

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

Korreferent: Prof. Dr.-Ing. ...

7. März 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, 7. März 2022

Geheimhaltung - Sperrvermerk

Die vorliegende Arbeit unterliegt bis [Datum] der Geheimhaltung. Sie darf vorher weder vollständig noch auszugsweise ohne schriftliche Zustimmung des Autors, des betreuenden Referenten bzw. der Firma [Firmenname und -sitz] vervielfältigt, veröffentlicht oder Dritten zugänglich gemacht werden.

Inhalt

1. Einleitung	4
1.1. Motivation und Aufgabenstellung	4
1.2. Vorgehensweise	5
 2. Grundlagen	 6
2.1. Machine Learning	6
2.1.1. Random Forest Classifier	8
2.1.2. Support Vector Classifier	8
2.1.3. Gradient Boosting Classifier.....	9
2.1.4. Neuronale Netzwerke	10
2.1.5. Long Short Term Memory	10
2.2. CO2 als Anwesenheitsindikator	12
2.3. Outlier Detection.....	12
2.4. Sensordaten	13
2.5. Datenbeschaffung und -Vorbereitung	13
2.5.1. Gruppierung	14
2.5.2. Zyklische Codierung	14
2.5.3. Deltas und Shift-Werte.....	15
2.5.4. Validierung	15
 3. Anwendung und Ergebnisse	 17
 4. Zusammenfassung und Ausblick	 19
 Quellenverzeichnis	 20
Abkürzungsverzeichnis	21
Abbildungsverzeichnis	22
Tabellenverzeichnis	23
 Anhang	 23
A. Quellcode	24
B. Rohdatenvisualisierungen	25

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 in Zuge solcher Optimierungsschritte als „Smart Buildings“ bezeichnet werden, rückt sie in den 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.

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

1.1. Motivation und Aufgabenstellung

Für nahezu alle Bereiche der Gebäudeautomatisierung stellt die Anwesenheit von Personen eine zentrale Variable dar. Da die direkte Messung von Anwesenheit über z.B. Infrarotsensoren nicht verlässlich und rechtlich problematisch ist, soll in dieser Arbeit untersucht werden, inwiefern Machine-Learning Algorithmen genutzt werden können, um eine genaue Erwartung über die Anwesenheit von Personen anhand von CO₂-Werten in der Raumluft zu treffen.

Die Motivation der Optimierung der Gebäudeautomatisierung existiert, da ein steigender CO₂-Gehalt der Raumluft nachweislich mit einer Abnahme der menschlich kognitiven Leistung einhergeht. Mehrere Studien 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-Werte als aussagekräftige Indikatoren für menschliche Präsenz angesehen werden können. Es gab keine Einschränkungen hinsichtlich dessen, welche konkreten Machine-Learning Algorithmen benutzt werden sollten.

Als Programmiersprache für das Projekt wurde Python gewählt. Python ist wegen seiner umfassenden Machine-Learning Bibliotheken und einfachen Auswertungstechniken anhand von z.B. Graphen und Statistiken gut für diesen Anwendungsfall geeignet.

1.2. Vorgehensweise

Das Projekt beschäftigte sich im Schwerpunkt mit den folgenden Arbeitsschritten:

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

Da es zwischen allen verfügbaren Datensets der einzelnen Räume und Machine-Learning-Models eine Vielzahl an Kombinationsmöglichkeiten gibt, war es ein Anspruch der Projektarbeit, ein möglichst übersichtliches, gut gekapseltes Python Programm zu erstellen, mit dem man einfach und schnell verschiedene Datensets verarbeiten und mit einer dem Forschungszweck angemessenen Anzahl von Machine-Learning Models 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 Model- spezifische Leistungsindikatoren als besonders Erkenntnisreich erachtet werden, wird dies in dieser Arbeit angemerkt.

2. Grundlagen

2.1. Machine Learning

Grundsätzlich beschreibt Machine Learning das Entwickeln mathematischer Modelle zur statistischen Auswertung von Daten. Dabei wird dem Modell anhand von Daten zu einem bestimmten Sachverhalt beigebracht, in einem Datenset Schemata zu erkennen, womit sich eine Erwartung über die Umstände des Datensets treffen lässt. Beispielsweise könnte ein solches Model aus einem Datenset mit der aktuellen Jahreszeit, Uhrzeit und Position der Sonne am Himmel trainiert werden, sodass es auch schließlich 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. der 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.

Im Wesentlichen kann Machine Learning in zwei Unterkategorien unterteilt 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 Model 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 Model 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 Bereiche einzuteilen und so aus den Feldern des Datensets bestimmte Abhängigkeiten abzuleiten.

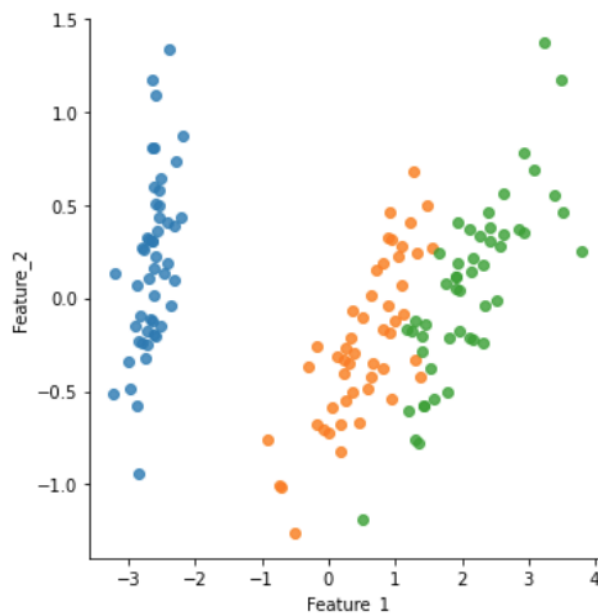


Abbildung 2.1.: Beispiel für Clustering

Bei der „*Dimensionality Reduction*“ versucht der Algorithmus das Datenset in einer Dimensionalität, also seiner 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 vorallem 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 herzustellen werden später trotzdem noch einzelne Ergebnisse von Clustering und Dimensionality Reduction betrachtet. Im Folgenden sollen die genutzten Modelle erklärt werden.

2.1.1. Random Forest Classifier

Random Forests stellen eine Unterkategorien der „*Decision Trees*“ dar. Decision Trees sind einfache Anordnungen von bestimmten Fragen, die über das Datenset gestellt werden, um eine Klassifikation zu erreichen.

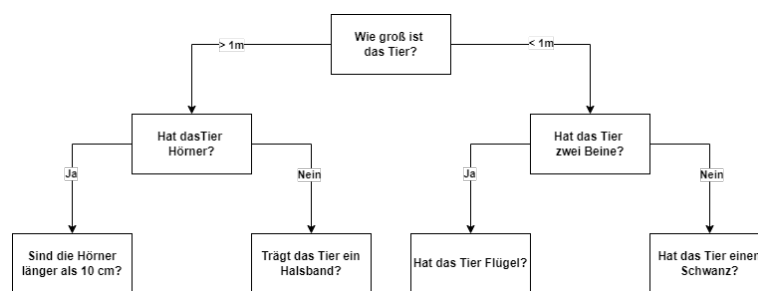


Abbildung 2.2.: Beispiel eines Decision Trees

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.

2.1.2. 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.

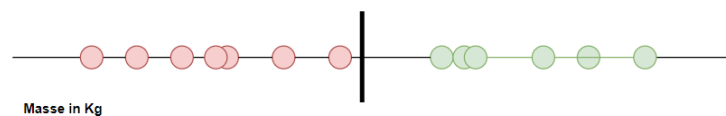


Abbildung 2.3.: Beispiel eines Support Vector Classifiers

In Abb. 2.3 ist der SVC ein Punkt auf einer eindimensionalen Linie, auf der das Gewicht in Kg von z.B. Mäusen 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 zwei-dimensionalen oder eine Ebene in einem drei-dimensionalen Koordinatensystem dar.

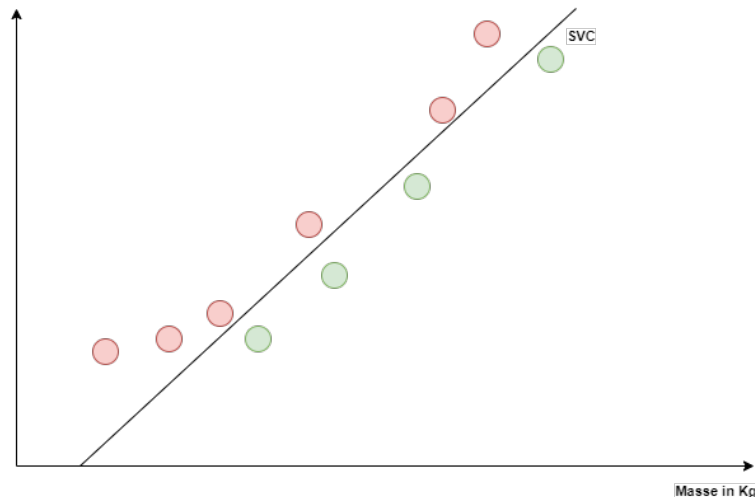


Abbildung 2.4.: Beispiel eines Support Vector Classifiers in der zweiten Dimension

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

2.1.3. Gradient Boosting Classifier

Der Gradient Boosting Classifier (GBC) versucht seine Erwartungen auf Grund von Abweichungen eines Labels vom Durchschnitt dieses Labels zu treffen. Erweitert man das Datenset im o.g. Beispiel um das Alter einer Maus, 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 Mäuse von einem bestimmten Alter über dem Durchschnittsgewicht liegen. Genauso liegen besonders junge Mäuse 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.

2.1.4. Neuronale Netzwerke

Die Funktionsweise eines Neuronalen Netzwerks ist direkt angelehnt an die Funktionsweise des menschlichen Gehirns. Einzelne Knotenpunkten (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 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 hier also zwei Neuronen für An- und Abwesenheit.

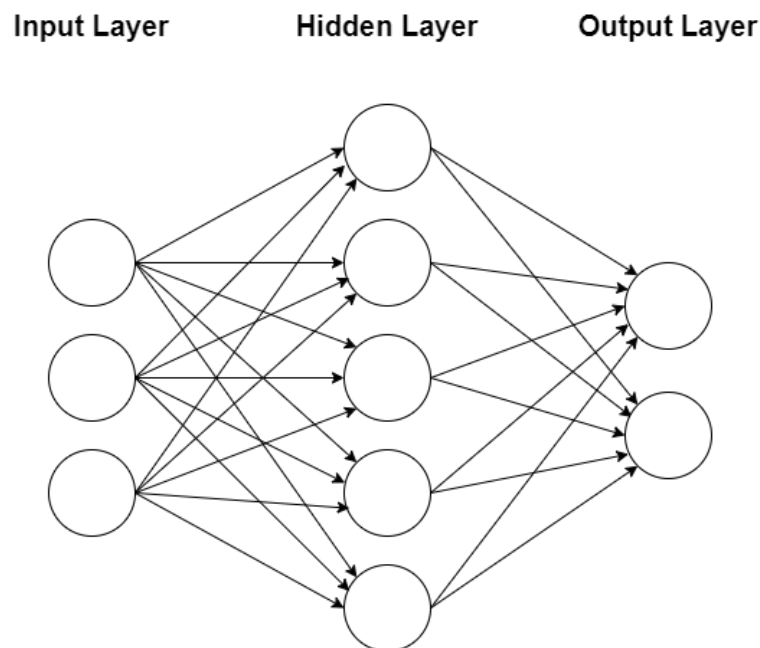


Abbildung 2.5.: Beispiel eines neuronalen Netzwerks

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.

2.1.5. Long Short Term Memory

Das Long Short Term Memory (LSTM) ist eine Abwandlung herkömmlicher Neuronalen 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 welches das Netzwerk bessere Rückschlüsse auf die Einordnung des aktuellen Input-Wertes ziehen kann.

Wird beispielsweise beim Satz „Das Auto ist Rot.“ das Wort „Das“ verarbeitet, kann das Netzwerk nun beim nächsten Schritt eine zusätzliche Erwartung treffen, dass das nächste

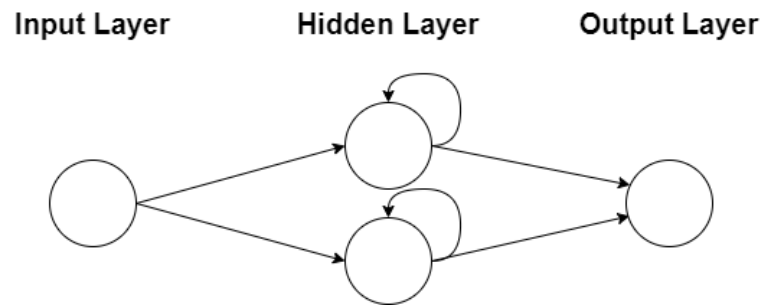


Abbildung 2.6.: Beispiel eines rekurrenten neuronalen Netzwerks

Wort wahrscheinlich nicht ebenfalls „Das“ sein wird. Ein LSTM kann auf diese Weise eine Vielzahl von Zeitschritten zurückblicken und 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.

2.2. CO2 als Anwesenheitsindikator

Der CO₂-Gehalt der Raumluft ist als sehr guter Indikator für menschliche Präsenz anzusehen. Anders als andere Umweltindikatoren wie Temperatur oder Luftfeuchtigkeit hat der CO₂-Gehalt die Eigenschaft, dass es in geschlossenen Räumen keine äußeren Einflussfaktoren für diesen Messwert gibt. In einem Büroraum kann der Mensch als alleinige Quelle für CO₂ angesehen werden.

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. Bei der Obergrenze von 1500 ppm entstehen beim Menschen erste Müdigkeitserscheinungen, weshalb dieser Wert in der Literatur als maximaler Richtwert für Innenräume gilt.

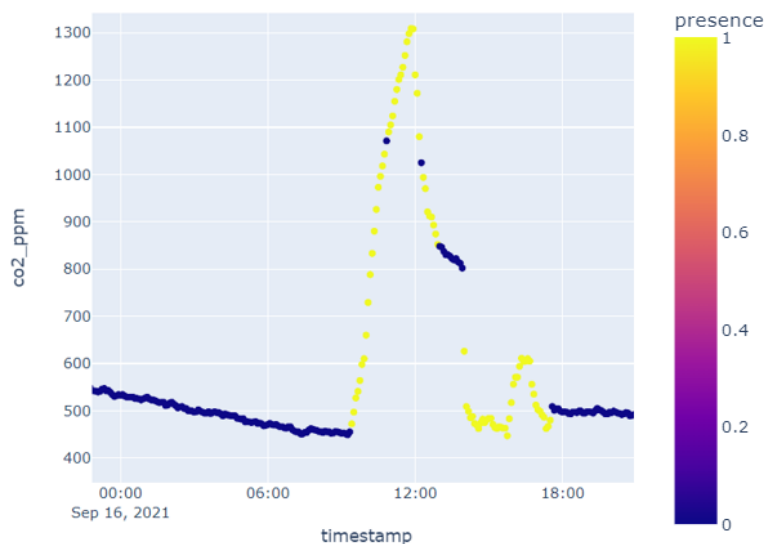


Abbildung 2.7.: CO₂ Gehalt der Raumluft über einen Tag

Es ist zu erkennen, dass die CO₂-Werte in einem normalen Büroraum innerhalb dieser empfohlenen Grenzen schwanken. Mit einigen kleinen Pausen sind fast durchgehend Personen anwesend, die täglich etwa um 13:00 den Raum lüften. Es ist auch klar zu erkennen, dass der CO₂-Gehalt während Abwesenheit durch die passive Lüftung des Raumes (Tür-/Fensterspalten) langsam wieder gegen den Grundwert bewegt.

2.3. 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 Leistung des Algorithmus deutlich senken. Um diese Ausreißer vor dem Training der Models 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 $x = (x_0, x_1, \dots, x_n)$ bestimmt man die Mediane der unteren und oberen Hälfte des Datensets Q_1 und Q_2 . Der IQR ergibt sich nun aus

$$IQR = Q_2 - Q_1 \quad (2.1)$$

Mit diesem Wert kann man nun die erlaubten Ober- und Untergrenzen des Datensets mit

$$Limit_{upper} = Q_2 + 1.5 * IQR \quad (2.2)$$

$$Limit_{lower} = Q_1 - 1.5 * IQR \quad (2.3)$$

bestimmen. Alle Werte die außerhalb dieser Grenzen liegen, können als Ausreißer betrachtet werden.

Ausreißer zu entfernen, ist hier wichtig, da die Sensoren Messfehler erzeugen können, oder gelegentlich zur Demonstration von starken Veränderungen in der direkten Nähe des Sensors gemessen wurde und es so in vielen Datensets kurzfristige CO₂-Werte gibt, die natürlich in geschlossenen Räumen nicht vorkommen.

2.4. Sensordaten

Wie bereits beschrieben, wurden die Sensordaten in mehreren Räumen der FH Aachen kontinuierlich gesammelt. Durch das Filtern nach der Beziehung des Raumes ergab sich folgende Datenstruktur als Ausgangslage:

Sensordaten		
Name	Format	Beschreibung
timestamp	timestamp	Zeitpunkt der Messung
co2_ppm	integer	CO ₂ -Wert
temperature_celsius	float	Temperatur in Grad Celsius
relative_humidity_percent	float	Luftfeuchtigkeit
presence	boolean	Aktivität des Bewegungssensors

2.5. 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 konnten die Daten jederzeit mit

einfachen SQL-Abfragen beschafft werden. Die Daten wurden so durch die gesamte Dauer des Projektes immer aktuell gehalten, damit alle Erkenntnisse immer auf der aktuellsten Datenlage basieren.

Die Datenvorbereitung oder Pre-Processing ist eine der wichtigsten Schritte bei der Anwendung von Machine Learning. Durch sie kann man beim Training des Models durch Bearbeitung bestehender Spalten oder Hinzufügen von zusätzlichen Spalten im Datenset Schwerpunkte setzen, die es den Algorithmen beim Training zum einen erleichtern, ihre Erwartungen zu präzisieren, zum anderen aber auch die Leistung beim Verarbeiten bestimmter Spalten zu steigern.

2.5.1. Gruppierung

Die Sensordaten wurden alle sechs Sekunden erfasst. Da sich weder CO₂-Gehalt noch Feuchtigkeits- oder Temperaturwerte der Raumluft so schnell nicht verändert, wurden die Daten direkt beim Drill per SQL zu zwei-Minuten-Intervallen zusammengefasst. Dabei werden über alle Spalten hinweg Durchschnittswerte gebildet, die dann nachher zu einem Datensatz zusammengefasst werden. Dies steigert die Leistung aller Algorithmen erheblich, da sich die Zahl der Datensätze um den Faktor 20 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 Datensatz dadurch nicht maßgeblich.

2.5.2. 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 86400 Sekunden bewegt. Aus diesem Wert wurden dann zwei neue Datenspalten „*hour_sin*“ und „*hour_cos*“ in das Dataset eingefügt welche sich durch

$$hour_sin = \sin(2 * \pi * x / x_{max}) \quad (2.4)$$

$$hour_cos = \cos(2 * \pi * x / x_{max}) \quad (2.5)$$

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

2.5.3. 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:

$$\Delta x_k = x_n - x_{n-k} \quad (2.6)$$

Die Erwartung ist hier, dass die Änderung der CO₂-, Temperatur- und Luftfeuchtheitswerte ein wichtigerer Indikator sein könnte, als die tatsächlichen Werte. In einem schlecht klimatisierten Raum könnten Grundwerte von z.B. CO₂ höher sein, als in anderen Räumen. Durch die hinzugefügten 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 Model-Genauigkeiten getestet.

2.5.4. 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 Model mit 80 Prozent der Daten trainiert wird, wobei die anderen 20 Prozent zurückgehalten werden, um daraufhin das fertig trainierte Model daran zu testen. Aus dem Ergebnis dieses Tests ergibt sich ein Prozentwert, der die allgemeine Genauigkeit des Models repräsentiert, d.h. wie verlässlich das Model die An- und Abwesenheit des Testsets selbst berechnen kann, wenn man ihm den tatsächlichen Wert vorenthält.

Dieser Gesamtwert ist wiederum der Durchschnitt der jeweiligen Genauigkeit beim Errechnen von An- und Abwesenheit. In diesem Anwendungsfall ist es durchaus wichtig, beide Genauigkeiten einzeln zu betrachten, da ein Arbeitstag nur etwa ein Drittel eines tatsächlichen Tages darstellt. Das Datenset ist also mit einem Verhältnis von etwa 8/16 in Richtung der Abwesenheit unausgeglichen. Dies lässt die Erwartung zu, dass das trainierte Model wesentlich besser darin sein wird, Abwesenheit zu erkennen, als Anwesenheit.

Hierzu wurde bei der Auswertung der Ergebnisse immer auch der *Classification Report* hinzugezogen, aus dem ersichtlich ist, wie genau das Model beide möglichen Labelwerte berechnen konnte.

Desweiteren war es entscheidend zu erkennen, dass das Datenset über mehrere Monate hinweg nicht perfekt uniform ist, da es in bestimmten Monaten zu verhältnismäßig vielen Urlaubstagen (z.B. Weihnachten/Silvester) und damit einer Häufung an Abwesenheits-Werten 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) implementiert. Bei einer CV wird das Datenset immernoch im gleichen Verhältnis aufgeteilt, allerdings mehrmals, sodass die Trainings- und Testsets jedes mal aus jeweils anderen Daten bestehen.



Abbildung 2.8.: Beispiel einer Kreuzvalidierung

Errechnet man nun die Durchschnittsgenauigkeit aller Iterationen der CV, ergibt sich eine Gesamtgenauigkeit, die das Model besser repräsentiert, weil es an einer größeren Anzahl an 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 Model selbst gefüllt werden sollte. Das Ergebnis musste dann in einem Graphen 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 Models.

3. Anwendung und Ergebnisse

Tabelle 3.1.: Anwendung und Ergebnisse

Stellung	$\frac{T_U}{^{\circ}C}$	$\frac{T_c}{^{\circ}C}$	$\frac{\Delta T}{^{\circ}C}$
senkrecht (0°)	27,3	69,8	42,5
waagrecht (90°)	26,6	70,6	44,0

Wie in Tabelle 3.2 zu sehen ist, ist es besser, Trennlinien nur dort einzusetzen, wo logische Grenzen liegen.

Tabelle 3.2.: Smartphone Sensordaten

Sensorinformation	Format	frequency [s^{-1}]
App identifier for vendor	int64	once per transfer
WIFI and network carrier IP addresses	int128	once per transfer
battery level	int8	0.1
Position information: latitude, longitude, altitude, speed, course, vertical position accuracy, horizontal position accuracy, floor level information	float32[8]	1
Heading information: heading.x, heading.y, heading.z, true heading, magnetic heading, heading accuracy	float16[6]	1
Accelerometer information: acceleration.x, acceleration.y, acceleration.z	float16[3]	2
Gyroscope information: rotationRate.x, rotationRate.y, rotationRate.z	float16[3]	2
altimeter information: relative altitude, pressure	float16[2]	1
timestamp	uint32	once per transfer
Temperature [$^{\circ}C$]	float16	1

4. Zusammenfassung und Ausblick

Quellenverzeichnis

- [Hartnett, 2018] Hartnett, K. (2018). Machine learning confronts the elephant in the room. Quanta Magazine, Online. <https://www.quantamagazine.org/machine-learning-confronts-the-elephant-in-the-room-20180920/>.
- [Le, 2018] Le, J. (2018). How to do semantic segmentation using deep learning. Online. <https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef>.

Abkürzungsverzeichnis

g	Gravitation in Nähe der Erdoberfläche
Nu	Nußelt-Zahl
ν_{Luft}	Kinematische Viskosität von Luft
Pr	Prandtl-Zahl
\dot{Q}	Wärmestrom
Ra	Rayleigh-Zahl
ρ_{Luft}	Dichte von Luft
T	Temperatur
T_{∞}	Umgebungstemperatur

Abbildungsverzeichnis

2.1. Beispiel für Clustering	7
2.2. Beispiel eines Decision Trees	8
2.3. Beispiel eines Support Vector Classifiers	8
2.4. Beispiel eines Support Vector Classifiers in der zweiten Dimension.....	9
2.5. Beispiel eines neuronalen Netzwerks	10
2.6. Beispiel eines rekurrenten neuronalen Netzwerks.....	11
2.7. CO2 Gehalt der Raumluft über einen Tag	12
2.8. Beispiel einer Kreuzvalidierung	16

Tabellenverzeichnis

3.1. Anwendung und Ergebnisse	17
3.2. Smartphone Sensordaten	18

A. Quellcode

1. Source 1
2. Source 2

B. Rohdatenvisualisierungen

1. Graustufen
2. Verteilungen