Indoor Positionsbestimmung mittels weighted extreme learning machine (WELM). Dafür wurde ein WELM-Modell trainiert. Auf diesem Modell wurden verschiedene Indikatoren getestet und verglichen welches die besten Ergebnisse liefert. Die Indikatoren sind received signal strength (RSS), signal strength dierence (SSD) und eine aus dem RSS erstellter signal tendency index (STI).

Bei dem STI handelt es sich um eine Standardisierte Variante der Fingerabdrücke, um Robustheit gegenüber variierenden Umgebungsdaten und Hardware unterschieden auszugleichen.

In echtwelt Daten besteht meistens eine unausgeglichenen Klassenverteilung. Die WELM kann besser als die ELM Regressions und Klassifikationsaufgaben mit ungleich verteilten Klassen lösen.

In der online Construction Phase wird eine Datenbank mit den standardisierten Fingerabdrücke erstellt und damit das STI-WELM Modell gelernt. Die zweite Phase, die online Location Phase, nutzt das gelernte Modell und bekommt echtzeit Daten um die Position des Nutzers zu ermitteln.

Kapitel 3

Methoden

Nach Russel und Norvig [8] ist maschinelles Lernen wie folgt definiert:

Im maschinellen Lernen soll ein Agent etwas lernen. Ein Agent lernt etwas, wenn er seine Leistung für zukünftige Aufgaben verbessern kann, indem er sein Arbeitsumfeld beobachtet. Es gibt verschiedene Arten des maschinellen Lernens. Man unterscheidet zwischen bestärkendem, unüberwachtem und überwachtem Lernen.

Das unüberwachte Lernen zeichnet sich dadurch aus, dass der Agent selbst keine Rückmeldung bekommt. Er entscheidet selbst was zusammengehört. Daher wird diese Methode vor allem für Clustering verwendet.

Beim bestärkendem Lernen (englisch reinforcement learning) erlernt der Agent selbst eine Strategie, um Belohnungen bzw. Bestrafungen zu maximieren bzw. zu minimieren.

Im überwachten Lernen gibt es Eingaben und das Ziel ist es, die Ausgabe vorherzusagen. Die Methoden der Klassifikation und Regression gehören zum überwachtem Lernen.

Die Trainingsdaten T im überwachten lernen bestehen aus Eingabe-Ergebnispaaren (x_i, y_i). Es wir versucht Y anhand von X vorher zu sagen. Das Y, welches vorhergesagt wird, wird mit $\hat{Y}$ bezeichnet. Eine Eingabe ist ein Vektor von beobachteten Eigenschaften, Attribute genannt. Die Variable X_j bezeichnet ein Attribut aus X.

Wenn das zu vorhersagende Ergebnis aus einer endlichen Menge G von Werten (z.b Rot, Gelb, Blau) ist, spricht man von Klassifikation. Wenn Y eine Zahl ist, wird das Lernproblem Regression genannt [8]. Es gibt verschiedene Methoden wie man über die selben Daten die Ergebnisse vorhersagen kann. Welche Methode sich gut eignet ist problemabhängig, weshalb in dieser Arbeit mehrere Methoden miteinander verglichen werden. Auf das Problem des Lagers bezogen würde man vermuten, dass sich k-Nearest Neighbors (kNN) besonders eignet, da alle Kisten nebeneinander stehen und es eine wirkliche Nachbarschaftsbeziehung gibt. Laut den Ergebnissen von Masoudinejad et al. [6] ist dies allerdings nicht der Fall. Dort schneidet kNN am schlechtesten ab.

Im folgenden werden die verwendeten Methoden betrachtet.

3.1 Naive Bayes

Naive Bayes Klassifizierer sind Wahrscheinlichkeit Klassifizierer. Sie basieren auf Bayes Theorem und der Annahme, dass gegeben Klasse $G = j$, die Attribute X_l unabhängig sind [3]. Ansonsten sind die Attribute nicht voneinander unabhängig. Die Wahrscheinlichkeit mit der ein Beispiel einer Klasse j zugeordnet wird, wird mit $f_j(X) = \prod_{l=1}^{p} f_{jl}(X_l)$ berechnet. Die Funktion f_{jl} berechnet dabei die Wahrscheinlichkeit, dass die Ausprägung des l-ten Attributes in der j-ten Klasse vorkommt. Die Klasse j für die f_j die höchste Wahrscheinlichkeit berechnet, wird als Klasse für das Beispiel gewählt.

3.2 k-Nearest Neighbors

k-Nearest Neighbors (kNN) ist ein Mustererkennungsalgorithmuss für Klassifikation und Regression [3]. Er betrachtet die k ähnlichsten Trainingsdaten zur Eingabe um $\hat{Y}$ vorherzusagen. Die Ähnlichkeit wird über eine Nachbarschaftsfunktion $N_k(x)$, was verschiedene Distanzmetriken sein können, ermittelt.

Bei der Klassifikation mit kNN wird die Klassenzugehörigkeit geprüft. Ein Beispiel x_i wird der am häufigst verwendeten Klasse zugewiesen, unter Betrachtung der k ähnlichsten Beispiele. Wenn $k = 1$, dann wird das Beispiel x_i der Klasse, in der das ähnlichste Beispiel ist, zugewiesen.

Der Einfluss der Beispiele auf die Klassifizierung kann auch gewichtet werde. Damit sind die Beispiele, die näher dran sind, wichtiger, als die, die weiter weg sind. Ein Beispiel für eine Gewichtung ist $\frac{1}{N_k(x)}$.

3.3 Decision Tree

Entscheidungsbäume (engl. Decision Trees) können für Klassifikation und Regression benutzt werden. In der Klassifikation sind die Blätter des Baumes, die Klassen aus G. Der Baum entsteht durch die Spaltung der Trainingsdaten anhand der vorhandenen Attribute und ihrer Ausprägungen [8]. Welches Attribut zu Spaltung benutzt wird, wird durch ein statistisches Kriterium (siehe Kapitel 6.1) ausgewählt.

In Abbildung 3.1 ist ein Beispiel Decision Tree, der auf den PhyNode Daten gelernt wurde, dargestellt. Zu Anschauungszwecken wurde eine geringe Tiefe von vier gewählt, wodurch nicht alle möglichen Klassen aufgeführt werden. Rapidminer teilt die Gruppen in jeder Spaltung in zwei Teile auf, wodurch Binärbäume als Struktur entstehen. In den Inneren Knoten steht, welches Attribut zur Spaltung benutzt wird und an den Kanten, wie die Merkmalsausprägung der einzelnen Gruppen ist.

Eine Eingabe wird einer Klasse zugeordnet, indem an der Wurzel des Entscheidungsbaumes geprüft wird welchem Kind die Merkmalsausprägung der Eingabe entspricht. Dies wird für jeden Knoten durchgeführt, bis ein Blatt und damit die Zuordnung zu einer Klasse erreicht wird.

Abbildung 3.1: Beispiel Entscheidungsbaum, maximale Tiefe 4, Information Gain Kriterium

3.4 Random Forest

Random Forest (RF) kann für Klassifikation und Regression benutzt werden. In der Klassifikation gibt es in Random Forest eine Menge von Decision Trees, wo jeder einzelne eine Klassenvorhersage macht. Das Prinzip basiert darauf, dass viele Bäume erstellt werden und das Endergebnis die Mittlung über die einzelnen Bäume ist [3]. Wenn dabei jedoch oft ähnliche Bäume enthalten sind, ist die durchschnittliche Erwartung des Random Forest genauso wie die Erwartung eines Baumes. Eine Verbesserung wird durch die Verringerung der Varianz σ^2 erzeugt. Eine Menge von B unabhängigen, gleich verteilten Zufallsvariablen mit jeweils der Varianz σ^2, hat eine durchschnittliche Varianz von $\frac{1}{B}\sigma^2$. Wenn die Zufallsvariablen gleich verteilt sind und eine paarweise Korrelation ρ haben, ist die durchschnittliche Varianz $\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$. Um die Varianz zu verringern muss die Korrelation zwischen den Bäumen so verringert werden, dass sich die Varianz nicht erhöht. Sei p die Anzahl an Attributen, dann soll vor jeder Spaltung eine Menge von $m \leq p$ Attributen ausgewählt werden, die die Kandidaten für die Spaltung ist. Je kleiner das m ist, desto kleiner ist die Korrelation zwischen den paaren der Bäume. Dadurch wird der Durchschnitt der Varianz verringert.

3.5 Lineare Regression

Lineare Regression ist ein Modell für Regression. Als Eingabe hat man einen Vektor $X^T = (X_1, X_2, ..., X_p)$ und es soll ein reellwertigs Y vorhergesagt werden.

Das Modell für lineare Regression wird durch $f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$ beschrieben.

Dabei gilt die Annahme, dass die Regressionsfunktion $f(X)$ linear ist oder das das Modell einer vernünftige Annäherung erzeugt [3].

Die X_j können aus verschiedenen Quellen stammen. Die β_j werden aus den Trainingsparametern berechnet.

Eine Methode zum schätzen der β ist least squares. Bei least squares werden die β so gewählt, dass die verbleibende Quadratsumme, residual sum of squares $RSS(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2$ minimiert wird.

Das Gauss-Markov Theorem besagt dabei, dass die β bei Verwendung von least squares die kleinste Varianz haben, als alle anderen linearen Schätzer, die nicht den Bias betrachten [3].

Die Vektor Lineare Regression beschreibt ein lineares Regression Modell, bei dem nicht eine Wert, sondern ein Vektor von Werten vorhergesagt werden soll.

3.6 Support Vector Machine

Die Support Vector Machine (SVM) [8] kann Klassifkations (SVC) und Regressions (SVR) Aufgaben lösen.

Für eine typische Klassifikationsaufgabe geht man davon aus, dass es zwei Klassen gibt. In der Einen legen die Werte mit $Y = 1$ und in der Anderen die Werte mit $Y = -1$. Alle Datenpunkt können im $\mathbb{R}^n$ abgebildet werden, die SVM soll eine Ebene (Hyperebene bei beliebigen Dimensionen) finden, die die Klassen möglichst gut voneinander trennt.

Der Bereich zwischen den Punkten der zwei Klassen, welcher um der Ebene liegt, wird Margin genannt. Je größer der Margin, desto mehr Platz ist zwischen einer Seite des Margin und der Ebene, was zu einer erhöhten Fehlertoleranz führt. Die Formel zur Berechnung der Ebene lautet $S := \{\overrightarrow{x} | \overrightarrow{w} * \overrightarrow{x} + b = 0\}$, wobei $\overrightarrow{w}$ ein Gewichtsvektor und b ein Versatz ist. Die Gewichtung und der Versatz sind so optimierbar, sodass der größte Margin gefunden wird, aber trotzdem alle Beispiele richtig klassifiziert werden. Die Beispiele, die auf dem Margin liegen, sind Stützvektoren. Sie haben den geringsten Abstand zur Eben und sind dadurch für die Klassifizierung wichtiger sind als die anderen Beispiele.

Für die SVM ist es wichtig, dass sich die Beispiele durch eine lineare Ebene teilen lassen. Wenn dies in der Eingabedimension nicht möglich ist, kann die SVM mit dem Kernel Trick die Daten in einen höherdimensionalen Raum überführen und dort die Ebene berechnen.

SVMs können auch für Regression benutzt werden. Dabei gibt es einen ϵ-Bereich um die Ebene, wenn in diesem Bereich ein Fehler liegt, wird er nicht in der Fehlerfunktion betrachtet [3].

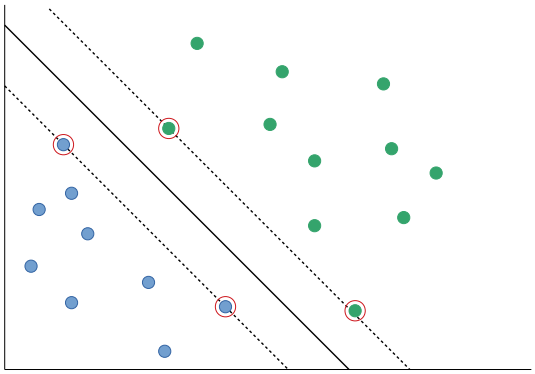

Abbildung 3.2: Die Klassen Grün und Blau getrennt durch eine Ebene schwarze Linie), der Bereich zwischen den gestichelten Linien ist der Margin. Die Ebene ist so gewählt, das der Abstand zu den Margin rändern maximal ist. Die rot umrandeten Punkte sind am nächsten an der Trennebene und bilden damit die Stützvektoren (support vectors)

3.7 Kreuzvalidierung

Um den durchschnittlichen Gesamtfehler zu berechnen eignet sich Kreuzvalidierung [3].

Bei der K-fachen Kreuzvalidierung werden die Daten in K gleich große Teile geteilt. Für jeden $k$ten Teil wird dann aus den $K-1$ anderen Teilen ein Modell erstellt und dies auf den $k$ten Teil getestet. Der Fehler wird für alle $k \leq K$ Teile ermittelt und der Durchschnitt dieser Fehler ist dann der durchschnittliche Gesamtfehler.

Bei N Daten und $K = N$ handelt es sich um eine leave-one-out Kreuzvalidierung weil man immer einen Datenteil gegen den Rest prüft.

Übliche Werte für die Kreuzvalidierung sind $K = 5$ oder $K = 10$. Für $K = N$ ist zwar der Bias gering, dafür kann aber die Varianz sehr groß sein. $K = 5$ führt zu einer geringen Varianz, aber der Bias kann, je nach Trainingsmethode, groß sein. Insgesamt kann jedoch gesagt werden das 5, oder 10-Fache Kreuzvalidierung ein guter Kompromiss ist [3].

3.8 Gridoptimierung

Bei der Gridoptimierung bzw Grid search, ist jeder Parameter, den man optimieren möchte eine eigene Dimension. In dieser Dimension überprüft man nicht alle Punkte, sondern nur ausgewählte, so dass man die Punkte auf dem Gitter testet. Am Ende wird die Kombination an Parametern gewählt, die das beste Ergebnis, von den getesteten Punkten, erzielt. Wenn das Optimum nicht auf dem Gitter liegt, kann es auch nicht gefunden werden. Aber der Suchraum wird dadurch kleiner, wodurch ein Optimum schneller gefunden werden kann.

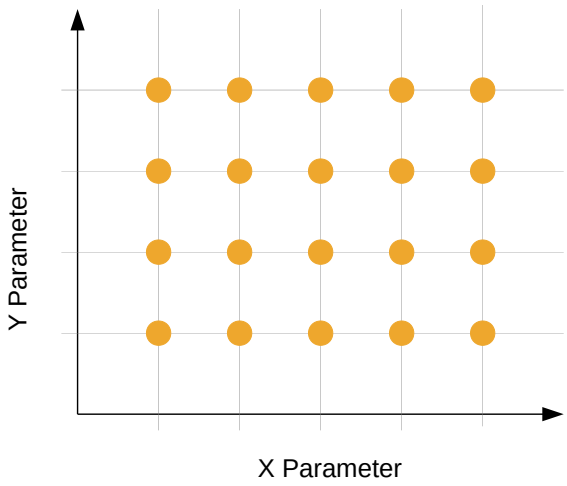

Abbildung 3.3: Schema einer Gridoptimierung, orange Punkte sind Parameterkombinationen, welche getestet werden

Kapitel 4

Daten

Die Daten, die im PhyNetLab für das Paper von Masoudinejad et al [6] gesammelt wurden, stehen unter http://phynetlab.com zur Verfügung. Dort findet sich auch eine Anleitung [9] für den Hackatron, in der der Versuchsaufbau nochmal genauer beschrieben wird.

4.1 Hardware

Die Hardware, die im PhyNetLab verwendet wird [6, 2] besteht aus PhyNodes und Accesspoints.

Die PhyNode ist ein ultra Low Power Internet-of-Things Gerät, also ein eingebettetes System, welches wenig Energie verbraucht. Der Energieverbrauch liegt zwischen weniger als $1\mu W$ und $5mW$. Der Mikrokontroller der PhyNode hat $2kB$ RAM und $64kB$ Ferroelectric Random Access Memory (FRAM), bleibt ohne Strom bestehen (persistenter Speicher), welcher sich auf zwei Teile mit $48kB$ und $16kB$ aufteilt. Da einiges an Speicher bereits für Betriebtssystem, Treiber und Top-Level-Anwendungen benutzt wird, stehen ca. $14kB$ FRAM für das maschinelle Lernmodell zur Verfügung.

Die PhyNode besitzen verschiedene Sensoren um ihre Umgebungsdaten zu messen. Dazu gehören zwei Umgebungslichtsensoren, die den Lux-Wert bestimmen, ein MEMS-Beschleunigungssensor (Micro Electromechanical System), der die Beschleunigung der drei Achsen und die Temperaturdaten misst. Des weiteren hat jede PhyNode eine Antenne um im $868MHz$ Band zu kommunizieren.

Die Accesspoints bestehen aus einem Rasberry Pi, welcher mit vier CC1200 Radio Chips über Serial Peripheral Interface (SPI) verbunden ist. Daher kann ein Accesspoint mit mehreren PhyNodes über verschiedene Kanäle zur Selben Zeit kommunizieren. Außerdem können die Accesspoints mit dem Lokalen Netzwerk oder über Ethernet mit dem Internet, verbunden werden. Da sie mit $230V$ oder einer Batterie betrieben werden können, können sie überall platziert werden, auch wenn kein direkter Stromanschluss vorhanden ist.

4.2 Rohdaten

Die Daten stammen aus dem Logistik Labor der TU Dortmund [6]. An den Lagereinheiten wurde eine PhyNode angebracht. Jede PhyNode speichert für einen Zeitpunkt, die drei RSSI Werte zu den verschiedenen Accesspoints, den Lux Wert des Lagerplatzes, Beschleunigung in x, y, und z - Richtung, eine Temperatur und die ID der PhyNode. Die Daten für das Maschinelle Lernen erhalten

zusätzlich zu den Daten der PhyNodes auch noch die Position der Lagereinheit, sowie Reihe, Spalte und Zeile der Lagerposition als einzelne Werte. Masoudinejad et al. [6] haben in ihrem Versuch zwei Reihen, wo die Kisten in 5 Spalten und 3 Zeilen stehen können. In dem Experiment gibt es 22 PhyNodes, wobei Nummer 21 und 22 Referenzboards sind und immer fest an einer Stelle, außerhalb der zu bestimmenden Positionen, stehen (siehe Figur 2 [9]). Eine Messreihe besteht immer aus den Ergebnissen aller PhyNodes, in einem bestimmten Zeitintervall. Der Datensatz besteht aus mehreren Messreihen zu unterschiedlichen Zeitpunkten und unterschiedlichen Lichtbedingungen.

4.3 Korrektur der Messwerte

Bei einigen PhyNodes gibt es in den Temperaturdaten starke Messabweichungen, für diese wurden die Temperaturdaten korrigiert. Da genügend normale Messwerte vorliegen wurden Mittelwert μ und Standardabweichung σ jeweils für die normalen und die abweichenden Messwerte der PhyNodes berechnet. Danach wurden die abweichenden Messwerte skaliert. Dafür wurden die abweichen auf Mittelwert 0 und Standardabweichung 1 z-transformiert. Anschließend wurden die standardisierten Messwerte in den Bereich der normalen Messwerte verschoben, in dem die Standardabweichung und der Mittelwert der standardisierten Werte auf den der normalen Messwerte berechnet wurde. Damit ist der Mittelwert und die Standardabweichung aller Daten in den PhyNodes gleich dem Mittelwert und der Standardabweichung die vorher nur die normalen Messergebnisse hatten.

	Mittelwert	Standartabweichung
normalen Messwerte	2031	32
abweichende Messwerte	28510	53

Tabelle 4.1: Mittelwert und Standartabweichung der normalen und abweichenden Temperaturmesswerte von PhyNode Nr.. 5

Utime	RSSI 1	RSSI 2	RSSI 3	LUX	ACC x	ACC y	ACC z	Temp	ID	POS	SI	CO	RO
1505906457555	51	64	60	82	-1051	-46	110	28431	5	231	2	3	1
1505906457555	51	64	60	82	-1051	-46	110	1982.9	5	231	2	3	1

Tabelle 4.2: Beispiel Daten vor und nach Skalierung, von PhyNode Nr.. 5, ohne Zuordnung von Tag und Lichtpegel. Die erste Zeile zeigt das Beispiel vor der Skalierung und die zweite Zeile das Beispiel nach der Skalierung.

Die Formel für die z-Transformation (Standardisierung) lautet $Z_i = \frac{X_i - \mu}{\sigma}$. Dabei ist Z_i der standardisierte Wert des Attributs X_i. Der Mittelwert ist μ und die Standartabweichung σ.

Mit den Werten aus der 4.1 kann man die Temperatur Werte aus 4.2 skalieren. Zuerst wird der Temperaturwert standardisiert. Dafür wird die normale z-Transformationsländer benutzt $Z_i = \frac{X_{iv} - \mu_v}{\sigma_v}$ wobei X_{iv} für das Attribut vor der Skalierung steht. Die Variablen μ_v und σ_v stehen für den Mittelwert und die Standartabweichung der abweichenden Messwerte. Im zweiten Schritt wird die Berechnung umgedreht $X_{in} = Z_i * \sigma_n + \mu_n$. Dabei ist X_{in} das Attribut nach der Skalierung und $\mu_n$und σ_n sind Standartabweichung und Mittelwert der normalen Messwerte.

Kapitel 5

Ansatz

Diese Arbeit beschäftigt sich mit der Positionsbestimmung einer Lagereinheit innerhalb eines Lagers mit Hilfe von maschinellem Lernen. Die Positionsbestimmung soll dezentral, von der Lagereinheit selbst erfolgen. Jede Lagereinheit ist dafür im Besitz von lokalen Daten. Die Daten bestehen dabei aus der Empfangsstärke von 3 WLAN Signalen, Temperatur und Lichtstärke. Die Empfangsstärke der WLAN Signale wird durch die Received Signal Strength Indication (RSSI) angegeben. Temperatur und die Lichtstärke werden durch Sensoren der Lagereinheit ermittelt. Die Sensordaten funktionieren dabei wie ein Fingerabdruck, wie in Liu et al. 2017 [4] beschrieben. Mit den vorhandenen Daten und einem vorher erlernten Model soll die Lagereinheit dann ermitteln, an welcher Position (Stellplatz) sie sich selbst innerhalb des Lagers befindet. Je nachdem wie man das Problem betrachtet, können dafür unterschiedliche maschinelle Lernverfahren eingesetzt werden. Wird die Position als ein festes Label behandelt, ist es ein Klassifikationsproblem, da die Menge der Label diskret ist. In dem Versuchsaufbau gibt es 30 mögliche Positionen. Wenn die Position als eine dreidimensionale Koordinate (Reihe (Si), Spalte (Co), Zeile (Ro)) gesehen wird, handelt es sich um ein Regressionsproblem.

Die Position einer Lagereinheit, in einem Lager, ist in der Logistik wie folgt definiert: Die erste Ziffer gibt an in welchem Regal sich die Lagereinheit befindet. Der Aufbau eines Regals ist ein Tabellensystem, wo die Spalten von links nach Rechts und die Reihen von Oben nach Unten durchnummeriert sind. Die zweite Ziffer gibt demnach die Spalte und die dritte Ziffer die Reihe an, in der sich die Lagereinheit befindet. Eine Beispiel einer solchen Lagerposition wird in Abbildung 5.1 gezeigt.

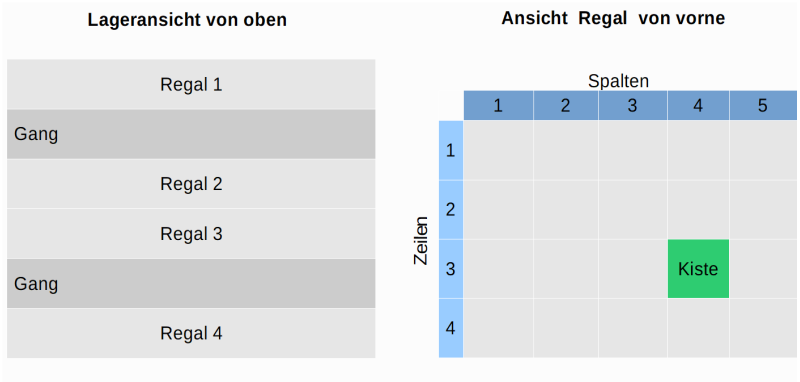

Abbildung 5.1: Links zeigt die Abbildung eines Lagers von oben mit 4 Regalen und dazwischen 2 Gängen. Rechts zeigt die Ansicht eines Regals von vorne mit 5 Spalten und 4 Zeilen und einer grünen Kiste. Wir nehmen an Regal 2 wird in der Vorderansicht gezeigt, dann steht die Kiste auf Position 243.

Die vorverarbeiteten Daten werden mit verschiedenen Modellen der Klassifikation und Regression untersucht. Es wird verglichen mit welchem Modell sich die besten Ergebnisse erzielen lassen. Für die Klassifikation werden Decision Trees, Random Forest kNN, Naive Bayes und SVM benutzt. Lineare Regression, lineare Vektor Regression und Regessions SVM's werden für das Regressionsproblem untersucht. Die Parameter der Modelle werden mit Grid Search optimiert und jeweils zehnfach Kreuzvalidiert, um belastbare Ergebnisse zu erhalten. Jedes Beispiel hat eine Zuordnung zu Tag und Lichtpegel des Versuchs, in dem es aufgenommen wurde. Anhand dieser Zuordnung werden die Daten in Test und Trainingsdaten für die Kreuzvalidierung geteilt. Dabei wird die Größe der kleinsten Gruppe als sample Größe genommen, damit keine Versuchsreihe mehr gelernt wird wie die kleinste.

Kapitel 6

Experimente

Für die Experimente wurden die vorverarbeiteten Daten genommen und zuerst alle Attribute, die nicht zum lernen geeignet sind entfernt. Für dir Klassifikation wurden UnixTime, die ID der PhyNode und die Si, Co und RO Attribute entfernt. Für die Regression wurde zusätzlich zur ID der PhyNode und der UnixTime noch die Lagerfachnummer POS entfernt, da die drei Koordinaten einzeln vorhergesagt werden sollen.

Vor dem Erstellen der eigentlichen Modelle wurde zunächst eine Parameteroptimierung für die einzelnen Modelle angefertigt. Die Performance der Kreuzvalidierung und die optimierten Parameter wurden geloggt.

Die Daten wurden vom Kreuzvaliderungsoperator aus Rapidminer anhand der Batches in Test und Trainingsdaten aufgeteilt. Da allerdings nicht jede Gruppe von Experimenten gleich groß war, wurde die Beispielgröße auf die der kleinsten Batch gesetzt. Hierdurch wird keine Umgebung mehr erlernt wird als eine andere.

In der Kreuzvalidierung wurde zunächst auf einem Teil der Daten die Modelle mit den Parametern trainiert und dann auf den Testbatches evaluiert.

Nach der Optimierung der Parameter wurden für jedes Verfahren die fünf besten Parameterkombinationen gewählt. Für diese wurde nochmal in einer Kreuzvalidierung ein Modell erstellt, dessen Ergebnisse verglichen werden.

Die Daten, Rapidminer Prozesse, erstellten Modelle und Ergebnisse sind in Anhang B beigelegt.

6.1 Klassifikation

Als Vergleichskriterium bei der Klassifikation wird die Accuracy genommen. Sie gibt an, wie viel Prozent der Testdaten richtig klassifiziert wurden. Bei zwei Klassen, positiv und negativ, gibt es vier Möglickkeiten wie Punkte klassifiziert werden können. True Positives sind die, die in der positive Klasse sind, und auch richtig klassifiziert wurden. False Posivives sind diejenigen, die in der nergativen Klasse liegen, jedoch der positiven Klasse zugeordnet wurden. False Negatives gehören zu der positiven Klasse, wurden aber negativ zugeordnet und die True Negatives gehören der negativen Klasse an und werden dort auch zugeordnet. Eine Möglichkeit dies darzustellen ist eine sogenannte Konfusionsmatrix (siehe Abbildung 6.1).

Die Accuracy wird mit $Accuracy = \frac{TP+TN}{all}$, also alle Beispiele, die richtig Klassifiziert wurden durch die Gesamtmenge an Beispielen, berechnet. Da es 30 mögliche Klassen in dem Versuchsaufbau gibt, ist die Konfusionsmatrix dementsprechend größer.

Klassen \ vorhergesagte Klassen	C	$-C$	Gesamt
C	True Positives (TP)	False Negatives (FN)	# Beispiele aus C
$-C$	False Positives (FP)	True Negatives (TN)	# Beispiele aus $-C$
Gesamt	# vorhergesagte C	# vorhergesagte $-C$	all

Abbildung 6.1: Beispiel Konfusionsmatrix (Warheitsmatrix, engl. Confusion matrix), # steht für Anzahl

kNN

Bei kNN wurde die Größe des k, ob die nächsten Nachbarn gewichtet werden sollen und welche Distanzmetrik benutzt wird, optimiert.

Die Größe des k gibt an wie viele Nachbarn betrachtet werden sollen. Der Weighted Vote gibt an ob die Distanz zu den Nachbarn gewichtet werden soll.

Die Attribute sind Zahlen, daher wurden numerische Distanzmetriken miteinander verglichen.

In einer grafischen Darstellung ist jedes Attribut eine Dimension. Zur Vereinfachung werden die Distanzmetriken für den zweidimensionalen Raum dargestellt. Bei kNN ist einer der beiden Punkte, zwischen denen die Distanz berechnet wird, immer fest. Es ist die Datenreihe, die gerade ihren nächsten Nachbarn sucht.

Die euklidische Distanz, ist die Distanz die direkt zwischen zwei Punkten liegt. Die Formel dafür ist $d_{(i)} = ||x_{(i)} - x_0||$ [3].

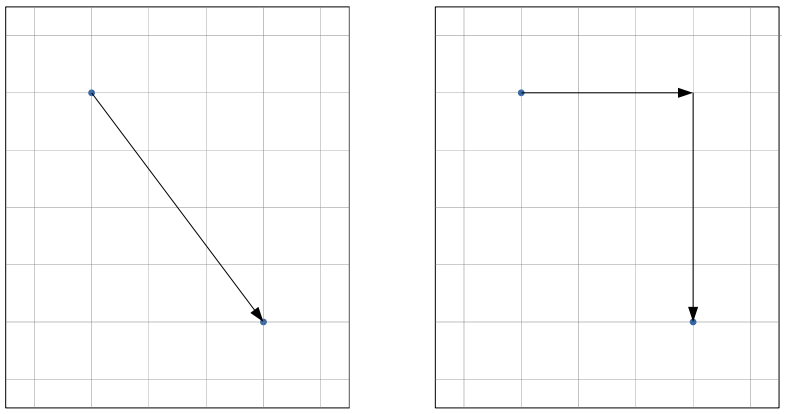

Abbildung 6.2: Darstellung einer Stadt, graue Linien stellen Straßen dar. Abstand zwischen zwei Punkten. Links: Beispiel Euklidische Distanz, Rechts: Beispiel Manhattan Distanz

Bei der Manhattan Distanz wird nicht die direkte Strecke zwischen den Punkten berechnet, da davon ausgegangen wird, dass der direkte Weg nicht möglich ist. Es soll der Weg zwischen Punkt A und B berechnet werde, wobei die Stadt quadratisch aufgebaut ist, wie Manhattan. Da der Weg durch die Gebäude blockiert ist, kann sich nur horizontal und vertikal, auf den Straßen, bewegt werden.

In Abbildung 6.3 ist zu sehen, dass während die Anzahl der betrachteten Nachbarn kaum ins Gewicht fallen, die benutzte Distanzmetrik einen Einfluss auf die Accuracy hat. Die euklidische Distanz funktioniert dabei wesentlich schlechter, als die Manhattan Distanz. Auch ob die Nachbarn gewichtet werden oder nicht, hat wenig Einfluss auf die erreichte Accuracy (Abbildung 6.4). Die besten fünf Modelle erreichen alle eine Accuracy zwischen 50-60%, allerdings nur mit einer sehr großen Standartabweichung.

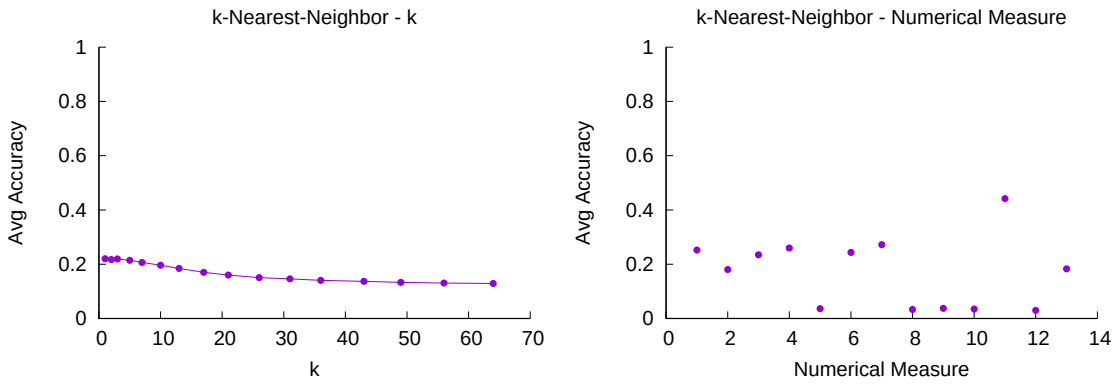

Abbildung 6.3: Durchschnittliche Accuracy der links: Werte für k (1 - 65), kNN; rechts: verschiedene Numerische Distanzmetriken (von links: Nr.. 7 Euklidische Distanz, Nr.. 11 Manhattan Distanz), kNN

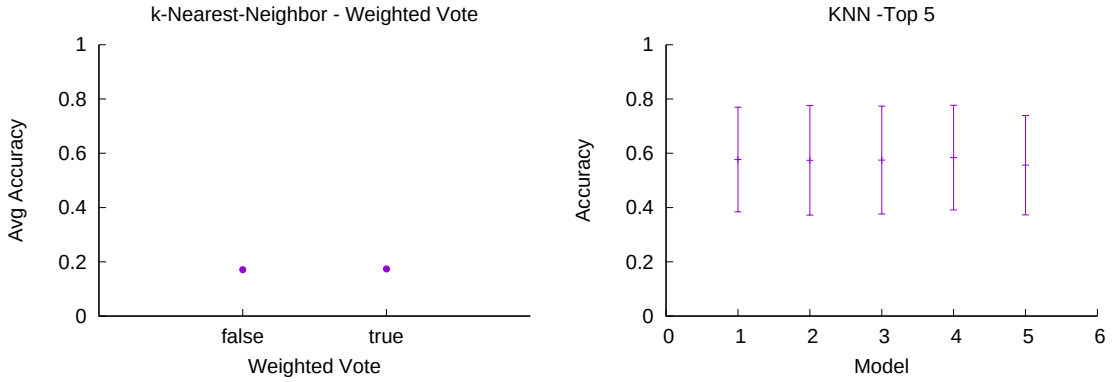

Abbildung 6.4: Durchschnittliche Accuracy der links: Gewichtung von Nachbarn, kNN; rechts: Vergleich der besten fünf Modelle, mit Betrachtung der Standartabweichung

Model Nr.	Accuracy	σ^2	k	Weighted Vote	Measure Type
1	0,557	0,193	2	True	Manhattan distance
2	0,574	0,202	1	True	Manhattan distance
3	0,575	0,199	3	True	Manhattan distance
4	0,584	0,193	1	False	Manhattan distance
5	0,556	0,183	5	True	Manhattan distance

Tabelle 6.1: Accuracy, Standardabweichung σ^2, und Parameter der Top 5 kNN Modelle

Decision Tree

Für Entscheidungsbäume wurde die Tiefe der Bäume und das Kriterium, das entscheidet an welchem Kriterium die Daten jetzt gespalten werden, optimiert.

Die Tiefe von Bäumen darf nicht zu groß sein, da man ansonsten entweder auswendig lernt oder der Baum so groß ist, dass man nicht mehr Effizient in ihm suchen kann. Wenn der Baum nicht tief genug ist, kann das Modell die Daten nicht richtig abbilden und damit nicht gut vorhersagen [8].

Bei der Optimierung konnte beobachtet werden, dass auch wenn der Parameter der maximalen Tiefe höher eingestellt war, nie Bäume mit einer Tiefe von mehr als 15 gebildet wurden. Daher wurden die Parameter und die Besten fünf Modelle über eine maximale Tiefe von 15 untersucht.

Entropie ist ein Maß für die Unsicherheit einer Zufallsvariable. Die Shannon-Entropie bezieht diese Unsicherheit auf den Informationsgehalt einer Nachricht. Für eine Zufallsvariable V, mit den Werten v_k, ist die Entropie wie folgt definiert. $H(V) = -\sum_k P(v_k) log_2 P(v_k)$, wobei Eta (H) die Entropie und $P(v_k)$ die Wahrscheinlichkeit für v_k ist [8].

Im folgenden werden die verschiedenen statistischen Teilungskriterien erklärt.

Für das Information Gain Kriterium wird die Entropie von allen Attributen berechnet. Das Ziel ist es den Baum so zu spalten, dass der Informationsgewinn am größten ist, dafür wird das Attribut mit der Geringsten Entropie geteilt [7]. Die Formel für Information Gain ist:

$$IG(T_1, ..., T_m) = H(T) - \sum_{i=1}^{m} \frac{|T_i|}{|T|} * H(T_i)$$

.

Gain Ratio betrachtet der Information Gain unter Berücksichtigung der erwarteten Entropie [7]. Der Baum wird am Attribut mit dem höchsten Gain Ratio geteilt.

$$GR(T_1, ...T_m) = \frac{IG(T_1, ..., T_m)}{-\sum_{i=1}^{m} \frac{|T_i|}{|T|} log_2 \frac{|T_i|}{|T|}}$$

Der Gini Index gibt an, wie in einem Attribut die Beispiele über die Klassen verteilt sind [3]. Auch hier wird das Attribut mit dem höchsten Gini Index gespalten. $Gini(T) = 1 - \sum_{j=1}^{k} P(j)^2$, dabei ist $P(j)$ die Wahrscheinlichkeit das T in der Klasse j liegt.

Beim Accuracy Kriterium wird das Attribut gesplittet, welches für den gesamten Baum die Accuracy am meisten erhöht.

Gain Ratio ist als Kriterium auf den vorhandenen Daten nicht so gut wie die anderen Kriterien. Die Tiefe des Baumes verbessert die Accuracy bis zur Tiefe von 15 immer weiter. Mit einer Accuracy von 88-91 % und einer geringen Standartabweichung, sind die Entscheidungsbaum modelle geeignet.

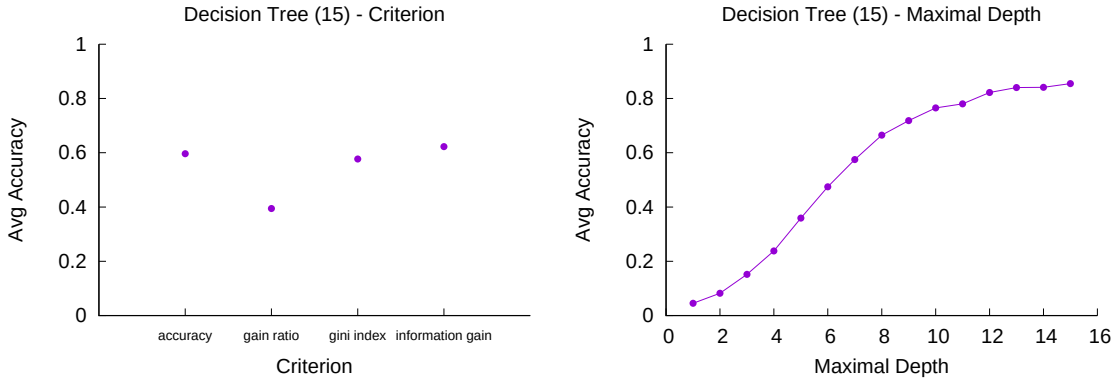

Abbildung 6.5: Durchschnittliche Accuracy der links: Kriterien, Decision Tree; rechts: maximalen Baumtiefe (1-15), Decision Tree

Model Nr.	Accuracy	σ^2	Max Depth	Criterion
1	0,90	0,04	13	information gain
2	0,88	0,05	11	information gain
3	0,91	0,05	15	information gain
4	0,88	0,06	13	gini index
5	0,90	0,05	12	information gain

Tabelle 6.2: Accuracy, Standardabweichung σ^2, und Parameter der Top 5 Decision Tree Modelle

Model Nr.	Accuracy	σ^2	Max Depth	Criterion	Num of Trees	Voting Strategy
1	0,944	0,040	23	gini index	45	confidence vote
2	0,948	0,044	38	information gain	35	confidence vote
3	0,949	0,044	13	information gain	50	majority vote
4	0,948	0,042	49	information gain	35	majority vote
5	0,930	0,047	49	gini index	45	majority vote

Tabelle 6.3: Accuracy, Standardabweichung σ^2, und Parameter der Top 5 Random Forest Modelle

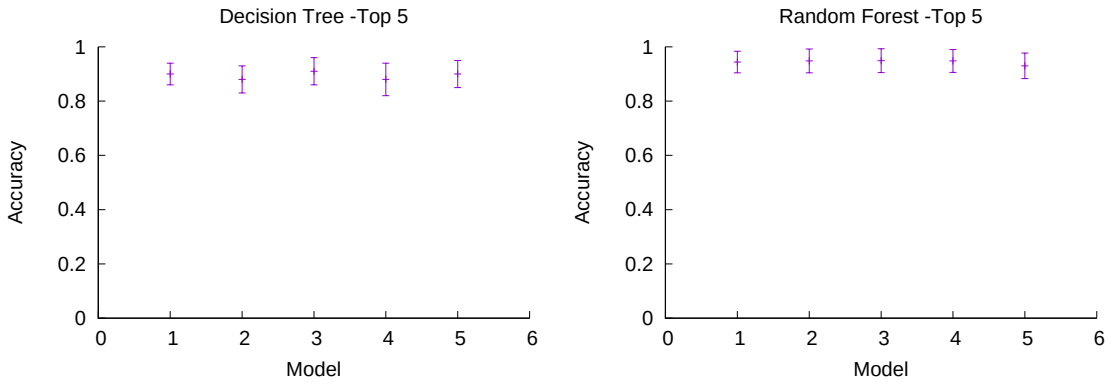

Abbildung 6.6: Links: Top 5 Decision Tree Modelle mit Betrachtung der Standartabweichung, Rechts: Top 5 Random Forest Modelle mit Betrachtung der Standartabweichung

Random Forest

Für Random Forest wurde die Anzahl an Bäumen, die maximale Tiefe der Bäume und das Kriterium zum Erstellen der Bäume optimiert. Außerdem wurde optimiert, ob die Attributsets für die einzelnen Bäume geraten werden sollen und die Strategie mit der im Random Forest die Klassen Vorhergesagt werden.

Die Verfahren zum erstellen der Bäume sind Information Gain, Gain Ratio, Gini Index und Accuracy. Sie funktionieren, genauso wie bei der Tiefe der Bäume, wie bei den Entscheidungsbäumen.

Number of Trees gibt an wie viele zufällige Bäume erstellt werden sollen. Die Voting Strategy gibt an wie über die verschiedenen Baummodelle vorhergesagt werden soll.

Beim Confidence Vote wird die Klasse gewählt, in die man das größte Vertrauen hat, das sie richtig ist.

Der Majority Vote nimmt die Klasse, die von den meisten Bäumen als Ergebnis klassifiziert wurde.

Die Anzahl an Attributen, die für jeden Split zur Verfügung stehen beträgt $log(m) + 1$, mit m als Anzahl an Attributen.

Die Maximale Tiefe und das Kriterium, verhält sich bei den Random Forest wie bei den Entscheidungsbäumen. Bei der Anzahl der Bäume, gibt es bei 6 einen Wechsel der Steigung, danach erhöht sich die Accuracy nur noch wenig. Sowohl Cofidence Vote, als auch Majority vote erzielen ähnliche Ergebnisse.

Random Forest hat mit einer Accuracy von über 90%, die beste Accuracy und die geringste Standartabweichung.

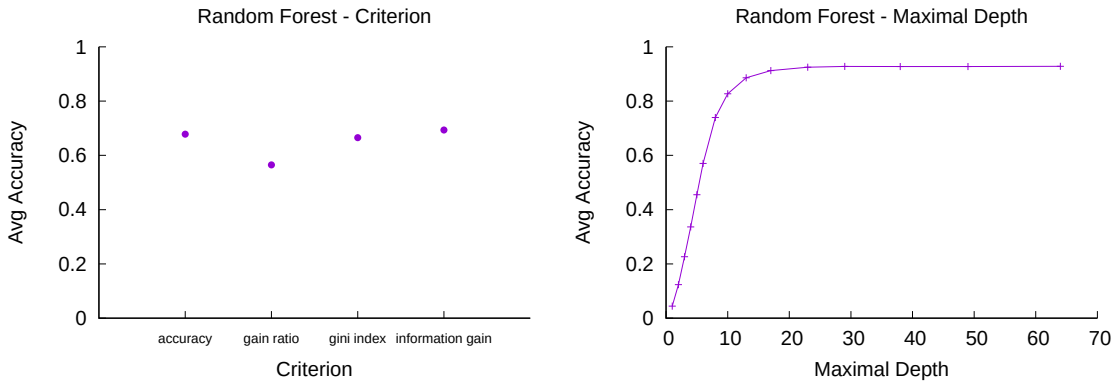

Abbildung 6.7: Durchschnittliche Accuracy der links: Kriterien, Random Forest; rechts: Maximale Tiefe der Bäume (1-64), Random Forest

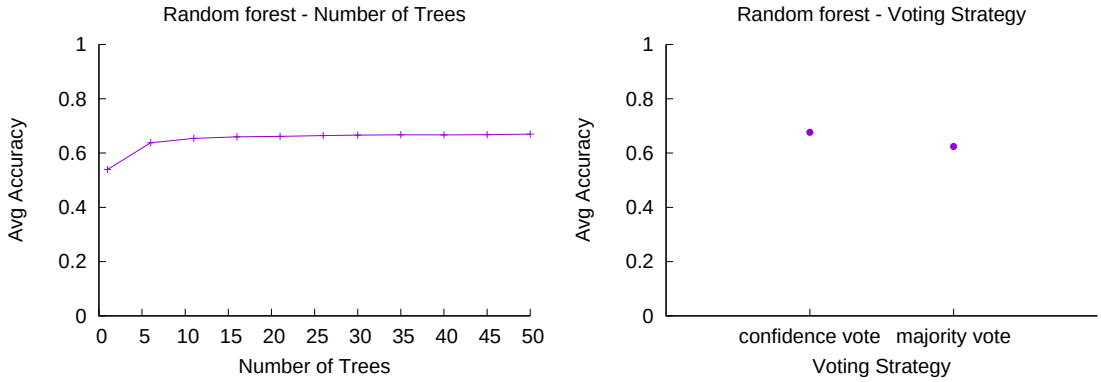

Abbildung 6.8: Durchschnittliche Accuracy der links: Anzahl an Bäumen (1-50), Random Forest; rechts: Voting Strategy, Random Forest

SVC

Für die klassifikations SVM wurde der Kernel, für den Kernel Trick ausgesucht. Benutzt wurde der LibSVM Operator von Rapidminer und daraus die C-SVC.

Eine SVM erstellt eine Ebene immer für die Trennung von zwei Klassen. Da im PhyNetLab 30 verschiedene Klassen vorhergesagt werden können, entscheiden 435 Schätzer zusammen, zu welcher Klasse ein Beispiel gehört [6].

Der polynomial Kernel (poly) [3] berechnet $K(x, x') = (\langle x, x' \rangle + c)^d$, wobei c eine Konstante und d der Polynomgrad ist. Der Grad d wird auch optimiert.

Gaussian radial base funktion (rbf) Kernel [3] berechnet $K(x, x') = e^{-\gamma ||x - x'||^2}$. Gamma kann optimiert werden.

Der rbf Kernel erzielt insgesamt bessere Ergebnisse als der poly Kernel. Der Degeree im poly Kernel ist mit 3 am besten. Gamma im rbf Kernel hat einen starken Einfluss auf das Ergebnis, je kleiner das Gamma, desto besser die Accuracy. Insgesamt gibt es starke Schwankungen zwischen den einzelnen Modellen, mit hoher Accuracy und großer Standartabweichung, bis hin zu niedriger Accuracy und dafür niedrige Standartabweichung.

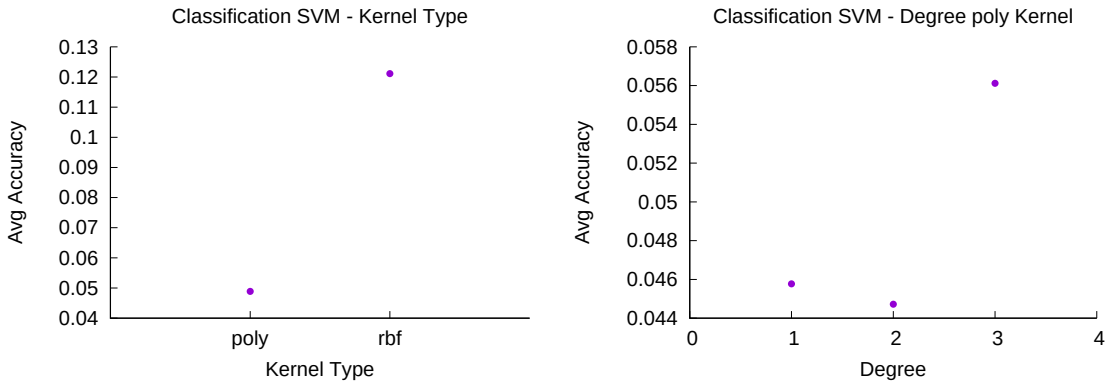

Abbildung 6.9: Links: durchschnittliche Accuracy des Kernel Type SVC. Rechts: durchschnittliche Accuracy des Degree (1-3), poly Kernel SVC

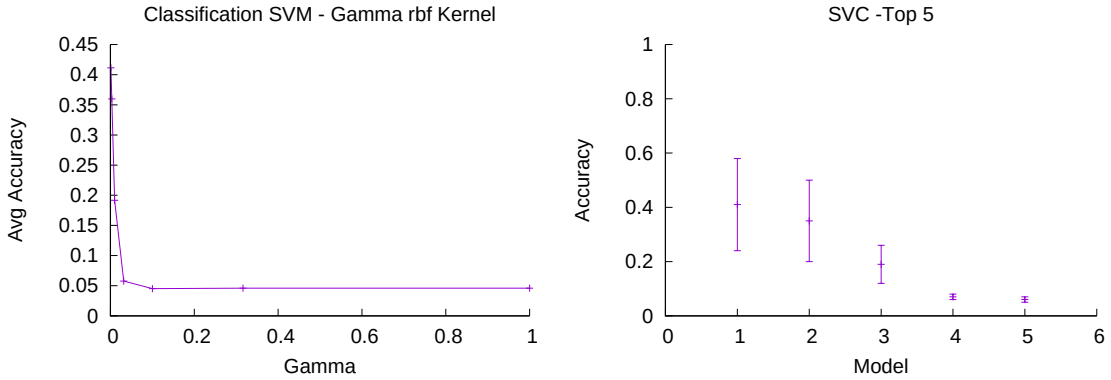

Abbildung 6.10: Links : durchschnittliche Accuracy von Gamma (0,001 - 1), rbf Kernel SVC. Rechts: Top 5 SVC Modelle

6.2 Regression

Als Fehlermaß bei der Regression wird der Mean Squared Error $SqE = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{X}_i)^2$ (im folgenden nur noch Squared Error) benutzt [3]. Der Squared Error ist die Summe über die quadrierte Euklidische Norm. Dieses Fehlermaß eignet sich besonders, da in Rapidminer die Si, Co und Ro Werte nur einzelne evaluiert werden können. Um die Modelle miteinander vergleichen zu können, kann die Summe

$$\frac{1}{n}\sum_{i=1}^{n}(si_i - \hat{si}_i)^2 + (co_i - \hat{co}_i)^2 + (ro_i - \hat{ro}_i)^2$$

als Fehler gebildet werden. Der Summierte Squared Error kann auch so geschrieben werden :

$$\frac{1}{n}\sum_{i=1}^{n}\left\|\begin{pmatrix} si_i \\ co_i \\ ro_i \end{pmatrix} - \begin{pmatrix} \hat{si}_i \\ \hat{co}_i \\ \hat{ro} \end{pmatrix}\right\|_2^2$$

Die Variablen si_i, co_i, ro_i sind die Si, Co und Ro Werte und $\hat{si}_i, \hat{co}_i, \hat{ro}_i$ sind die vom Modell vorhergesagten Si, Co und Ro Werte.

Model Nr.	Accuracy	σ^2	Kernel Type	Gamma	Degree
1	0,41	0,17	rbf	0,001	x
2	0,35	0,15	rbf	0,00316	x
3	0,19	0,07	rbf	0,01	x
4	0,07	0,01	rbf	0,0361	x
5	0,06	0,01	poly	x	3

Tabelle 6.4: Accuracy, Standardabweichung σ^2, und Parameter der Top 5 SVC Modelle

Der Durchschnittliche Summierte Squared Error (SumSQ Error) der Parameter, wurde für Si, Co und Ro getrennt berechnet. Da für alle drei Koordinaten die Parameter im selben Bereich getestet wurden, sind sie in den Grafiken zusammen aufgeführt.

Lineare Regression

Für die lineare Regression wurde in der Kreuzvalidierung jeweils ein Modell für Si, Co und Ro trainiert, und diese dann evaluiert. Als Fehlermaß wurde der Squared Error benutzt.

Die minimale Toleranz gibt an ab wann Attribute mit zu hoher Kollinearität nicht mehr betrachtet werden.

Eine Beschränkungsmethode in der linearen Regression ist die Ridge Regression. Dabei werden Regressionsparameter w unter Betrachtung ihrer Größe beschränkt [3]. Der Ridge Parameter soll so gewählt werden, dass die berechnete quadratische Bestrafung minimiert wird.

Die Minimale Toleranz bleibt bis 0,8 , für Si Co und Ro, ungefähr gleich, danach erhöht sich der Fehler für Co und Ro stark.

Der Ridge Parameter verhält sich für Si, Co und Ro verscheiden. Alle Modelle haben erreichen einen Summierten Squared Error zwischen 7 und 8. Die Standartabweichung der einzelnen Koordinaten ist gering und der Fehler für Si, kleiner als der von Co, und Ro hat den größten Fehler.

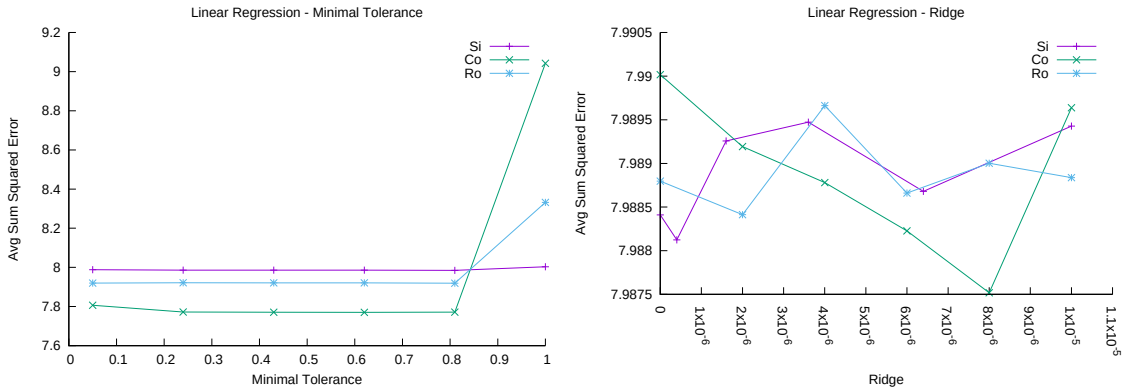

Abbildung 6.11: Links: durchschnittlicher Sum Squared Error der Minimalen Toleranz (0,05 - 1), für Si, Co und Ro, lineare Regression. Rechts: durchschnittlicher Sum Squared Error des Ridge (1,0E-9 - 1,0E-5), für Si, Co und Ro, lineare Regression

Model Nr.	SumSQ Error	Si MinTol	Co MinTol	Ro MinTol	Si Ridge	Co Ridge	Ro Ridge
1	7,65	0,24	0,43	0,43	4,01E-7	2,00E-06	1,00E-05
2	7,72	1	0,43	0,43	4,01E-7	6,00E-06	1,00E-05
3	7,69	0,62	0,05	0,24	6,40E-06	8,00E-06	2,00E-06
4	7,73	0,62	0,24	0,43	3,60E-06	1,00E-05	1,00E-09
5	7,79	0,05	0,05	0,24	1,00E-05	1,00E-05	1,00E-09

Tabelle 6.5: Sum Squared Error und Parameter der Top 5 lineare Regressions Modelle

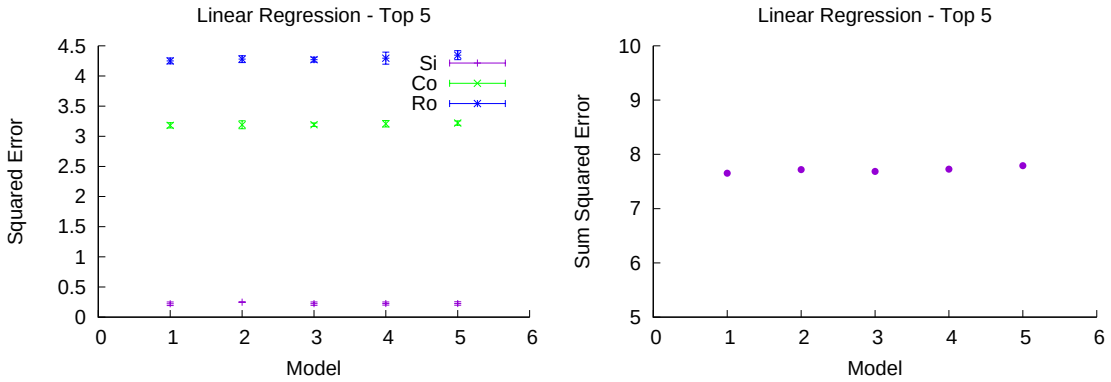

Abbildung 6.12: Links: Squared Error der Top 5 lineare Regressions Modelle, mit Betrachtung der Standartabweichung. Rechts: Sum Squared Error der Top 5 lineare Regressions Modelle

Vektor Regression

In der Kreuzvalidierung wurde ein Vektor Lineare Regression Modell trainiert. Da es allerdings keinen Operator für die Auswertung von Vektoren gibt, musste diese nach Si, Co, Ro getrennt erfolgen. Auch hier wurde der Squared Error als Fehlermaß benutzt.

Die Betrachtung des Bias kann in der linearen Regression für bessere Ergebnisse sorgen. Anhand des Bias kann ein Wert berechnet werden, der dann auf Ergebnisse aufgerechnet wird um sie in eine bestimmte Richtung zu verschieben. Der Use Bias Parameter ist ein Boolean, wenn er True ist, wird der Verschiebungswert berechnet, ansonsten wird der Bias nicht betrachtet.

Auch hier wird als Beschränkungsmethode die Ridge Regression verwendet und damit kann der Ridge Parameter optimiert werden.

Der Ridge Parameter schwankt sehr stark, am besten ist er bei ungefähr 6,0E-8. Es wird immer Use Bias benutzt, weil es den Fehler stark verkleinert. Auch hier liegt der Summierte Squared Error zwischen 7 und 8 und die Standartabweichung verhält sich ähnlich wie bei der linearen Regression.

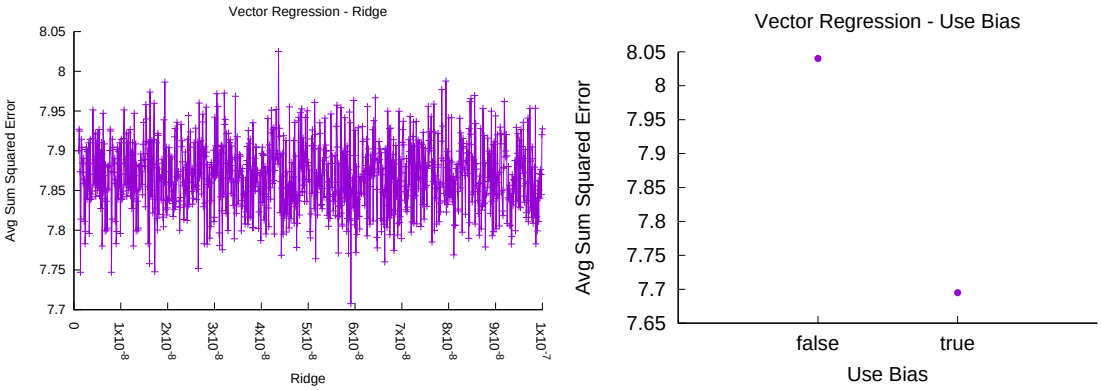

Abbildung 6.13: Links: durchschnittlicher Summierte Squared Error des Ridge (1,0 E-9 – 1,0 E-7), für Si, Co und Ro, Vektor Lineare Regression. Rechts: durchschnittlicher Summierte Squared Error des Use Bias, für Si, Co und Ro, Vektor Lineare Regression

Model Nr.	SumSQ Error	Use Bias	Ridge
1	7,67	True	3,43E-8
2	7,62	True	4,47E-8
3	7,67	True	8,62E-8
4	7,75	True	2,65E-8
5	7,74	True	3,83E-8

Tabelle 6.6: Sum Squared Error und Parameter der Top 5 vector lineare Regressions Modelle

SVR

Bei der Regressions SVM werden als Kernel der poly und der rbf Kernel getestet. Es wird eine $\epsilon-$SVR verwendet. Für Si, Co und Ro wurden jeweils wieder einzelne Modelle erstellt und der Squared Error als Fehlermaß benutzt.

Im poly Kernel wird der Grad und p, die Größe des $\epsilon-$Bereiches, optimiert.

Im rbf Kernel wird das Gamma optimiert und das p bleibt auf 0,1.

Auch bei den SVR ist der rbf Kernel besser als der poly Kernel. Beim poly Kernel erzielt der Degree von 3 und p Werte unter 0,5 für den poly Kernel die Besten Ergebnisse.

Der Gamma Wert Verhält sich für die Koordinaten und über den getesteten Bereich sehr verschieden. Für Co sind geringe Gamma Werte gut und für Ro erzielt 0,1 den besten Fehler.

Die Modelle verhalten sich sowohl im Summierten Squared Error, sowie im Squared Error und der Standartabweichung, der Koordinaten, sehr unterschiedlich. Modell 2 (Vergleiche Tabelle 6.7) hat einen guten Summierten Squared Error und eine geringe Standartabweichung in den Parametern.

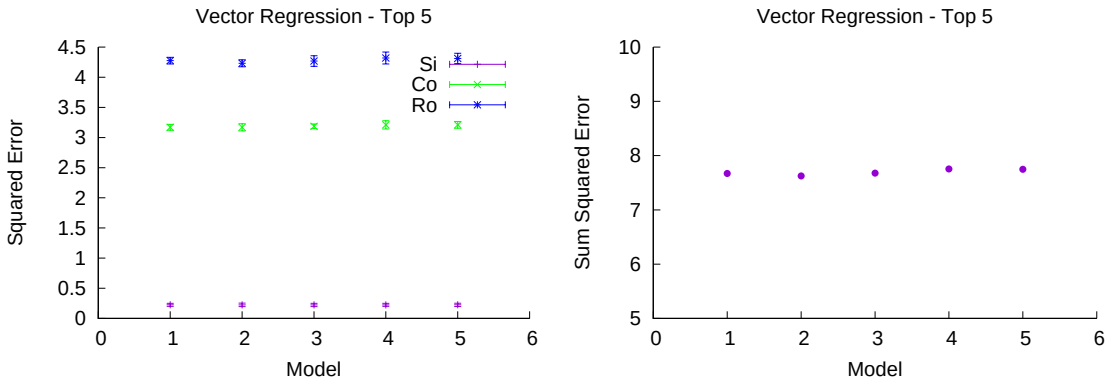

Abbildung 6.14: Links: Squared Error der Top 5 vector lineare Regressions Modelle, mit Betrachtung der Standartabweichung. Rechts: Sum Squared Error der Top 5 vector lineare Regressions Modelle

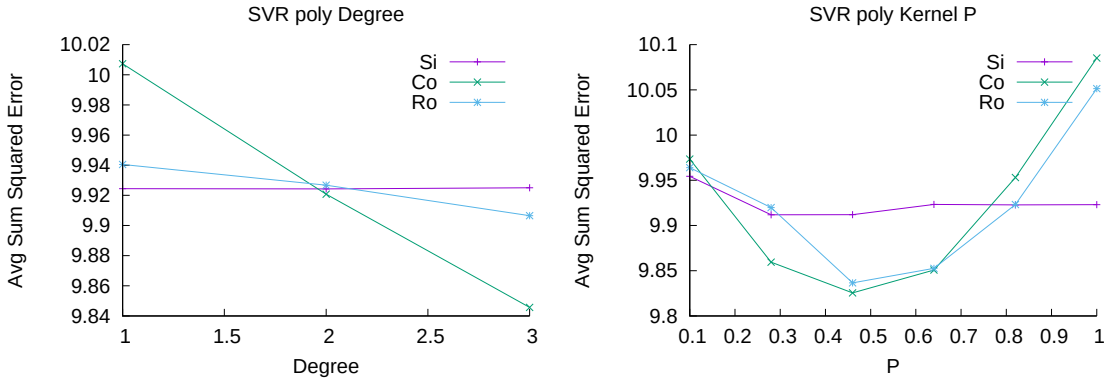

Abbildung 6.15: Links: durchschnittlicher Summierten Squared Error des Degree (1-3), für Si, Co und Ro, SVR. Rechts: durchschnittlicher Summierten Squared Error des p (0.1 – 1), für Si, Co und Ro, poly Kernel SVR

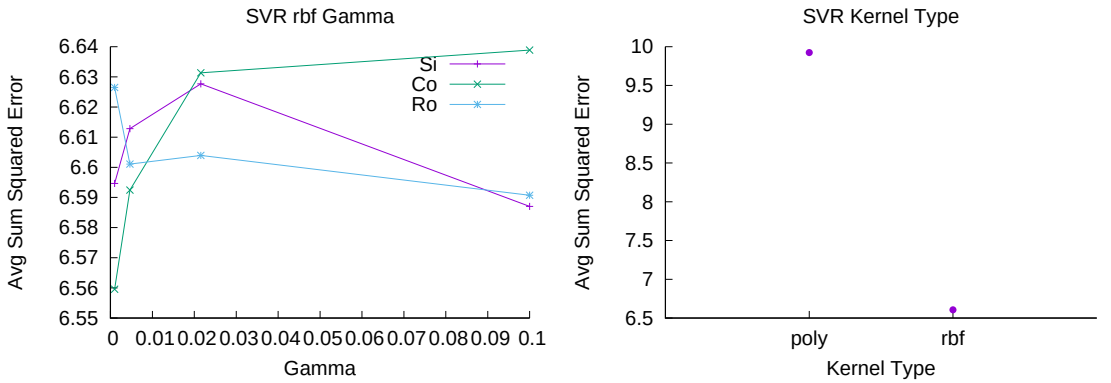

Abbildung 6.16: Links: durchschnittlicher Summierter Squared Error des Gamma (0.001 – 0.1), für Si, Co und Ro, rbf Kernel SVR. Rechts: durchschnittlicher Summierter Squared Error des Kernel Types, für Si, Co und Ro, SVR

Model Nr.	SumSQ Error	Kernel Type	Si γ	Co γ	Ro γ
1	7,36	rbf	0,001	0,0046	0,001
2	6,72	rbf	0,001	0,001	0,0046
3	7,76	rbf	0,022	0,022	0,022
4	8,31	rbf	0,022	0,001	0,1
5	8,14	rbf	0,001	0,001	0,1

Tabelle 6.7: Summierter Squared Error und Parameter der Top 5 SVR Modelle mit $p = 0,1$

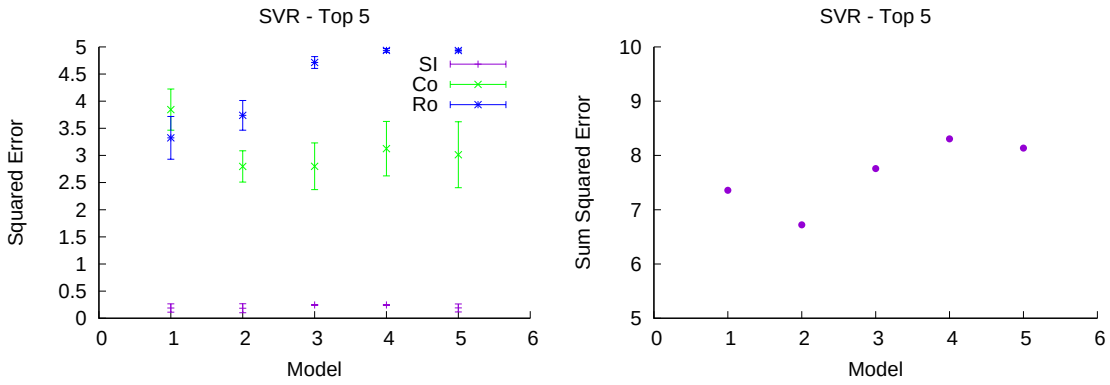

Abbildung 6.17: Links: Squared Error der Top 5 SVR Modelle, mit Betrachtung der Standartabweichung. Rechts: Sum Squared Error der Top 5 SVR Modelle

Kapitel 7

Fazit

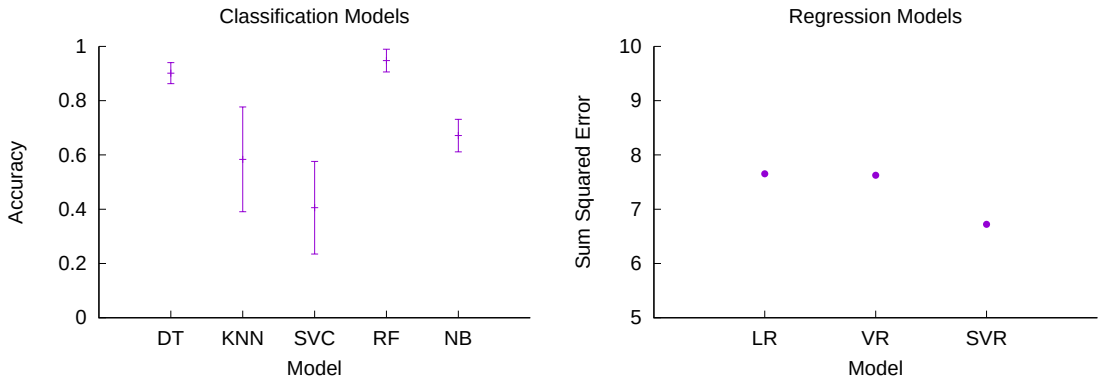

Abbildung 7.1: Links: Vergleich Decision Tree, kNN, SVC, Random Forest und Naive Bayes nach Accuracy, mit Betrachtung der Standartabweichung. Rechts: Vergleich der Regressions Modelle, Lineare Regression, Vector Lineare Regression und SVR über den Summierten Quadratischen Fehler.

In der Arbeit wurde untersucht, ob die Positionen von Lagereinheiten in einem Lager, anhand von Sensordaten mit Hilfe von maschinellen Lernverfahren vorhergesagt werden können.

Die Ergebnisse der Experimente haben das Folgende gezeigt:

KNN und SVC haben von den Klassifikationsmodellen die höchste Standartabweichung, und erzielen dabei keine gute Accuracy. Die Decision Trees und Random Forest haben beide eine gute Accuracy und eine geringe Standartabweichung. Naive Bayes hat keine hohe Standartabweichung, allerdings auch keine besonders gute Accuracy.

Bei den Regressionsmodellen erzielte die SVR den kleinsten Fehler. Zwischen linearer Regression und Vektor Regression ist kaum ein Unterschied im Fehler zu erkennen.

Dabei ist zu beachten, dass sowohl eine geringe Standartabweichung, als auch eine gute Accuracy bzw. ein kleiner summierter Mean Squared Error wichtig ist. Andere Prozesse in Lagern sind darauf angewiesen die richtige Position der zu benutzenden Lagereinheit zu kennen.

Von den untersuchten Modellen waren die Random Forests am geeignetsten, was jedoch auf realitätsnahen Experimenten geprüft werden muss.

Wie bei den OS-ELM [11] beschrieben, gibt es eine Differenz zwischen der Phase, in der die Daten gesammelt werden und dem Live-Betrieb. Dies wurde in den PhyNode Experimenten nicht beachtet, sollte in weiteren Ansätzen jedoch bedacht werden.

Auch wenn die mit Rapidminer erstellten Modelle zu groß sind, lassen sich mit anderen Implementierungen der gewählten Verfahren sehr wohl geeignete Modelle für die Verwendung auf ressourcenbeschränkter Hardware erstellen. Einige der erstellten Modelle erzielten durchaus gute Resultate, wodurch eine sinnvolle Lösung des Problems möglich scheint.

Anhang A

Weitere Informationen

Variable	Belegung
X	Eingabe
Y	Ergebnis
T	Trainingsdaten
$\hat{Y}$	Vorhergesagtes Y
$x_i, ..., x_n$	Beispiele
X_j	j-tes Attribut
$\hat{G}$	Vorhergesagte Klasse
σ	Varianz
μ	Mittelwert
σ^2	Standartabweichung
si,co,ro	Seite, Spalte, Reihe als Koordinaten
$\hat{si}$,$\hat{co}$,$\hat{ro}$	Vorhergesagte Seite, Spate und Reihe

Anhang B

Datenträger

Abbildungsverzeichnis

Literaturverzeichnis

[1] Breiman, L, JH Friedman und RA Olshen: *stone, CJ (1984).* Classification and regression trees, 85.

[2] Falkenberg, Robert, Mojtaba Masoudinejad, Markus Buschhoff, Aswin Karthik Ramachandran Venkatapathy, Daniel Friesel, Michael ten Hompel, Olaf Spinczyk und Christian Wietfeld: *PhyNetLab: An IoT-based warehouse testbed.* In: *Computer Science and Information Systems (FedCSIS), 2017 Federated Conference on*, Seiten 1051–1055. IEEE, 2017.

[3] Hastie, Trevor, Robert Tibshirani und Jerome Friedman: *The elements of statistical learning: data mining, inference and prediction.* Springer, 2 Auflage, 2009.

[4] Liu, Hui, Houshang Darabi, Pat Banerjee und Jing Liu: *Survey of wireless indoor positioning techniques and systems.* IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 37(6):1067–1080, 2007.

[5] Madigan, David, E Einahrawy, Richard P Martin, W-H Ju, Parameshwaran Krishnan und AS Krishnakumar: *Bayesian indoor positioning systems.* In: *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, Band 2, Seiten 1217–1227. IEEE, 2005.

[6] Masoudinejad, Mojtaba, Aswin Karthik Ramachandran Venkatapathy, David Tondorf, Danny Heinrich, Robert Falkenberg und Markus Buschhoff: *Machine Learning Based Indoor Localisation Using Environmental Data in PhyNetLab Warehouse.* In: *Smart SysTech 2018; European Conference on Smart Objects, Systems and Technologies*, Seiten 1–8. VDE, 2018.

[7] Quinlan, J. R.: *Induction of decision trees.* Machine Learning, 1(1):81–106, Mar 1986.

[8] Russell, S. und P. Norvig: *Artificial Intelligence: A Modern Approach.* Prentice Hall, Third Auflage, 2010.

[9] SFB 876: *PhyNetLab.* `http://phynetlab.com/PhyNetLab.pdf`. abgerufen am 03.12.2018.

[10] Zou, Han, Baoqi Huang, Xiaoxuan Lu, Hao Jiang und Lihua Xie: *A robust indoor positioning system based on the procrustes analysis and weighted extreme learning machine.* IEEE Transactions on Wireless Communications, 15(2):1252–1266, 2016.

[11] Zou, Han, Hao Jiang, Xiaoxuan Lu und Lihua Xie: *An online sequential extreme learning machine approach to WiFi based indoor positioning.* In: *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, Seiten 111–116. IEEE, 2014.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 3. Dezember 2018

Sabrina Schawohl