FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

FH Aachen

Fachbereich
Elektrotechnik und Informationstechnik

Bachelorarbeit

Prognose der Anwesenheit von Personen für die Gebäudeautomatisierung mittels Umweltsensordaten

Alexander Loosen
Matr.-Nr.: 3167353

Referent: Prof. Dr-Ing. Ingo Elsen

20. April 2022

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Aachen, 20. April 2022

Geheimhaltung - Sperrvermerk

Die vorliegende Arbeit unterliegt bis zum 25.04. der Geheimhaltung. Sie darf vorher weder vollständig noch auszugsweise ohne schriftliche Zustimmung des Autors oder des betreuenden Referenten vervielfältigt, veröffentlicht oder Dritten zugänglich gemacht werden.

Inhalt

1. Einleitung

Gebäudeautomatisierung bezeichnet die automatische Steuerung und Regelung von Gebäudetechnik wie Heizung, Lüftung oder Beleuchtung. Während sie bisher hauptsächlich für die Optimierung der Energieeffizienz von gewerblichen und öffentlichen Gebäuden genutzt wurde, welche im Zuge solcher Optimierungsschritte als „Smart Buildings" bezeichnet werden, rückte sie im Verlauf der letzten Jahren zunehmend unter dem Begriff „Smart Home" auch in den privaten Bereich.
Die beiden Begriffe stehen in den letzten Jahren so im Vordergrund, weil eine Verbesserung der Energieeffizienz durch bauphysikalische Maßnahmen, wie verminderte Wärmeverluste durch bessere Isolation, an ihre Grenzen gestoßen sind.

In Deutschland haben private Haushalte einen Anteil am gesamten Energieverbrauch des Landes von etwa 29%[1]. Davon nimmt die Erzeugung der Raumwärme einen Anteil von etwa 68%[2] ein. Vor allem bezogen auf den Energieverbrauch durch das Heizen stellt die Gebäudeautomatisierung also einen der größten Sektoren dar, in denen eine Optimierung der Prozesse entscheidend wäre. Beispielsweise können Wärmeverluste über die Nacht hinweg um 20% verringert werden, wenn abends die Rolläden heruntergelassen werden.

Zur weiteren Steigerung der Energieeffizienz ist es also nötig, die Gebäudetechnik insofern automatisch anzusteuern, sodass sog. Performance-Gaps vermieden werden. Performance-Gaps stellen eine Diskrepanz im Energieverbrauch eines Gebäudes zwischen einem theoretischen Soll-Wert zu einem tatsächlichen Ist-Wert dar.

1.1. Motivation und Aufgabenstellung

Für nahezu alle Bereiche der Gebäudeautomatisierung stellt die Anwesenheit von Personen eine zentrale Variable dar. Es ist entscheidend bei einem Optimierungsprozess zu erkennen, zu welchen Zeitpunkten sich Menschen in einem Raum aufhalten. Beispielsweise werden in vielen gewerblich genutzen Gebäuden grundsätzlich alle Räume beheizt - unabhängig davon, ob Personen tatsächlich anwesend sind. Eine Erkennung von menschlicher Anwesenheit ist bei der Optimierung also essenziell.

Machine Learning stellt außerdem eine wichtige Komponente der Gebäudeautomatisierung dar, weil durch den Lernprozess Lüftungen oder Heizungen an- oder ausgeschaltet werden können, bevor bestimmte Ereignisse tatsächlich eintreten. So können regelmäßig genutzte Räume angemessen klimatisiert werden, bevor sie von Personen betreten werden, sodass ein Maximum aus Energieeffizienz und Komfort entsteht.

Da die direkte Messung von Anwesenheit über z.B. Infrarot- oder Kamerasensoren rechtlich problematisch ist, soll in dieser Arbeit untersucht werden, inwiefern Machine-Learning

[1] [Umweltbundesamt, 2022]
[2] [Umweltbundesamt, 2020]

Algorithmen genutzt werden können, um menschliche Präsenz anhand von CO2-, Temperatur und Luftdruckwerten der Raumluft festzustellen.

Die Motivation der Optimierung der Gebäudeautomatisierung besteht, da ein steigender CO2-Gehalt der Raumluft nachweislich mit einer Abnahme der menschlich kognitiven Leistung einhergeht. Mehrere Studien[1] [2] konnten belegen, dass sowohl sprachliche als auch logisch-mathematische Fähigkeiten abnehmen, sobald der CO2-Gehalt der Raumluft bestimmte Werte überschreitet.
Um eine angemessene Datengrundlage zu schaffen, wurden in diversen Büro-Räumen der FH Aachen Temperatur-, Luftfeuchtigkeits-, Infrarot- und CO2-Sensoren angebracht, deren Messungen kontinuierlich auf einer Datenbank gespeichert wurden. Der Zeitraum der Messwerte begann Mitte 2021.
In allen Räumen sind täglich ein oder mehrere Personen im Rahmen eines ca. 8-stündigen Arbeitstages anwesend, weshalb die Temperatur-, Luftfeuchtigkeits- und CO2-Messwerte über einen Tag hinweg Schwankungen aufweisen. Diese sollen von einem Machine Learning Modell mit der direkten Präsenzmessung des Infrarotsensors gegenübergestellt werden, sodass das fertig trainierte Modell anhand der Messwerte von Temperatur, Luftfeuchtigkeit und CO2 Aussagen über menschliche Präsenz treffen kann.
Die Frage, ob ein in einem Raum trainiertes Modell auch präzise Aussagen über menschliche Anwesenheit in einem anderen Raum treffen kann, stellt eine besondere Bedeutung dar.

Es gab keine Einschränkungen hinsichtlich dessen, welche konkreten Machine-Learning Algorithmen benutzt werden sollen.
Als Programmiersprache für das Projekt wurde Python mit der *„scikit-learn"*-Bibliothek gewählt. scikit-learn implementiert eine große Menge an Machine Learning Algorithmen zusammen mit einfachen Auswertungstechniken wie Graphen und Statistiken und ist deshalb für diesen Anwendungsfall gut geeignet.

[1] [Snow et al., 2019]
[2] [Riham Jaber et al., 2017]

1.2. Vorgehensweise

Das Projekt beschäftigt sich im Schwerpunkt mit den folgenden Arbeitsschritten:

- Datenbeschaffung durch Datenbankzugriffe per SQL
- Analyse und Vorbereitung der Daten (Pre-Processing)
- Trainieren von Machine-Learning Modellen anhand der vorbereiteten Datensets
- Ergebnisauswertung durch Gegenüberstellung verschiedener Datensets und Modellen

Da es zwischen allen verfügbaren Datensets der einzelnen Räume und Machine-Learning-Modellen eine Vielzahl an Kombinationsmöglichkeiten gibt, ist die Projektarbeit mit dem Anspruch angelegt, ein möglichst übersichtliches, gut gekapseltes Python Programm zu erstellen, das einfach und schnell verschiedene Datensets verarbeiten und mit einer dem Forschungszweck angemessenen Anzahl von Machine-Learning Modellen auszuwerten.

Um einen Vergleich der Ergebnisse zu ermöglichen, sollen diese klar und verständlich dargestellt werden. Da nicht bei allen Algorithmen die gleichen Leistungsindikatoren genutzt werden, sollen hauptsächlich nur jene Indikatoren betrachtet werden, die bzgl. aller Algorithmen auch gleiche Bedeutung haben. Falls modellspezifische Leistungsindikatoren als besonders erkenntnisreich erachtet werden, wird dies in dieser Arbeit angemerkt.

2. Stand der Technik für Anwesenheitserkennung/-prognose

Man kann verschiedene Techniken zur Anwesenheitserkennung und -prognose in Innenräumen in *aktive* und *passive* Verfahren unterteilen.

Aktive Verfahren messen die menschliche Anwesenheit etwa mit Kamera- oder Infrarotsensoren, welche den Menschen direkt im Raum erkennen können. Ein Kamerasensor nimmt dabei kontinuierlich Bilder des Raumes auf, während ein Algorithmus im Hintergrund versucht ein 3D-Modell eines menschlichen Skeletts auf sich bewegende Teile des Bildes zu legen. Bewegen sich über mehrere Bilder hinweg die Bereiche auf und unmittelbar neben dem 3D-Modell, registriert das System eine Anwesenheit.

Infrarotsensoren dagegen funktionieren eher wie ein klassischer Bewegungsmelder, welcher Unterschiede in der Infrarotstrahlung eines Raumes erkennt. Durch die generierte Körperwärme des menschlichen Körpers verursacht ein Mensch, der sich durch den Sensor bewegt, einen Ausschlag des Sensors.

Passive Verfahren orientieren sich an den physikalischen Eigenschaften des Raumes. Anstatt die Präsenz des Menschen direkt zu messen, wird versucht, diese anhand von Veränderungen der Eigenschaften der Raumluft oder elektromagnetischer Strahlung des Raumes zu erkennen.
Verändert sich beispielsweise durch menschliche Anwesenheit die Luftfeuchtigkeit in einem Raum in sehr geringem Maß, wird die Laufzeit einer Ultraschallwelle, die durch diesen Raum geschickt wird, leicht verringert, da die Schallgeschwindigkeit in dichteren Medien zunimmt.
Die Systeme unterscheiden sich neben der Art der Messung auch deutlich in der Komplexität der Implementierung. Während Ultraschallsensoren wegen der verhältnismäßig geringen Schallgeschwindigkeit keine besonders hohe Genauigkeit besitzen müssen, sind die an einen Mikrowellensensor gestellten Ansprüche wesentlich höher, da sich Mikrowellen mit deutlich höherer Geschwindigkeit im Raum ausbreiten. Mikrowellen erreichen in Atemluft Geschwindigkeiten von ca. $300 \cdot 10^6\, m/s$, Schallwellen lediglich ca. $330\, m/s$. Mit Mikrowellen können Bewegungen durch Ausnutzung des Doppler-Effektes einer elektromagnetischen Welle erkannt werden. Bewegen sich Objekte oder Personen auf den Sensor zu, wird das Echo der Welle gestaucht, wodurch die Bewegung erkannt wird.
Hier verdeutlicht sich nochmals die Motivation dieser Arbeit, da Sensoren für Temperatur, Luftfeuchtigkeit oder CO2 weder teuer sind, noch besondere Ansprüche an Positionierung im Raum oder Kalibrierung besitzen.

3. Untersuchte Verfahren

3.1. Machine Learning

Machine Learning ist ein Teilbereich der künstlichen Intelligenz und beschreibt das Entwickeln mathematischer Modelle zur statistischen Auswertung von Daten.
Dabei wird dem Modell anhand von Trainingsdaten beigebracht, Gesetzmäßigkeiten zu erkennen, um danach Erwartungen über bisher unbekannte Daten treffen zu können. Beispielsweise könnte ein solches Modell aus einem Datenset mit aktueller Jahreszeit, Uhrzeit und Position der Sonne am Himmel trainiert werden, sodass es schließlich auch in einem anderen Datenset aus Jahreszeit und Position der Sonne Rückschlüsse auf die Uhrzeit treffen kann.
Als Vorbild für diesen „Lernvorgang" dient das menschliche Gehirn, welches ebenfalls versucht zwischen bestimmten Input-Parametern wie z.B. Form und Farbe eines Gegenstandes eine Beziehung herzustellen, um das beobachtete Objekt in Zukunft schneller kategorisieren zu können.
Da eine Vielzahl von effektiven Machine Learning Algorithmen existiert, ist es essenziell, sich mit den Stärken und Schwächen einzelner Herangehensweisen zu befassen.

In dieser Arbeit sollen zwei Unterkategorien des Machine Learning genauer betrachtet werden:

- *Supervised Learning*
- *Unsupervised Learning*

Supervised Learning bedeutet zwischen bestimmten Feldern eines Datensets eine Beziehung zu einem sog. Label herzustellen, welches als eine Art Ergebnis aus den Eingabewerten gesehen werden kann. Ein so trainiertes Modell kann dann neue, ihm vorher unbekannte Datensets, mit einem Label versehen - etwa wie in dem o.g. Beispiel, wo Jahreszeit und Sonnenposition die Eingabewerte und die Uhrzeit das Label darstellen. Der Begriff „*supervised*" ergibt sich daraus, dass das Datenset, mit dem das Modell trainiert wird, diese Labels gegeben hat, sodass das Modell sich bei jedem Schritt des Lernvorgangs selbst korrigieren kann, falls eine Fehleinschätzung getroffen wurde. Bei einer sog. „*Klassifizierung*" sind diese Labels fest vorgegeben, während sie in der „*Regression*" kontinuierlicher Natur sind. Im Kontext dieser Arbeit wäre das Ergebnis einer Klassifizierung eine „1" für Anwesenheit und eine „0" für Abwesenheit, während das Ergebnis einer Regression eine Wahrscheinlichkeit auf Anwesenheit zwischen 0.0 und 1.0 darstellen würde.

Beim „*Unsupervised Learning*" versucht das Modell ohne Referenz zu einem bestimmten Label Zusammenhänge zwischen bestimmten Feldern des Datensets herzustellen. Solche Modelle arbeiten vorrangig mit „*Clustering*" und „*Dimensionality Reduction*".
„*Clustering*"-Algorithmen versuchen ein Datenset in kleinere Teilmengen einzuteilen und so aus den Feldern des Datensets bestimmte Abhängigkeiten abzuleiten.

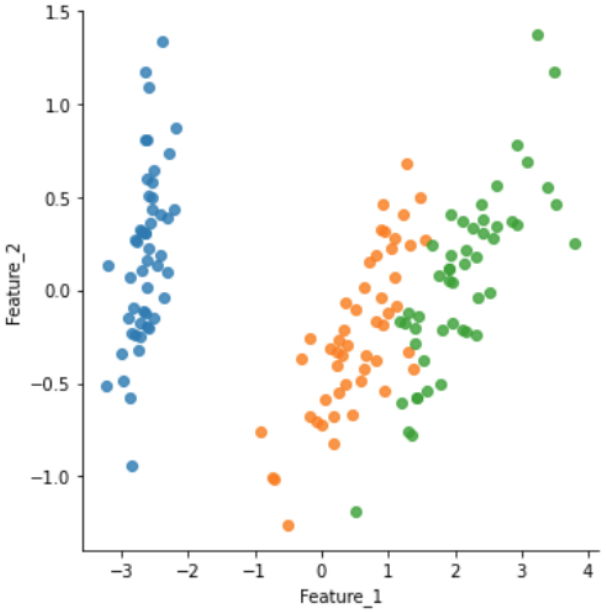

Abbildung 3.1.: Beispiel für Clustering

Bei der „*Dimensionality Reduction*" versucht der Algorithmus das Datenset in seiner Dimensionalität, also in der Anzahl an Feldern, zu reduzieren. Es wird also die Frage gestellt, ob sich in einem bestehenden Datenset auch mit weniger Feldern Abhängigkeiten feststellen lassen. Dieser Schritt wird vor allem für Modelle benutzt, die sensibel gegenüber hoher Dimensionalitäten sind, sodass das Datenset vor dem Training in seiner Dimensionalität heruntergebrochen werden kann.
Im Rahmen des Projektes wurden hauptsächlich Klassifizierungs-Algorithmen genutzt, da ein Großteil der Datensets Labels zur Überprüfung hatte.
Um einen Vergleich zu ermöglichen, werden später trotzdem noch einzelne Ergebnisse von Clustering und Dimensionality Reduction betrachtet. Im Folgenden sollen die genutzten Modelle erklärt werden.

3.1.1. Random Forest Classifier

Random Forest Classifier RFC stellen eine Unterkategorie der „*Decision Trees*" dar. Decision Trees sind einfache Anordnungen von bestimmten Fragen, die über das Datenset gestellt werden, um eine Klassifikation zu erreichen.

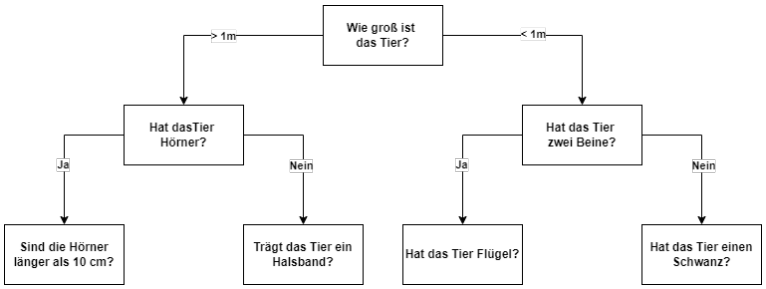

Abbildung 3.2.: Beispiel eines Decision Trees

Erstellt man ein „*Ensemble*" aus Decision Trees, die Erwartungen über einen zufällig gewählten Teil des Datensets treffen können, entsteht ein Random Forest. Der Random Forest Classifier versucht, eine Menge einfacher Schätzfunktionen über einen komplexeren Sachverhalt „abstimmen" zu lassen. Während sich in einem einzelnen Entscheidungsbaum Fehleinschätzungen entwickeln können, sinkt die Chance auf eine solche Fehleinschätzung, je mehr unabhängige Entscheidungsbäume man befragt.

3.1.2. Gradient Boosting Classifier

Der *Gradient Boosting Classifier* GBC ist eine Abwandlung eines RFC und versucht, seine Erwartungen aufgrund von Abweichungen eines Labels vom Durchschnitt dieses Labels zu treffen.
Versucht man beispielsweise aus dem Gewicht eines Tieres dessen Alter zu bestimmen, wird ein GBC als Ausgangswert den Durchschnitt aller Label-Werte, also dem Gewicht, berechnen. Danach werden die Abweichungen aller Label-Werte zu diesem Durchschnitt gebildet. Diese Abweichungen werden nun in Beziehung zu den anderen Spalten des Datensets gesetzt.

Beispielsweise könnte man so davon ausgehen, dass ausgewachsene Tiere von einem bestimmten Alter über dem Durchschnittsgewicht liegen. Genauso liegen besonders junge Tiere wahrscheinlich immer einen ähnlichen Wert unter dem Durchschnittsgewicht. So wurde zwischen dem Label *Gewicht* und der Spalte *Alter* eine Beziehung hergestellt. In weiteren Iterationen orientiert sich der GBC immer an der Abweichung zum Durchschnittswert des vorherigen Baumes. So werden die getroffenen Erwartungen über mehrere Iterationen immer präziser.

3.1.3. Support Vector Classifier

Der *Support Vector Classifier* SVC versucht in einem Datenset anhand von bestimmten Cut-Off-Values klare Grenzen zwischen Werten zu finden, sodass man alle Messwerte ober- und unterhalb der Grenze eindeutig klassifizieren kann.

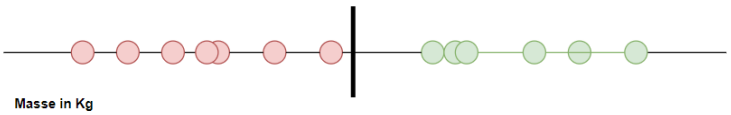

Abbildung 3.3.: Beispiel eines Support Vector Classifiers

In Abb. 3.3 ist der SVC ein Punkt auf einer eindimensionalen Linie, auf der das Gewicht in kg von z.B. Tieren in „unter-" und „übergewichtig" unterteilt wird. Dieser Punkt ist Ergebnis aller Verhältnisse der einzelnen Datenpunkte zueinander. Durch sog. „*Kernel Funktionen*" versucht der Algorithmus nun Beziehungen in höheren Dimensionen zu finden, wie z.B. $Masse^2$, $Masse^3$ usw. . Der SVC stellt dann in diesen Dimensionen eine Linie in einem zweidimensionalen oder eine Ebene in einem dreidimensionalen Koordinatensystem dar.

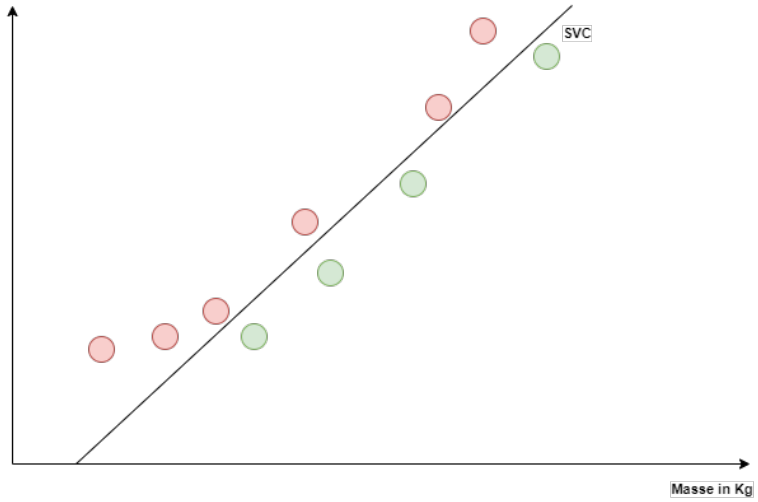

Abbildung 3.4.: Beispiel eines Support Vector Classifiers in der zweiten Dimension

Da der SVC die Verhältnisse aller Datenpunkte zueinander betrachtet, ist er sehr anfällig für Abweichungen in den Daten, was bei der Datenvorbereitung und der Auswertung beachtet werden muss.

3.1.4. Logistische Regression

Die *Logistische Regression* LR ordnet Daten anhand einer Wahrscheinlichkeit einem bestimmten Label zu. Trotz des Teilnamens „Regression" handelt es sich um einen Klassifizierungsalgorithmus. Der Name kommt daher, dass die berechnete Wahrscheinlichkeit zu einem kontinuierlichen Label zwischen 0 und 1 führt, was durch die Form einer Sigmoid-Funktion ausgedrückt wird. Der Algorithmus eignet sich wegen dieser Eigenschaft besonders für das Klassifizierungsproblem der Projektarbeit, bei der ein Wahrheitswert, wie „Anwesenheit" oder „Abwesenheit" untersucht werden soll.
Nutzt man logistische Regression beispielsweise zur Abbildung einer Beziehung zwischen Gewicht von Tieren in Gramm zu einer Wahrscheinlichkeit für Übergewicht, könnte das entstehende Modell wie folgt aussehen.

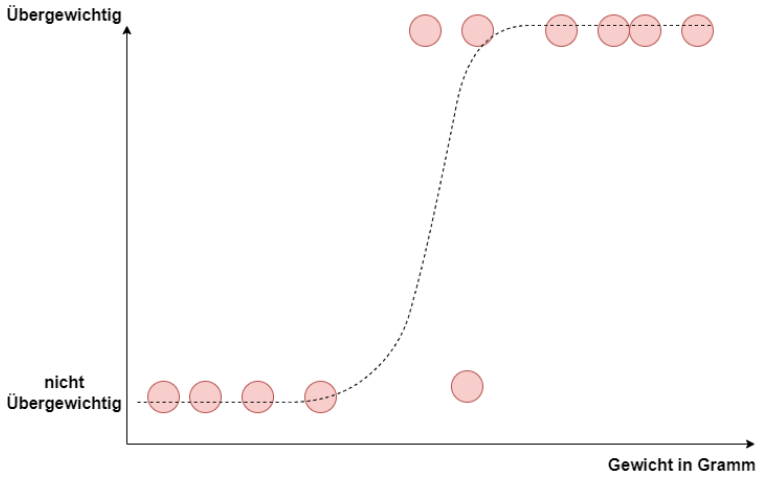

Abbildung 3.5.: Beispiel logistischer Regression

Die Sigmoid-Funktion wird grundsätzlich durch

$$p = \frac{1}{1 + e^{-(x-\mu)/s}} \tag{3.1}$$

beschrieben, wobei x der Eingabewert in Gramm darstellt und μ der Mittelpunkt der Kurve beschreibt, der sich aus dem Verhältnis von Eintrittswahrscheinlichkeit p_1 und Gegenwahrscheinlichkeit p_0 ergibt.

$$p(\mu) = 0.5 = \frac{p_1}{p_0} \tag{3.2}$$

Zusätzlich beschreibt s einen Skalierungsparameter, mit dem die Form der Kurve flacher oder steiler werden kann, was angibt, wie eindeutig die eingegebenen Daten zugeordnet werden können. Wären im o.g. Beispiel also alle roten Punkte jeweils links und rechts von der Mitte einsortiert, wäre die Sigmoid-Funktion in der Mitte sehr steil, da alle Daten anhand des Mittelpunktes klar getrennt werden könnten.

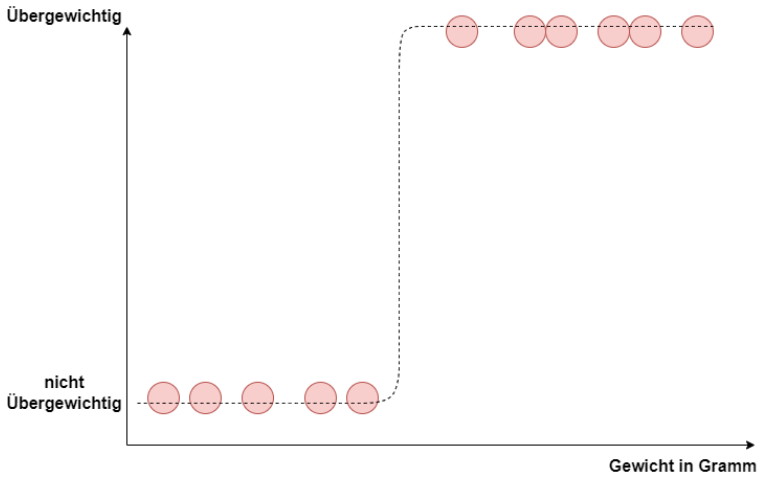

Abbildung 3.6.: Logistische Regression mit deutlicher Skalierung

Anhand der optimierten Sigmoid-Funktion können nun neue Daten klassifiziert werden, wobei die Wahrscheinlichkeit bei $p < 0.5$ auf 0 und $p >= 0.5$ auf 1 gerundet wird.

3.1.5. K-Nearest-Neighbours Clustering

Der *K-Nearest-Neighbours*-Algorithmus KNN betrachtet den Abstand von einem gegebenen Punkt p zu einer Menge an Punken in einem Datenset und ordnet ihn gemäß dieser Abstände einer bestimmten Kategorie zu. Die Zuordnung geschieht anhand der Kategorie der größten Menge zum nächsten Nachbarn.

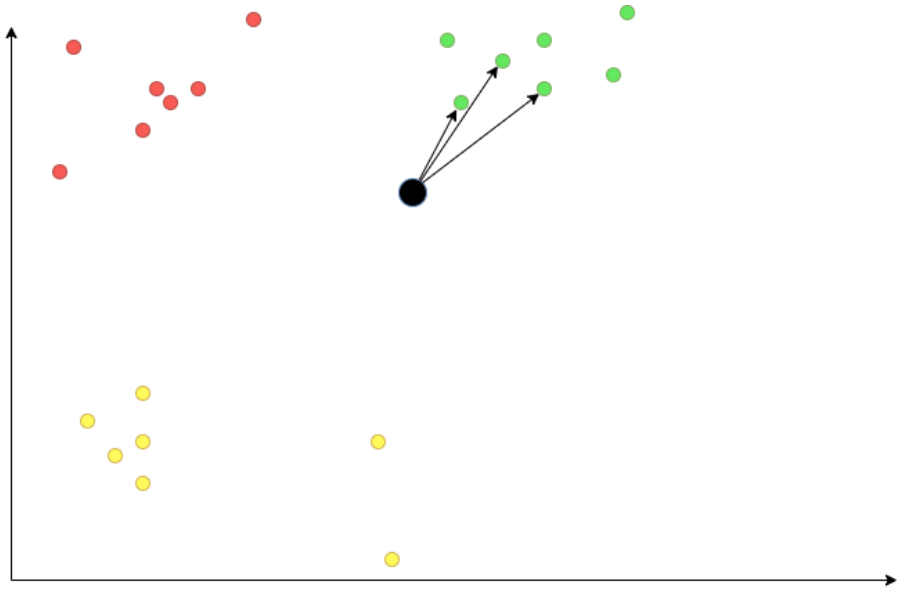

Abbildung 3.7.: Beispiel eines KNN

Im oben gezeigten Beispiel würde ein Punkt anhand seiner drei nächsten Nachbarn als „grün" kategorisiert. Die Anzahl der gewünschten Nachbarn, die zu untersuchen sind, ist frei wählbar. Da die Abstandsberechnung über den *Euklidischen Abstand* berechnet wird, funktioniert dieser Algorithmus auch in höheren Dimensionen, denn

$$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2 + ... + (n2 - n1)^2} \tag{3.3}$$

lässt sich für beliebig viele Dimensionen fortsetzen.

3.1.6. K-Means Clustering

Der *K-Means* Algorithmus legt eine bestimmte Anzahl von neuen Datenpunkten an eine zufällige Position des Datensets. Die herumliegenden Datenpunkte werden nun anhand ihrer Abstände diesen sog. *Clustern* zugeordnet (in Abb. 3.8 als vertikale Linien dargestellt).
Die Abstände werden auch hier durch die Euklidische Abstandsformel berechnet.

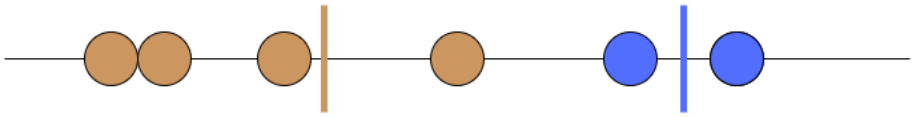

Abbildung 3.8.: Beispiel des K-Means Algorithmus

Da der Algorithmus die optimale Verteilung von Clustern nicht erkennen kann, wird die Position der Cluster gespeichert und neue Cluster an zufällige Positionen gelegt. Sind die durchschnittlichen Abstände der Datenpunkte zu den neuen Clustern geringer als vorher, werden die gespeicherten Cluster nun in Richtung der neuen Cluster verschoben (Abb. 3.9).

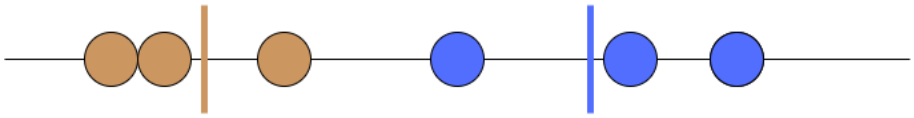

Abbildung 3.9.: Beispiel des K-Means Algorithmus

Die Anzahl an Schritten die der Algorithmus durchläuft kann vorgegeben werden. Allgemein sollte der Algorithmus so lange neue Schritte machen, bis der Abstand der gespeicherten und neuen Cluster zueinander einen bestimmten Minimalwert unterschreitet.

3.1.7. Neuronale Netzwerke

Die Funktionsweise eines neuronalen Netzwerks ist direkt angelehnt an die Funktionsweise des menschlichen Gehirns. Einzelne Knotenpunkte (Neuronen) werden mithilfe von gewichteten Verbindungen verknüpft, sodass das Netzwerk versucht Eingabewerte bestimmten Ausgabewerten zuzuordnen. Diese Zuordnung der Ein- und Ausgabewerte im Input- und Output-Layer geschieht nicht direkt, sondern durch ein oder mehrere *Hidden Layer*, dessen Neuronenzahl üblicherweise über der Anzahl der Neuronen im Input Layer liegt. Die Anzahl der Neuronen im Output-Layer entspricht der Anzahl an Ergebnissen, die sich aus dem Input ergeben können. Im Beispiel der Anwesenheitsanalyse entspräche das also zwei Neuronen für An- und Abwesenheit.

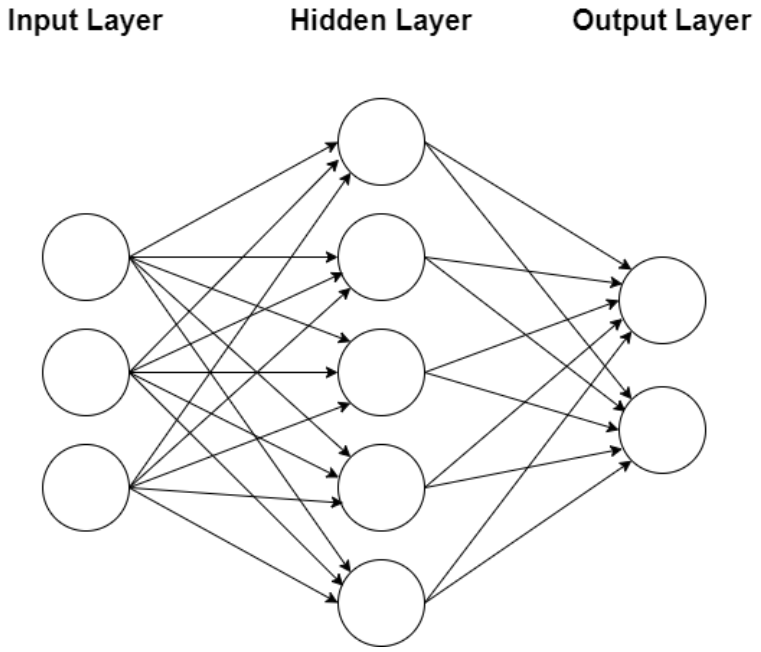

Abbildung 3.10.: Beispiel eines neuronalen Netzwerks

Liegt an einem Neuron eine Information an, wird diese als Eingabe einer Aktivierungsfunktion φ genutzt, die mithilfe eines bestimmten Schwellwertes bestimmt, ob dieses Neuron aufgrund der Eingabe aktiviert wird. Über eine bestimmte Anzahl von Iterationen werden die Gewichtungen zwischen den einzelnen Neuronen stärker oder schwächer.

3.1.8. Long Short Term Memory

Das *Long Short Term Memory* LSTM ist eine Abwandlung herkömmlicher neuronaler Netzwerke. Es handelt sich um ein *rekurrentes* neurales Netzwerk, was bedeutet, dass jedes Neuron seine Ausgabewerte auch wieder als Eingabewerte nutzt. Der Begriff *Memory* rührt daher, dass durch diese Rückkopplung eine Art Gedächtnis entsteht, durch die das Netzwerk bessere Rückschlüsse auf die Einordnung des aktuellen Input-Wertes ziehen kann.

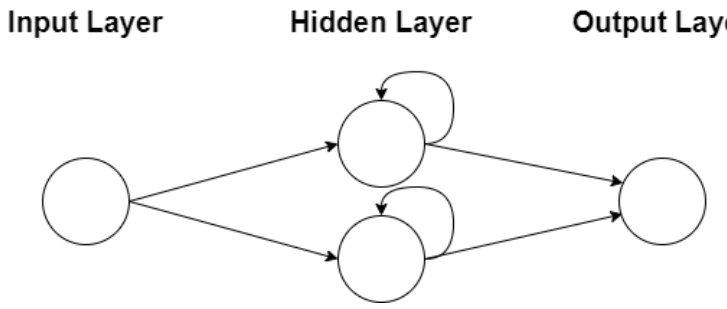

Abbildung 3.11.: Beispiel eines rekurrenten neuronalen Netzwerks

Bei einem LSTM verfügt jedes Neuron zusätzlich über einen Zell-Status C_t, welcher durch einen Input x_t verändert werden kann. Auf den Zell-Status nehmen *Input*-, *Forget*- und *Output*-Gates Einfluss. Diese drei Gates besitzen jeweils ein eigenes neuronales Netzwerk σ.

Das Input-Gate bestimmt, ob der Zell-Status anhand des anliegenden Inputs verändert werden darf. Das Forget-Gate gibt an, ob der Status der Zelle zurückgesetzt und somit das „Gedächtnis" der Zelle gelöscht wird. Es stellt die zentrale Komponente des LSTMs dar. Das Output-Gate regelt, ähnlich wie Aktivierungsfunktion normaler NNs, ob der anliegende Input zu einem Output der Zelle führt.

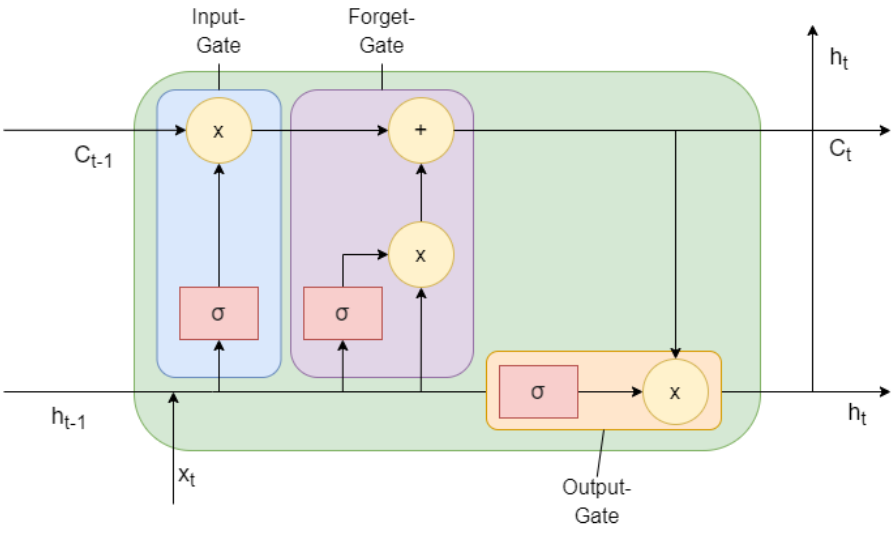

Abbildung 3.12.: Vereinfachter Aufbau einer LSTM-Zelle

Ein LSTM kann auf diese Weise eine Vielzahl vorangegangener Zeitschritte einbeziehen. Es verlässt sich so nicht direkt auf einen gegebenen Input, sondern auf einen langen Verlauf von bereits verarbeiteten Input-Werten. LSTMs sind deshalb besonders interessant für Probleme bei denen Beziehungen zwischen kontinuierlichen Datenwerten gebildet werden müssen.

Man gehe man von einem LSTM aus, das über eine Liste an Namen verfügt und anhand eines Datensatzes gelernt hat, zwei Namen mit der Beziehung „sah" zu z.B. „Jonas sah Jakob." zu verknüpfen.
Das LSTM hat nun intern einen Status, der angibt, dass auf einen Namen mit hoher Wahrscheinlichkeit das Wort „sah" und mit niedriger Wahrscheinlichkeit ein zweiter Name folgt. Beginnt man nun einen neuen Satz mit „Jakob", werden als mögliche nächste Worte „Jonas", „Jakob" und „sah" ausgewählt. Bisher ähnelt dieser Verlauf einem normalen neuronalen Netzwerk. Beim LSTM steht aber nun wegen des vorherigen Durchlaufs der Name „Jakob" im Forget-Gate, sodass „Jakob" aus der verfügbaren Wortauswahl für das nächste Wort gelöscht wird.

3.2. CO2 als Anwesenheitsindikator

Der Anteil von CO2 in frischer Atemluft beträgt zwischen 350 und 450 ppm. Es gibt in Deutschland und auch Europa keine grundsätzlich festgelegten Grenzwerte für akzeptable Raumluft, vielmehr raten Gesundheitsämter verschiedener Länder Grenzwerte zwischen 1200 und 1500 ppm einzuhalten. Ab der Obergrenze von 1500 ppm zeigen sich beim Menschen erste Müdigkeitserscheinungen, weshalb dieser Wert in der Literatur als maximaler Richtwert für Innenräume gilt.

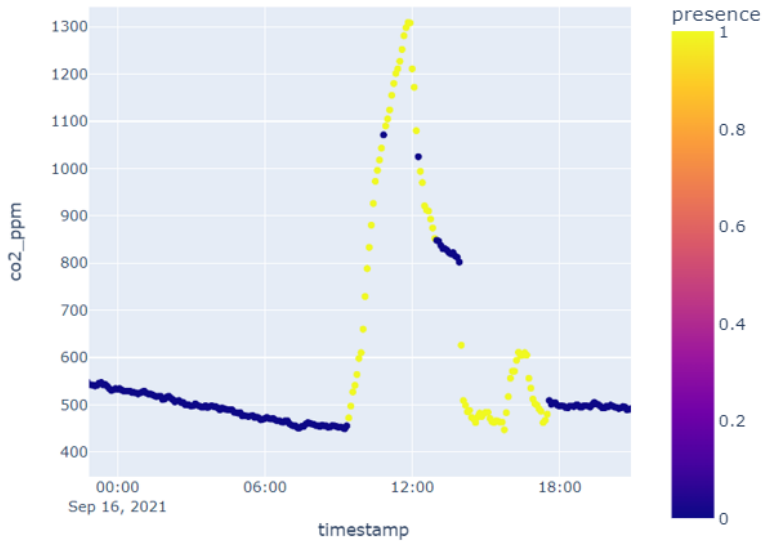

Abbildung 3.13.: CO2-Gehalt der Raumluft über einen Tag

Es ist zu erkennen, dass die CO2-Werte in einem normalen Büroraum innerhalb dieser empfohlenen Grenzen schwanken. Abgesehen von einigen kleinen Pausen sind fast durchgehend Personen anwesend, die täglich etwa um 12:00 Uhr den Raum lüften. Es ist auch klar zu erkennen, dass sich der CO2-Gehalt während der Abwesenheit von Personen durch die passive Lüftung des Raumes (Tür-/Fensterspalten) langsam wieder gegen den Grundwert bewegt.

3.3. Luftfeuchtigkeit und Temperatur

Auch wenn Luftfeuchtigkeit und Temperatur Indikatoren für menschliche Präsenz in einem Raum sein können, unterscheidet sich deren Aussagekraft von der des CO2-Gehaltes der Raumluft in einem wichtigen Faktor.
Sie werden beide stark von äußeren Umweltfaktoren wie dem aktuellen Wetter oder der Jahreszeit beeinflusst. Beide Werte wurden zunächst in das Training aller Modelle miteinbezogen; es wird allerdings im folgenden Kapitel gezeigt, dass diese beiden Werte zusammen mit dem CO2-Wert wenig aussagefähig sind und deshalb später nicht weiter genutzt werden.

3.4. Sensordaten

Wie bereits beschrieben, wurden die Sensordaten in mehreren Räumen der FH Aachen kontinuierlich gesammelt. Durch das Filtern nach den einzelnen Räumen kann für jeden Raum ein Datenset nach folgender Struktur erzeugt werden:

Tabelle 3.1.: Sensordaten

Sensordaten		
Name	Format	Beschreibung
timestamp	timestamp	Zeitpunkt der Messung
co2_ppm	integer	CO2-Wert
temperature_celsius	float	Temperatur in Grad Celsius
relative_humidity_percent	float	Luftfeuchtigkeit
presence	boolean	Aktivität Bewegungssensor

4. Technische Umsetzung

4.1. Datenbeschaffung und -Vorbereitung

Die Daten wurden lokal auf einem der FH-Server in Form eines *Hadoop Distributed File System* (HDFS) gespeichert. Mithilfe von Apache Drill waren die Daten jederzeit durch einfache SQL-Abfragen abrufbar. Die Daten wurden so während der gesamten Dauer des Projektes immer aktuell gehalten, damit alle Erkenntnisse jeweils auf der aktuellsten Datenlage basieren.

Die Datenvorbereitung (Pre-Processing) ist einer der wichtigsten Schritte bei der Anwendung von Machine Learning. Dadurch besteht die Möglichkeit, beim Training des Modells durch Bearbeitung bestehender Spalten oder Hinzufügen zusätzlicher Spalten im Datenset Schwerpunkte zu setzen, die es den Algorithmen beim Training zum einen erleichtern, ihre Erwartungen zu präzisieren, zum anderen aber auch deren Leistung bei der Verarbeitung bestimmter Spalten zu steigern.

4.1.1. Gruppierung

Die Sensordaten wurden alle sechs Sekunden erfasst. Da sich weder CO2-Gehalt noch Feuchtigkeits- oder Temperaturwerte der Raumluft so schnell nicht verändern, wurden die Daten direkt beim Drill per SQL zu zwei-Minuten-Intervallen zusammengefasst. Dabei werden über alle Spalten hinweg Durchschnittswerte gebildet, die dann zu einem Datensatz zusammengefasst werden. Dies steigert die Leistung aller Algorithmen erheblich, weil sich dadurch die Zahl der Datensätze stark verringert. Da sich, wie oben erwähnt, der CO2-Gehalt der Raumluft in einem Intervall von zwei Minuten kaum merklich verändert, verringert sich die Genauigkeit des gesamten Datensets dadurch nicht maßgeblich.

4.1.2. Zyklische Codierung

Zyklische Codierung wird immer dort verwendet, wo Daten sich in wiederholenden Schemata bewegen. Diese Schemata, wie z.B. die Zahlenumbrüche bei einer Uhrzeit, sind für Algorithmen nicht direkt ersichtlich und sind zudem für Computer nicht leicht zu verarbeiten. Durch eine Encodierung in Sinus- und Cosinus-Werte können diese Zusammenhänge vereinfacht werden.
Hierzu wurde der Timestamp zuerst in Sekunden übersetzt, sodass sich ein bestimmter Zeitpunkt eines Tages immer zwischen 0 und 86399 Sekunden bewegt. Aus diesem Wert wurden dann zwei neue Datenspalten „*hour_sin*" und „*hour_cos*" in das Datenset eingefügt, welche sich aus

$$hour\_sin = sin(2 * \pi * x/x_{max}) \tag{4.1}$$

$$hour\_cos = cos(2 * \pi * x/x_{max}) \tag{4.2}$$

ergeben. So kann jede Tageszeit einer eindeutigen Kombination aus Sinus- und Cosinus-Werten zwischen 0 und 1 zugeordnet werden.

4.1.3. Deltas und Shift-Werte

Desweiteren wurden von den Spalten „*co2_ppm*", „*temperature_celsius*" und „*relative_humidity_percent*", die tatsächlich Rückschlüsse auf die Präsenz zulassen, zusätzliche Delta- und Shift-Spalten angelegt.

Ein Shift-Wert bedeutet lediglich, dass in einer Zeile x_n des Datensets zusätzlich, neben den aktuellen Werten, auch Werte von k Zeilen zuvor, also x_{n-k} stehen. So haben alle Algorithmen direkten Zugriff auf Vergangenheitswerte der ausgewählten Spalten.

Delta-Spalten stellen, dem Namen nach, Deltas zu vorherigen Werten dar:

$$\Delta x_k = x_n - x_{n-k} \tag{4.3}$$

Die Erwartung ist hier, dass die Änderung der CO2-, Temperatur- und Luftfeuchtigkeitswerte ein wichtigerer Indikator sein könnte, als die tatsächlichen Werte. In einem schlecht klimatisierten Raum könnten Grundwerte von z.B. CO2 höher sein, als in Räumen, in denen regelmäßig gelüftet wird. Unabhängig dieser Grundwerte könnte man bei einem starken Anstieg wesentlich einfacher menschliche Präsenz erkennen, ohne dass zunächst die vorangegangenen Werte überprüft werden müssen. Durch die hinzufügten Deltas werden diese Grundwerte ignoriert und Rückschlüsse auf die aktuelle Präsenz sind besser möglich.
Im Zuge der Projektarbeit wurden verschiedene Kombinationsmöglichkeiten von Delta- und Shiftwerten mit Zeitschritten zwischen zwei Minuten und einer Stunde mit Hinblick auf Verbesserungen der Modell-Genauigkeiten getestet.

4.1.4. Outlier Detection

Wie bereits erwähnt, spielen Datenausreißer für die Ergebnisse mancher Algorithmen eine große Rolle. Überall wo z.B. aus einer Reihe von Datenwerten Durchschnittswerte berechnet werden, würden Ausreißer in den Daten das Ergebnis verfälschen und die Aussagekraft des Algorithmus deutlich senken. Um diese Ausreißer vor dem Training der Modelle zu beseitigen, wurde das Verfahren des *Interquartilabstands* (IQR nach der englischen Bezeichnung *Interquartile Range*) gewählt.
Der IQR gibt die Intervallgröße an, die ein Wert vom Median einer Datenreihe abweichen darf. Bei einer der Größe nach sortierten Datenreihe $x = (x_0, x_1, ..., x_n)$ bestimmt man die Mediane der unteren und oberen Hälfe des Datensets Q_1 und Q_2. Der IQR ergibt sich nun aus

$$IQR = Q_2 - Q_1 \tag{4.4}$$

Mit diesm Wert können jetzt die erlaubten Ober- und Untergrenzen des Datensets mit

$$Limit_{upper} = Q_2 + 1.5 * IQR \tag{4.5}$$
$$Limit_{lower} = Q_1 - 1.5 * IQR \tag{4.6}$$

bestimmt werden. Alle Werte die außerhalb dieser Grenzen liegen, können als Ausreißer betrachtet werden.
Ausreißer zu entfernen, ist hier wichtig, da die Sensoren Messfehler erzeugen können und einige Modelle dadurch, wie bereits beschrieben, stark beeinflusst werden können.

4.2. Validierung

Über das Projekt hinweg wurden verschiedene Validierungsmethoden verwendet. Grundsätzlich unterteilt man das Datenset in ein Trainings- und Testset mit einem Verhältnis von etwa 80-20. Das bedeutet, dass das Modell mit 80 Prozent der Daten trainiert wird, wobei die anderen 20 Prozent zurückgehalten werden, um daraufhin das fertig trainierte Modell daran zu testen. Aus dem Ergebnis dieses Tests ergeben sich verschiedene Werte, die die allgemeine Genauigkeit des Modells repräsentieren, d.h. wie verlässlich das Modell die An- und Abwesenheit des Testsets selbst berechnen kann, wenn man ihm den tatsächlichen Wert vorenthält.
Allgemeine Basis für die errechneten Genauigkeiten bildet die sog. *Confusion Matrix*.

Confusion Matrix		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

Abbildung 4.1.: Beispiel einer Confusion Matrix

Diese zeigt, inwiefern die Genauigkeit des Modells neben tatsächlich richtig erkannten Werten zusätzlich von false negatives und false positives beeinflusst wird.

Die *Accuracy Score* ist eine einfache Methode zur Bestimmung der Modellqualität und berechnet sich aus der Summe der richtigen Ergebnisse geteilt durch die Summe aller Ergebnisse.

$$AccuracyScore = \frac{TP + TN}{TP + TN + FP + FN}$$

Die *Recall Score* beschreibt die Genauigkeit bezogen auf alle erkannten positiven Ergebniswerte.

$$RecallScore = \frac{TP}{TP + FN}$$

Die *Precision Score* gibt die allgemeine Menge an positiven Werten an, die hätten erkannt werden müssen.

$$PrecisionScore = \frac{TP}{TP + FP}$$

Die *F1-Score* ist der Durchschnittswert aus Recall- und Precision-Score und liefert im Allgemeinen eine gute Auskunft über die Qualität eines Modells.

$$F1\_Score = \frac{2}{\frac{1}{PrecisionScore} + \frac{1}{RecallScore}} = \frac{2 \cdot (PrecisionScore \cdot RecallScore)}{PrecisionScore + RecallScore}$$

In diesem Anwendungsfall ist es durchaus wichtig, Werte zur Anwesenheit von Personen genauer zu betrachten als Werte zur Abwesenheit, da die tägliche Arbeitszeit nur etwa ein Drittel eines ganzen Tages ausmacht. Da von 24 Stunden ca. 16 Stunden Werte zur Abwesenheit und nur etwa 8 Stunden Werte zur Anwesenheit enthalten, ist das Datenset also mit einem Verhältnis von etwa 1/3 in Richtung der Abwesenheit unausgeglichen. Dies lässt die Erwartung zu, dass das trainierte Modell wesentlich besser darin sein wird, Abwesenheit von Personen zu erkennen, als Anwesenheit.
Hierzu wurde bei der Auswertung der Ergebnisse immer auch der *Classification Report* hinzugezogen, aus dem ersichtlich ist, wie genau das Modell beide möglichen Labelwerte berrechnen konnte.

Desweiteren war es entscheidend zu erkennen, dass das Datenset über mehrere Monate hinweg nicht perfekt uniform ist, da in bestimmten Monaten verhältnismäßig viele Urlaubstage (z.B. Weihnachten/Silvester) vorkommen und es damit zu einer Häufung an Abwesenheitswerten kommt. Wenn die Daten nun im o.g. Verhältnis aufgeteilt werden und dann viele Daten des Testsets in solchen Monaten liegen, könnte es beim Ergebnis zu Verzerrungen kommen.
Hierfür wurde eine *Kreuzvalidierung* (engl. Cross Validation CV) implementiert. Bei einer CV wird das Datenset immer noch im gleichen Verhältnis aufgeteilt, allerdings mehrmals, sodass die Trainings- und Testsets jedes Mal aus jeweils anderen Daten bestehen.

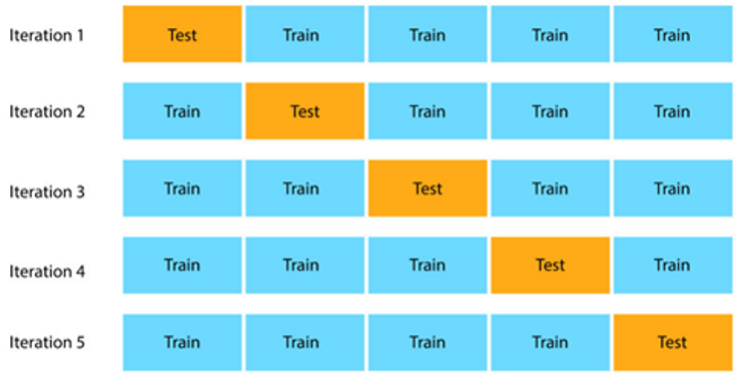

Abbildung 4.2.: Beispiel einer Kreuzvalidierung

Errechnet man nun die Durchschnittsgenauigkeit aller Iterationen der CV, ergibt sich eine Gesamtgenauigkeit, die das Model besser repräsentiert, weil es mit einer größeren Anzahl von verschiedenen Daten trainiert und getestet wurde.

Da es für manche Räume keine Datensätze mit Labelwerten gab, musste die Validierung dieser Datensätze augenscheinlich erfolgen. Hierfür wurde in ein Datenset eine Labelspalte eingefügt, die dann vom trainierten Modell selbst gefüllt werden sollte. Das Ergebnis musste schließlich als Graph gezeichnet und per Hand validiert werden. Da sich das Ergebnis der Berechnung in diesem Anwendungsfall, wie oben gezeigt, sehr übersichtlich als Graph darstellen lässt, war diese Methode der Validierung zwar nicht perfekt, lieferte aber trotzdem einen ausreichenden Eindruck über die Qualität des trainierten Modells.

4.2.1. Over- und Underfitting

Over- und Underfitting sind Probleme, die bei allen Machine Learning Algorithmen auftreten können und bezeichnen im Allgemeinen einen falschen Umgang mit dem Datenset. *Overfitting* bedeutet, dass das Modell mit zu vielen, sich zu stark ähnelnden, Daten trainiert wurde, wodurch es in der Lage ist, nur neue Daten richtig zu kategorisieren, die dem Trainingsset besonders ähnlich sind. Overfitting kann während einer Kreuzvalidierung oder der Validierung anhand eines anderen Datensets als dem Trainingsset erkannt werden.

Um Overfitting zu verhindern, kann das Datenset vereinfacht werden, um so die Beziehungen zwischen den für die Erwartungsberechnung relevanten Spalten zu stärken.

Underfitting bedeutet dagegen, dass es das Modell nicht geschafft hat, zwischen den gegebenen Daten Beziehungen herzustellen, sodass das Modell unbekannte Daten richtig kategorisieren kann. Hier reicht es normalerweise aus, dem Modell mehr Daten zur Verfügung zu stellen und in der Datenvorbereitung Felder einzufügen, die dem Modell bestimmte Beziehungen zwischen Datenfeldern hervorheben sollen.

4.2.2. Parameter Tuning

Parameter Tuning beschreibt einen Schritt der Modell-Optimierung, der normalerweise stattfindet, nachdem ein funktionierendes Modell erstellt wurde, das mit bereits verarbeiteten Daten gute Ergebnisse liefert. Jedes Modell besitzt Parameter, die bei der Erstellung festgesetzt werden. Diese Parameter haben großen Einfluss auf den Trainingsprozess, weshalb es sich anbietet, die Genauigkeiten mehrerer Modelle mit verschiedenen Parameter-Kombinationen zu testen.
Das im Scikit-Learn enthaltene *GridSearchCV*-Modul bietet die Möglichkeit, einem Modell eine Vielzahl von verschiedenen Parameter-Optionen zu übergeben. Mit diesen werden dann automatisch Modelle trainiert und per Kreuzvalidierung verglichen. Am Ende liefert das Modul die Parameter-Kombination, mit der das beste Ergebnis erzielt wurde.

```
param_test = {
    'n_estimators':[25, 50, 100, 250],
    'max_depth': [2, 5, 10, 15],
    'min_samples_split':[25, 150, 500],
    'min_samples_leaf':[50, 75, 125],
    'max_features':[5, 6, 7, 8, 9],
    'subsample':[0.75, 0.9, 0.95]
     }
```

Abbildung 4.3.: Beispiel einer Sammlung von Parameter-Optionen

Im obenstehenden Beispiel kann man erkennen, dass für bestimmte Parameter (rot) eine Sammlung an erlaubten Werten (grün) übergeben wird. Alle Kombinationen aus diesen Parameter-Optionen werden dann vom GridSearchCV-Modul ausgewertet.
Die Verbesserungen gegenüber Standard-Parametern bewegen sich normalerweise im niedrigen einstelligen Prozent-Bereich.

5. Anwendung und Ergebnisse

5.1. Feature Vektor

Der Feature Vektor FV beschreibt das Ergebnis des Datensets nach der Vorverarbeitung. Die benutzten Felder werden bei jedem Anwendungsfall zunächst vom Programmierer selbst gewählt. Durch die o.g. Schritte ergibt sich eine Vielzahl an möglichen zusätzlichen Features, um die das Datenset erweitert werden kann. Es muss also zunächst ermittelt werden, welche dieser möglichen Features wirklich aussagekräftig sind, um den FV nicht unnötig zu überladen.
Einige Algorithmen geben nach dem Training Auskunft darüber, in welchem Maß ein Feature in die Berechnung des Labels einfließt. Somit wurden zur Wahl eines FVs, der nach dem Training mit einem Modell eine möglichst hohe Genauigkeit liefert, verschiedene Kombinationen von möglichen Feldern durch Verknüpfung von Temperatur, Luftfeuchtigkeit, CO2 mit jeweiligen Delta- und Shiftwerten vorgenommen. Anschließend werden die Modellgenauigkeiten einzelner FVs miteinander verglichen.
Ziel war es, mit einem möglichst kleinen Vektor eine möglichst hohe Genauigkeit zu erreichen, da bei den meisten Algorithmen eine direkte Abhängigkeit zwischen Dauer der Berechnung und Dimensionalität des FV besteht.

In diesem Fall wurden die Feature Importances eines Random Forest genutzt, um Auskunft darüber zu erhalten, wie relevant bestimmte Felder für die Erwartungsberechnung sind. Diese werden errechnet, indem geprüft wird, wie viele Datensätze des Sets bestimmte Knoten des Baumes erreichen. Wenn ein beliebiger Decision Tree eine große Menge des Datensets allein über seinen rechten Teilbaum klassifizieren kann, werden die darin enthaltenen Knoten schwerer gewichtet, da sie in einem höheren Maß in die endgültige Entscheidung einfließen.

Diese Eigenschaft wurde zur Ermittlung des FVs genutzt, indem verschiedene Modelle mit einem FV-Kandidaten trainiert wurden und anschließend sowohl die Genauigkeiten, als auch die Feature Importances gegenübergestellt wurden. Features mit geringer Relevanz wurden entfernt und die Modellgenauigkeiten erneut geprüft, um so über mehrere Iterationen hinweg einen FV zu finden, dessen Felder möglichst hohe Aussagekraft besitzen, während die Anzahl nötiger Felder möglichst gering ist.

Hierzu wurden die Sensordaten zunächst um Delta- und Shiftwerte erweitert, um den Modellen die Möglichkeit zu geben die aktuellen Werte den vorangegangenen Messungen gegenüberzustellen. Mit diesem FV wurde dann ein Random Forest trainiert und dessen Feature Importances ausgewertet.

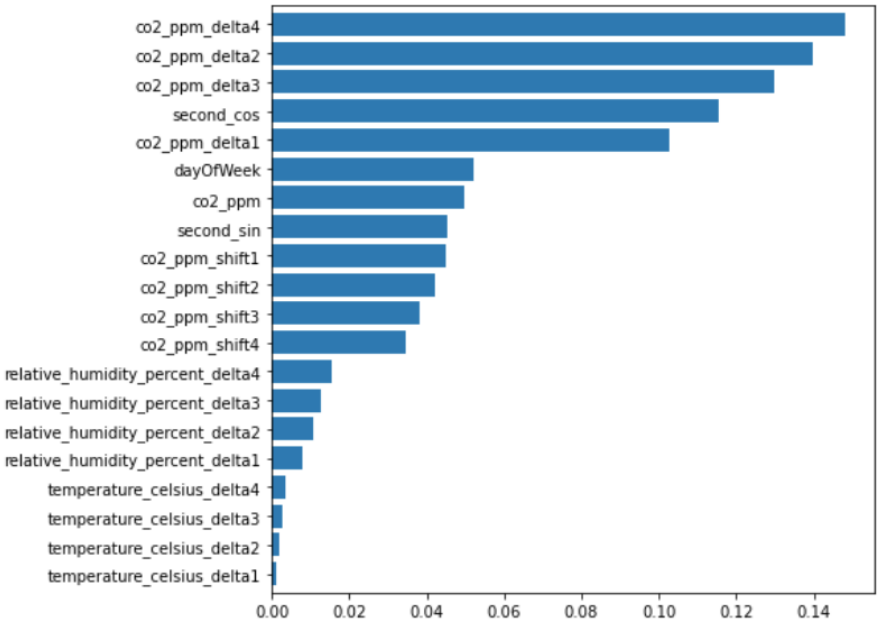

Abbildung 5.1.: Feature Importances eines Random Forest

Wie bereits im vorherigen Kapitel angedeutet, sieht man in Abb. 5.1, dass die Temperatur- und Luftfeuchtigkeitswerte für die Errechnung der Labels kaum zu Rate gezogen werden. Weder Grund- noch Deltawerte weisen eine hohe Relevanz für das Modell auf.
Es ist anzunehmen, dass Temperatur und Luftfeuchtigkeit an sich durchaus nützlich für solche Betrachtungen sein können, allerdings werden sie durch die CO2-Werte in ihrer Relevanz übertroffen, da CO2 im Kontext menschlicher Präsenz in Innenräumen eine deutlich höhere Aussagekraft besitzt.
Die CO2-Werte sind für die Modelle einfacher zu klassifizieren. Die Messwerte eines CO2-Sensors sind unabhängig von äußeren Einflüssen wie Jahreszeiten oder dem aktuellen Wetter und sind deshalb, wie in Abb. 5.1 gezeigt, entscheidender Faktor bei der Berechnung.

Nach anschließender Exkludierung von Temperatur und Luftfeuchtigkeit wurden neue Deltas und Shifts für die CO2-Werte eingefügt und die Feature Importances erneut betrachtet.

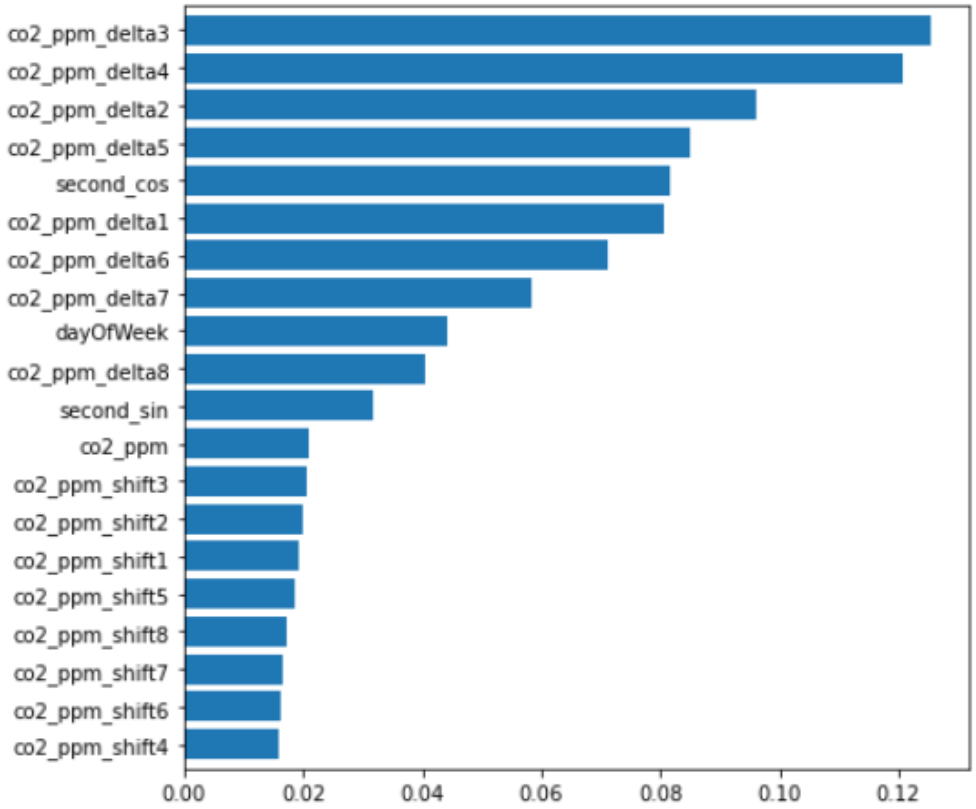

Abbildung 5.2.: Feature Importances eines Random Forest

Es ist in Abb. 5.2 zu erkennen, dass die CO2-Shift-Werte ebenfalls nicht maßgeblich in die Berechnung einfließen. Diese weisen allerdings noch deutlich höhere Feature Importances auf, als die Temperatur- und Luftfeuchtigkeitswerte.
Der Graph bestätigt die Annahme, dass beim CO2 die Deltawerte deutlich ausschlaggebender sind, als die tatsächliche Messung zu einem bestimmten Moment.

Wenn diese Werte nur in solch geringem Maß zur Berechnung beitragen, muss ebenfalls die Schlussfolgerung daraus, dass die Genauigkeit auch ohne diese Werte innerhalb einer gewissen Toleranz konstant bleibt, überprüft werden.
Im Folgenden werden beispielhaft vier verschiedene Feature Vektoren beschrieben, aus denen der Vektor, der im weiteren Verlauf des Projektes benutzt werden sollte, abgeleitet wurde.

Tabelle 5.1.: Vergleich Feature Vektoren

Feature Vektor	Beschreibung
FV1	CO2, Temperatur und Luftfeuchtigkeit mit Shift-Werten
FV2	CO2, Temperatur und Luftfeuchtigkeit ohne Shift-Werte
FV3	Nur CO2 mit Shift-Werten
FV4	Nur CO2 ohne Shift-Werte

Mit diesen vier Vektoren wurden nun verschiedene Modelle trainiert und deren Genauigkeiten gegenübergestellt.

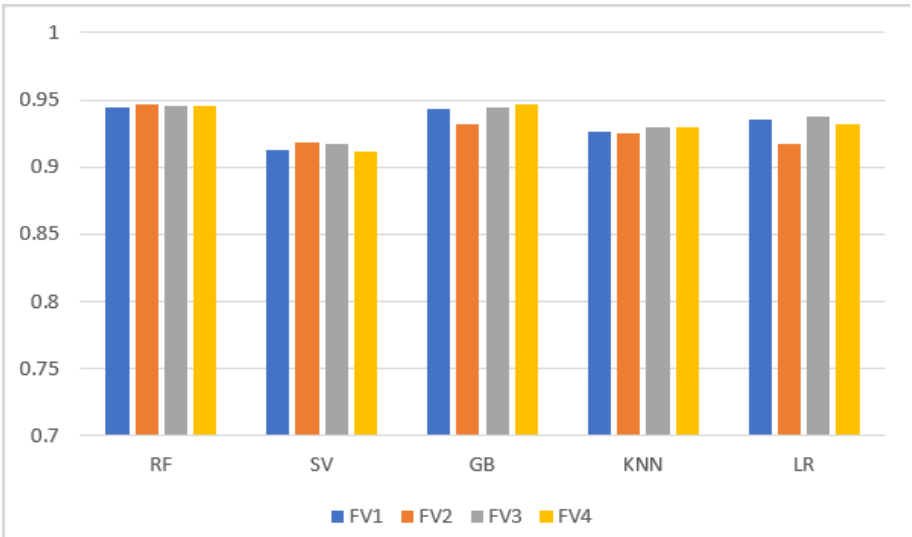

Abbildung 5.3.: Vergleich der Genauigkeiten

Wie deutlich sichtbar ist, sind die Genauigkeiten fast identisch. Diese Gegenüberstellung wurde über das Projekt hinweg mehrfach wiederholt und ergab immer ähnliche Unterschiede.
Dies war Anlass, sowohl die Temperatur- und Luftfeuchtigkeitswerte, als auch die Shift-Werte der CO2-Messungen nicht weiter zu betrachten, um den FV für Modelle mit hoher Empfindlichkeit gegenüber der Dimensionalität zu optimieren.

Aus diesen Betrachtungen ergab sich folgender FV, der für den weiteren Verlauf des Projektes genutzt wurde:

Tabelle 5.2.: Feature Vektor

Feld	Beschreibung
second_sin	timestamp-Sinusanteil
second_cos	timestamp-Cosinusanteil
day_of_week	Wochentag der Messung
co2_ppm	CO2-Wert
co2_ppm_delta1	Delta zum CO2-Wert vor 2 Minuten
co2_ppm_delta2	Delta zum CO2-Wert vor 4 Minuten
co2_ppm_delta3	Delta zum CO2-Wert vor 6 Minuten
co2_ppm_delta4	Delta zum CO2-Wert vor 8 Minuten
co2_ppm_delta5	Delta zum CO2-Wert vor 10 Minuten
co2_ppm_delta6	Delta zum CO2-Wert vor 12 Minuten
co2_ppm_delta7	Delta zum CO2-Wert vor 14 Minuten
co2_ppm_delta8	Delta zum CO2-Wert vor 16 Minuten

5.2. Ergebnisse

Die ausgewählten Algorithmen konnten mit dem ausgewählten FV bei der Erwartungsberechnung über das gesamte Datenset eine Genauigkeit von etwa 94% erreichen.
Eine *Confusion Matrix* (Abb. 5.4), welche die errechneten Werte den tatsächlichen Labelwerten gegenüberstellt, zeigt, dass im hier benutzten Beispiel ein Random Forest Classifier selbst bei der starken Unausgeglichenheit des Datensets ähnlich viele Fehler bei sowohl An- als auch Abwesenheit von Personen macht.
In den Feldern oben links und unten rechts ist jeweils zu sehen, wann das Modell eine richtige Erwartung für jeweils Ab- und Anwesenheit errechnet hat, während die Felder oben rechts und unten links jeweils die Menge falscher Erwartungen für beide Werte zeigen.

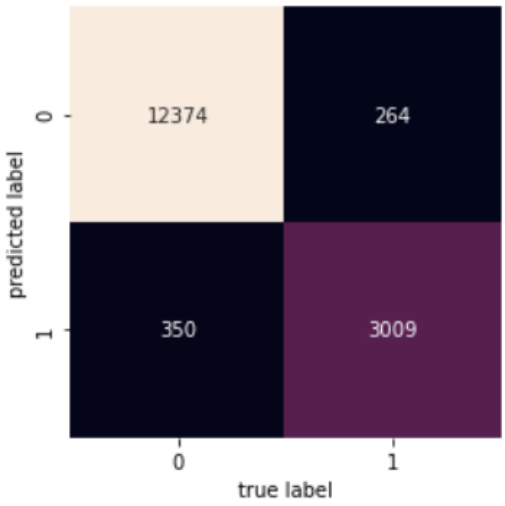

Abbildung 5.4.: Confusion Matrix eines RFC

Bezogen auf die Fehlerquote ist allerdings zu erkennen, dass die Berechnung von Abwesenheit mit wesentlich höherer Genauigkeit erfolgt, als die Berechnung von Anwesenheit. In Abb. 5.4 liegt die Genauigkeit Für Abwesenheit etwa bei 97.5%, während die Genauigkeit für Anwesenheit bei lediglich ca. 91.9% liegt.

Dies liegt daran, dass sehr viele Datenpunkte für Abwesenheit zwischen ca. 18:00 Uhr und 07:00 Uhr des nächsten Tages aufgezeichnet werden. Während dieser Zeit sind nie Personen anwesend, weshalb die Modelle diesen Teil des Datensets leicht klassifizieren können.
Während dem Arbeitstag zwischen ca. 07:00 Uhr und 17:00 Uhr existieren sowohl Messwerte für An- als auch für Abwesenheit, weshalb es hier deutlich schwieriger ist den richtigen Labelwert zu errechnen.

Dieser Trend lies sich bei allen Modellen beobachten. Die Genauigkeit bei der Errechnung von Abwesenheit lag immer höher, als bei Anwesenheit.

Die Gegenüberstellung der Durchschnittsgenauigkeiten aller Modelle mit Precision-, Recall- und F1-Scores in Abb. 5.5 zeigt, dass die Decision Tree Modelle bei diesem Klassifikationsproblem mit sehr hoher Genauigkeit sowohl An-, als auch Abwesenheit berechnen können. Zusätzlich wies das KNN Clustering ähnlich hohe Genauigkeiten auf.

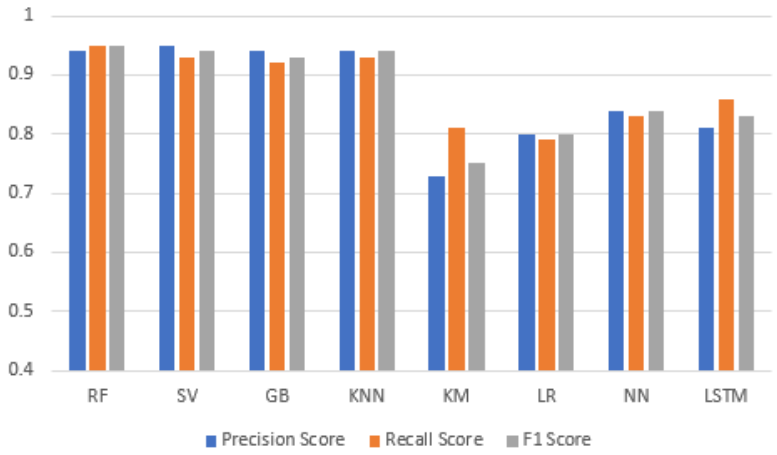

Abbildung 5.5.: Durchschnittsgenauigkeiten aller Modelle

Als Metrik wurde hier der Durchschnitt der Genauigkeiten für An- und Abwesenheit gewählt, da die Datensets wie bereits beschrieben An- und Abwesenheit stark unausgeglichen sind. Wenn man die Genauigkeiten nach der Menge der jeweiligen Datenpunkte (s. Abb. 5.4) gewichtet, erhöht das die Genauigkeit aller Modelle deutlich, ohne dass sich deren Aussagekraft steigert.

Dieser Sachverhalt zeigt sich besonders deutlich durch die Darstellung der tatsächlichen Genauigkeiten bezogen auf die beiden Labelwerte in Abb. 5.6. Alle Modelle sind deutlich besser in der Erkennung von Abwesenheit.

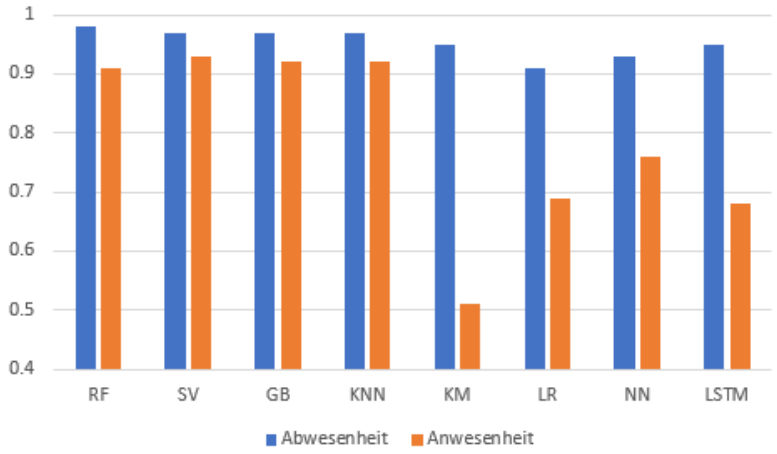

Abbildung 5.6.: Genauigkeiten einzelner Labelwerte aller Modelle

Besonders auffällig ist die niedrige Genauigkeit des K-Means Modells bei der Erkennung von Anwesenheit. Diese wird in 5.2.3 genauer betrachtet.

Ebenfalls aufschlussreich, ist die grafische Gegenüberstellung der erwarteten Labelwerte mit den tatsächlichen Labelwerten indem man das ursprüngliche Datenset mit *timestamp* und *CO2-Wert* zeichnet. Dabei markiert gelb die Anwesenheitund blau die Abwesenheit von Personen.

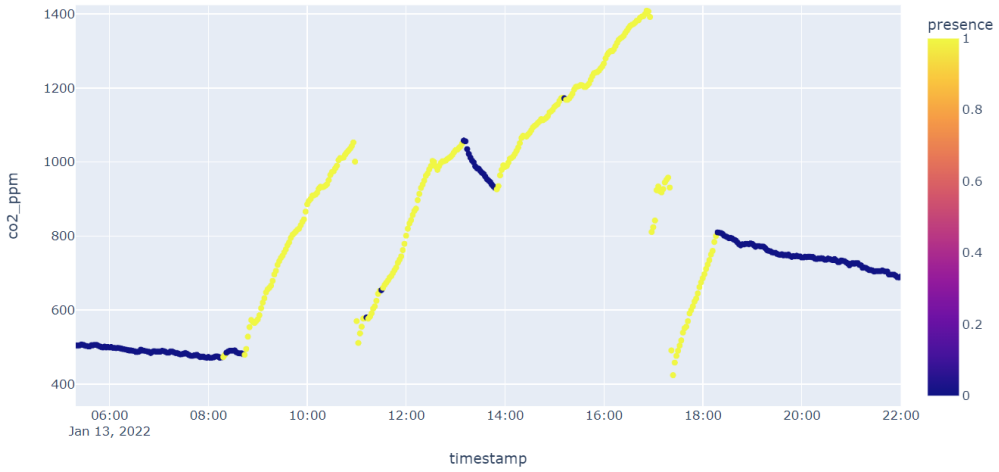

Abbildung 5.7.: Messwerte des 13. Januar

5.7 zeigt den Verlauf der CO2-Messungen über einen Arbeitstag. Die blaue und gelbe Einfärbung stellt die Messung des Infrarotsensors dar.
Ersetzt man aus dieser Datenreihe die tatsächlichen Messwerte des Infrarotsensors durch die errechneten Anwesenheitswerte des Modells, zeigt sich die hohe Genauigkeit in 5.8 durch die neue Einfärbung der Datenreihe deutlich. Bis auf einige kleine Fehler trifft das Modell eine sehr präzise Aussage über die aktuelle Anwesenheit.

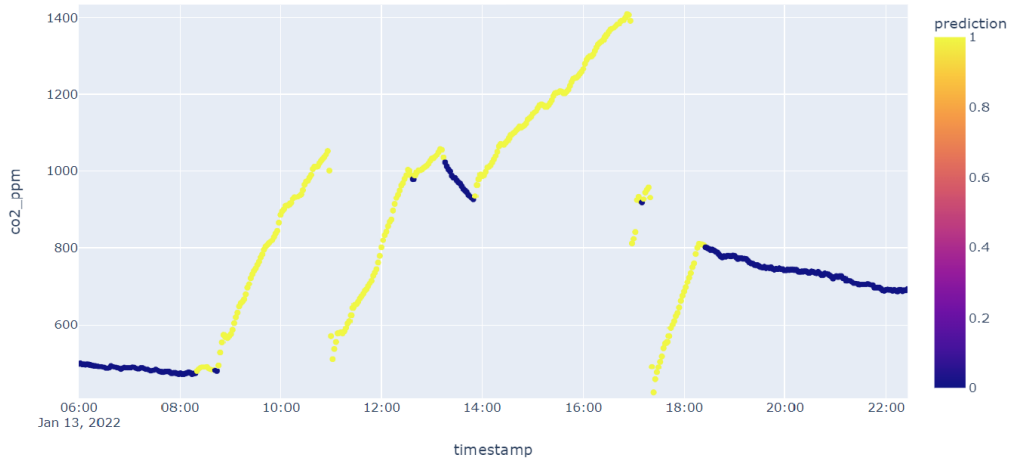

Abbildung 5.8.: Erwartungsberechnung des 13. Januar

Dieses Ergebnis konnte über weitere Kombinationen verschiedener Räume und Modelle weiterhin bestätigt und repliziert werden.

Beim Parameter Tuning der ausgewählten Modelle wurden als Parameter-Optionen ausschließlich Werte gewählt, die nicht den Standardwerten der Modelle entsprechen. Somit sollte das GridSearchCV-Modul ausschließlich in der Menge von Parametern nach Möglichkeiten suchen, die nicht den Standard-Einstellungen entsprechen.
Hier ist anzumerken, dass das K-Means Modell kein Parameter Tuning unterstützt und deshalb hier nicht betrachtet wird. Wie beschrieben, werden beim K-Means Algorithmus zufällige Datenpunkte in das Datenset gelegt, weshalb nicht jede Ausführung des Modells die gleiche Genauigkeit liefert. Es ist deshalb auch durch manuelle Veränderung der Parameter sehr schwer Verbesserungen der Genauigkeit herbeizuführen.
Neuronale Netze tauchen hier ebenfalls nicht auf, da diese Art von Modell vom GridSearchCV-Modul nicht unterstützt wird. Die Layer des NN und LSTM wurden manuell erzeugt, wobei deren Anordnung bereits so gewählt wurde, dass die Genauigkeiten möglichst maximiert werden.

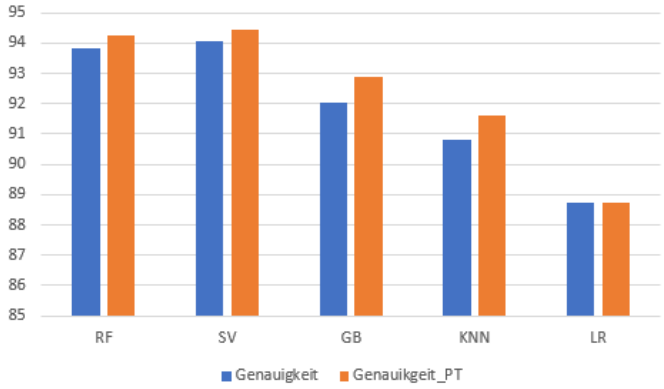

Abbildung 5.9.: Ergebnisse des Parameter Tuning

Es ist zu sehen, dass das Parameter Tuning die Genauigkeit der Modelle im erwarteten Rahmen erhöht oder verringert hat. Dies bestätigt die Annahme, dass dieser Anwendungsfall ein typisches Klassifizierungsproblem darstellt, weshalb die Modelle nicht mehr maßgeblich verbessert werden können. Die Standardparameter der einzelnen Modelle sind im Scikit-learn bereits so eingestellt, dass eine große Zahl an Klassifizierungsproblemen ohne weiteres Tuning zufriedenstellend gelöst werden kann. Der Schritt des Parameter Tunings sollte trotzdem grundsätzlich immer durchgeführt werden, um zu erkennen, ob man durch eine einfache Änderung der Parameter eine Verbesserung des Modells erzielen kann.

Zusätzlich wurden die *ROC-Curves* (ROC: englisch für *receiver operating characteristic* bzw. deutsch Operationscharakteristik) gezeichnet, um zu erkennen, inwiefern die Ergebnisse zufällig oder errechnet sind.
Hierbei berechnet sich die True Positive Rate auf der y-Achse aus

$$\frac{TP}{TP - FN} \tag{5.1}$$

während die False Positive Rate auf der x-Achse durch

$$\frac{FP}{FP + TN} \tag{5.2}$$

angegeben wird.

Der Optimalwert wird hier durch in der grünen Kurve gezeigt, welche den Fall darstellt, dass das Modell die Labels immer richtig berechnet und es nie zu Fehleinschätzungen kommt.
Liegt ein Modell näher an der schwarzen Diagonale in der Mitte bedeutet das, dass True- und False-Positives jeweils 50% der Ergebnismenge ausmachen. Das Modell liegt dann genauso oft richtig, wie es Fehleinschätzungen trifft. Je näher die einzelnen Modelle an der schwarzen Linie liegen, desto weniger aussagefähig sind sie.

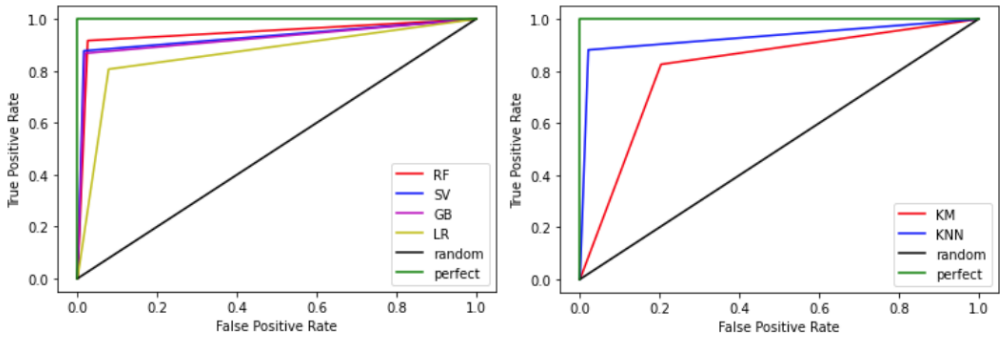

Abbildung 5.10.: ROC-Curves der Decision Tree und Clustering Modelle

In 5.10 ist zu sehen, wie alle Modelle sehr ähnliche Ergebnisse aufweisen. Keines der Modelle weist Anzeichen dafür auf, dass Ergebnisse wegen Underfitting erraten werden. Alle Modelle zeigen eine hohe Rate von True-Positives und sind somit in den meisten Fällen in der Lage, aus den Input-Werten das richtige Ergebnis für die Präsenz abzuleiten. Es ist auch zu erkennen, dass für diesen Anwendungsfall Decision Tree Modelle am besten abschneiden. Von den Clustering Modellen ist das K-Nearest-Neighbours Modell erheblich genauer als das K-Means Modell.

5.2.1. Umgang mit fehlenden Präsenz-Labels

Da zur Auswertung auch eine große Menge an Datensätzen ohne Präsenz-Label zur Verfügung stand, konnte die Qualität der Ergebnisse für diese Datensets nicht rechnerisch quantifiziert werden. Die Ergebnisse sind hierbei aufgrund der fehlenden Label lediglich augenscheinlich zu bewerten, da eine mathematische Berechnung der Genauigkeit nicht möglich ist.
In diesem Kontext stellt sich die Frage, inwiefern ein auf einen Büroraum trainiertes Modell auch richtige Erwartungen für einen Wohnraum treffen kann. Da das Modell unter anderem auf eine Erkennung der aktuellen Tageszeit trainert wird, muss geprüft werden, inwiefern ein solches Modell auch Aussagen über CO2-Werte treffen kann, dessen Veränderungen außerhalb der bisher trainierten Arbeitszeiten von etwa 7 Uhr morgens bis 17 Uhr nachmittags liegen. Außerdem ist der Wohnraum in der Regel deutlich größer als ein Büro und weist in diesem Fall eine größere Anzahl anwesender Personen auf, wodurch sich die ermittelten Deltas der CO2-Werte erheblich von denen der Büroräume unterscheiden.

Eines der Datensets zeichnete kontinuierliche Werte aus einem Wohnzimmer auf, in dem sich über einen Tag hinweg maximal drei Personen befanden. Die Zeiträume, zu denen in diesem Wohnzimmer anhand der CO2-Werte Anwesenheit zu erkennen war, unterschieden sich deutlich von denen der Datensets der FH Aachen. Die starken Anstiege des CO2-Wertes waren bei diesem Datenset viel mehr über den Tag verteilt, wodurch sich die Datenreihe für diesen Test besonders anbot.
Auf dieses Datenset wurde das auf die Büroräume trainierte SVC Modell angewendet, da dieses bisher die besten Ergebnisse bei der Unterscheidung von An- und Abwesenheit während eines Arbeitstages liefern konnte.

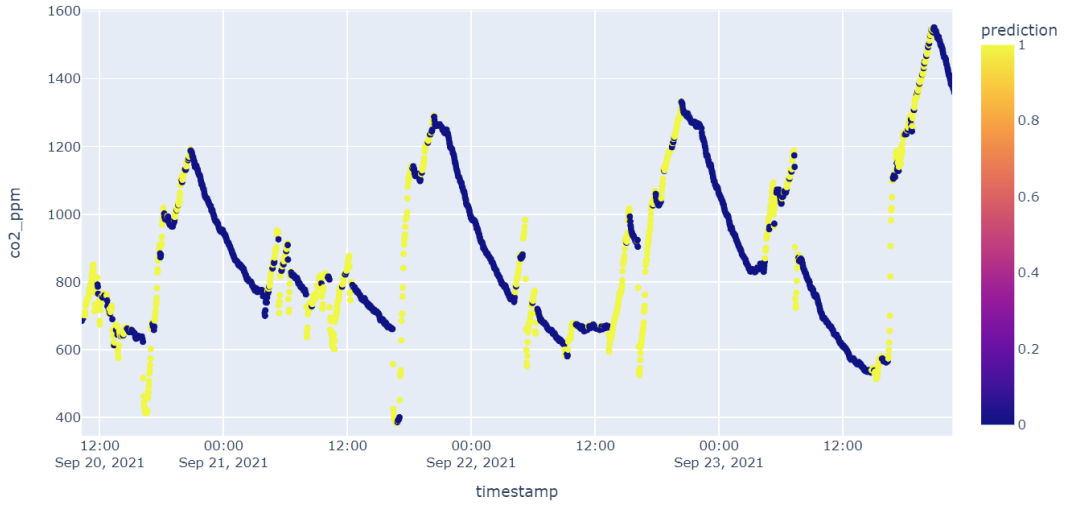

Abbildung 5.11.: Büroraum-Modell trifft Erwartungen für Wohnzimmer

Auch hier wurden mit den errechneten Anwesenheitswerten des SVC Modells die Einfärbungen der einzelnen Datenpunkte vorgenommen. Wie in 5.11 anhand der

Einfärbung zu erkennen ist, liegt die Vermutung nahe, dass das Modell auch unabhängig von der Tageszeit und den bekannten Gegebenheiten eines normalen Arbeitstages Anwesenheit erkennen kann. Vor allem deutliche Anstiege und Abfälle zeigen sich merklich durch kontinuierliche gelbe bzw. blaue Einfärbung.

5.2.2. Neuronale Netzwerke

Die beiden implementierten neuronalen Netze konnten eine ähnliche Leistung wie die bereits oben genannten Algorithmen erzielen. Beide Modelle wurden über so viele Iterationen (Epochen) trainiert, bis ersichtlich war, dass die Genauigkeit gegen einen Wert konvergiert. Zur Auswertung wurden hier die *Accuracy*- und *Loss*-Funktionen genutzt. Die Genauigkeit beschreibt wie bei den anderen Modellen das Verhältnis zwischen richtigen und falschen Berechnungen.
Die Loss-Funktion hingegen gibt die Abweichung einer Schätzung zum tatsächlichen Wert an. Diese beiden Werte werden nach jeder Epoche ausgewertet, wobei zunächst der Erfolg während des Trainigs als *train* angegeben wird und danach das Modell an einem zufälligen Teil des Validierungssets *val* getestet wird. Liegen diese beiden Kurven nah beieinander, kann dies als ein klares Anzeichen für ein funktionierndes Modell ohne Over- oder Underfitting gewertet werden, da sowohl beim Training als auch bei der Validierung die Genauigkeiten Ähnlichkeiten aufweisen, während zugleich die Loss-Funktion minimiert wird.

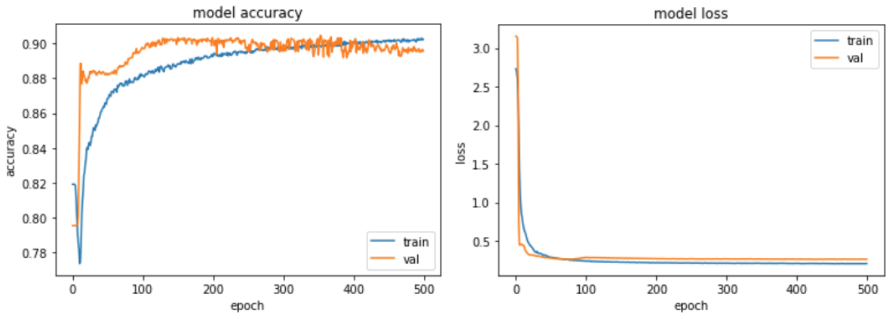

Abbildung 5.12.: Ergebnisse des NN

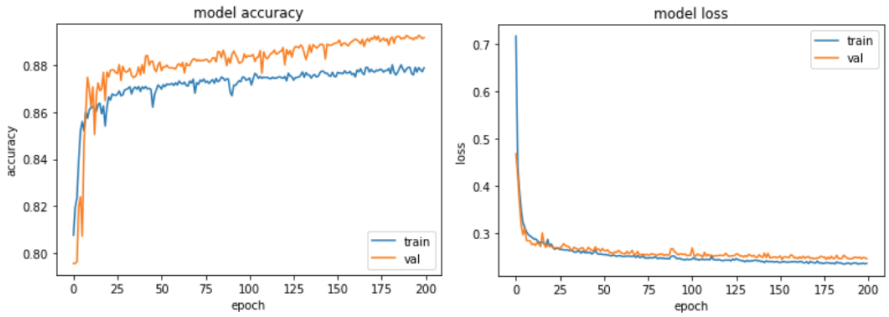

Abbildung 5.13.: Ergebnisse des LSTM

Wie 5.12 und 5.13 zeigen, weisen beide Arten neuronaler Netzwerke diese Eigenschaften auf. Die Genauigkeiten beider Modelle konvergierten auch nach verschiedenen Epochen-Werten etwa bei 89%.

Die ROC-Curves zeigen auch hier, dass die durchschnittliche Modellgenauigkeit nicht als Metrik ausreicht, um die Qualität eines Modells festzustellen. Bei diesen Modellen zeigen sich deutlich größere Abweichungen zwischen den Genauigkeiten für die Berechnung von Ab- und Abwesenheit. Wie in Abb. 5.14 zu sehen, liefern beide Modelle hohen Genauigkeiten bei der Errechnung von Anwesenheit.

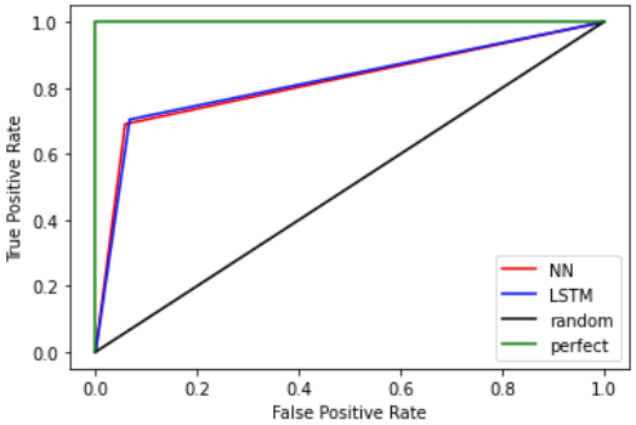

Abbildung 5.14.: ROC-Curves aller Modelle

Zieht man den Classification Report in Tab. 5.3 hinzu, wird dieser Sachverhalt noch deutlicher.

Tabelle 5.3.: Genauigkeitsauswertung des NN

Präsenz	Precision	Recall	F1
Abwesenheit	0.96	0.92	0.94
Anwesenheit	0.74	0.83	0.78

Tabelle 5.4.: Genauigkeitsauswertung des LSTM

Präsenz	Precision	Recall	F1
Abwesenheit	0.93	0.93	0.93
Anwesenheit	0.73	0.72	0.72

Die in den Abb. 5.12 und 5.13 errechnete Durchschnittsgenauigkeit von ca. 89% ist in Richtung der Abwesenheitswerte gewichtet. Auch neuronale Netze scheinen, ähnlich die das K-Means Modell, Probleme bei der Unterscheidung von An- und Abwesenheit während des Arbeitstages zu haben. Nur bei der Klassifizierung der Abwesenheitswerte außerhalb des Arbeitstages liefern diese Modellen befriedigende Ergebnisse.

5.2.3. Ergebnisse des K-Means Modells

Von allen benutzten Modellen ergaben sich bei dem K-Means Modell die größten Schwierigkeiten bei der Klassifikation des Datensets. Während die Genauigkeit bei der Erwartungsberechnung von Abwesenheit bei 94% lag, konnte das Modell Anwesenheiten nur zu 50% erkennen, was bedeutet, dass das Modell das Ergebnis errät, anstatt es zuverlässig zu berechnen.
Wegen der Unausgeglichenheit des Datensets sind ca. zwei Drittel des Datensets sehr einfach zu klassifizieren, da es im ganzen Datenset nie Anwesenheiten zwischen ca. 18:00 Uhr und 07:00 Uhr des nächsten Tages gibt. Bei diesem Teil der Daten wies das Modell eine hohe Genauigkeit von 94% auf. So werden also ca. 67% des Datensets mit enorm hoher Genauigkeit klassifiziert.
Selbst wenn das Modell beim letzten Drittel der Daten, also dem Teil des Datensets bei dem es zwischen An- und Abwesenheit unterscheiden muss, rät und somit nur mit einer Genauigkeit von 50% das richtige Label errechnet, suggeriert eine daraus resultierende Gesamtgenauigkeit von etwa 80% ein funktionierendes Modell. Allerdings zeigt sich hier, warum auch die Gegenüberstellung von True und False Positives in der Modellauswertung von entscheidender Wichtigkeit sein kann.

Tabelle 5.5.: Genauigkeitsauswertung des K-Means Modells

Präsenz	Precision	Recall	F1
Abwesenheit	0.94	0.79	0.86
Anwesenheit	0.50	0.82	0.62

Wie anhand der Precision-Score in 5.5 zu sehen ist, tritt hier genau der oben beschriebene Fall ein. Das Modell erkennt Abwesenheiten nahezu perfekt, während die übrigen Anwesenheiten nicht klar erkannt werden.

Zur genaueren Untersuchung und Veranschaulichung der Ergebnisse hilft eine Dimensionalitätsreduktion, bei der das Datenset auf zwei Dimensionen reduziert wurde und An- und Abwesenheiten in einem Graphen darstellt werden.

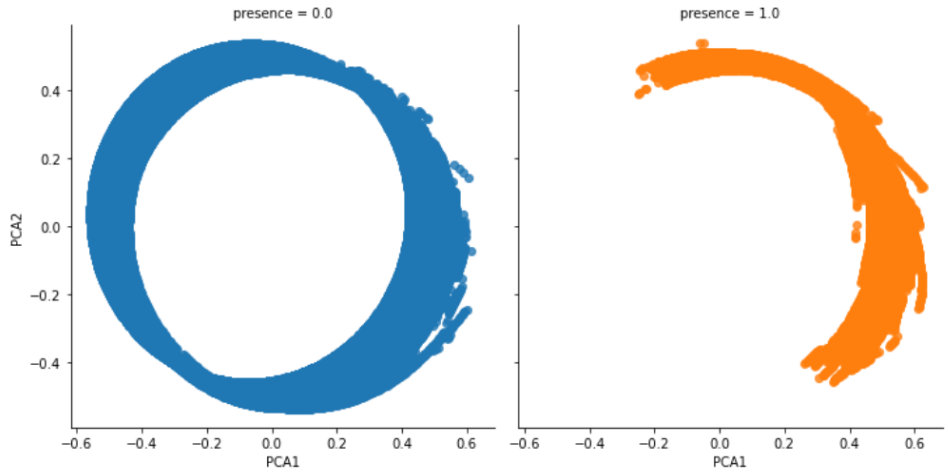

Abbildung 5.15.: Reduktion des Datensets auf zwei Dimensionen

Die Menge der Anwesenheiten (orange) ist in 5.15 nahezu identisch mit der rechten Teilmenge der Abwesenheiten (blau). Das heißt, dass alle Datenpunkte, die in der linken Hälfte des blauen Datensets liegen, sehr einfach klassifiziert werden können.
Legt man beide Datensets übereinander, ist es nun ohne die Einfärbung nahezu unmöglich zwischen An- und Abwesenheit zu unterscheiden.

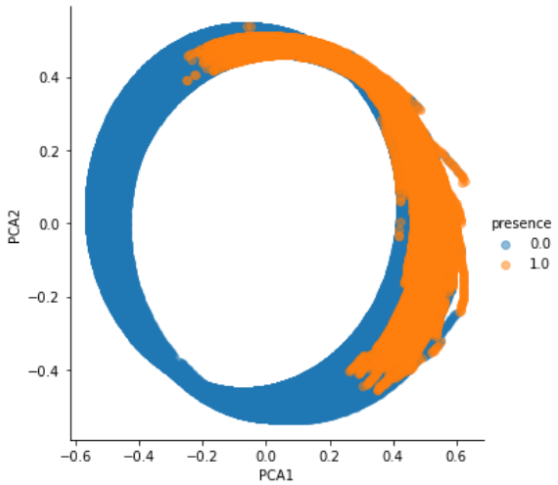

Abbildung 5.16.: Datenreduktion ohne Trennung

Da die Datenpunkte im rechten Teilbild kaum zu unterscheiden sind, schafft es das Modell nicht, An- und Abwesenheit während eines Arbeitstages zu unterscheiden. Währenddessen sind die Abwesenheiten im linken Teilbild sehr einfach zu klassifizieren. Diesen Sachverhalt spiegelt auch die Auswertung des *Classification Report* in Tabelle 5.5 wieder.

6. Zusammenfassung und Ausblick

6.1. CO2 als Anwesenheitsindikator

Die Genauigkeit aller Modelle bei der Präsenzerkennung während der gesamten Durchführungsphase des Projektes lag zwischen 75% und 94%. Hierbei wiesen Decision Tree Modelle für diesen Anwendungsfall die höchsten Genauigkeiten auf. Die Beziehung zwischen der aktuellen Tageszeit und dem CO2-Gehalt der Luft erwies sich für das Training von Machine Learning Algorithmen als sehr nützlich. Resultate vorangegangener Forschung unterstützen dieses Ergebnis ebenfalls[1].
Weiterhin konnte gezeigt werden, dass alle Modelle zusätzlich in der Lage sind, menschliche Präsenz außerhalb der trainierten Uhrzeiten korrekt zu erkennen, wenn die Modelle mit Deltas zu Vergangenheitswerten des CO2-Gehalts trainiert werden. Die Präsenzerkennung ist also vollständig unabhängig von den Umständen des Trainingssets und kann sowohl für andere Räume, als auch zu anderen Tageszeiten effektiv genutzt werden. Der CO2-Gehalt der Luft kann somit als geeigneter Indikator für menschliche Präsenz in Innenräumen angesehen werden.

Zudem wurde gezeigt, dass der Einsatz von CO2-Sensoren bei Gebäudeautomatisierungssystemen zweckmäßig ist. Ein weiterer Vorteil besteht darin, dass diese kostengünstig und leicht zu implementieren sind.

6.2. Mögliche Verbesserungen

Die Genauigkeit von Machine Learning Modellen kann allgemein durch die Hinzugabe von mehr Daten in ein Modell verbessert werden.
Da der Datenbestand, auf die sich diese Arbeit stützt, über die Bearbeitungszeit des Projektes kontinuierlich vergrößert wurde, wuchs damit auch die Anzahl an Anhaltspunkten für Zusammenhänge zwischen den Datenfeldern stetig an. Besonders nützlich wäre das Sammeln von Messdaten über ein oder mehrere Jahre hinweg. Somit hätten die Modelle eine Chance, Gewohnheiten in z.B. Urlaubstagen, Feiertagen oder auch wiederkehrenden Meetings korrekt einzuordnen, sodass Gebäudeautomatisierungssysteme ein Maximum an Komfort und Energieeffizienz herstellen können.

[1] Bsp. [Hanfstaengl et al., 2019]

Die Anwendung eines Modells auf andere Räume lies die Vermutung zu, dass diese Art von Präsenzerkennung das Potential aufweist, anhand eines Traningssets auch eine Vielzahl anderer Datensets klassifizieren zu können. Durch Aufzeichnung von Daten aus anderen Räumen könnte diese Vermutung genauer untersucht werden.

Da Infrarotsensoren inhärent nicht in der Lage sind kontinuierliche Präsenz zu erkennen, wenn sich die Personen im Raum nicht bewegen, kam es innerhalb des Datensets immer wieder zu kleinen Messfehlern, die durch Gruppierung und Durchschnittsberechnung der Präsenzwerte behoben werden mussten. Es ist klar davon auszugehen, dass eine genauere Datenlage auch die Qualität der trainierten Modelle steigern würde.

Zusätzlich existierten im Datenset durch seltene Probleme mit der Hardware längere Datenreihen mit falschen Labelwerten, welche während der Bearbeitung ausgeschlossen werden mussten. Da sich dadurch die für das Training geeignete Datenmenge verringerte, ist auch hier davon auszugehen, dass das Aussortieren dieser Werte mit einer Verringerung der Modellqualität einherging.

Quellenverzeichnis

[Hanfstaengl et al., 2019] Hanfstaengl, L., Parzinger, M., Wirnsberger, M., Spindler, U., and Wellisch, U. (2019). Identifying the presence of people in a room based on machine learning techniques using data of room control systems. Conference.

[Irvan B. Arief-Ang, 2017] Irvan B. Arief-Ang, Flora D. Salim, M. H. (2017). Cd-hoc: Indoor human occupancy counting using carbon dioxide sensor data. Online.

[Koehrsen, 2018] Koehrsen, W. (2018). Hyperparameter tuning the random forest in python. *Towards Data Science*.

[Riham Jaber et al., 2017] Riham Jaber, A., Dejan, M., and Marcella, U. (2017). The effect of indoor temperature and co2 levels on cognitive performance of adult females in a university building in saudi arabia. *Energy Procedia*, 122.

[Ronaghan, 2018] Ronaghan, S. (2018). The mathematics of decision trees, random forest and feature importance in scikit-learn and spark. *Towards Data Science*.

[Roy, 2020] Roy, B. (2020). All about feature scaling. *Towards Data Science*.

[Snow et al., 2019] Snow, S., Boyson, A. S., Paas, K. H., Gough, H., King, M.-F., Barlow, J., Noakes, C. J., and m.c. schraefel (2019). Exploring the physiological, neurophysiological and cognitive performance effects of elevated carbon dioxide concentrations indoors. *Building and Environment*, 156.

[Umweltbundesamt, 2020] Umweltbundesamt (2020). Endenergieverbrauch der privaten haushalte. Online. https://www.umweltbundesamt.de/daten/private-haushalte-konsum/wohnen/energieverbrauch-privater-haushalte/.

[Umweltbundesamt, 2022] Umweltbundesamt (2022). Energieverbrauch nach energieträgern und sektoren. Online. https://www.umweltbundesamt.de/daten/energie/energieverbrauch-nach-energietraegern-sektoren.

[VanderPlas, 2016] VanderPlas, J. (2016). *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media, Incorperated.

Abkürzungsverzeichnis

RFC	Random Forest Classifier
GBC	Gradient Boosting Classifier
SVC	Support Vector Classifier
LR	Logistic Regression
KNN	K-Nearest-Neighbours
KM	K-Means
$LSTM$	Long Short Term Memory
$HDFS$	Hadoop Distributed File System
IQR	Interquartile Range
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Positive
FV	Feature Vektor
ROC	Reciever Operating characteristic

Abbildungsverzeichnis

Tabellenverzeichnis

A. Quellcode

Hier steht der Quellcode