

**Entwicklung und Analyse von Algorithmen für die
automatische Indoor-Ortung basierend auf WiFi-Fingerprints**

Bachelorarbeit

Name des Studiengangs

Angewandte Informatik

Fachbereich 4

vorgelegt von

Friedrich Völkers

Datum:

Berlin, 12.08.2024

Erstgutachter: Prof. Dr. Alexander Huhn

Zweitgutachter: Prof. Dr.-Ing. Thomas Schwotzer

Inhaltsverzeichnis

1 Einleitung	1
1.1 Motivation und Zielsetzung	1
1.2 Aufbau der Arbeit	1
2 Grundlagen	2
2.1 WiFi-Fingerprinting	2
2.2 Indoor-Ortung: Offline- und Online-Phase	2
2.3 SSID, BSSID und RSSI	2
2.3.1 SSID	2
2.3.2 BSSID	2
2.3.3 RSSI	2
2.4 Verhalten von RSSI-Werten in Bezug auf Entfernung	3
3 Algorithmen und Methoden	4
3.1 K-Nearest Neighbors (KNN)	4
3.1.1 Algorithmusbeschreibung	4
3.1.2 Distanzmetriken (euklidisch, Sorensen)	4
3.1.3 Gewichtung (uniform vs. distance)	6
3.1.4 Parameter: Anzahl der Nachbarn	6
3.2 Support Vector Machines (SVM)	7
3.2.1 Algorithmusbeschreibung	7
3.2.2 Kernel (linear, RBF)	7
3.2.3 Parameter: Regularisierungsparameter C und Kernel-Parameter gamma .	8
3.3 Random Forest	8
3.3.1 Algorithmusbeschreibung	9
3.3.2 Parameter: Anzahl der Bäume (n_estimators)	9

4 Systemarchitektur und Implementierung	10
4.1 Architekturübersicht (App, API, Datenbank)	10
4.1.1 Implementierung der Android App (BVG Detection)	10
4.1.2 Implementierung der API	10
4.2 Datenbankstruktur (MariaDB, SQLite)	11
4.2.1 Tabelle <i>rooms</i>	11
4.2.2 Tabelle <i>measurements</i>	13
4.2.3 Tabelle <i>routers</i>	13
4.2.4 Tabelle <i>measurement_router</i>	13
4.2.5 Validierung und Vermeidung von Duplikaten	14
4.2.6 MariaDB auf der VM	14
4.2.7 SQLite auf den mobilen Geräten	14
4.3 Implementierung der Android App (BVG Detection)	14
4.3.1 Wiederinbetriebnahme der App	14
4.3.2 Gerätekompatibilität und allgemeine Informationen	14
4.3.3 Aufnahme von WiFi-Fingerprints	15
4.3.4 Datenaustausch	15
4.4 Bluetooth-Datenaustausch zwischen Geräten	15
4.4.1 Standortbestimmung	16
4.5 Implementierung der API und Datenbank	16
4.5.1 API-Endpunkte und Kommunikation	16
4.5.2 Code-Integration des ESP32 für die Standortbestimmung	16
5 Datenerhebung und -analyse	17
5.1 Datensammlung und -aufbereitung	17
5.1.1 Beschreibung der gesammelten Daten	17
5.1.2 Methodik zur Untersuchung der Algorithmen	17
5.1.3 Detaillierte Beschreibung des Testprogramms	17

5.2	Vergleich der Algorithmen und Parameter	19
5.2.1	Untersuchung der Genauigkeit in Abhangigkeit von Parametern	19
5.2.2	Strategien zum Umgang mit fehlenden Werten	23
5.2.3	Einfluss der verwendeten Router	24
5.2.4	Filterung von RSSI-Werten	25
5.2.5	Skalierung von RSSI-Werten	27
5.3	Erweiterte Untersuchungen	30
5.3.1	Einfluss der Anzahl der Fingerprints auf die Genauigkeit	30
5.3.2	Verbesserung der Vorhersagegenauigkeit durch Fingerprints außerhalb der Raume	30
6	Fazit und Ausblick	32
6.1	Zusammenfassung der Ergebnisse	32
6.2	Diskussion der Ergebnisse	32
6.3	Ausblick auf zukunftige Arbeiten	32
A	Erganzende Informationen	I
A.1	Zusatzliche Daten	I

Code Beispiele

5.1 .yaml Konfigurationsdatei 19

Abbildungsverzeichnis

2.1	RSSI vs. Distance	3
3.1	Euclidean vs. Sorensen distance metrics	6
3.2	SVM	7
3.3	Distance vs. Uniform weights	8
4.1	Datenbankstruktur	12
4.2	Diagramm der Datenbanktabellen und ihrer Beziehungen	12
5.1	Beispielabbildung 1	17
5.2	Distance vs. Uniform weights	19
5.3	Scale vs. Auto bei SVM RBF	20
5.4	Vergleich der Parameter in Abhangigkeit der Anzahl der Messungen	21
5.5	Vergleich der durchschnittlichen Genauigkeit der Parameter	22
5.6	Vergleich der Genauigkeit in Abhangigkeit der Strategie zum Umgang mit fehlenden Werten	24
5.7	Auswahl der Router	25
5.8	Vergleich der Genauigkeit in Abhangigkeit des Schwellenwerts fur die Anzahl der Messungen eines Routers	26
5.9	Router Rssi Threshold	27
5.10	Value Scaling Strategies	28
5.11	KNN Uniform vs. Distance im Durchschnitt	28
5.12	KNN Uniform vs. Distance pro Raum	29
5.13	Skalierungsstrategien nach Algorithmus und Parameter pro Anzahl Messungen pro Raum	30
5.14	Ergebnisse unter Betrachtung von Messungen außerhalb der Rume	31

Tabellenverzeichnis

3.1	RSSI-Werte für die drei Router und die Räume	4
3.2	Berechnete Distanzen für die gegebenen Wertepaare	5

1 Einleitung

1.1 Motivation und Zielsetzung

1.2 Aufbau der Arbeit

2 Grundlagen

Quelle 1: Access Points Service Set Identifier (SSID) for Localization and Tracking
Quelle 2: RSSI-Based Indoor Localization With the Internet of Things
Quelle 3: Survey on Indoor localization System and Recent Advances of WIFI Fingerprinting Technique

2.1 WiFi-Fingerprinting

Gute Grundlagenquelle: Overview of WiFi fingerprinting-based indoor positioning -> Was ist Indoor Ortung, was gibt es für Möglichkeiten, Offline/Online Phase, was für Modelle gibt es?

2.2 Indoor-Ortung: Offline- und Online-Phase

2.3 SSID, BSSID und RSSI

2.3.1 SSID

Der Service Set Identifier (SSID) ist ein eindeutiger Bezeichner, der ein drahtloses Netzwerk kennzeichnet. SSIDs sind alphanumerische Zeichenfolgen, die vom Netzwerkadministrator festgelegt werden. Die SSID ermöglicht es Nutzern, zwischen verschiedenen WiFi-Netzwerken zu unterscheiden und das gewünschte Netzwerk auszuwählen. In einem Indoor-Lokalisierungsszenario kann die SSID als Landmarke für Access Points genutzt werden, um die Position eines Benutzers zu bestimmen (Quelle 1, Kapitel 1, Seite 5460).

2.3.2 BSSID

Der Basic Service Set Identifier (BSSID) ist eine eindeutige Kennung für jeden Access Point innerhalb eines WiFi-Netzwerks. Diese Kennung besteht aus der MAC-Adresse des Access Points, die fest zugewiesen und unveränderlich ist. In Netzwerken mit mehreren Access Points, die dieselbe SSID nutzen, ermöglicht die BSSID die genaue Unterscheidung der einzelnen Access Points. Dies ist besonders wichtig für die Lokalisierung und Netzwerkverwaltung, da es die Identifikation und Verfolgung spezifischer Access Points ermöglicht (Quelle 1, Kapitel 1, Seite 5460).

2.3.3 RSSI

Der Received Signal Strength Indicator (RSSI) ist ein Maß für die Stärke des empfangenen WiFi-Signals an einem bestimmten Punkt. RSSI-Werte werden von der Netzwerkkarte des Geräts gemessen und in der App aufgezeichnet. Diese Werte spielen eine entscheidende Rolle bei

der Erstellung von Fingerprints für die Indoor-Lokalisierung, da die Signalstärke als Indikator für die Entfernung zwischen dem Gerät und dem Access Point genutzt wird. Schwächere RSSI-Werte deuten dabei auf eine größere Entfernung hin (Quelle 1, Kapitel 1, Seite 5464).

2.4 Verhalten von RSSI-Werten in Bezug auf Entfernung

Quelle 2:

Die RSSI-Werte nehmen mit zunehmender Distanz zwischen Sender und Empfänger ab, und können mit dem Pfadverlustmodell beschrieben werden. Das Pfadverlustmodell beschreibt den Zusammenhang zwischen der Entfernung und der Signalstärke, und wird durch die folgende Gleichung dargestellt:

$$RSSI = -10n \log_{10}(d) + C \quad (2.1)$$

wobei n der Pfadverlustexponent ist, der je nach Umgebung variiert, d die Distanz zwischen Sender und Empfänger ist und A der RSSI-Wert in einem Meter Entfernung vom Sender ist.

Umgebung der Quelle: Im Raum sind viele Computer und eine große Anzahl an WiFi- und BLE-Geräten, Ein Forschungslabor mit den Abmessungen 10,8 m x 7,3 m.

Ergebnisse: n = 2.013, C = -49,99 dBm

Die Path-Loss-Modell-Gleichung mit den Parametern $n = 2.013$ und $C = -49.99$ ist:

In Abbildung 2.1 ist der Zusammenhang zwischen RSSI und Entfernung dargestellt. Dieser Abschnitt dient dazu um eine Vorstellung davon zu bekommen, wie sich RSSI-Werte in Bezug auf die Entfernung verhalten.

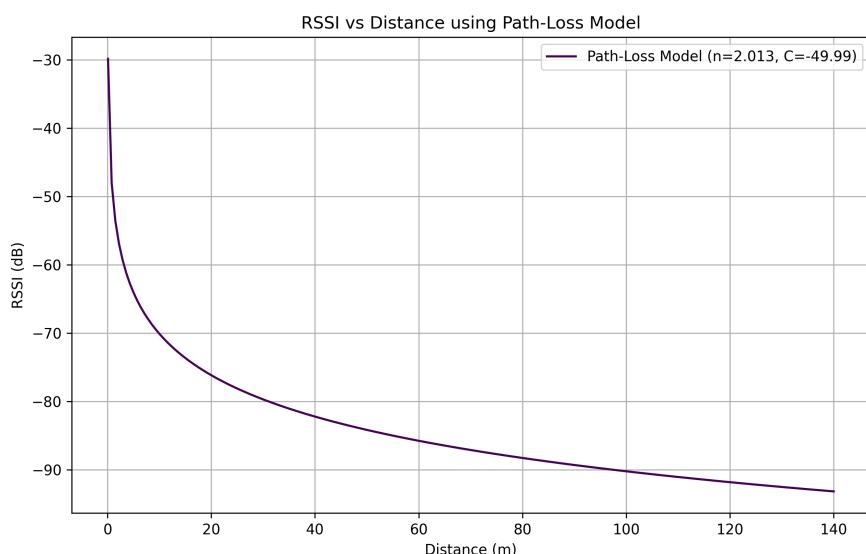


Abbildung 2.1: RSSI vs. Distance

3 Algorithmen und Methoden

Router	Raum 1	Raum 2	Raum 3	Unbekannt (1)
1	-73.67 dBm	-91.12 dBm	-69.37 dBm	-73.51 dBm
2	-104.99 dBm	-64.47 dBm	-89.65 dBm	-103.41 dBm
3	-67.03 dBm	-105.38 dBm	-88.40 dBm	-70.35 dBm

Tabelle 3.1: RSSI-Werte für die drei Router und die Räume

Quellen für Auswahl

- <https://arxiv.org/html/2111.14281> -> KNN, SVM
- A Wireless Fingerprint Location Method Based on Target Tracking (Kapitel 3, Seite 3)
-> KNN, SVM, Random Forest
- <https://onlinelibrary.wiley.com/doi/full/10.1155/2017/6268797> -> KNN, SVM, Random Forest. In Table 2 stehen die Genauigkeiten der Algorithmen

3.1 K-Nearest Neighbors (KNN)

Quellen:

Quelle 1: <https://www.ibm.com/de-de/topics/knn>

Quelle 4: Comprehensive analysis of distance and similarity measures for Wi-Fi fingerprinting indoor positioning systems

Quelle 5: Hechenbichler, Schliep: - Weighted k-Nearest-Neighbor Techniques and Ordinal Classification

3.1.1 Algorithmusbeschreibung

Quelle 1: Der k-nearest neighbor (KNN) Algorithmus ist ein überwachter Lernklassifikator, der auf dem Konzept der Nähe basiert und zur Lösung von Klassifikations- und Regressionsproblemen verwendet werden kann. Der Algorithmus funktioniert so, dass ein Datenpunkt mit den vorhandenen Datenpunkten in den Trainingsdaten verglichen wird und die Distanz zu jedem Datenpunkt berechnet wird. Basierend auf diesen Distanzen werden die k Datenpunkte ausgewählt, die den kleinsten Abstand haben. Aus diesen k Datenpunkten wird dann die Klasse bestimmt, die am häufigsten vertreten ist. Hierbei reicht bereits eine relative Mehrheit aus (wenn z.B. 4 Klassen vertreten sind, kann ein Anteil von mehr als 25 % ausreichend sein).

3.1.2 Distanzmetriken (euklidisch, Sorensen)

Quelle 1: Für die Berechnung der Distanzen können verschiedene Distanzmetriken verwendet werden. Die am häufigsten verwendete Distanzmetrik ist der euklidische Abstand. In dieser

Arbeit wurde entschieden, sowohl die euklidische Distanz als auch die Sorensen-Distanz zu verwenden. Bei der euklidischen Distanz wird das Quadrat der Abstände (Betrag der Differenz) zwischen zwei Werten gebildet, über alle Wertepaare aufsummiert und abschließend die Quadratwurzel dieser Summe gezogen (siehe 3.1).

$$\text{distance}_{\text{euclidean}}(P, Q) = \sqrt{\sum_{i=1}^d (P_i - Q_i)^2} \quad (3.1)$$

Quelle 4: Bei der Sorensen-Distanzfunktion werden die Abstände der Datenpunkte aufsummiert und durch die Summe der Wertepaare zweier Datenpunkte geteilt (siehe 3.2). Grund für die Wahl dieser beiden Metriken: Die Arbeit "Comprehensive analysis of distance and similarity measures for Wi-Fi fingerprinting indoor positioning systems" konnte damit gute Ergebnisse erzielen. Der euklidische Abstand ist weit verbreitet und entspricht auch der bisherigen Implementierung in der App.

$$\text{distance}_{\text{sorensen}}(P, Q) = \frac{\sum_{i=1}^d |P_i - Q_i|}{\sum_{i=1}^d (P_i + Q_i)} \quad (3.2)$$

In Abbildung 3.1 sind die Heatmaps für die beiden Distanzmetriken dargestellt.

Damit die Unterschiede zwischen den beiden Distanzmetriken besser sichtbar sind, wurden die Distanzen zwischen allen möglichen Wertepaaren zwischen 0 und -100 berechnet und in Abbildung 3.1 dargestellt. Wie zu erkennen ist, sind diese beiden symmetrisch zu der Geraden zwischen den beiden Punkten (0, 0) und (-100, -100). Wie zu erkennen ist, sind die Werte für die euklidische Distanz auch symmetrisch zu der Geraden zwischen den beiden Punkten (0, -100) und (-100, 0). Dementsprechend ist die Distanz alleine von der Differenz der beiden Werte abhängig. Bei der Sørensen-Distanz muss bei kleineren Werten der Abstand größer sein, um die gleiche Distanz zu erreichen wie bei kleineren Werten.

Um dies zu verdeutlichen, betrachten wir die Berechnung der Distanzen für zwei Beispielwertepaare:

Gegeben:

- Wertepaar 1: $P = -30 \text{ dBm}$, $Q = -40 \text{ dBm}$
- Wertepaar 2: $P = -70 \text{ dBm}$, $Q = -80 \text{ dBm}$

Wertepaar	Sørensen-Dice Distanz	Euklidische Distanz
-30 dBm, -40 dBm	-0.142857	10
-70 dBm, -80 dBm	-0.066667	10

Tabelle 3.2: Berechnete Distanzen für die gegebenen Wertepaare

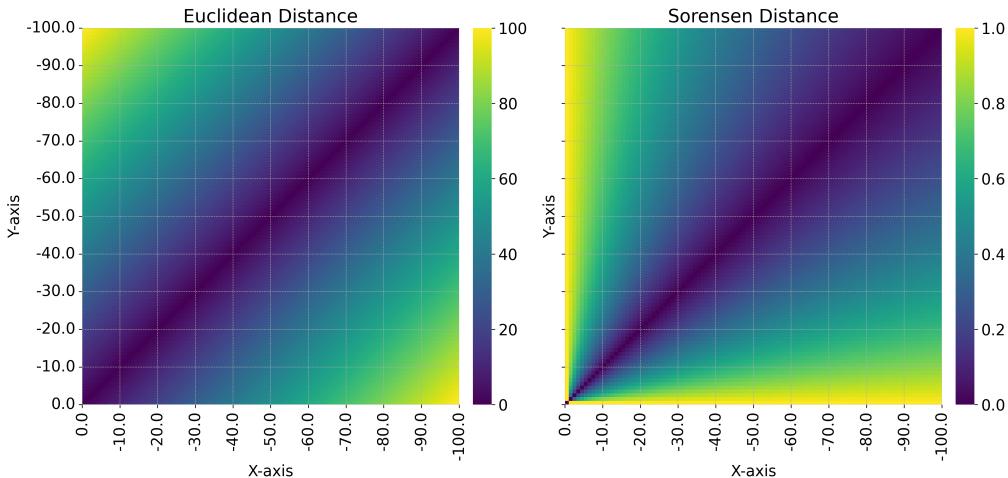


Abbildung 3.1: Euclidean vs. Sorensen distance metrics

3.1.3 Gewichtung (uniform vs. distance)

Quelle 5:

- Die Erweiterung basiert auf der Idee, dass Beobachtungen im Trainingsset, die besonders nahe an der neuen Beobachtung (y, x) liegen, ein höheres Gewicht in der Entscheidungsfindung erhalten sollten als weiter entfernte Nachbarn.
- Im klassischen kNN beeinflussen nur die k nächsten Nachbarn die Vorhersage.
- Der Einfluss jedes der k nächsten Nachbarn ist gleich, obwohl die individuelle Ähnlichkeit zu (y, x) stark variieren kann.
- Um dieses Ziel zu erreichen, müssen die Distanzen, die bei der Suche nach den nächsten Nachbarn im ersten Schritt verwendet werden, in Ähnlichkeitsmaße umgewandelt werden, die als Gewichte verwendet werden können.

3.1.4 Parameter: Anzahl der Nachbarn

Quelle 1: Der Parameter k legt fest, wie viele nächste Nachbarn für die Klassifizierung ausgewählt werden. Für $k = 1$ wird nur der nächste Nachbar ausgewählt, wodurch kein Mehrheitsvotum stattfindet. Bei kleinen Werten für k kann eine hohe Varianz und eine geringe Verzerrung auftreten, während bei größeren Werten die Varianz geringer und die Verzerrung höher ist. Die Auswahl von k hängt stark von den vorhandenen Trainingsdaten ab. Bei Daten mit Ausreißern und Rauschen wird empfohlen, höhere Werte für k zu wählen, da in diesen Fällen bessere Ergebnisse erzielt werden. Zudem wird empfohlen, ungerade Werte für k zu wählen, um Unentschieden beim Mehrheitsvotum zu vermeiden.

3.2 Support Vector Machines (SVM)

In Abbildung 3.2 ist ein SVM dargestellt.

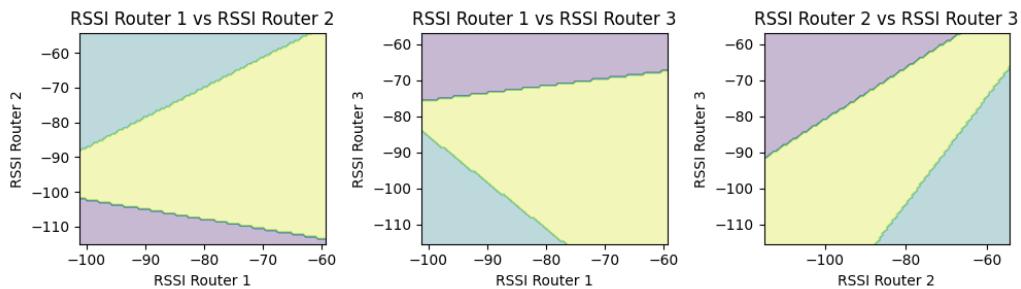


Abbildung 3.2: SVM

Quellen:

Quelle 6: <https://www.bigdata-insider.de/was-ist-eine-support-vector-machine-a-880134>

Quelle 7: <https://www.ibm.com/topics/support-vector-machine>

Quelle 8: <https://wires.onlinelibrary.wiley.com/doi/full/10.1002/wics.49>

Quelle warum gerade RBF: An Indoor Localization of WiFi Based on Support Vector Machines

3.2.1 Algorithmusbeschreibung

Quelle 7: Der Support Vector Machine (SVM) Algorithmus ist ein überwachter maschineller Lernalgorithmus, der verwendet wird, um Daten zu klassifizieren. Die Klassifizierung erfolgt, indem die optimale Trennlinie – die sogenannte Hyperebene – zwischen den Datenpunkten gefunden wird. Diese Hyperebene wird so positioniert, dass der maximale Abstand zwischen den Datenpunkten der beiden Klassen erreicht wird. SVM kann sowohl mit linear trennbaren Daten als auch mit nicht linear trennbaren Daten arbeiten. Bei nicht linearen Daten wird der sogenannte Kernel-Trick angewendet, bei dem die Daten mithilfe einer Kernel-Funktion in einen höherdimensionalen Raum transformiert werden.

3.2.2 Kernel (linear, RBF)

Quelle 7: Bei der linearen Klassifizierung werden die Datenpunkte durch eine Linie oder Ebene getrennt und nicht transformiert. Bei nicht linearen SVMs wird zuerst die Kernel-Methode angewendet, und anschließend werden die Daten linear getrennt. In dieser Arbeit wurde entschieden, ein lineares und ein nicht lineares SVM zu vergleichen. Für das nicht lineare SVM wurde der RBF-Kernel gewählt, da in der Arbeit "Device-Free Presence Detection and Localization With SVM and CSI Fingerprinting" mit diesem Kernel die besten Ergebnisse unter den nicht linearen SVMs erzielt wurden.

3.2.3 Parameter: Regularisierungsparameter C und Kernel-Parameter gamma

Quelle 7: Mit dem Parameter C kann der Margin angepasst werden. Ein größerer Wert verengt den Margin für eine minimale Fehlklassifizierung und ein größerer Wert für C erweitert den Margin, sodass mehr Fehlklassifizierungen zugelassen werden. Gamma kontrolliert den Einfluss eines einzelnen Datenpunktes. Ein höherer Wert sorgt für eine kleinere Einflussreichweite, wodurch die Entscheidungsgrenzen enger werden. Der Parameter Gamma kann nur bei nicht linearen SVMs verwendet werden.

Quelle für die Auswahl der Parameter Werte: Semi-Supervised Classification by Low Density Separation

3.3 Random Forest

Gute quelle: WiFi Indoor Localization with CSI Fingerprinting-Based Random Forest -> Wie wichtig ist welcher Hyperparameter?

In Abbildung 3.3 ist ein Entscheidungsbaum dargestellt.

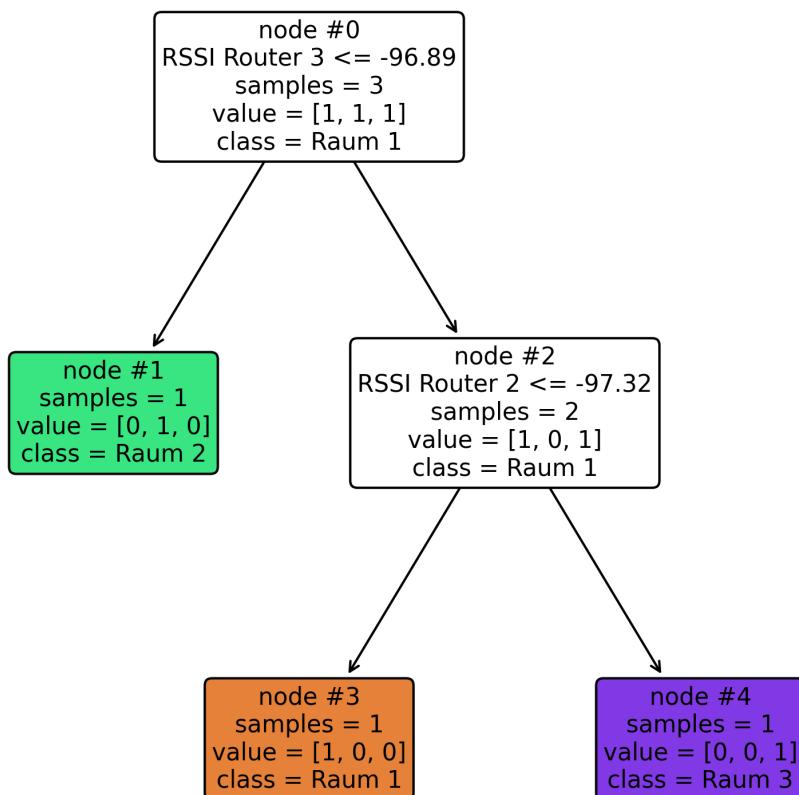


Abbildung 3.3: Distance vs. Uniform weights

Quelle:

Quelle 9: <https://builtin.com/data-science/random-forest-algorithm>

3.3.1 Algorithmusbeschreibung

Quelle 9: Der Random Forest Algorithmus ist ein überwachter Lernalgorithmus, der eine Menge an Entscheidungsbäumen erstellt. Ein Entscheidungsbaum besteht aus Knoten, welche eine Bedingung zu einem der Merkmale aus dem Datenset abfragen. Zum Beispiel sowas wie: Ist der RSSI-Wert von Access Point 1 größer gleich -80 dBm? Je nachdem wie diese Frage beantwortet wird, geht es dann zum nächsten Knoten usw. Am Ende jedes Strangs

3.3.2 Parameter: Anzahl der Bäume (n_estimators)

4 Systemarchitektur und Implementierung

4.1 Architekturübersicht (App, API, Datenbank)

4.1.1 Implementierung der Android App (BVG Detection)

- **App-Architektur:** Geben Sie eine Übersicht über die Architektur der Android-App und deren Hauptkomponenten.
- **Funktionalitäten:** Beschreiben Sie die Hauptfunktionen der App, einschließlich Fingerprint-Aufnahme und Raumvorhersage.
- **Technologien und Bibliotheken:** Erklären Sie die verwendeten Technologien und Bibliotheken zur Implementierung der App.

4.1.2 Implementierung der API

Die API wurde mithilfe des Frameworks Flask entwickelt und läuft auf der virtuellen Maschine

- **API-Design:** Die API wurde mit Flask entwickelt und dient der Verwaltung und Abfrage von WiFi-Fingerprints zur Indoor-Ortung. Die Hauptendpunkte sind:
 - `/measurements` (POST): Hinzufügen eines neuen Fingerprints.
 - `/measurements/batch` (POST): Hinzufügen mehrerer Fingerprints im Batch.
 - `/measurements/all` (GET): Abrufen aller gespeicherten Fingerprints.
 - `/measurements/reset` (POST): Zurücksetzen aller gespeicherten Fingerprints und zugehöriger Daten.
 - `/measurements/predict` (POST): Vorhersage des Raumes basierend auf empfangenen WiFi-Daten.
 - `/ping` (GET): Überprüfen, ob der Server läuft.
 - `/measurements/room_names` (GET): Abrufen aller Messungs-IDs mit ihren zugehörigen Raumnamen.

Jeder Endpunkt hat eine spezifische Funktion und dient entweder der Datenspeicherung, -abfrage oder -vorhersage.

- **Funktionalitäten:** Die API bietet folgende Hauptfunktionen:

- **Fingerprint-Speicherung:** Die Funktion `handle_add_measurement` ermöglicht das Hinzufügen eines neuen Fingerprints zur Datenbank. Hierbei werden Informationen wie Raumname, Geräte-ID, Zeitstempel und Routerdaten gespeichert.
- **Batch-Speicherung:** Mit der Funktion `handle_add_measurements_batch` können mehrere Fingerprints in einem Schritt hinzugefügt werden, was die Effizienz bei der Datenerfassung erhöht.

- **Raumvorhersage:** Die Funktion `handle_predict_room` verwendet verschiedene Machine Learning Algorithmen (KNN, Random Forest, SVM), um basierend auf empfangenen WiFi-Daten den wahrscheinlichsten Raum vorherzusagen. Hierbei werden verschiedene Parameter wie der Algorithmus, die Anzahl der Nachbarn (k-Wert) und Gewichtungsstrategien berücksichtigt.
- **Datenabfrage:** Über die Endpunkte `/measurements/all` und `/measurements/room_names` können alle gespeicherten Fingerprints sowie die zugehörigen Raumnamen abgerufen werden.
- **Daten-Reset:** Mit `handle_reset_measurements` können alle gespeicherten Fingerprints und zugehörige Daten zurückgesetzt werden, was besonders bei Testläufen nützlich ist.
- **Sicherheitsaspekte:** Die API berücksichtigt verschiedene Sicherheitsaspekte, um den Schutz der Daten zu gewährleisten:
 - **Authentifizierung und Autorisierung:** Um sicherzustellen, dass nur autorisierte Benutzer auf die API zugreifen und Änderungen vornehmen können, sollten Authentifizierungs- und Autorisierungsmechanismen wie API-Schlüssel oder OAuth implementiert werden. In der aktuellen Implementierung fehlt diese Funktionalität und sollte daher als zukünftige Verbesserung berücksichtigt werden.
 - **Datenvalidierung:** Bei der Verarbeitung eingehender Anfragen wird die Integrität der Daten überprüft. Fehlende oder fehlerhafte Daten führen zu Fehlermeldungen und die Anfragen werden abgelehnt (z.B. `handle_add_measurement` überprüft, ob alle erforderlichen Felder vorhanden sind).
 - **Logging und Monitoring:** Die API enthält Logging-Funktionalitäten, die Informationen über eingehende Anfragen und ausgehende Antworten protokollieren. Dies hilft bei der Überwachung der API-Nutzung und beim Debugging.
 - **Fehlerbehandlung:** Die API behandelt verschiedene Fehlerzustände, wie z.B. das Überschreiten der Zeitlimits für Anfragen oder Netzwerkprobleme, und liefert entsprechende Fehlermeldungen an den Client.

4.2 Datenbankstruktur (MariaDB, SQLite)

Es wurde eine relationale Datenbank gewählt, da viele Einträge wie Räume und Router mehrfach vorkommen und so die Größe der Datenbank nicht linear wächst mit jeder neuen Messung.

Die entwickelte Datenbankstruktur besteht aus vier zentralen Tabellen: `rooms`, `measurements`, `routers` und `measurement_router`. Diese Struktur wurde gewählt, um eine effiziente und flexible Speicherung sowie Verwaltung der für die Raumbestimmung erforderlichen Daten zu gewährleisten. Die einzelnen Tabellen und deren Beziehungen sind in Abbildung 4.1 dargestellt.

4.2.1 Tabelle `rooms`

Die Tabelle `rooms` speichert Informationen über alle aufgenommenen Räume. Sie enthält die folgenden Felder:

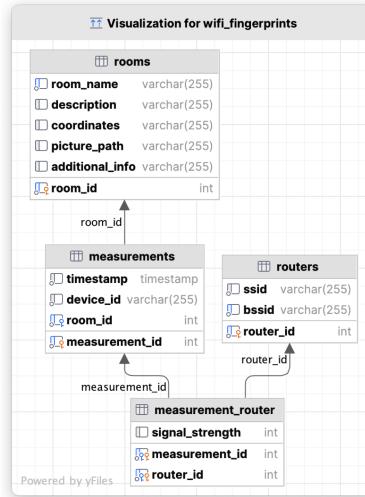


Abbildung 4.1: Datenbankstruktur

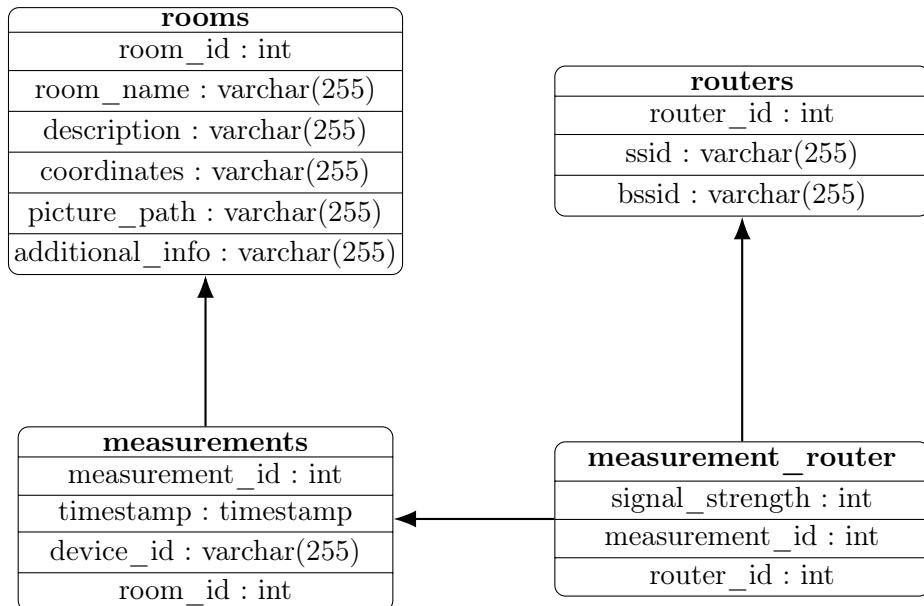


Abbildung 4.2: Diagramm der Datenbanktabellen und ihrer Beziehungen

- *room_id*: Primärschlüssel, der automatisch inkrementiert wird.
- *room_name*: Name des Raumes, der eindeutig sein muss.
- *description*: Beschreibung des Raumes.
- *coordinates*: Koordinaten des Raumes.
- *picture_path*: Pfad zu einem Bild des Raumes.
- *additional_info*: Weitere Informationen zum Raum.

Die Felder *description*, *coordinates*, *picture_path* und *additional_info* sind vorhanden, da die Datenbank auch auf mobilen Geräten läuft und die App diese Daten speichert. Dies ermöglicht eine umfassende Dokumentation und einfache Identifikation der Räume.

4.2.2 Tabelle *measurements*

In der Tabelle *measurements* werden alle Messungen gespeichert. Die Felder umfassen:

- *measurement_id*: Primärschlüssel, der automatisch inkrementiert wird.
- *timestamp*: Zeitstempel der Messung.
- *device_id*: Kennung des Geräts, das die Messung durchgeführt hat.
- *room_id*: Fremdschlüssel, der auf die *room_id* in der Tabelle *rooms* verweist.

Bei der Hinzufügung einer neuen Messung wird überprüft, ob der Raum bereits existiert und gegebenenfalls hinzugefügt. Ebenso wird überprüft, ob der Router bereits in der Tabelle *routers* vorhanden ist.

4.2.3 Tabelle *routers*

Die Tabelle *routers* enthält Informationen zu allen Routern, die bei den Messungen erfasst wurden:

- *router_id*: Primärschlüssel, der automatisch inkrementiert wird.
- *ssid*: SSID des Routers.
- *bssid*: BSSID des Routers, die eindeutig sein muss.

Die Eindeutigkeit der BSSID wird durch einen einzigartigen Schlüssel sichergestellt. Dies ist notwendig, um sicherzustellen, dass jeder Router nur einmal in der Datenbank gespeichert wird.

4.2.4 Tabelle *measurement_router*

Die Tabelle *measurement_router* stellt die Beziehung zwischen den Messungen und den Routern dar und speichert die Signalstärke:

- *measurement_id*: Fremdschlüssel, der auf die *measurement_id* in der Tabelle *measurements* verweist.
- *router_id*: Fremdschlüssel, der auf die *router_id* in der Tabelle *routers* verweist.
- *signal_strength*: Signalstärke, die bei der Messung erfasst wurde.

Die Primärschlüssel der Tabelle *measurement_router* bestehen aus der Kombination von *measurement_id* und *router_id*, um sicherzustellen, dass jede Kombination aus Messung und Router nur einmal gespeichert wird.

4.2.5 Validierung und Vermeidung von Duplikaten

Um zu verhindern, dass Messungen doppelt hinzugefügt werden, wird überprüft, ob bereits eine Messung mit derselben *device_id* und demselben *timestamp* existiert. Dies stellt sicher, dass jede Messung einzigartig ist und keine redundanten Daten in der Datenbank gespeichert werden.

Zusammenfassend bietet die gewählte Datenbankstruktur eine robuste Grundlage für die Speicherung und Verwaltung der Daten, die für die Raumbestimmung mittels WiFi Fingerprints erforderlich sind. Die Struktur ermöglicht eine effiziente Verarbeitung und Abfrage der Daten, was für die Echtzeitanwendung in mobilen Geräten essenziell ist.

4.2.6 MariaDB auf der VM

- **Datenbankdesign:** Beschreiben Sie das Design der MariaDB-Datenbank und die wichtigsten Tabellen.
- **Speicherung von Fingerprints:** Erklären Sie, wie Fingerprint-Daten in der MariaDB gespeichert und verwaltet werden.
- **Abfragen und API-Interaktion:** Diskutieren Sie die wichtigsten Abfragen und die Interaktion mit der API.

4.2.7 SQLite auf den mobilen Geräten

- **Datenbankdesign:** Beschreiben Sie das Design der SQLite-Datenbank und die wichtigsten Tabellen.
- **Lokal gespeicherte Daten:** Erklären Sie, welche Daten lokal auf den mobilen Geräten gespeichert werden und warum.
- **Synchronisation mit MariaDB:** Diskutieren Sie die Mechanismen zur Synchronisation der SQLite-Daten mit der MariaDB.

4.3 Implementierung der Android App (BVG Detection)

4.3.1 Wiederinbetriebnahme der App

TODO: Aus der ersten Version holen...

4.3.2 Gerätetkompatibilität und allgemeine Informationen

TODO: Welche Versionen werden unterstützt (Android/SDK...)

4.3.3 Aufnahme von WiFi-Fingerprints

TODO: Screenshots bei der Aufnahme von Fingerprints und mit der Liste der Fingerprints

4.3.4 Datenaustausch

4.4 Bluetooth-Datenaustausch zwischen Geräten

- **Funktionalität:** Der Bluetooth-Datenaustausch zwischen Geräten ermöglicht es, WiFi-Fingerprints direkt zwischen zwei Geräten auszutauschen, ohne eine zentrale Datenbank oder API verwenden zu müssen. Dies ist besonders nützlich, wenn eine schnelle und lokale Synchronisation der Daten erforderlich ist, beispielsweise in Umgebungen ohne Internetzugang. Die Bluetooth-Kommunikation ermöglicht es, dass Gerät A Fingerprints an Gerät B senden kann und umgekehrt. Dies erweitert die Flexibilität und Anwendungsmöglichkeiten der App erheblich.
- **Technische Umsetzung:** Die technische Umsetzung des Bluetooth-Datenaustauschs in der Android-App erfolgt durch die Verwendung der BluetoothAdapter- und BluetoothSocket-Klassen von Android. Der Prozess beinhaltet die folgenden Schritte:
 - **Initialisierung:** Der BluetoothAdapter wird initialisiert, um das Bluetooth-Modul des Geräts zu steuern. Falls Bluetooth nicht aktiviert ist, wird der Benutzer aufgefordert, es zu aktivieren.
 - **Paaren von Geräten:** Eine Liste der gekoppelten Bluetooth-Geräte wird angezeigt, und der Benutzer kann ein Gerät auswählen, mit dem eine Verbindung hergestellt werden soll.
 - **Verbindung herstellen:** Die App startet einen Server-Socket (AcceptThread) auf einem Gerät, der auf eingehende Verbindungen wartet. Ein anderes Gerät kann sich mit diesem Server-Socket verbinden (ConnectThread).
 - **Datenübertragung:** Nachdem die Verbindung hergestellt ist, wird ein ConnectedThread gestartet, der die Datenübertragung verwaltet. Die Fingerprint-Daten werden in JSON-Format konvertiert und in Blöcken über die Bluetooth-Verbindung gesendet. Der Empfang und das Zusammenfügen der Daten erfolgt ebenfalls in diesem Thread.
 - **Verarbeitung empfangener Daten:** Die empfangenen Daten werden geparsst und in die lokale SQLite-Datenbank eingefügt oder aktualisiert. Der BluetoothHelper kümmert sich um die Verarbeitung der empfangenen JSON-Daten und deren Integration in die bestehende Datenbankstruktur.
- **Anwendungsfälle:** Der Bluetooth-Datenaustausch bietet mehrere Vorteile und Anwendungsfälle:
 - **Schnelle lokale Synchronisation:** Ideal für Umgebungen ohne Internetzugang, da die Daten direkt zwischen den Geräten ausgetauscht werden können.
 - **Datensicherung und -wiederherstellung:** Ein Benutzer kann seine Fingerprint-Daten auf ein anderes Gerät übertragen, um eine Datensicherung zu erstellen oder Daten wiederherzustellen.

- **Kollaborative Datensammlung:** Mehrere Benutzer können parallel Fingerprint-Daten sammeln und diese anschließend miteinander teilen, um eine umfassendere Datenbasis zu schaffen.
- **Redundanz und Ausfallsicherheit:** Durch den Austausch der Daten zwischen Geräten wird sichergestellt, dass die Daten auch bei einem Geräteausfall nicht verloren gehen.

4.4.1 Standortbestimmung

TODO: Screenshots mit den Einstellungen, dem erfolgreichen Ergebnis

4.5 Implementierung der API und Datenbank

Hier kommt eig. der Teil hin der aktuell in Implementierung der API steht.

4.5.1 API-Endpunkte und Kommunikation

TODO: Liste aller Endpunkte..

4.5.2 Code-Integration des ESP32 für die Standortbestimmung

TODO: Code des ESP32, MicroPythin ist toll

5 Datenerhebung und -analyse

5.1 Datensammlung und -aufbereitung

5.1.1 Beschreibung der gesammelten Daten

- **Datenerhebungsmethoden:** Erklären Sie, wie die Daten gesammelt wurden (z.B. welche Geräte, Orte und Zeitpunkte).
- **Datensätze und Umfang:** Beschreiben Sie den Umfang und die Struktur der gesammelten Datensätze.
- **Datenqualität und -validität:** Diskutieren Sie die Qualität und Validität der Daten.

Es wurden insgesamt 407 Messungen in 35 Räumen durchgeführt. Die Messungen wurden zwischen dem 8. Mai 2024 und dem 31. Mai 2024 durchgeführt. Im Schnitt wurden bei jeder Messung 12,8 Router erfasst. Im Schnitt wurden 11,6 Messungen pro Raum durchgeführt. Die Anzahl der Messungen pro Raum variiert zwischen 2 und 19. Es wurden insgesamt 318 Router erfasst.

Die Anzahl der Messungen pro Raum sind in Abbildung 5.1 dargestellt.

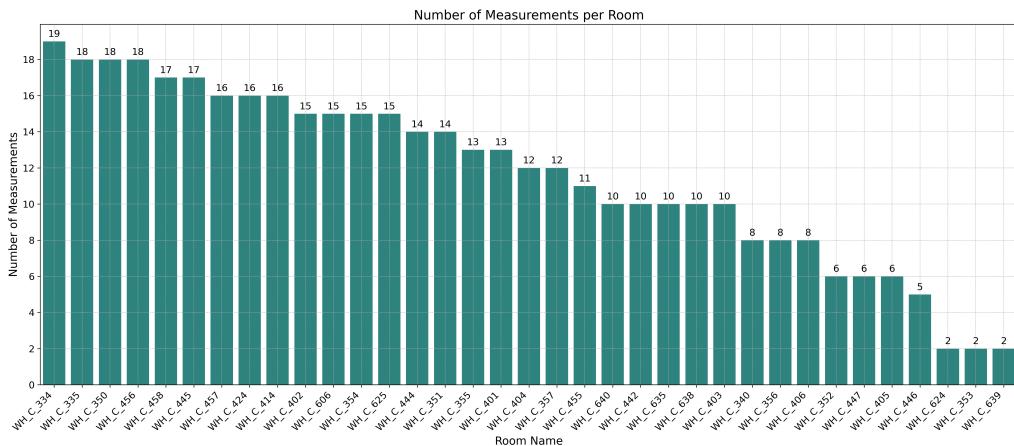


Abbildung 5.1: Beispielabbildung 1

- Verwendete Geräte (Doogee Y8, Google Pixel 8, Samsung Galaxy A?!)

5.1.2 Methodik zur Untersuchung der Algorithmen

5.1.3 Detaillierte Beschreibung des Testprogramms

- **Programmübersicht:** Das Testprogramm dient dazu, die Leistungsfähigkeit verschiedener Machine Learning Algorithmen zur Indoor-Ortung basierend auf WiFi-Fingerprints zu

evaluieren. Es lädt Konfigurationsparameter aus einer YAML-Datei, ruft Daten von einer API ab, filtert diese nach bestimmten Kriterien und vergleicht die Vorhersagen der Algorithmen unter verschiedenen Parametereinstellungen. Die Ergebnisse werden anschließend in CSV-Dateien gespeichert.

- **Ablauf und Logik:** Der Ablauf des Programms lässt sich wie folgt zusammenfassen:
 - **Konfiguration laden:** Mit der Funktion `load_config` wird die Konfigurationsdatei `config.yaml` geladen, die alle notwendigen Einstellungen und Parameter für die Tests enthält.
 - **Erstellen des Ausgabeordners:** Die Funktion `create_output_directory` erstellt einen neuen Ordner, in dem die Ergebnisse gespeichert werden. Der Ordnername enthält einen Zeitstempel, um die Ergebnisse eindeutig zu identifizieren.
 - **Daten abrufen:** Die Funktion `fetch_data` ruft die WiFi-Fingerprint-Daten von einer angegebenen URL ab.
 - **Daten filtern:** Falls in der Konfiguration Räume oder Korridore angegeben sind, werden die Daten mit der Funktion `filter_data` entsprechend gefiltert.
 - **Parameterkombinationen generieren:** Mit der Funktion `generate_parameter_combinations` werden alle möglichen Kombinationen der angegebenen Parameterwerte erstellt.
 - **Vorhersagen und Vergleiche:** Die Funktion `compare_predictions` vergleicht die Vorhersagen der Algorithmen mit den tatsächlichen Raumdaten. Die Ergebnisse werden in CSV-Dateien gespeichert, wobei jede Datei eine spezifische Parametereinstellung repräsentiert.
- **Erweiterbarkeit und Anpassung:** Das Programm ist so aufgebaut, dass es leicht erweitert und angepasst werden kann. Neue Parameter oder Algorithmen können einfach in die YAML-Konfigurationsdatei hinzugefügt werden. Die modularen Funktionen ermöglichen es, einzelne Teile des Programms bei Bedarf zu modifizieren oder zu ersetzen, ohne die gesamte Codebasis ändern zu müssen.

Konfiguration über YAML-Dateien

- **Konfigurationsdateien:** Die Konfiguration des Programms erfolgt über eine YAML-Datei (`config.yaml`), die alle notwendigen Parameter und Einstellungen enthält. Diese Datei ermöglicht eine flexible Anpassung des Testprozesses, ohne dass der Quellcode selbst verändert werden muss.
- **Parameter und Einstellungen:** Die YAML-Datei enthält folgende Hauptparameter:
 - **url_fetch:** Die URL, von der die WiFi-Fingerprint-Daten abgerufen werden.
 - **url_predict:** Die URL der API, die für die Vorhersagen verwendet wird.
 - **num_measurements:** Die Anzahl der Messungen, die verarbeitet werden sollen.
 - **rooms und corridors:** Listen von Raum- und Korridornamen, die in die Analyse einbezogen werden sollen.
 - **parameter_sets:** Verschiedene Sätze von Parametern, die für die Vorhersagen verwendet werden. Jeder Satz enthält spezifische Werte für Algorithmen und deren Einstellungen.

- **Anwendungsbeispiele:** Ein typisches Beispiel für die YAML-Konfigurationsdatei sieht wie folgt aus:

```

1 url_fetch: "http://127.0.0.1:5000/measurements/all"
2 url_predict: "http://127.0.0.1:5000/measurements/predict"
3 num_measurements: 10
4 rooms: ["WH_C_334", "WH_C_335"]
5 corridors: ["WH_C_352"]
6 parameter_sets:
7   - name: "test"
8     parameters:
9       router_selection: ['all']
10      handle_missing_values_strategy: [-100]
11      router_presence_threshold: [0]
12      router_rssi_threshold: [-100]
13      value_scaling_strategy: ['none']
14      weights: ["distance"]
15      algorithm:
16        knn_euclidean:
17          k_value: [7]
18

```

Listing 5.1: .yaml Konfigurationsdatei

Diese Konfiguration spezifiziert, dass die Daten von `http://127.0.0.1:5000/measurements/all` abgerufen und die Vorhersagen bei `http://127.0.0.1:5000/measurements/predict` gemacht werden sollen. Es sollen 10 Messungen verarbeitet werden, die entweder in den angegebenen Räumen oder Korridoren aufgenommen wurden. Der Parameter `parameter_sets` definiert verschiedene Sätze von Parametern, die für die Vorhersagen verwendet werden.

5.2 Vergleich der Algorithmen und Parameter

5.2.1 Untersuchung der Genauigkeit in Abhängigkeit von Parametern

1. Auswahl der nicht konkret betrachteten Parameter bei KNN und SVM

KNN (weights: distance vs. uniform):

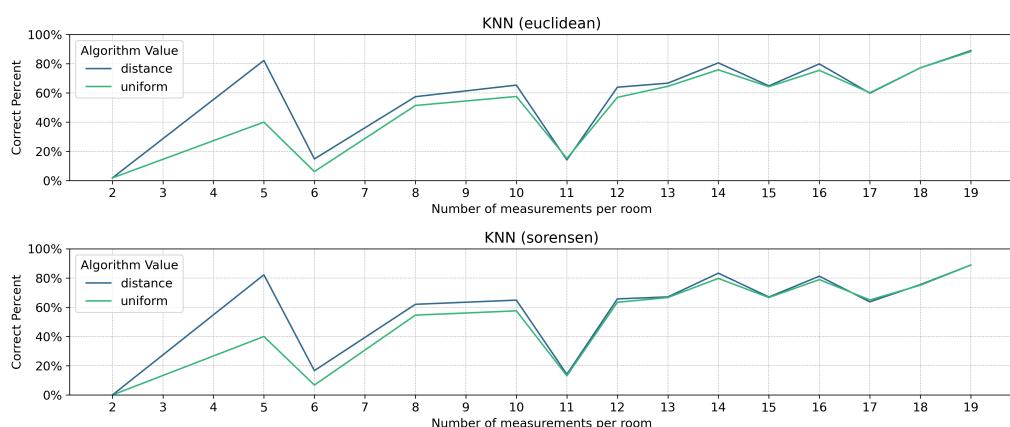


Abbildung 5.2: Distance vs. Uniform weights

Ergebnis: Distance ist besser als Uniform

SVM (RBF) (gamma: scale vs. auto):

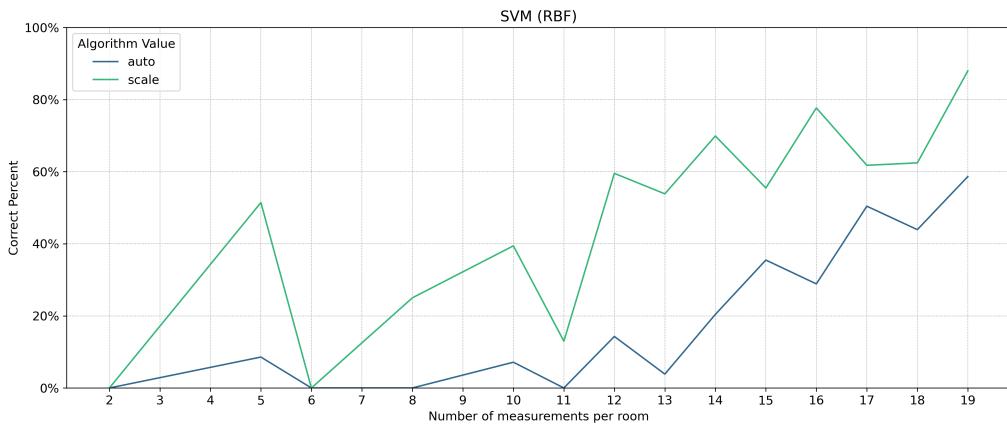


Abbildung 5.3: Scale vs. Auto bei SVM RBF

Ergebnis: Scale ist besser als Auto

TODO: In welchem Wertebereich liegen die Werte?

TODO:

- Es muss auch irgendwo stehen was der average und der weighted average ist und warum beides in den Plots gezeigt wird!

In Abbildung 5.4 sind die besten Parameter für die Algorithmen in Abhängigkeit der Anzahl der Messungen dargestellt. Die Parameter wurden durch die Anwendung des Testprogramms auf die gesammelten Daten ermittelt. Die Parameter wurden so gewählt, dass die Genauigkeit der Algorithmen maximiert wurde. In Abbildung 5.5 ist die durchschnittliche Genauigkeit der Parameter dargestellt. Die Genauigkeit wurde durch die Anwendung des Testprogramms auf die gesammelten Daten ermittelt.

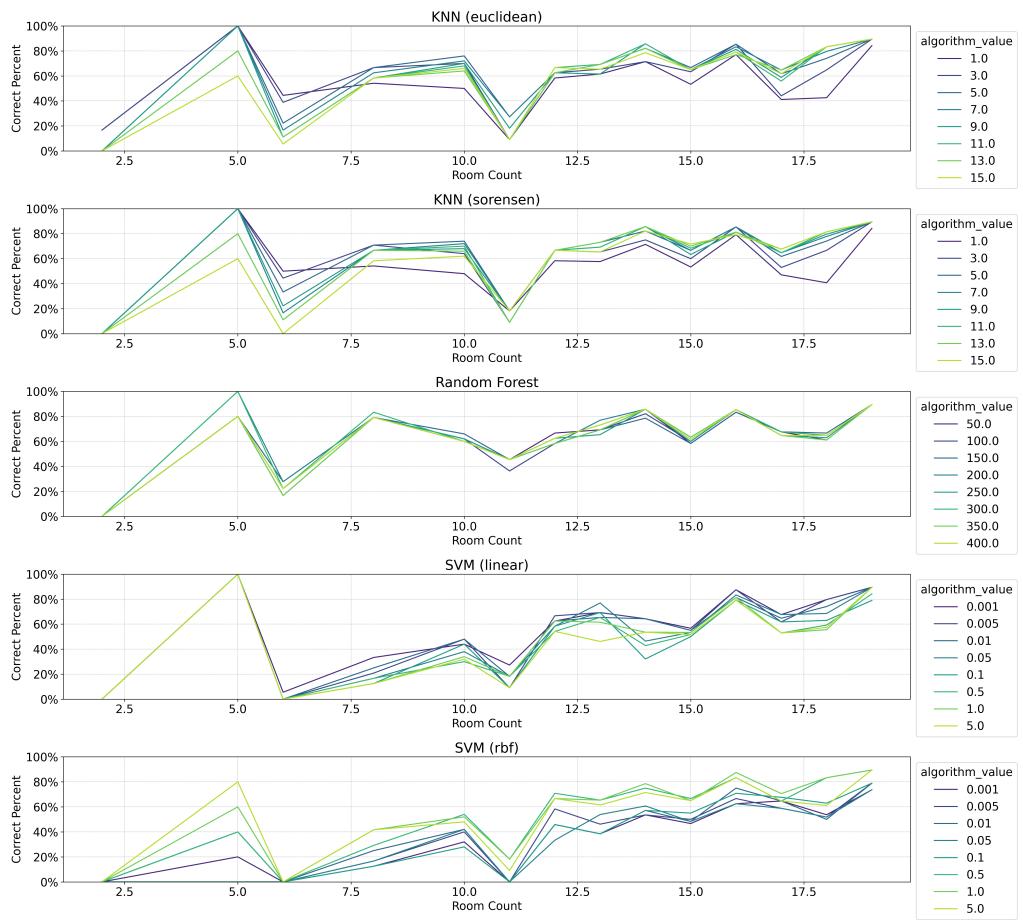


Abbildung 5.4: Vergleich der Parameter in Abhängigkeit der Anzahl der Messungen

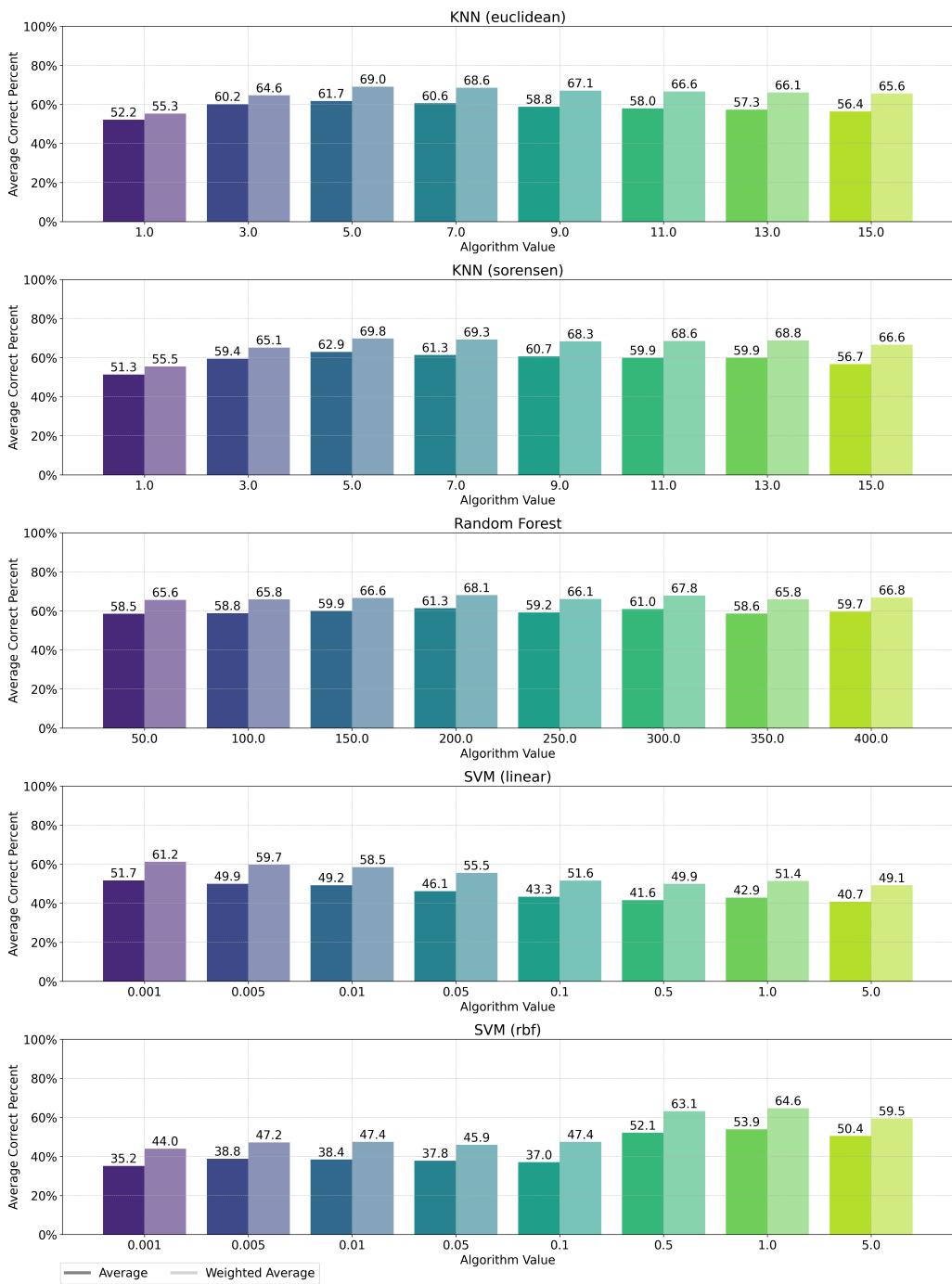


Abbildung 5.5: Vergleich der durchschnittlichen Genauigkeit der Parameter

Warum diese Parameter?

- KNN: 1, 3, 5, 7, 9, 11, 13, 15 -> Weil ungerade (siehe Quelle bei KNN), im Wertebereich der Anzahl an Messungen pro Raum
- RF: 50, 100, 150, 200, 250, 300, 350, 400 Wurde gewählt, weil 407 Messpunkte existieren
- SVM: 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5 -> Quelle: Supervised Learning-Based Indoor Positioning System Using WiFi Fingerprints Seite 62

Ergebnisse:

- KNN: 5, 7, 9
- RF: 100, 200, 300
- SVM (RBF): 0.5, 1, 5
- SVM (Linear): 0.001, 0.005, 0.01

KNN Bei der KNN-Analyse wurden die Werte 5, 7 und 9 für k ausgewählt. Wie in Abbildung 5.4 zu sehen ist, zeigen die Genauigkeiten, dass bei Räumen mit wenigen Messungen kleinere k -Werte bessere Ergebnisse liefern, während bei Räumen mit mehr Messungen größere k -Werte vorteilhafter sind. Diese Tendenz ist sowohl bei der euclidean- als auch bei der sorensen-Metrik zu beobachten.

Abbildung 5.5 zeigt, dass bei average/euclidean die Werte 3, 5 und 7 die besten Ergebnisse liefern, während bei weighted average/euclidean die Werte 5, 7 und 9 am besten abschneiden. Bei average/sorensen erzielen die Werte 5, 7 und 9 die besten Ergebnisse, und bei weighted average/sorensen sind es die Werte 5, 7 und 13, wobei die Genauigkeit bei $k = 9$ und $k = 11$ kaum von der Genauigkeit bei $k = 13$ abweicht.

Generell ist die Genauigkeit bei mehreren Messungen pro Raum weniger stark von k abhängig als bei wenigen Messungen pro Raum, mit Ausnahme von sehr kleinen k -Werten wie 1 oder 3. Daher wurden für die weiteren Analysen die k -Werte 5, 7 und 9 gewählt.

Random Forest Wie in Abbildung 5.4 zu sehen ist, hat der Wert für $n_estimators$ beim Random-Forest-Algorithmus keinen großen Einfluss. Aus diesem Grund wurden die Werte 100, 300 und 500 gewählt, um eine gleichmäßige Verteilung zu gewährleisten.

SVM (linear) Interessanterweise wurden bei der linearen SVM unabhängig vom Parameter C alle Räume korrekt vorhergesagt. Sollte man vielleicht in Betracht ziehen, alle Räume mit nur 5 Messungen zu verwenden und den SVM-Algorithmus anzuwenden? Die Ursache könnte Zufall sein (weniger Messungen führen zu größerer Streuung) oder der Algorithmus ist tatsächlich so gut. Ansonsten nimmt die Genauigkeit mit zunehmendem Wert für C ab. Die besten Werte bei average und weighted average wurden für die Werte 0,001, 0,005 und 0,01 erzielt. Daher wurden für die weitere Analyse die Werte $C = 0.001, 0.005$ und 0.01 gewählt.

SVM (rbf) Generell lässt sich in Abbildung 5.4 erkennen, dass größere Werte für C bessere Ergebnisse liefern als kleinere. Die besten Ergebnisse konnten bei average und weighted average für $C = 1.0$ erzielt werden. Daher wurden für die weitere Analyse die Werte $C = 0.5, 1.0$ und 5.0 gewählt.

5.2.2 Strategien zum Umgang mit fehlenden Werten

In Abbildung 5.6 ist die Genauigkeit der Algorithmen in Abhängigkeit der Strategie zum Umgang mit fehlenden Werten dargestellt. Die Strategien wurden durch die Anwendung des Testprogramms auf die gesammelten Daten ermittelt. Die Genauigkeit wurde durch die Anwendung des Testprogramms auf die gesammelten Daten ermittelt.

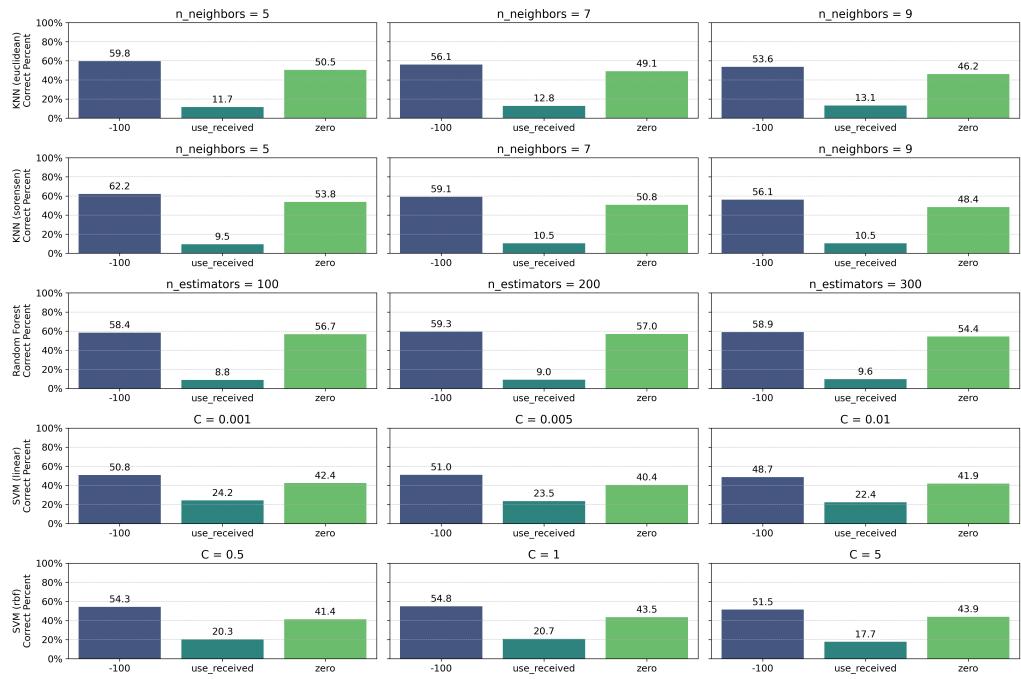


Abbildung 5.6: Vergleich der Genauigkeit in Abhängigkeit der Strategie zum Umgang mit fehlenden Werten

5.2.3 Einfluss der verwendeten Router

Die Ergebnisse der verschiedenen Strategien zur Auswahl der Router sind in Abbildung 5.7 dargestellt.

- Alle Router sind besser als nur die eduroam Router
- Unterschiede nehmen mit zunehmener Anzaahl der Messungen eines Raums ab.
- Es wird sich trotzdem für alle Router entschieden, da die Unterschiede hauptsächlich bei wenigen Messungen auftreten und dort eine größere Streuung erwartet wird und bei den eduroam Routern sichergestellt werden kann, dass diese nicht die Position verändern

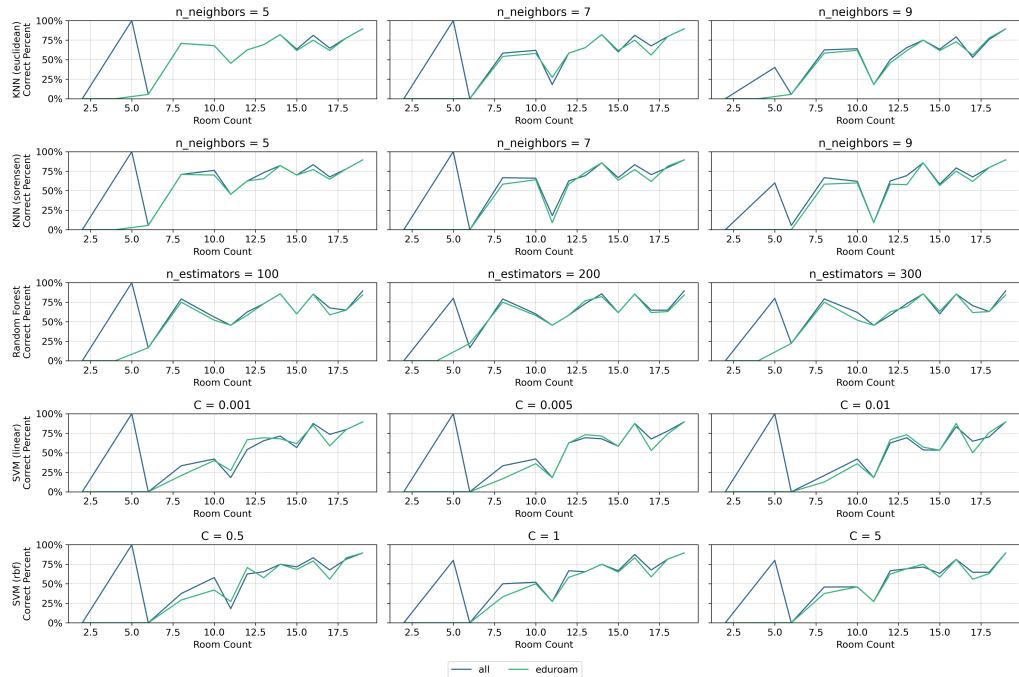


Abbildung 5.7: Auswahl der Router

5.2.4 Filterung von RSSI-Werten

Berücksichtigung nur häufig auftretender Router

Die Ergebnisse der verschiedenen Strategien zur Auswahl des Router Presence Threshold sind in Abbildung 5.8 dargestellt.

- Router die in mehreren Messungen eines Raums vorhanden sind, sind aussagekräftiger als Router die nur in wenigen Messungen vorhanden sind
- Also: In wie viel Prozent der Messungen muss ein Router vorhanden sein, damit er berücksichtigt wird
- Das bedeutet, dass bei wenig Messungen die Auswirkungen gering sein sollten
 - bei allen Algorithmen ist 0 am besten
 - Interessant: Bei allen Algos haben die Genauigkeiten in Abhängigkeit der Messungen pro Raum einen ähnlichen Verlauf und sind mit zunehmendem Router Presence Threshold schlechter, AUSSER bei SVM. Dafür sind bei wenigen Messungen die größeren Router Presence Thresholds besser und bei mehr Messungen die kleineren. -> Deswegen wurde hier 0.25 verwendet, da dieser Wert im Schnitt die besten Ergebnisse erzielt
 - SVM-Kippunkt bei 10

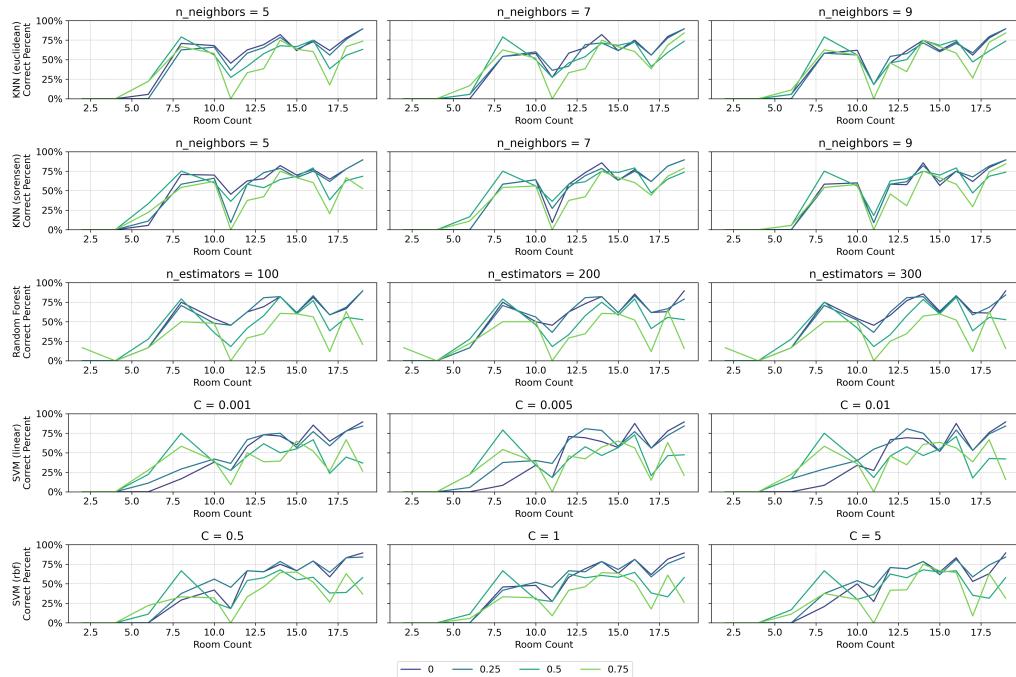


Abbildung 5.8: Vergleich der Genauigkeit in Abhängigkeit des Schwellenwerts für die Anzahl der Messungen eines Routers

Ausschluss von Routern mit schwachem Signal

Die Ergebnisse der verschiedenen Strategien zur Auswahl des Router RSSI Threshold sind in Abbildung 5.9 dargestellt.

- Idee: sehr kleine RSSI Werte werden ignoriert/so getan, als wäre die bei der Messung nicht dabei
- Gedanke dahinter: Router mit größeren RSSI-Werten sind aussagekräftiger und sollten dadurch mehr Einfluss haben
- Ergebnis: Bei allen Algorithmen ist -100 am besten -> Router mit geringen RSSI-Werten haben einen größeren Einfluss als vermutet
- Idee ist nicht von mir, sondern kommt aus dem Paper: Quelle: Comprehensive analysis of distance and similarity measures for Wi-Fi fingerprinting indoor positioning systems
- Die Thresholds sind: -100, -90, -80, -70, -60, -50, -40

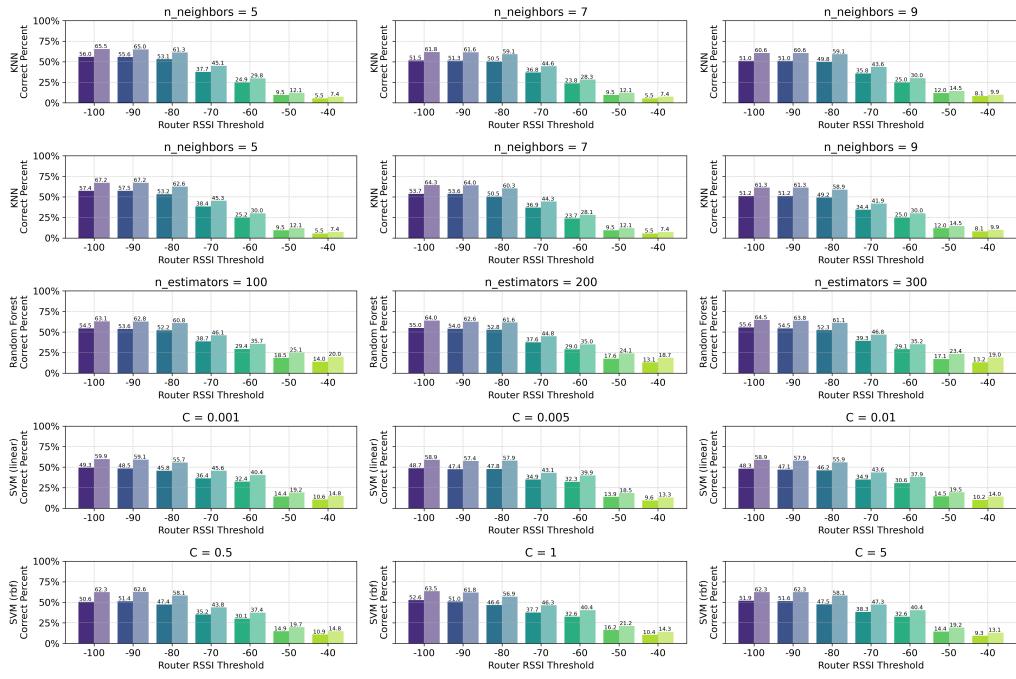


Abbildung 5.9: Router Rssi Threshold

5.2.5 Skalierung von RSSI-Werten

Formeln

Quelle XX: Comprehensive analysis of distance and similarity measures for Wi-Fi fingerprinting indoor positioning systems

Die Idee hinter der Werteskaliierung stammt aus der Quelle XX und basiert auf der Erkenntnis, dass die RSSI-Werte nicht linear verteilt sind. Durch eine geeignete Skalierung kann der Zusammenhang zwischen Entfernung und RSSI-Wert besser abgebildet werden. In der vorliegenden Arbeit wurden drei Skalierungsmethoden aus der Quelle XX implementiert und verglichen.

Grundlage jeder Skalierung ist die positive Darstellung der Werte. Hierfür wird von allen RSSI-Werten aus den Trainings- und Testdaten der niedrigste gemessene RSSI-Wert minus 1 subtrahiert:

$$\text{Positiv}_i(x) = \text{RSS}_i - \min \quad (5.1)$$

wobei \min der niedrigste RSS-Wert minus 1 ist, der alle Fingerabdrücke und WAPs in der Datenbank berücksichtigt.

Anschließend werden die Werte auf drei verschiedene Arten skaliert: linear, exponentiell und potenziert.

Lineare Normalisierung:

Die linear normalisierten Werte werden berechnet als:

$$\text{NullBisEinsNormalisiert}_i(x) = \frac{\text{Positiv}_i(x)}{-\min} \quad (5.2)$$

Exponentielle Skalierung:

Die exponentiell skalierten Werte werden berechnet unter Verwendung von $\alpha = 24$:

$$\text{Exponentiell}_i(x) = \frac{\exp\left(\frac{\text{Positiv}_i(x)}{\alpha}\right)}{\exp\left(\frac{-\min}{\alpha}\right)} \quad (5.3)$$

Potenzierte Skalierung:

Die potenzierten Werte werden berechnet unter Verwendung von $\beta = e$:

$$\text{Potenz}_i(x) = \left(\frac{\text{Positiv}_i(x)}{-\min}\right)^\beta \quad (5.4)$$

In Abbildung 5.10 sind die verschiedenen Skalierungsmethoden dargestellt.

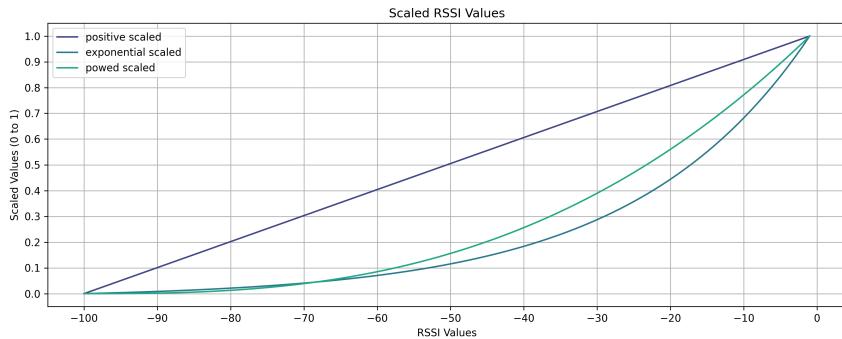


Abbildung 5.10: Value Scaling Strategies

Nochmal KNN Uniform vs. Distance

Durch die Skalierung könnte es sein, dass die ursprüngliche Wahl der distance KNN Methode nicht mehr die beste ist. Aus diesem Grund wird vor der Untersuchung der Skalierungsmethoden nochmal der Vergleich zwischen distance und uniform durchgeführt in Zusammenhang mit den Skalierungsmethoden.

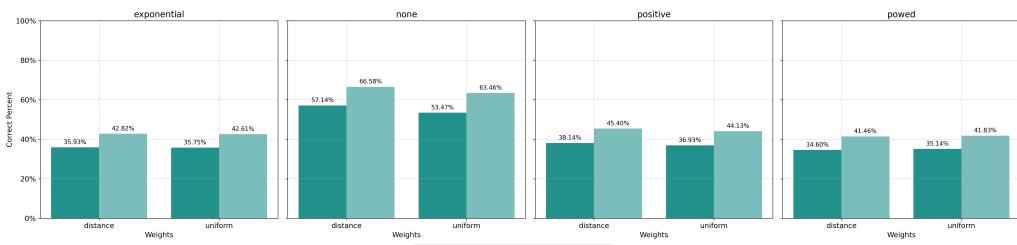


Abbildung 5.11: KNN Uniform vs. Distance im Durchschnitt

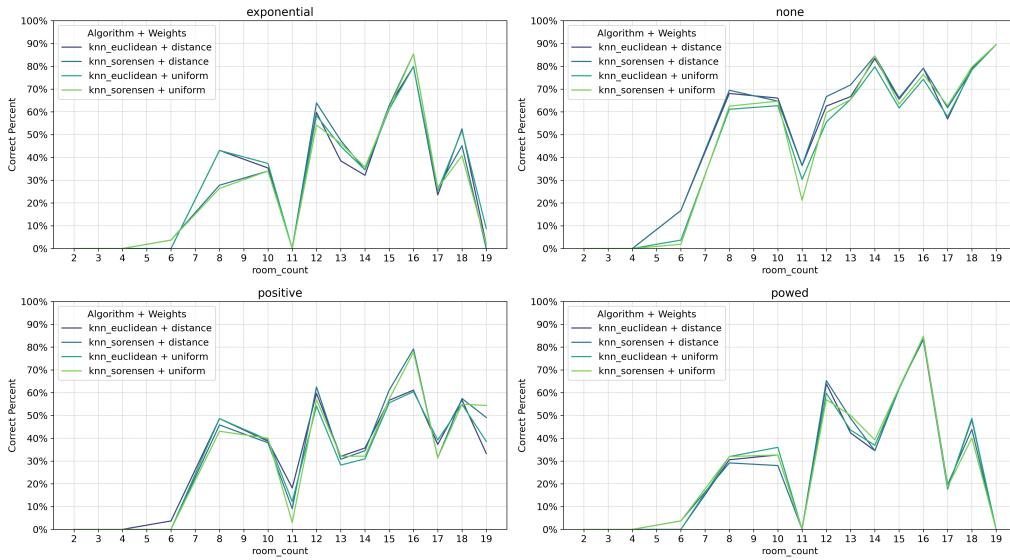


Abbildung 5.12: KNN Uniform vs. Distance pro Raum

Wie zu erkennen ist, ist distance weiterhin besser als uniform. Aus diesem Grund wird weiterhin die distance weights Methode verwendet.

In Abbildung 5.13 folgendes zu erkennen:

- In den meisten fällen sehr ähnlicher Verlauf -> Es gibt Unterschiede bei den Skalierungsmethoden, aber bei den Skalierungsmethoden sind die Ergebnisse bei euclidean/sorensean und distance/uniform sehr ähnlich
- Bei exponential: distance ist besser als uniform bei wenigen Messungen pro Raum
- Bei none: distanz ist etwas besser als uniform bei weniger Messungen. Abstand nimmt aber ab mit der Anzahl an Messungen
- Bei exponential, powd (und auch leicht bei positive): Nimmt die Genauigkeit mit zunehmender Anzahl an Messungen wieder ab. Bisher hatten die Räume mit den meisten Messungen immer die größten Genauigkeiten. In diesem Fall haben die Räume mit 16 Messungen die höchste Genauigkeit und danach nimmt es wieder ab. Bei exponential und powd ist die Genauigkeit bei n = 19 sogar wieder bei fast allen Kombinationen aus distance und weights bei 0%!

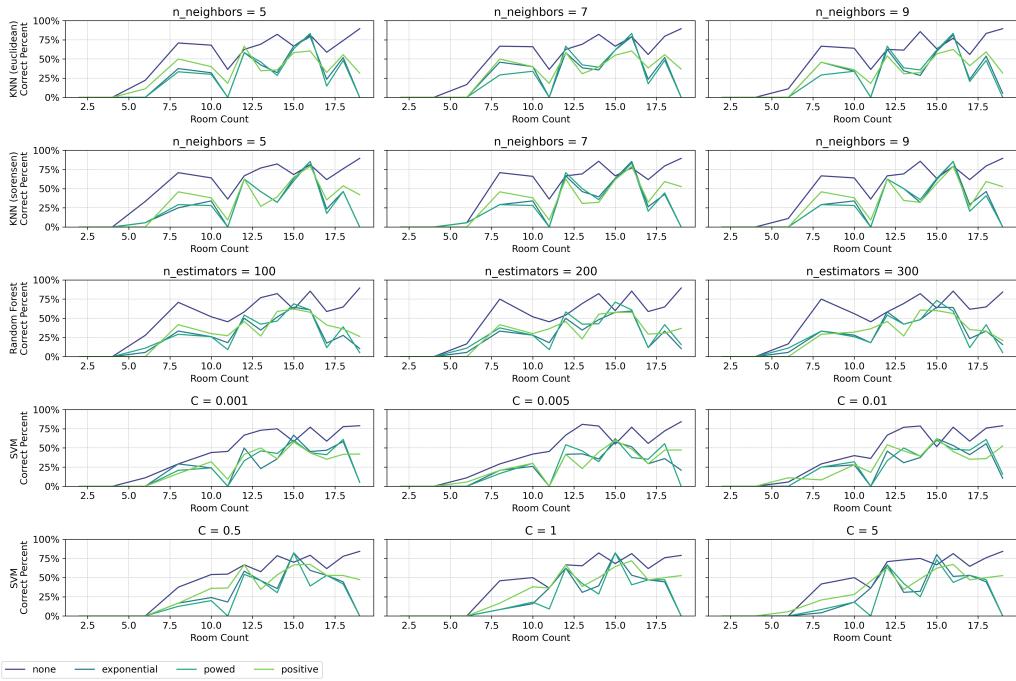


Abbildung 5.13: Skalierungsstrategien nach Algorithmus und Parameter pro Anzahl Messungen pro Raum

5.3 Erweiterte Untersuchungen

5.3.1 Einfluss der Anzahl der Fingerprints auf die Genauigkeit

In der folgenden Untersuchung wird im Detail untersucht, wie sich die Anzahl an Messungen auf die Genauigkeit der Ergebnisse wiederspiegelt. Dafür werden in diesem Fall die Trainingsdaten erweitern, sodass in 4 nahegelegenen Räumen je 20 Messungen gemacht wurden. Aufgrundlage dieser Messungen wird nun überprüft, wie sich der bis jetzt beste Algorithmus schlägt.

TODO: Lineplot mit den Ergebnissen (So wie die Heatmap, aber halt nur eine Achse)

5.3.2 Verbesserung der Vorhersagegenauigkeit durch Fingerprints außerhalb der Räume

Für eine noch bessere Genauigkeit wurde jetzt untersucht, wie sich die Genauigkeit verbessert, wenn zusätzlich zu den Fingerabdrücken innerhalb der Räume auch Fingerabdrücke außerhalb der Räume verwendet werden und somit auch vorhergesagt werden kann, dass kein korrekter Raum ermittelt werden konnte. Das ist sinnvoll, da in diesen Fällen die ESP32 keine Messdaten für falsche Räume veröffentlichen bzw. in weniger Fällen falsche Ergebnisse veröffentlichen kann.

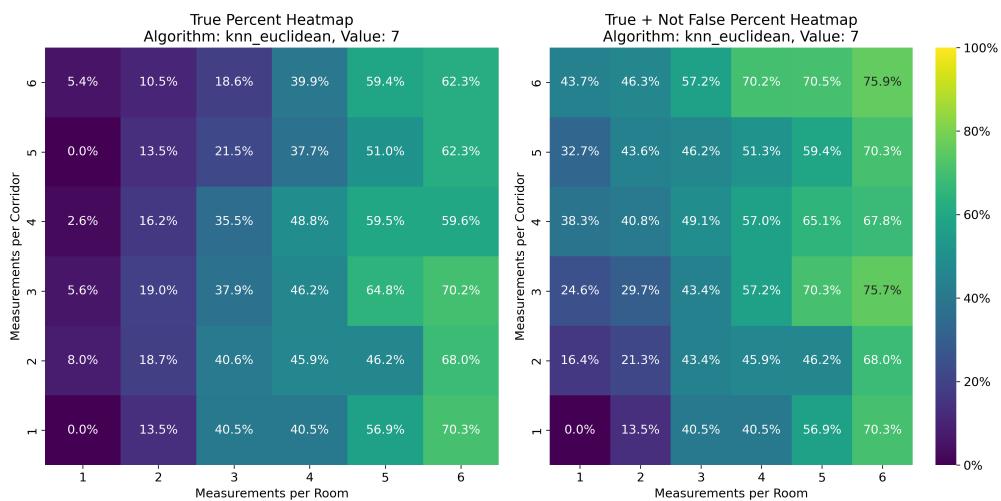


Abbildung 5.14: Ergebnisse unter Betrachtung von Messungen außerhalb der Räume

6 Fazit und Ausblick

6.1 Zusammenfassung der Ergebnisse

6.2 Diskussion der Ergebnisse

6.3 Ausblick auf zukünftige Arbeiten

A Ergänzende Informationen

A.1 Zusätzliche Daten

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipisci scing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Eidesstattliche Versicherung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige Hilfe Dritter angefertigt habe. Sämtliche Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Quellen entnommen sind, habe ich als solche kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Berlin, 14. August 2024

Ort, Datum



Friedrich Völkers