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**

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

16. 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, 16. 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

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 exisitert, 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 Projekte 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 einzlnen 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. 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 mit Kamera-, Infrarot-, Mikrowellen-, oder Ultraschallsensoren. Durch diese kann menschliche Präsenz sowohl direkt (Kamera-, Infrarotsensoren) oder indirekt über Veränderungen in den physischen Eigenschaften des Raumes festegestellt werden. 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 verhätlnismäßig geringen Schallgeschwindigkeit keine besonders hohe Genauigkeit besitzen müssen, sind die an einen Mikrowellensensor gestellten Ansprüche wesentlich höher. Mit diesen können Bewegungen durch Ausnutzung des Doppler-Effektes einer elektromagnetischen Welle erkannt werden. elektromagnetische Wellen sind im gegensatz zu Schallwellen etwa um einen Faktor von einer Million schneller, weshalb Messungen sehr viel genauer sein müssen, um die gleiche Aussagekraft zu besitzen.
Hier verdeutlicht sich nochmals die Motivation dieser Arbeit, da CO2-Sensoren weder teuer sind, noch besondere Ansprüche an die Positionierung im Raum oder Kallibrierung besitzen.

3. Untersuchte Verfahren

3.1. 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 traniertes 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 traniert 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.

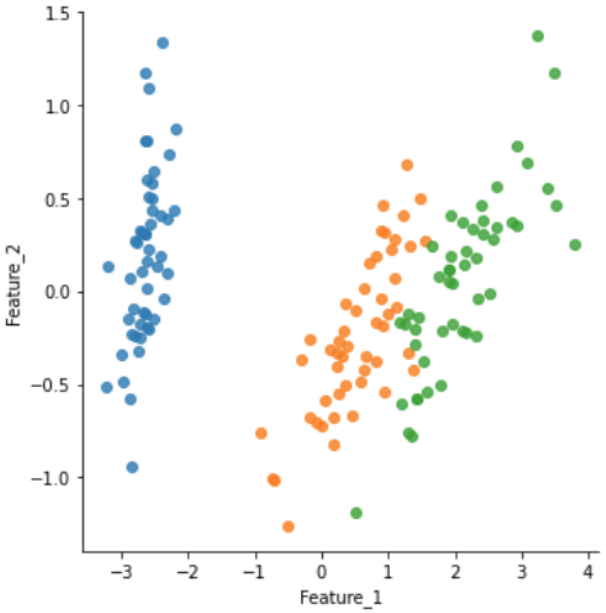

Abbildung 3.1.: Beispiel für Clustering

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.

3.1.1. 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.

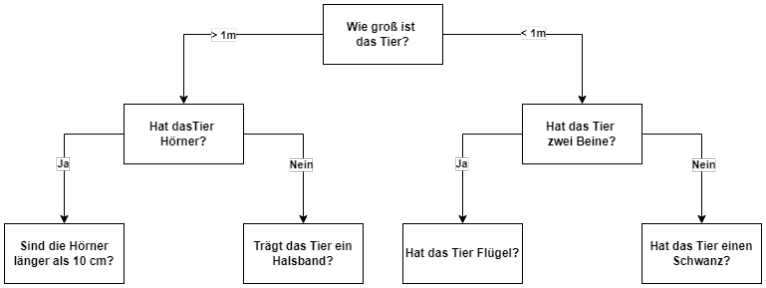

Abbildung 3.2.: Beispiel eines Decision Trees

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 Gradien 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.

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.

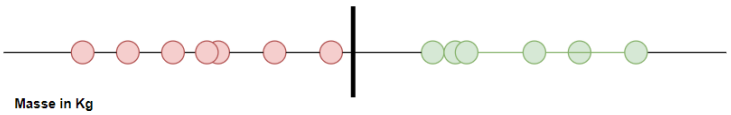

Abbildung 3.3.: Beispiel eines Support Vector Classifiers

In Abb. 3.3 ist der SVC ein Punkt auf einer eindimensionalen Linie, auf der das Gewicht in Kg von z.B. 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.

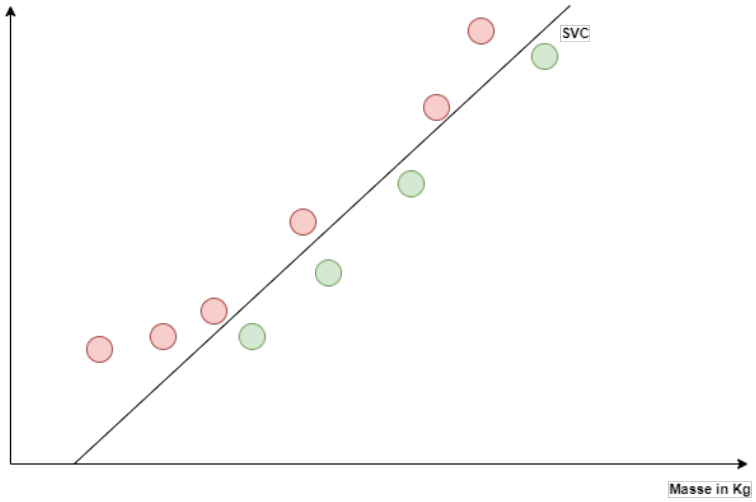

Abbildung 3.4.: Beispiel eines Support Vector Classifiers in der zweiten Dimension

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

3.1.4. 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 geschiet 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.

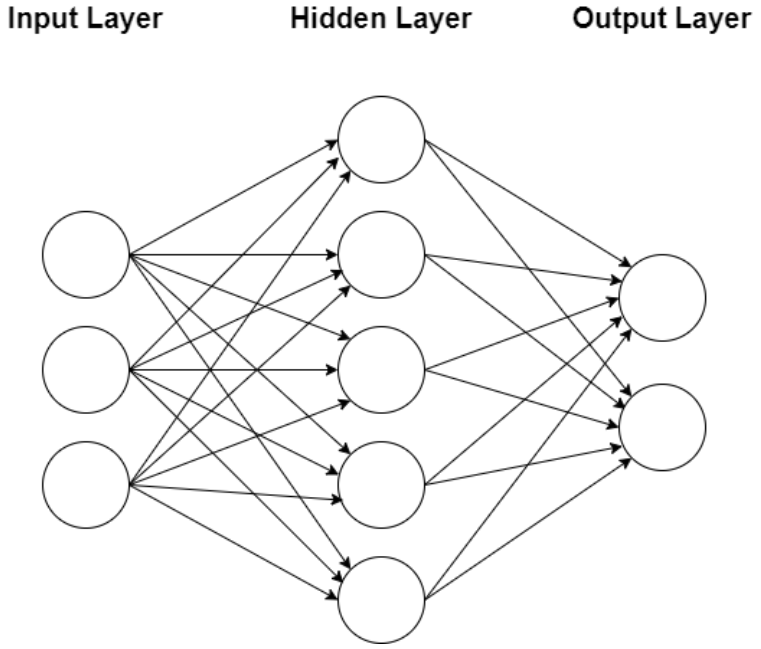

Abbildung 3.5.: Beispiel eines neuronalen Netzwerks

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.

3.1.5. 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

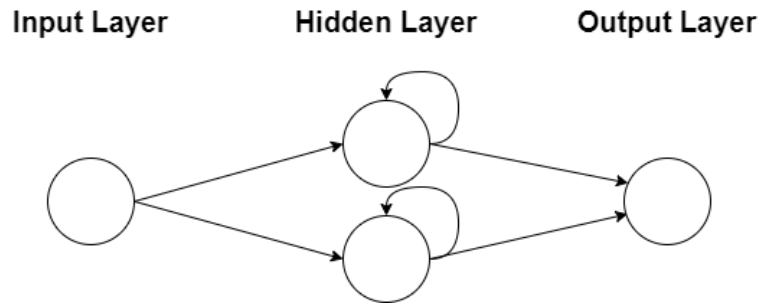

Abbildung 3.6.: Beispiel eines rekurrenten neuronalen Netzwerks

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.

3.2. 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.

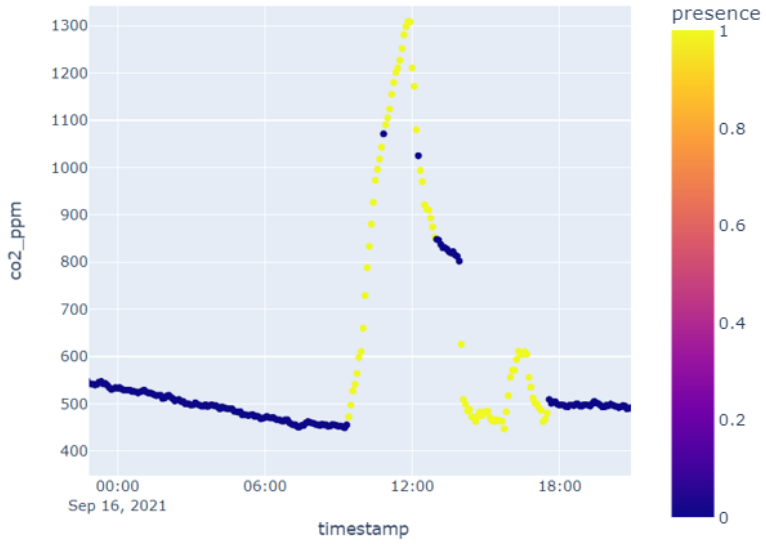

Abbildung 3.7.: CO2 Gehalt der Raumluft über einen Tag

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 12:00 den Raum lüften. Es ist auch klar zu erkennen, dass sich der CO2-Gehalt während Abwesenheit 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, unterliegen sie in ihrer Nützlichkeit dem CO2-Gehalt der Raumluft in einem wichtigen Faktor. Sie sich beide stark beeinlusst, von äußeren Umweltfaktoren wie dem aktuellen Wetter oder der Jahreszeit. 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-Werte nicht von besonderem Nutzen

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 der Beziehung des Raumes ergab sich folgende Datenstruktur als Ausgangslage:

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 des Bewegungssensors

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 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 Leitung beim Verarbeiten 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 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 Datensetz 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 86400 Sekunden bewegt. Aus diesem Wert wurden dann zwei neue Datenspalten „*hour_sin*" und „*hour_cos*" in das Datenset eingefügt welche sich durch

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

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

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 anderen Räumen. 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 Model-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 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 $x = (x_0, x_1, ..., x_n)$ 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 kann man nun die erlaubten Ober- und Untergrenzen des Datensets mit

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

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 geatmet wurde und es so in vielen Datensets kurzfristige CO2-Werte gibt, die natürlich in geschlossenen Räumen nicht vorkommen.

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 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 ergeben sich verschiedene Werte, die 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.
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 sich die Genauigkeit des Modells neben tatsächlich richtig erkannten Werte aus false negatives und false positives zusammensetzt.
Die *Accuracy Score* ist die meistgenutzte Metrik für Klassifikation 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, die hätten erkannt werden müssen.

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

Die *F1-Score* ist der Durchschnittswert aus Recall- und Precision-Score und gibt 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, Anwesenheitswerte genauer zu betrachten als Abwesenheitswerte, 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 berrechnen konnte.

Desweiteren war es entscheidend zu erkennen, dass das Datenset über mehrere Monate hinweg nicht perfekt uniform ist, da es in bestimmten Monaten zu verhätlnismäßig 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.

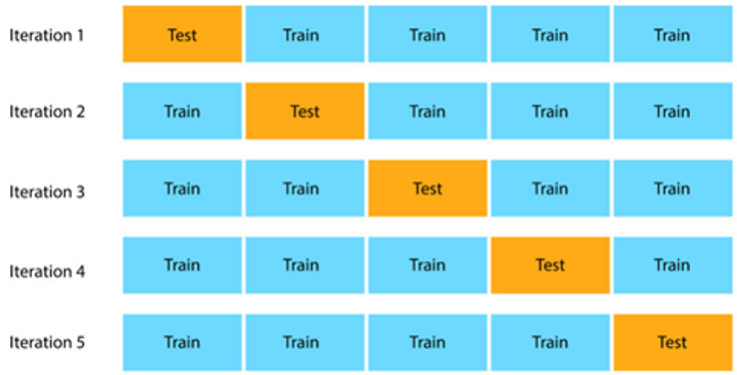

Abbildung 4.2.: Beispiel einer Kreuzvalidierung

Errechnet man nun die Durchschnittsgenauigkeit aller Iterationen der CV, ergibt sich eine Gesamtgenauigkeit, die das Model besser repräsentiert, weil es 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.

4.2.1. Parameter Tuning

Parameter Tuning beschreibt einen Schritt der Modell-Optimierung, der normalerweise stattfindet, nachdem ein funktionierendes Modell, das mit bereits verarbeiteten Daten gute Ergebnisse liefert, erstellt wurde. 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 Paramter-Kombinationen zu testen.
Das im Scikit-Learn enthaltene *GridSearchCV*-Modul bietet die Möglichkeit einem Modell eine Vielzahl an verschiedenen Parameter-Optionen zu übergeben. Mit diesen werden kann 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 werden. 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 und kann auch zu einer leichten Verschlechterung führen, wenn die Standard-Parameter der einzelnen Modelle bereits gut gewählt waren.

5. Anwendung und Ergebnisse

5.1. Feature Vektor

Der Feature Vektor FV beschreibt das Ergebnis des Datensets nach der Vorverarbeitung. Durch die o.g. Schritte ergibt sich eine Vielzahl an möglichen zusätzlichen Features, mit denen man das Datenset versehen könnte. Es sollte zunächst ermittelt werden, welche diese 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 einfließt. Zur Wahl eines FVs der nach dem Training mit einem Modell eine möglichst hohe Genauigkeit liefert, wurden verschiedene Kombinationen durch Verknüpfung von Temperatur, Luftfeuchtigkeit, CO2 mit jeweiligen Delta- und Shiftwerten vorgenommen. Ziel war es, mit einem möglichst kleinen Vektor eine möglichst hohe Genauigkeit zu erreichen, da die meisten Algorithmen eine direkte Abhängigkeit zwischen Dauer der Berechnung und Dimensionalität des FV haben.

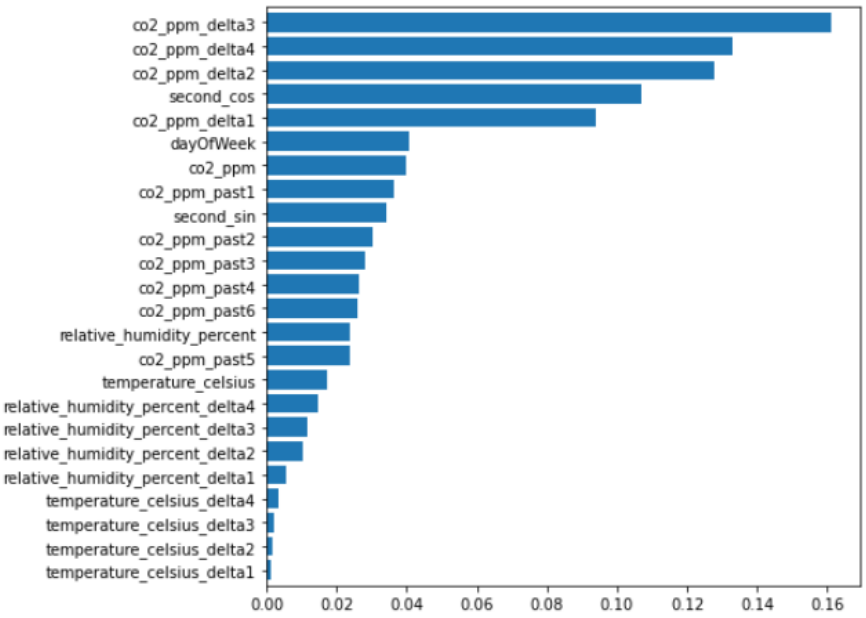

Abbildung 5.1.: Feature Importances eines Random Forest

In der o.g. Grafik sind neben den ursprünglichen Datenwerten nun auch Delta- und Shiftwerte zu sehen. Wie bereits im vorherigen Kapitel angedeutet, sieht man hier klar, dass die Temperatur- und Luftfeuchtigkeitswerte für die Errechnung der Labels kaum zu Rate gezogen werden. Weder die Grund- noch Deltawerte weisen eine hohe Wichtigkeit für das Model auf.

Nach Exkludierung von Temperatur und Luftfeuchtigkeit wurden neue Deltas und Shifts für die CO2-Werte eingefügt und die Feature Importances erneut betrachtet. Es ist zu erkennen, dass auch die CO2-Werte von n Minuten zuvor ebenfalls nicht maßgeblich in die Berechnung einfließen.

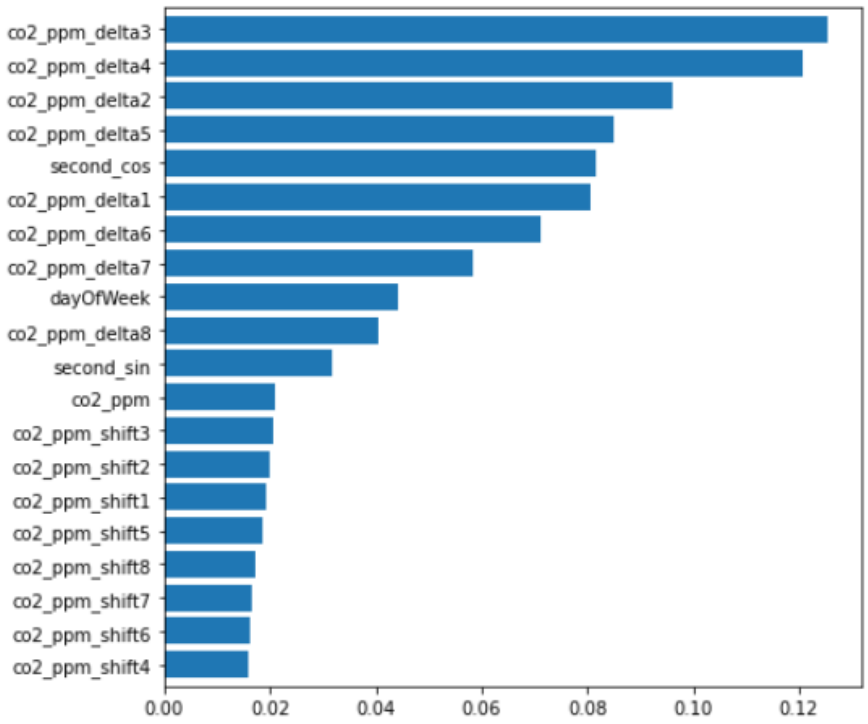

Abbildung 5.2.: Feature Importances eines Random Forest

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.

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

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

Feature-Vektor	RF (Random Forest)	SV (Support Vector)	GB (Gradient Boost)	LR (Logistic Regression)
FV1	0.944805	0.912594	0.943618	0.935428
FV2	0.946477	0.918528	0.931867	0.916739
FV3	0.945256	0.917241	0.944554	0.938136
FV4	0.945725	0.912095	0.947024	0.931573

Wie deutlich sichtbar ist, sind die Genauigkeiten fast identisch. Dies war Anlass sowohl die Temperatur- und Luftfeuchtigkeitswerte, als auch die Shift-Werte der CO2-Messungen nicht weiter zu betrachten.

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

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

Da nicht alle Algorithmen auf die gleiche Weise evaluiert und grafisch dargestellt werden können, sollen die Ergebnisse zu

- Clustering Modellen
- Decision Tree Modellen
- Neuralen Netzwerken

einzeln betrachtet.

5.2.1. Clustering Modelle

5.2.2. Decision Tree Modelle

Die ausgewählten Algorithmen konnten mit dem ausgewählten FV bei der Erwartungsberechnung über das gesamte Datenset eine Genauigkeit von etwa 94% erreichen. Eine *Confusion Matrix*, welche die errechneten Werte den tatsächlichen gegenüberstellt, zeigt, dass im hier benutzten Beispiel, ein Random Forest Classifier selbst bei der starken Unausgeglichenheit des Datensets ähnlich viele Fehler in sowohl An- als auch Abwesenheit macht.

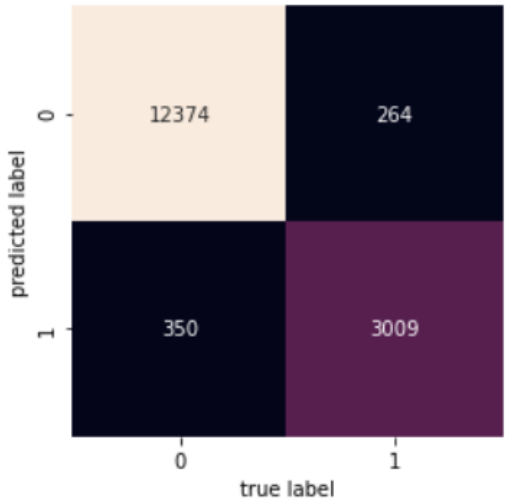

Abbildung 5.3.: Confusion Matrix eines RFC

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 falschen Erwartung für beide Werte zeigen.

Es ist ebenfalls aufschlussreich die erwarteten Labelwerte mit den tatsächlichen grafisch gegenüberzustellen, indem man das ursprüngliche Datenset mit timestamp und CO2-Wert zeichnet. Dabei wird die gemessene Anwesenheit farblich als gelb für an- und blau für abwesend eingefärbt.

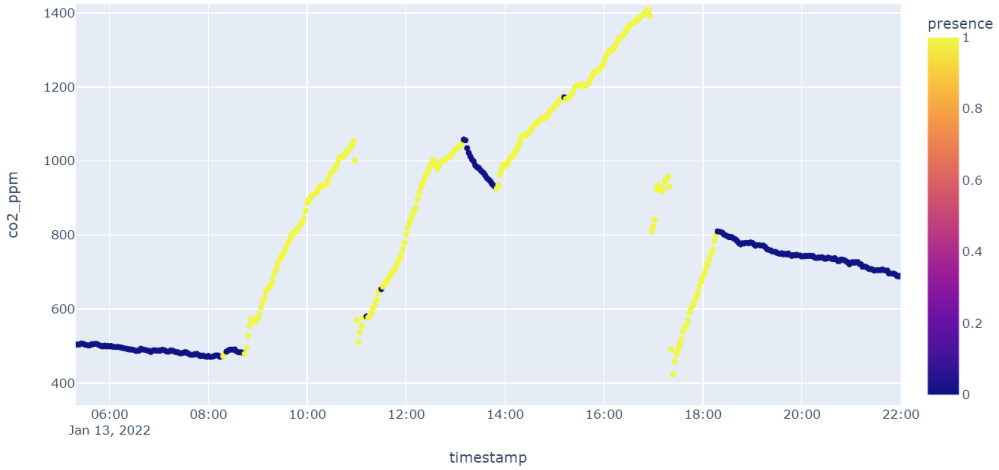

Abbildung 5.4.: Messwerte des 13. Januar

Löscht man aus dieser Datenreihe die tatsächlichen Labelwerte und fügt stattdessen die errechneten Anwesenheitswerte des Modells ein, zeigt sich die hohe Genauigkeit deutlich. Bis auf einige kleine Fehler trifft das Modell eine sehr präzise Aussage über die aktuelle Anwesenheit.

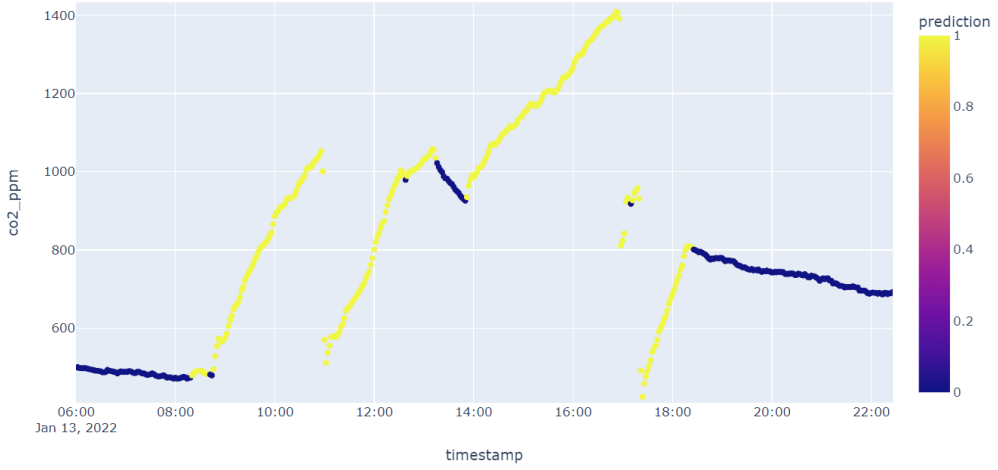

Abbildung 5.5.: Messwerte des 13. Januar

Dieses Ergebnis konnte über eine Vielzahl von Kombinationen verschiedener Räume und Modelle weiterhin bestätigt und repliziert werden. Die Tatsache, dass die Genauigkeiten aller Modelle so nah beieinander liegen, rührt daher, dass dieses Datenset ein typisches Klassifizierungsproblem darstellt, wodurch die sich wiederholenden Schemata in den CO2-Werten von einer Vielzahl von Algorithmen schnell erkannt und verarbeitet werden können.

Beim 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 möglichen suchen, die nicht den Standard-Einstellungen entsprechen.

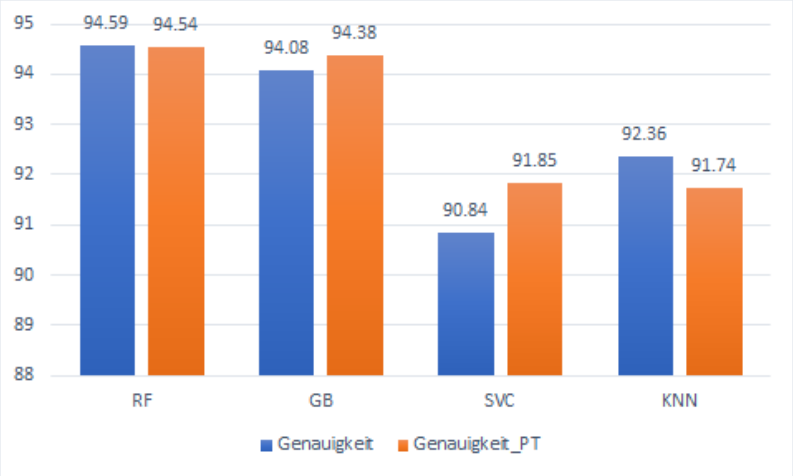

Abbildung 5.6.: Ergebnisse des Parameter Tuning

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. 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 hervorbringen kann.

Zusätzlich wurden die *ROC-Curves* der True- und False-Positives gegeneinander gezeichent um zu erknennen inwiefern die Ergebnisse zufällig oder errechnet sind.

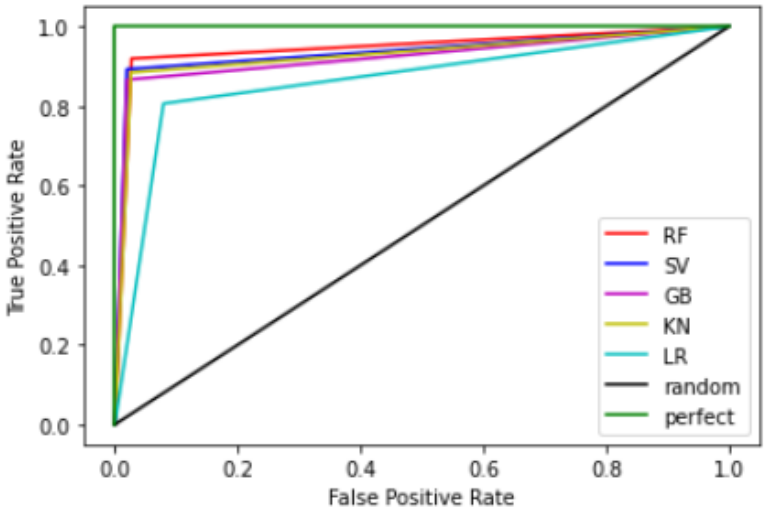

Abbildung 5.7.: ROC-Curves aller Modelle

Man sieht deutlich, wie alle Modelle sehr ähnliche Ergebnisse aufweisen. Keines der

Modelle weist anzeichen auf, dass Ergebnisse wegen mangelnder Informationen erraten werden.

5.2.3. Neurale Netzwerke

6. 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

Tabellenverzeichnis

A. Quellcode

1. Source 1
2. Source 2

B. Rohdatenvisualisierungen

1. Graustufen
2. Verteilungen