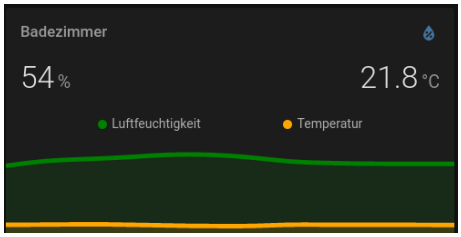
OPTIMIERUNG DER LÜFTUNG ZUR VERHINDERUNG VON SCHIMMELBILDUNG IN BADEZIMMERN BEI MINIMIERUNG DES WÄRMEVERLUSTS

JONATHAN GÄRTNER



Erarbeitung einer anwenderfreundlichen Lüftungsempfehlung mithilfe von Informationstechnologie

Dezember 2022 – version 2.5

Jonathan Gärtner: *OPTIMIERUNG DER LÜFTUNG ZUR VERHINDERUNG VON SCHIMMELBILDUNG IN BADEZIMMERN bei Minimierung des Wärmeverlusts,* Erarbeitung einer anwenderfreundlichen Lüftungsempfehlung mithilfe von Informationstechnologie, © Dezember 2022

ABSTRACT

Schimmel erhöht unter anderem das Risiko an Asthma zu erkranken um 50% [1], wobei "Lüften [...] die wichtigste Maßnahme zum Entfernen von Schadstoffen, Feuchtigkeit und Nässe aus Gebäuden [ist]."[2]. Hinzu kommt die Energiekrise, bei der "jeder Kubikmeter Gas, der nicht verfeuert wird, hilft"[3] Aber wie lange sollte man lüften, um dieses Ziel zu erreichen? Und in welcher Konfiguration (also gekippt oder ganz geöffnet)? Durch eine optimale Lüftung kann der Wärmeverlust minimiert werden. Außerdem wird die Handhabung durch klare Anweisungen, wie lange gelüftet werden muss, vereinfacht. Da dies von der lokalen Situation abhängig ist, gibt es darauf keine generelle Antwort. Deshalb wurde eine Applikation entwickelt, welche durch ein internes Modell des Vorgangs, ökologische und sehr komfortable Lüftungsvorschläge erteilt. Das Modell kann außerdem noch speziell an die eigene Situation angepasst werden, indem mithilfe von Optimizern die Parameter des Modells an die Messwerte angepasst werden. Dabei werden durch die Gestaltung des Modells nur relativ wenige Daten benötigt. Dies ermöglicht eine effiziente Verhinderung von Schimmel, bei gleichzeitiger Minimierung des Lüftens, wodurch Energie gespart wird.

1 Heseltine und Rosen, *WHO guidelines for indoor air quality : dampness and mould* S. 70.

2 Nach WHO aus World Health Organization (WHO) Europe, "Fachliche und politische Empfehlungen zur Verringerung von Gesundheitsrisiken durch Feuchtigkeit und Schimmel" S 7.

3 Bundeswirtschaftsminister Robert Habeck Habeck, "Habeck zur Gas-Frühwarnstufe" zuletzt besucht am 3.1.22

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— Donald E. Knuth

DANKSAGUNG

Vielen Dank allen die mich unterstützt haben: Herrn Dr. Andres für die vielen über den Schulstoff hinausgehende Einblicke in die Mathematik und meinen Eltern, die die Arbeit auf Verständlichkeit gegengelesen haben und mir den nötigen Freiraum gegeben haben.

Danke an die Entwickler von Texstudio, Languagetool, Python, Pandas und Numpy.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

1 EINFÜHRUNG

1.1 IDEENFINDUNG

Im Bad meiner Familie hat sich vor einer Weile Schimmel gebildet, was nicht nur sehr eklig war, sondern auch einigen Aufwand nach sich gezogen hat (da es in diesem Fall tatsächlich nicht hauptsächlich an der Luftfeuchtigkeit, sondern an der undichten Dusche lag), damals habe ich mich gefragt, ob sich Schimmel durch optimiertes Lüften besser vermeiden ließe. Im Laufe des Projektes kam dann verstärkt durch die sich anbahnende Energiekrise noch ein Aspekt zur Optimierung des Temperaturverlusts hinzu.

1.2 REALISIERUNG

Die Realisierung hat rein im privaten Raum stattgefunden. Nach ausführlicher Datensammlung in zwei Badezimmern wurden Faktoren, die das Lüftverhalten beeinflussen verifiziert und ein grobes Modell entwickelt, welches schließlich mithilfe von Optimierungsverfahren an die Messdaten angenähert wurde.

1.3 AKTUELLER STAND

In den meisten Bädern stehen Temperatur- und Luftfeuchtigkeitsmessgeräte, allerdings werden diese meist nur sehr sporadisch genutzt und das Lüften läuft eher nach Gefühl. Zwar gibt es prinzipiell Faustregeln wie lange man lüften sollte, aber so gut wie keine genaueren Regeln und Verfahren. Eine Lösung wäre das ständige ablesen des Messgeräts und das schließen der Fenster, sobald der angezeigte Wert auf ein unkritisches Niveau gefallen ist, dass ist aber nicht besonders praktikabel. Bei der hier vorgeschlagenen Lösung würde man bspw. nach dem Duschen die aktuelle Luftfeuchtigkeit und Temperatur angeben und das Modell würde sagen in wievielen Minuten man die Fenster schließen soll.

2 ENTWICKLUNG DES MODELLS

2.1 METHODIK

Um aus Daten irgendwelche Muster und Modelle abzuleiten zu können, werden heutzutage oft neuronale Netze verwendet. Das Problem dabei ist, das diese Modelle so viele Parameter haben, dass es oft schwer fällt zu verstehen was das Modell genau macht und ob es tatsächlich die Problemstellung löst die man hat[1], was es auch schwer macht zu garantieren das es in Spezialfällen nicht komplett absurde Werte (bspw. über 100% Luftfeuchtigkeit) liefert. Des Weiteren benötigt ein neuronales Netz mehr Trainingsdaten, da mehr Parameter optimiert werden müssen. Nichts desto trotz wäre ein neuronales Netz ebenfalls ein valider Ansatz für dieses Problem. Aber ich habe mich, aus den genannten Gründen, gegen neuronale Netze entschieden, sondern verwende ein Modell mit einigen Parametern, welche Daten mit der "least-square" Methode, ähnlich der Backpropagation in einem neuronalen Netzen, optimiert. Dabei wird der quadrierte Fehler immer weiter minimiert zwischen Ausgabe des Modells und den Trainingsdaten. Dazu im Folgenden mehr:

2.1.1 *Least Squares*

Bezüglich der Optimierung geht man folgendermaßen vor: Als Beispiel eine einfache Funktion $f(n) = a_0 \cdot \frac{1}{n}$, wobei a_0 der Parameter ist, welcher optimiert werden soll. Jetzt haben wir bspw. einige Messdaten, welche das Modell nachbilden soll, bspw. soll $f(1) = 0,5$ sein. Daraus ergibt sich in dem Fall, dass $a_0 = 0,5$ gelten muss. Das funktioniert bei mehreren Parametern allerdings nicht mehr. Ein weiterer Faktor ist, dass man insbesondere bei Messungen aus der realen Welt davon ausgehen muss, dass in den Trainingsdaten Messfehler sind und einige Einflüsse auf das Geschehen, seien sie noch so klein, außen vorgelassen wurden.

Es geht also nicht um das Finden des einen richtigen Ergebnisses sondern um eine Optimierung, eine Näherung. Dazu wird aus dem Satz Parametern, mit am Anfang zufälligen Startwerten[2], der Fehler berechnet, welcher sich über alle Trainingsdaten ergibt. Mit Fehler ist die Differenz zwischen gemessenen (bzw. gewünschten) Wert und dem Wert welchen das Modell voraussagt gemeint. Im Beispiel von oben wäre der Fehler für den Startwert, $a_0 = 1$, 0,5. Jetzt verändert man die Werte etwas und misst den Fehler wieder, ist er größer geworden ist man in die falsche Richtung gegangen. Damit nähert man sich immer mehr dem nächsten lokalem Minimum an. Dabei könnte man natürlich in irgendeinem lokalen Minimum statt im globalen landen, aber das lässt sich nicht wirklich vermeiden, außer durch anpassen der Startwerte, in eine passendere Richtung. Das kann allerdings genauso neuronalen Netzen passieren. Bei der ganzen Methode wird übrigens nicht der Fehler selbst, sondern sein Quadrat verwendet, damit der Fehler nie negativ ist, denn sonst würde dort das Optimum einer Minimierung liegen. Deswegen heißt die Methode auch "least squares" Methode.

1 Bspw. gibt es "overfitting" wobei das Netz die Trainingsdaten "auswendig lernt" und deswegen nicht auf neuen Daten funktioniert oder Fälle wo die Trainingsdaten nach anderen Dingen unterschieden wurden, als man es sich eigentlich gewünscht hat.

2 Bzw. i.d.R. setzt man sie in eine Größenordnung, von der man schätzt, dass sie stimmen könnte, um im richtigen Minima zu landen und evtl. auch die Optimierung zu beschleunigen.

Um das mathematisch zu formulieren: Wir betrachten beispielhaft ein Modell, welches durch folgende Funktion beschrieben wird $f(a,t,x) = a_0 \frac{1}{a_1 \cdot t} + x$, diese soll minimiert werden. Dabei enthält a die zu optimierenden Parameter. Jetzt wird eine Funktion $g(\overrightarrow{a})$ formuliert, welche die Summe des quadrierten Fehlers abbildet. In der Praxis übergibt man dieser nur noch die Parameter, da die Trainingsdaten für diese Funktion immer gleich bleiben (während der Optimierung). Das sieht so aus:

$$g(\overrightarrow{a}) = \sum_{i=0}^{n_{Mess}} (y_i - f(\overrightarrow{a}, t_i, x_i))^2$$

Diese Funktion soll jetzt durch Anpassen des Vektors $\overrightarrow{a}$ minimiert werden. Dabei ist y_i der Wert, den f bei diesen Eingabewerten haben soll, also i.d.R. der tatsächliche Messwert. Dieser Teil besteht grob gesagt darin, dass versucht wird in die Richtung mit geringerer Steigung zu gehen, um so den Gradienten (faktisch die Ableitung) hinabzusteigen und in einem Minimum zu landen. Wie genau, also bspw. wie weit pro Schritt man "hinabsteigt", ist recht komplex und übersteigt meine mathematischen Fähigkeiten. Für diese Arbeit werden Optimizer als eine Art "Black Box" behandelt und nur angewendet.

2.1.2 *absolute Luftfeuchtigkeit*

Die relative Luftfeuchtigkeit setzt sich aus zwei Größen zusammen. Erstens die maximale Luftfeuchtigkeit (der größtmögliche Druck des Wasserdampfes in der Luft), ab welcher die Luft übersättigt wäre und ein Teil zu flüssigem Wasser kondensieren würde. Diese Größe ist von der Temperatur abhängig, denn je wärmer die Luft ist, desto mehr Wasser kann sie aufnehmen.
Die Beziehung zwischen maximalen Wasserdampfdruck und Temperatur wird doch die sogenannte Taupunktkurve dargestellt. Als Funktion für diesen Graphen wird eine Näherung der sogenannten Magnus Formel[3] verwendet wird, die folgendermaßen aussieht: $e_s = C_1 \exp\left(\frac{A_1 \cdot t}{B_1 + t}\right)$
Wobei e_s den maximalen Wassergehalt in der Luft angibt (maximale Luftfeuchtigkeit), $C_1 = 610,94$ Pa, $A_1 = 17,625$, $B_1 = 243,04$ °C. t ist die Temperatur in °C.

Den Prozentsatz zwischen aktueller Luftfeuchtigkeit und maximaler Luftfeuchtigkeit nennt man dann relative Luftfeuchtigkeit, welche das Schimmelwachstum mit am meisten beeinflusst (Siehe auch 2.3.1.1). Gemessen wird in der Regel direkt die relative Luftfeuchtigkeit, da dies messtechnisch einfacher ist. Das geschieht mit einem sogenannten Hygrometer, welches bspw. die Änderung des Widerstands eines Leiters misst, welcher durch die Feuchtigkeit beeinflusst wird[4]. Das gilt auch für die in diesem Projekt verwendeten Sensoren.
Um also aus der relativen Luftfeuchtigkeit (R_L) die absolute Luftfeuchtigkeit e zu berechnen gilt folgende Rechnung:

$$e = \frac{R_L}{100} \cdot e_s$$

2.2 DATEN SAMMELN

Für eine Optimierung werden natürlich erst Daten benötigt. Dabei ist die Frage an welche Daten man kommt und welche für diese Modellierung hilfreich sind, also Einfluss auf das zu

3 Lawrence, "The Relationship between Relative Humidity and the Dewpoint Temperature in Moist Air: A Simple Conversion and Applications" Seite. 226 Gleichung 6.
4 Weber, *Technische Feuchtemessung: in Gasen und Festkörpern* S.90.

modellierende Verhalten haben. Für dieses Projekt sind das die folgenden Daten: Temperatur und Luftfeuchtigkeit in den Badezimmern und außen gemessen, sowie Zeitpunkt (Start und Ende) von Duschen und Lüften. Des Weiteren wurde notiert wie die Fenster beim Lüften geöffnet waren, also wieviel Fläche nach außen geöffnet war.
Natürlich gibt es noch weitere Faktoren die das Lüftverhalten beeinflussen, wie wahrscheinlich der Luftfluss im Bad (sehr aufwendig zu analysieren) oder der Luftdruck (wahrscheinlich nicht sehr einflussreich), aber eine noch breitere Analyse von beeinflussenden Faktoren hätte vermutlich nicht zu großen Verbesserungen geführt und den Umfang dieser Arbeit teilweise erheblich vergrößert.

2.2.1 *Datengewinnung*

Zur Gewinnung der erwähnten Daten werden Temperatur- und Luftfeuchtigkeitsmessgeräte in den Bädern bei uns Zuhause angebracht. In diesem Fall der *Mi Temperature and Humidity Monitor 2* von Xiaomi. Allerdings sind Daten, so wie man sie vom Hersteller her einsehen kann, fast unbrauchbar. Glücklicherweise gibt es ein Projekt namens Telink Flashers[5] mit welchem man eine neue Firmware flashen kann, welche dann lesbar BLE-Broadcasts schicken.

Diese BLE Signale werden aufgefangen und zur Speicherung weiterverarbeitet Dazu wurde ein ESP-32 verwendet, da dieser die benötigten Schnittstellen (WLAN und BLE) besitzt und von ESPHome unterstützt wird. ESPHome kann die Signale auslesen[6] und an weiterleiten Home Assistant, wo die Daten aggregiert werden weitergeleitet.

Home Assistant ist ein System zur Heimautomation, welches direkt eine Integration für ESPHome zur Verfügung stellt. Home Assistant bietet auch die Möglichkeit Automatisierungen zu ”bauen”, sodass man hier bspw. alle 5 Minuten die Raumdaten in eine CSV-Datei schreiben kann. Auch hier ist die Konfiguration einfach[7].

Allerdings hat sich bei den ersten Analysen herausgestellt das diese 5 Minuten zu grob für die Analysen sind, weshalb später dazu übergegangen wurde die Daten mit einer einfachen SQL-Anfrage aus der Datenbank von Home-Assistant zu extrahieren[8].
Diese Daten bieten bereits einige Möglichkeiten zur Analyse. Um aber wirklich Luftfeuchtigkeitspeaks und -tiefen dem Lüften bzw. Duschen zuzuordnen, das ganze also mit den bereits erwähnten Start- und Endzeitpunkten des Duschens bzw. Lüftens zu “synchronisieren“, wurden Listen in den Bädern geführt. Die Messdaten während dieser Zeiträume wurden dann nach Digitalisierung dieser Listen selektiert und gespeichert[9].

Dabei werden Datenpunkte bei denen Teile fehlen (bspw. die Daten draußen oder in dem Bad) gelöscht. Es kann passieren das Teile fehlen, wenn bspw. die Verbindung zur Wetterstation abgebrochen ist oder ein BLE-Broadcast nicht angekommen ist.

2.3 ZIELSETZUNG DER OPTIMIERUNG

Bspw in 1.3 wurde ein “kritischer Wert“ erwähnt, also in dieser Arbeit der Punkt ab dem sich kein Schimmel mehr bilden und festsetzen kann. Dafür erstmal einige Informationen über Schimmel:

5 https://atc1441.github.io/TelinkFlasher.html zuletzt abgerufen am 11.6.2022.
6 Der Code zur Konfiguration von ESPHome zu diesem Zweck ist unter A.1.1 einsehbar.
7 siehe Abschnitt A.1.2.
8 Siehe dafür A.1.3
9 Dies geschieht durch den Code in A.1.4

2.3.1 *Schimmel und Nässe*

Schimmelpilze bestehen aus vielen kleinen Fäden und ernähren sich hauptsächlich von abgestorbenen organischen Stoffen. Sie vermehren sich über Sporen, welche im Verdacht stehen gesundheitsschädlich zu sein. In Bezug auf Schimmel wird eine 50% höhere Rate für Husten und akutem Asthma festgestellt, für obere Atemwegsentzündungen sogar 52%[10].

Das Wachstum von Schimmel wird von vielen Faktoren beeinflusst, maßgebliche Ursachen für Schimmelwachstum sind allerdings hauptsächlich: Erhöhte relative Innenraumluftfeuchtigkeit (Siehe auch 2.1.2) (59%), Eindringen von Regenwasser (41%), Mangelhaftes Lüften (41%), Unzureichende Beheizung (35%), Beschädigte Baumaterialien (29%)[11].

Die Prozentzahl gibt an wie viele der von der Quelle ausgewerteten Fallstudien diesen Faktor als Problem bei der Schimmelvermeidung behandelt haben. Letztendlich ist das größte Problem also Feuchtigkeit in vielen Variationen, sowie fehlendes Lüften und zu niedrige Temperatur. Das sind alles Punkte, die durch effizienteres Lüften angegangen werden können.

2.3.1.1 *Luftfeuchtigkeit*

Wie bereits gesehen hat die Luftfeuchtigkeit (siehe auch Abschnitt 2.1.2) einen großen Einfluss auf das Wachstum des Schimmels. Aber schon bei der Schimmelbildung ist diese bedeutsam, so beziehen Sporen am Anfang fast ihren ganzen Wasserbedarf aus der Luft, weshalb nicht nur das Wachstum mit Lüften verhindert wird, sondern auch das Festsetzen selbst. Die notwendige relative Luftfeuchtigkeit liegt für die Bildung von Schimmel bei über 70%. Es gibt zwar Schimmelpilzarten, die sich schon darunter festsetzen können, diese kommen allerdings nicht in Gebäuden vor[12].

Hinzu kommen weitere weniger bekannte Probleme von feuchten Räumen: Beispielsweise wachsen viele Bakterien und Pilze, welche mit Asthma und weiteren Atemwegsproblemen assoziiert werden, erst ab einer gewissen Luftfeuchtigkeit. Außerdem halten sich einige Allergene bei höherer Luftfeuchtigkeit länger in der Luft.[13]

2.3.1.2 *Temperatur*

Die Temperatur spielt zwar bei der Bestimmung der relative Luftfeuchtigkeit bereits eine Rolle. Aber es gibt auch unabhängig davon einen idealen Temperaturbereich für den Schimmel bei circa 30°C, er gedeiht allerdings auch zwischen 0 und 50°C.[14]

2.3.2 *Schlussfolgerung*

Das Ziel ist es unter 70% Luftfeuchtigkeit zu sein, da dann sich kein Wasser an den Wänden absetzen[15] kann und kein Schimmel entsteht. Die Temperaturgrenzen für Schimmel kann man nicht umgehen, da dies auch die für den Menschen angenehmen Bereich sind und außerdem die Notwendigkeit zu heizen verringert werden soll.

10 Heseltine und Rosen, *WHO guidelines for indoor air quality : dampness and mould* Seite 70.
11 World Health Organization (WHO) Europe, "Fachliche und politische Empfehlungen zur Verringerung von Gesundheitsrisiken durch Feuchtigkeit und Schimmel" S.11.
12 Sedlbauer und Krus, "Schimmelpilz aus bauphysikalischer Sicht" S. 5.
13 Heseltine und Rosen, *WHO guidelines for indoor air quality : dampness and mould* S. XII.
14 Sedlbauer und Krus, "Schimmelpilz aus bauphysikalischer Sicht" S. 4.
15 Da die Luft nicht gesättigt ist.

2.4 DAS MODELL

Um zu wissen wann das kritische Niveau unterschritten wird brauchen wir ein Modell das den Verlauf der Luftfeuchtigkeit voraussagt. Dieses können wir dann nach der Zeit umstellen, um so zu einer gewünschten Luftfeuchtigkeit eine benötigte Wartezeit zu bekommen. Natürlich könnte man auch direkt mit einem solchen Modell anfangen. Aber bei der Auftragung Zeit->Luftfeuchtigkeit ist es einfacher sich vorzustellen was passiert, dies vermeidet Fehler.

2.4.0.1 *Absolute Luftfeuchtigkeit*

Dazu wird mit wenigen Parametern gestartet: Statt relativer Luftfeuchtigkeit und Temperatur wird im Modell die absolute Luftfeuchtigkeit (Siehe Abschnitt 2.1.2) verwendet, wodurch bereits einige Abhänigkeiten wegfallen. Für die anderen Parameter werden Einschränkungen bei den Messpunkten vorgenommen, sodass in dieser Auswertung bei allen Messungen die Verhältnisse (v.a. die Außenluftfeuchtigkeit), die aktuell noch außen vor gelassen werden, sehr ähnlich sind. Das ist nötig, da man nicht, wie bei physikalischen Experimenten üblich, alle Faktoren bis auf einen konstant halten kann, da bspw. das Wetter nicht beeinflussbar ist.

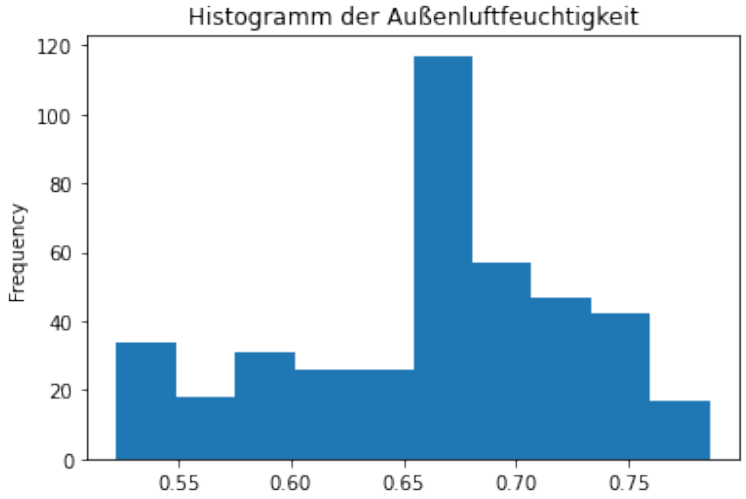


Abbildung 1: Histogramm der absoluten Außenluftfeuchtigkeit. Eigene Darstellung

Aus dem Histogramm (siehe Abbildung 1) ergibt sich, dass am häufigsten absolute Außenluftfeuchtigkeiten zwischen 0,65 und 0,7 gemessen wurden. Deshalb werden die entsprechenden Messreihen verwendet, um möglichst wenig Einfluss durch Fluktuationen zu haben. Geplottet ergibt sich Abbildung 2.

Beim Versuch, eine gute Beschreibung für den Luftfeuchtigkeitsverlauf zu finden, entsteht folgende Funktion (bereits im Plot eingezeichnet):

$$f(t) = k_0 \frac{1}{(t + k_2)^{k_1}} + k_3$$

Die Notwendigkeit des Parameters k_2 ist am einfachsten zu erklären, dieser ist notwendig, da $\frac{1}{n}$ bei $n = 0$ undefiniert ist. Um also jeglichen nicht negativen Wert für die Funktion einsetzen zu können, muss dieser Wert positiv sein. Der Wert wurde unter die Potenz gezogen, da man dadurch k_2 einfach auf 1 setzen kann. Dadurch ist die Funktion für alle positiven Zahlen definiert und man kann sich sicher sein, dass der Bruch immer eine Zahl von 1 bis 0 ergibt. Dadurch sind k_0 und k_3 einfacher unabhängig von dem Bruch betrachtbar, dazu gleich mehr.

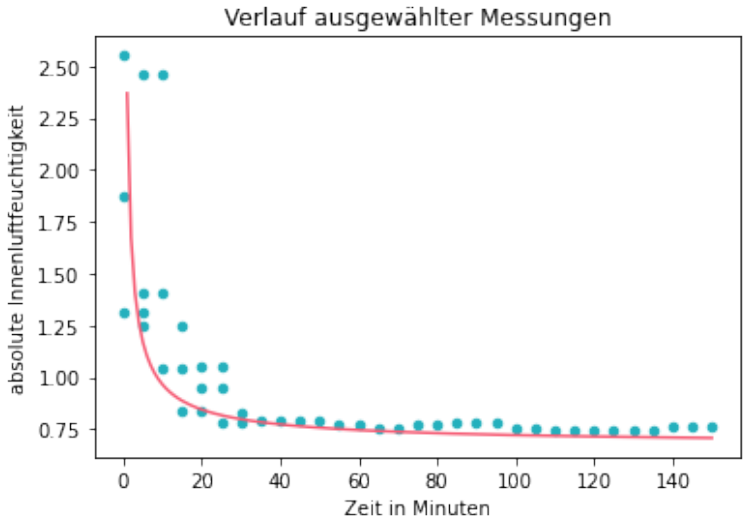


Abbildung 2: Verlauf der absoluten Luftfeuchtigkeit. Eigene Darstellung

k_1 ändert die Steigung der Funktion, denn die Luftfeuchtigkeit fällt nicht exakt mit der Steigung eines normalen $\frac{1}{n}$ ab. Hier ist wahrscheinlich auch die Stellschraube für Einflüsse wie die Fenstergröße, da diese sich auf die Luftaustauschgeschwindigkeit auswirkt.

Was k_3 angeht, wirkt es anfangs, ohne weitere Anhaltspunkte, so, als müsste man diesen einfach optimieren, aber tatsächlich ist es deutlich einfacher. Wenn man den letzten Plot nimmt und noch die Außenluftfeuchtigkeit einzeichnet ergibt sich Abbildung 3.

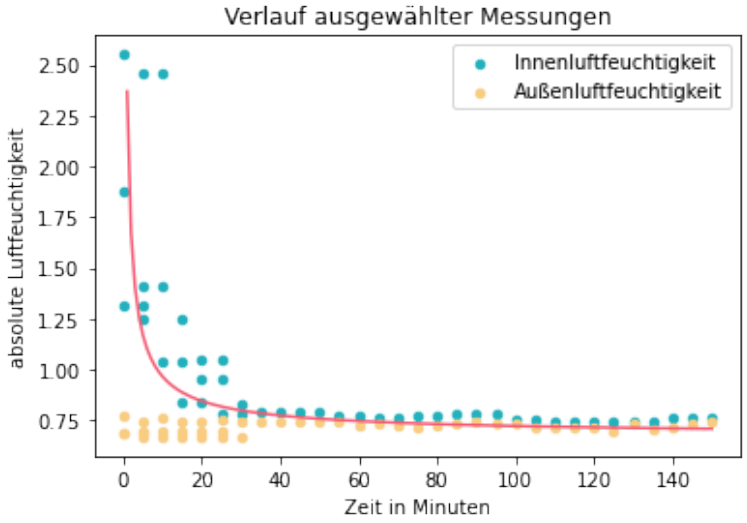


Abbildung 3: Verlauf von absoluter Außen- und Innenluftfeuchtigkeit. Eigene Darstellung

k_3 ist also einfach nur die aktuelle Außenluftfeuchtigkeit. ($k_3 = h_{out}$). Folglich:

$$f(t) = k_0 \frac{1}{(t+1)^{k_1}} + h_{out} \tag{1}$$

Was k_0 angeht, ist es der Sinn dieses Parameters, die Funktion beim ersten Luftfeuchtigkeitspunkt zu starten (bei dem der Bruch den Wert 1 hat). Dieser ist bekannt und wird als Funktionsparameter übergeben. k_0 muss also die Luftfeuchtigkeit zum Zeitpunkt $t = 0$ enthalten, außerdem muss

$$f(0) = k_0 \frac{1}{(0+1)^{k_1}} + h_{out} = k_0 + h_{out} = h_0$$

gelten. Daraus folgt das

$$k_0 = h_0 - h_{out} \tag{2}$$

gelten muss. Mit (1) in (2) folgt:

$$f(t) = (h_0 - h_{out}) \frac{1}{(t+1)^{k_1}} + h_{out}$$

Man sieht auch, dass für $f(t) \xrightarrow{t\to\infty} h_{out}$, was dem erwarteten Verhalten entspricht, da sich die Luftfeuchtigkeit im Bad der Luftfeuchtigkeit außerhalb des Bades annähert.

2.4.0.2 *Temperatur*

Damit existiert ein Modell für die absolute Luftfeuchtigkeit, in der Anwendung wird allerdings die relative benötigt, da diese das Schimmelwachstum beeinflusst[16]. Die Umrechnung wurde bereits in Abschnitt 2.1.2 besprochen. Daraus wird allerdings klar, dass außerdem ein Modell für die Entwicklung der Temperatur benötigt wird. Dazu wird zunächst wieder der Verlauf über die Zeit betrachtet. Siehe Abbildung 4.

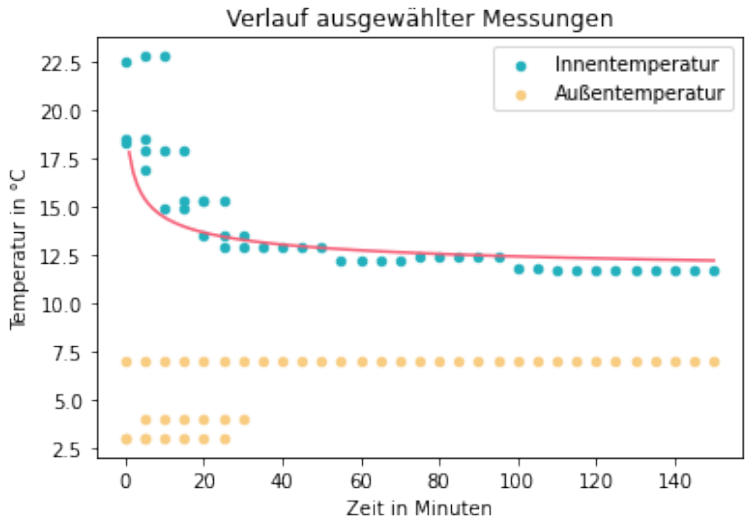


Abbildung 4: Verlauf der Innen- und Außentemperatur. Eigene Darstellung

Man sieht, dass der Verlauf dem der absoluten Luftfeuchtigkeit sehr ähnlich sieht, abgesehen davon, dass er deutlich langsamer absteigt und deshalb die Außentemperatur im Plot nicht von der Innentemperatur erreicht wird, trotzdem kann man sich, aufgrund der Thermodynamik, sicher sein, dass Außen- und Innentemperatur sich auf Dauer angleichen werden. Das schlägt sich einfach in der Steigung nieder. Wenn man die bereits bei der Luftfeuchtigkeit gewonnenen Informationen nutzt, entsteht wieder ein Modell, welches im Kern aus einer antiproportionalen Funktion besteht, sowie einem Parameter, welcher die Steigung bestimmt und einer Addition um 1 in der Potenz, damit die Funktion bei $t = 0$ definiert ist.

$$f_T(t) = (T_0 - T_{out}) \frac{1}{(t+1)^{k_5}} + T_{out}$$

16 Da die relative Luftfeuchtigkeit bestimmt wie einfach sich Wasser aus der Luft ziehen lässt

2.4.0.3 *Fensterfläche*

Allerdings ist bei beiden Modellen noch unklar wie genau die Steigung aussieht, deswegen wird als nächstes betrachtet wie sich die Fenstergröße auf Luftfeuchtigkeit und Temperatur auswirkt.

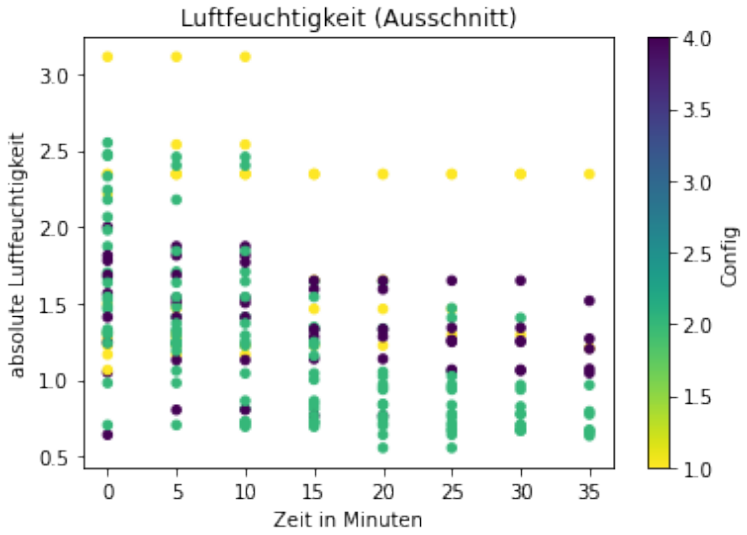


Abbildung 5: Verlauf der absoluten Luftfeuchtigkeit nach Fensteröffnung. Eigene Darstellung

Die Zahlen 1-4 kodieren wie die Fenster geöffnet sind: 1 bedeutet ein Fenster geöffnet, 2 zwei geöffnet, 3 eins geklappt und 4 dementsprechend zwei geklappt. Andere Szenarien wurden nicht probiert, da die Daten eindeutig genug sind. Zum Trainieren des Modells werden die Codes in die tatsächliche offene Fensterfläche umgerechnet. Die Daten für die Luftfeuchtigkeit sieht man in Abbildung 5. Dabei ist klar zu erkennen, dass je mehr Fensterfläche zum Lüften verfügbar ist, desto schneller sinkt die Luftfeuchtigkeit.

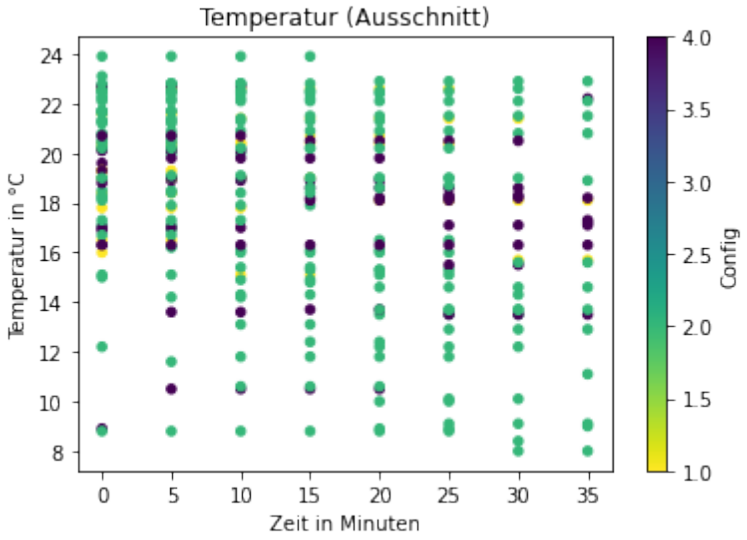


Abbildung 6: Verlauf der Temperatur nach Fensteröffnung. Eigene Darstellung

Bei den Temperaturdaten in 6 ist es nicht so eindeutig. Es ist eine starke Streuung aller Werte zu sehen, trotzdem sind klare Trends erkennbar, so ist die 4 relativ nah beieinander, ebenso die wenigen Einser. Die geöffnete Fensterfläche spielt also durchaus eine Rolle, wenn auch keine so große wie bei der Luftfeuchtigkeit. Aber das wirkt sich dann entsprechend auf den vorstehenden Parameter aus. Da sich die Fensterfläche auf die Steigung auswirkt, also darauf, wie schnell sich die Innenluftfeuchtigkeit an die Außenluftfeuchtigkeit anpasst, erweitert sich das Teilmodell der Luftfeuchtigkeit folgendermaßen:

$$f_h(t) = (h_0 - h_{out}) \frac{1}{(t+1)^{k_1 \cdot F_{größe}}} + h_{out}$$

Und entsprechend auch für das Temperaturmodell:

$$f_T(t) = (T_0 - T_{out}) \frac{1}{(t+1)^{k_5 \cdot F_{größe}}} + T_{out}$$

2.4.0.4 *Raumgröße*

Was die Raumgröße angeht, ist es relativ klar, dass je größer das Volumen ist, desto weniger fällt eine steigende absolute Luftfeuchtigkeit ins Gewicht, da die Luftfeuchtigkeit sich über das ganze Volumen verteilt. Die Außenluftfeuchtigkeit ändert sich von einem Duschen bspw. nicht. Gleichzeitig muss, wenn die Luftfeuchtigkeit erst mal über einem kritischen Wert steigt, mehr Luftfeuchtigkeit entweichen, um die Luftfeuchtigkeit des ganzen Raums wieder zu senken. Es handelt sich also um einen Faktor, der die Steigung dämpft. Dies ist leider nicht an den Daten zu belegen, da die Räume, die zum Erheben von Daten benutzt wurden, eine relativ ähnliche Größe haben und die Fenstergrößen sehr unterschiedlich sind, was einen Vergleich schwierig macht. Das Teilmodell erweitert sich also folgendermaßen:

$$f_h(t) = (h_0 - h_{out}) \frac{1}{(t+1)^{\frac{k_1 F_{größe}}{k_3 R_{größe}}}} + h_{out}$$

Dasselbe gilt natürlich auch für die Temperatur:

$$f_T(t) = (T_0 - T_{out}) \frac{1}{(t+1)^{\frac{k_5 \cdot F_{größe}}{k_6 \cdot R_{größe}}}} + T_{out}$$

2.4.1 *Zusammenfügen der Teilmodelle*

Nun kann man eine Vorhersage für die Luftfeuchtigkeit bekommen. Dazu rechnet man die gegebene Temperatur und die relative Luftfeuchtigkeit mit der bereits in Abschnitt 2.1.2 besprochenen Formel in die absolute um und setzt diese in das Luftfeuchtigkeitsmodell ein. Zusammen mit dem Wert, der sich eingesetzt aus dem Temperaturmodell ergibt, kann man nun wieder auf die relative Luftfeuchtigkeit zu einem gegebenen Zeitpunkt schließen.

Bei der tatsächlichen Anwendung möchte man natürlich, statt aus einer Zeit die Luftfeuchtigkeit zu bekommen, aus einer gewünschten Luftfeuchtigkeit die benötigte Zeit bekommen. Das wurde allerdings während des Erstellens des Modells nicht beachtet, da viele der Visualisierungen sonst nicht so gut funktioniert hätten und es schwerer wäre nachzuvollziehen was die Funktion jetzt genau darstellt, stattdessen wird ein Intervallverfahren genutzt, um in

kleinen Zeitschritten das Modell zu befragen, bis die Luftfeuchtigkeit unter dem gewünschten Wert ist. Letztendlich also eine while-Schleife, die minutenweise die Luftfeuchtigkeit, die das Modell vorhersagt, prüft. Man könnte das natürlich auch als binary-search gestalten, was äußerst effizient wäre. Also also bei der Hälfte der maximal benötigten Zeit (das wäre ein Schätzwert aus den Trainingsdaten) prüfen, wenn es darüber ist die Mitte der Hälfte danach bzw. wenn es darunter ist die Mitte der Hälfte davor, wodurch man in jedem Schritt die Hälfte aller übrigen Optionen als Grenzpunkt ausschließt. Aber es werden in den allermeisten Fällen weniger als 60 Schritte (in Minuten) sein und Angaben unter einer Minute (bspw. 5,43 Minuten) sind auch für keinen Anwender praktisch, weshalb sich es nicht lohnt das zu implementieren (auch wenn es recht einfach ist). Die Implementierung ist im Code der Webapp einsehbar (Siehe Abschnitt A.3).

Absolut wichtig ist auch, dass diese beiden Modelle, wenn man Sie zusammenfügt, theoretisch zeitweise relative Luftfeuchtigkeiten über 100% erreichen könnten. Das ist in Realität natürlich nicht möglich. Es kann passieren wenn die Temperatur schneller als in Realität fällt, bzw. die Luftfeuchtigkeit im Modell zu langsam. Das kann allerdings nur mit sehr abnormen Eingaben passieren, da der Optimizer die Modelle ja trainiert hat, um die Realität möglichst exakt darzustellen. Außerdem wurde das Modell für das Training so programmiert, dass es für Prognosen über 100% Luftfeuchtigkeit einen möglichst großen Fehler, also "Anweisung" sich zu ändern, ausgibt.

2.5 ANALYSE FÜR DAS DUSCHEN

Bei der Betrachtung der Daten fällt auf das es keinen wirklich klaren Trend gibt, zwar steigt es irgendwie nach oben, aber nicht sehr klar.

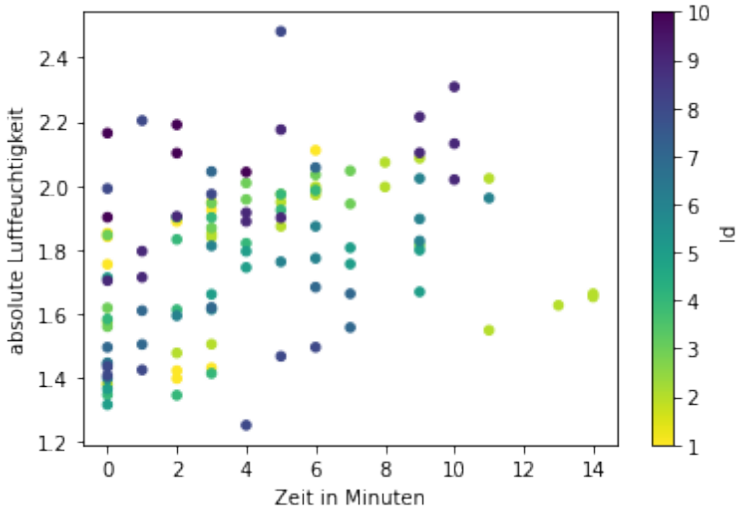


Abbildung 7: absoluter Luftfeuchtigkeitsverlauf. Eigene Darstellung

Man sieht, dass die Werte zwar schon von Anfang an deutlich über den kritischen 65% liegen können, aber gleichzeitig gibt es ganz viele Punkte, die auch niedriger liegen ohne dass es sich auf Dauer ausdünnen würde. Die durchschnittliche Luftfeuchtigkeit ist allerdings höher als die der Gesamtdaten:

```
Über alle Daten hinweg: 57.518542726420186
Während des Duschens: 59.33955223880597
```

(Die Analyse ist in relativer Luftfeuchtigkeit, nicht absoluter).

Das liegt wahrscheinlich daran das wir hier wichtige Faktoren, wie bspw. Wärme des Wassers und Durchflussmenge des Duschkopfs, nicht betrachten, von denen aber auch nicht klar ist wie man sie einfach beschaffen könnte. Außerdem reden wir hier von Zeiträumen von etwa 10 Minuten, was viel kleinere Zeitintervalle als bei Lüften sind, wodurch kleine Fluktuationen mehr die Daten durcheinander bringen.

2.6 TRAINING DES MODELLS

In diesem Teil werden die Parameter der beiden Teilmodelle zur Lüftung optimiert, um diese anwendbar zu machen.

Nachdem das Modell erstellt wurde, müssen nun die Parameter wie in Abschnitt 2.1.1 erwähnt optimiert werden. Der Code dazu findet sich in Abschnitt A.2.

Trainiert man einfach auf allen validen Daten (also ohne Not-a-number Werte), würde man auch mit den Daten trainieren, welche Stunden nach Start des Lüftens aufgezeichnet wurden. Diese sind dann faktisch nur abhängig von den Verhältnissen draußen, deshalb ist es sinnvoll diese Werte herauszunehmen, sodass die Ergebnisse nicht verzerrt werden. Aber ab wann können wir die Daten ignorieren?

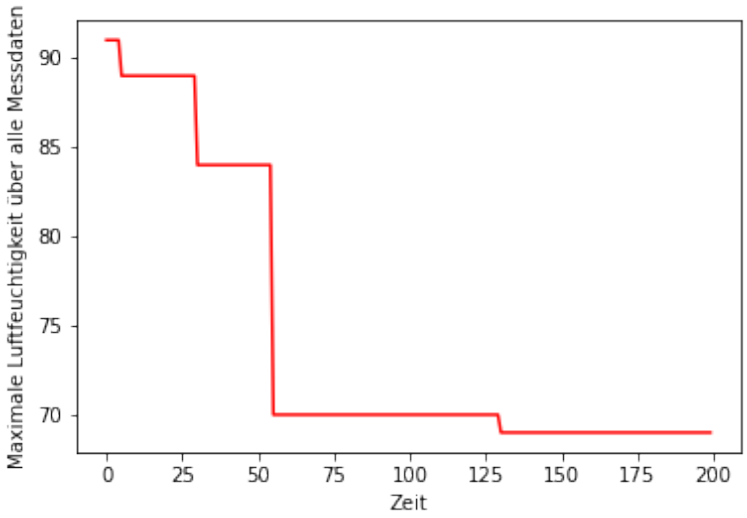


Abbildung 8: Maximal zu erwartende Luftfeuchtigkeit nach Zeit. Eigene Darstellung

Wie man in dem Abbildung 8 sieht, liegen die Luftfeuchtigkeiten im Extremfall ab 55 Minuten unter oder auf der Grenze. Zur Sicherheit werden beim Training allerdings die ersten 70 Minuten betrachtet. Alle Daten danach müssen wir also nicht berücksichtigen, da unsere Funktion nicht wieder steigen kann. Es wurden außerdem einige Daten herausgenommen, bei welchen die absolute Luftfeuchtigkeit wieder angefangen hat zu steigen. Das liegt mutmaßlich daran das es ein paar Mal vorgekommen ist das jemand während des Lüftens geduscht hat, das kann das Modell natürlich nicht voraussagen. Jetzt liegen die Verläufe wie erwartet zum Teil über, und zum Teil unter den korrekten Werten, sind aber in Summe ein sehr guter Fit. Exemplarisch in den Abbildungen 9 und 10. In dem zweiten Bild fällt auf, dass die relative Luftfeuchtigkeit wieder steigt. Das passiert wenn die Temperatur im Verhältnis schneller fällt als die Luftfeuchtigkeit und deckt sich bspw. in diesem Fall mit den Daten.

Für das Luftfeuchtigkeitsmodell ergeben sich hierbei die Parameter:

$$f_h(t) = (h_0 - h_{out}) \frac{1}{(t+1)^{\frac{49,9737675 \cdot F_{größe}}{2,28452262 \cdot R_{größe}}}} + h_{out}$$

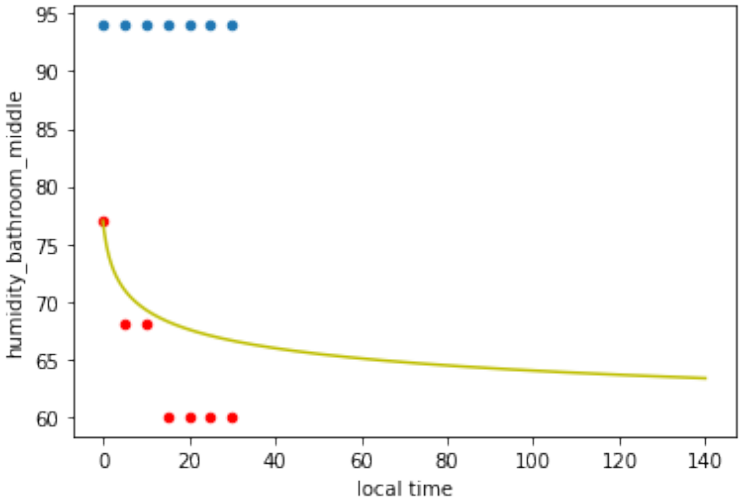


Abbildung 9: Beispiel Verlauf. Eigene Darstellung

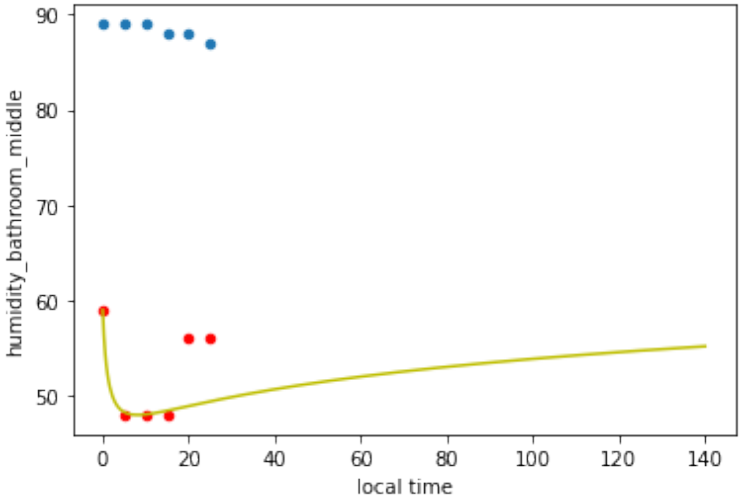


Abbildung 10: Beispiel Verlauf 2. Eigene Darstellung

Für das Temperaturmodell sieht es mit denselben Daten (also nur den ersten 55 Minuten) auch gut aus. Die Parameter sind folgende:

$$f_T(t) = (T_0 - T_{out}) \frac{1}{(t+1)^{\frac{49,3679433 \cdot F_{größe}}{8,39747826 \cdot R_{größe}}}} + T_{out}$$

Es gilt zu beachten, dass die einzelnen Teile zusammen trainiert wurden, um möglichst akkurat zu sein. Es wurde also nicht der Teil mit der absoluten Luftfeuchtigkeit alleine trainiert und dann der Teil der Temperatur, sondern beides zusammen.

2.7 WEBAPP

Die Webapp besteht aus 3 Reitern: Prognose, Anpassen und Einstellungen (als Zahnrad Icon). Beim einfachen Nutzen wird nach Eingabe der erforderlichen Daten ein voraussichtlicher Luftfeuchtigkeitsverlauf ausgegeben, sowie die genaue Zeit, bis der kritische Punkt unterschritten ist.

Unter "Anpassen" kann man eigene Daten eingeben und dann werden im Browser des Gerätes die Parameter angepasst und damit das Modell auf die lokale Situation maßgeschneidert.

Unter "Einstellungen" kann man nicht nur die Werte, welche sich nicht ändern, wie Raumgröße festlegen, sondern auch die Einstellungen sichern und wiederherstellen.

Dabei ist die ganze App, bis auf Anfrage der Wetterdaten, absolut lokal und funktioniert theoretisch auch offline (dann muss man selbst Wetterdaten liefern). Es werden auch keine Cookies oder andere Trackingtechnologien verwendet. Der steuernde Java Script Code ist im Abschnitt A.3 zu finden und die App können sie unter `https://hrgaertner.github.io/vent-optimization` ausprobieren. Ein Screenshot ist in Abbildung 11 zu sehen.

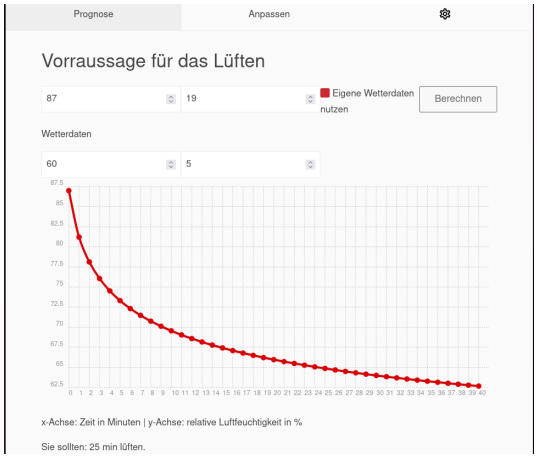


Abbildung 11: Die entwickelte Webapp. Eigene Darstellung

2.8 HOME-ASSISTANT INTEGRATION

Es wurde eine Home-Assistant Integration entwickelt, welche aus den gegebenen Temperatur und Luftfeuchtigkeitssensoren einen virtuellen Sensor erstellt, welcher ausgibt wie lange gelüftet werden sollte, das ermöglicht passgenau Benachrichtigung und allgemein höheren Komfort. Bei mir wird bspw. immer über Signal mitgeteilt das die Fenster geöffnet werden müssen und wie lange sie offen bleiben müssen (Siehe Abbildung 12). Das ist praktischer und genauer als sich nur benachrichtigen zu lassen wenn der Sensor unter 70% gefallen ist, da der Sensor, mit einer Batterie läuft und deshalb nur alle paar Minuten Messdaten liefert in der normalen Einstellung (beim sammeln der Trainingsdaten mussten die Batterien deutlich öfter ausgetauscht werden.). Außerdem schafft es eine bessere Planbarkeit, statt irgendwann möglichst schnell zu machen zu müssen. Das ist aber natürlich für die meisten Menschen keine Option, da deren Messgeräte nicht vernetzt sind, aber nichtsdesto trotz sehr praktisch, wenn anwendbar.[17]

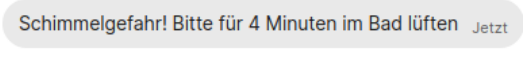


Abbildung 12: Information durch Home-Assistant über Signal. Eigene Darstellung

17 Der Code den essentiellen Teil des Codes ist unter A.4 einsehbar, die ganze Integration, sowie Informationen zur Installation sind auf `https://github.com/HrGaertner/HA-vent-optimization`

3 FAZIT UND AUSSICHT

Wie sich gezeigt hat braucht man zwar durchaus einige Daten, um sinnvolle Voraussagen zu erreichen. Wenn man diese allerdings hat, bekommt man ein relativ robustes Modell, welches sich eignet, um den Temperaturverlust beim Lüften zu minimieren.

3.1 TESTSZENARIO

Ich möchte noch anmerken, dass es aufgrund der zur Datensammlung zu Verfügung stehenden Bäder natürlich nicht möglich war alle Szenarien durchzuprobieren. Bspw. ein Durchlüften mit Fenstern auf beiden Seiten des Raumes war nicht möglich. Deshalb ist dieses Projekt natürlich nur im Rahmen der genannten Grenzen eine Optimierung. Allgemein ist es empfehlenswert, das Modell etwas an die lokalen Verhältnisse mit eigenen Daten anzupassen.

3.1.1 *Nutzung in anderen Räumen*

Prinzipiell funktioniert die Modellierung auch in anderen Räumen, die unter hoher Luftfeuchtigkeit leiden, da es für die Modellierung irrelevant ist wie es zu dieser Luftfeuchtigkeit kam. Dementsprechend ist ein anwenden sinnvoll, aber es sollte am Besten noch mal auf den Raum trainiert werden. In den Meisten solcher Räume, kann man aber wahrscheinlich (bio-) Antischimmelfarbe nutzen[1], was in Bädern eher keine Option ist.

3.2 AUSSICHT

Für die Zukunft interessant wäre interessant, den Fall zu behandeln, in welchem die Außentemperatur höher als die Innentemperatur ist. Dabei ginge es dann nicht um eine Temperaturverlustsminimierung, sondern um eine Minimierung der Temperaturzunahme, allerdings ist dieses Szenario nicht so drängend wie die Verlustminimierung. Aktuell kann das Modell zwar eine solche Situation darstellen, ist aber nicht besonders darauf trainiert und minimiert den Zuwachs nicht. Man sollte also nicht darauf vertrauen. Auch sinnvoll für die Zukunft wäre statt der Erstellung des Modells mithilfe von Graphen, die Beziehungen zwischen den einzelnen Faktoren mit Korrelationen (genauer Korrelationskoeffizienten) herauszubekommen, das würde es auch ermöglichen zu sehen, ob man bspw. das Quadrat eines Faktors benötigt. Außerdem werden aktuell Testdaten gesammelt um die Genauigkeit des Modells, sowohl in Bädern, als auch in anderen Räumen zu bestimmen. Außerdem möchte ich die Webapp noch besser für Mobilgeräte verwendbar machen. Diese Punkte möchte ich in Zukunft weiterführen und verbessern.

Vielen Dank und probieren Sie es gerne aus.

1 Trotzdem ist es nicht gute für die meisten Materialien (bspw. Holz) nicht gut wenn di Luftfeuchtigkeit so hoch ist.

APPENDIX

Der gesamte Code, sowie Teile der Daten und früherer Code sind auf Github unter https://github.com/HrGaertner/vent-optimization komfortabel einsehbar.

A.1 CODE

A.1.1 *ESP-Home Konfiguration*

Teile dieses Codes wurden aus der Vorlage von ESPHome übernommen.

```
esphome:
name: Ble-esp

esp32:  # Specify controller used
board: esp32dev
framework:
type: arduino

# Enable logging
logger:

# Enable Home Assistant API
api:

wifi:
ssid: !secret wifi_ssid
password: !secret wifi_password

esp32_ble_tracker:  # Component needed to connect to Xiaomi
   BLE

sensor:  # Configuration to connect to two of the temperature
   and humidity measurement devices
# Two of those xiaomi devices one for each bathroom
- platform: xiaomi_lywsd03mmc
mac_address: "BLE mac-address"
bindkey: "aquired bindkey provided by Telink"
temperature:
name: "Badezimmer Temperatur"
humidity:
name: "Badezimmer Luftfeuchtigkeit"
battery_level:
name: "Badezimmer Batterielever"

- platform: xiaomi_lywsd03mmc
mac_address: "BLE-Mac-Address"
bindkey: "aquired bindkey provided by Telink"
```

```
    temperature:
    name: "Badezimmer oben Temperatur"
    humidity:
    name: "Badezimmer oben Luftfeuchtigkeit"
    battery_level:
    name: "Badezimmer oben Batterielevel"
```

A.1.2 *Home Assistant Automation*

```
- id: '1635153260161'  # unique ID to indentify automation
alias: Write BLL Values to csv # Name given to it by me
description: ''
trigger:
- platform: time_pattern
hours: '*' # Every
minutes: /5 # fifth minute
seconds: '0' # At zero seconds
condition: []  # No "ifs"
action:  # If triggered write these sensor values in the csv
    format to the file specified through the service called
    writebllvalues
- service: notify.writebllvalues
data:
message: '{{now().strftime("%d.%m.%Y")}}-{{now().strftime("%H
    :%M:%S")}};{{states.sensor.badezimmer_temperatur.state
    }};{{states.sensor.badezimmer_luftfeuchtigkeit.state}};{{
    states.sensor.badezimmer_oben_temperatur.state}};{{states.
    sensor.badezimmer_oben_luftfeuchtigkeit.state}};{{states.
    sensor.wupws_temp.state}};{{states.sensor.wupws_humidity.
    state}};'
mode: single  # Do not run again if the command is still
    running when triggered again
```

Es ist allerdings auch möglich solche Automationen visuell zu konfigurieren

A.1.3 *Später genutzte SQL-Anfrage*

```
import sqlite3
import pandas as pd
con = sqlite3.connect('home-assistant_v2.db')

cur = con.cursor()

data = pd.DataFrame({"temperature": ["unavailable"], "humidity": ["
    unavailable"], "time":[0]})

for row in cur.execute('SELECT state, last_changed, entity_id FROM states
    WHERE entity_id LIKE "sensor.badezimmer_%";'):
if row[-1].split("_")[-1] == "temperatur":
data = pd.concat([data, pd.DataFrame({"temperature": [row[0]], "humidity"
    : [data["humidity"].iloc[-1]], "time":[row[1]]})], ignore_index=True)
```

```
    elif row[-1].split("_")[-1] == "luftfeuchtigkeit":
    data = pd.concat([data, pd.DataFrame({"temperature": [data["temperature"
        ].iloc[-1]], "humidity": [row[0]], "time":[row[1]]})], ignore_index=
        True)
    con.close()

    data["humidity"] = pd.to_numeric(data["humidity"], errors='coerce')
    data["temperature"] = pd.to_numeric(data["temperature"], errors='coerce')
    data["time"] = pd.to_datetime(data["time"], format="%Y-%m-%d %H:%M:%S")
    data.to_csv("shower.csv", sep=";")
```

A.1.4 *Importieren der Daten*

```
    import pandas as pd
    import datetime
    import re

    data = pd.read_csv("data_bll.csv", sep=";")  # Load
    data["time"] = pd.to_datetime(data["time"], format="%d.%m.%Y-%H:%M:%S")
        # Make time to type time

    dush = pd.read_csv("dush.csv", sep=";")
    dush["time"] = pd.to_datetime(dush["time"], format="%Y-%m-%d %H:%M:%S")
        # Make time to type time

    to_process = pd.read_csv("data.../duschen_oben.csv")
    shower_max = max(dush["Id"])
    to_process

    # for shower a bit different for vent
    showering = []

    current_ID = shower_max +1

    for i in range(len(to_process)):
    year, month = 2022, to_process["Monat"][i]
    day_start, day_end = to_process["Tag"][i], to_process["Tag"][i]
    if to_process["Minuten"][i] > to_process["Minute Ende"][i]:
    time_end = to_process["Stunde"][i]+1
    else:
    time_end = to_process["Stunde"][i]
    start = datetime.datetime(year, month, day_start, to_process["Stunde"][i
        ], to_process["Minuten"][i])
    end = datetime.datetime(year, month, day_end, time_end, to_process["
        Minute Ende"][i])
    time_mask = (data['time'] >= start) & \
    (data['time'] <= end)
    showering.append(data[time_mask].copy())
    showering[-1]["Id"] = current_ID
    current_ID += 1
```

```
35          showering = pd.concat(showering, ignore_index=True)
```

A.2 TRAININGSCODE

```
        import numpy as np
        import matplotlib.pyplot as plt
        import scipy.optimize as sco
        import pandas as pd
5
        vent = pd.read_csv("vent.csv", sep=";")
        vent["time"] = pd.to_datetime(vent["time"], format="%Y-%m-%d %H:%M:%S")
            # Make time to type time

        vent = vent[vent["Id"] !=10] # Dropping line going up (look at
            documentation)
10      vent = vent[vent["local time"] <70]

        # Needed to have one column regardless of which room the venting was in
        def temperature_deduction(row):
        if row["room"] == "up":
15      return row["temperature_bathoroom_upstairs"]
        elif row["room"] == "mid" or row["room"] == "middle":
        return row["temperature_bathroom_middle"]
        else:
        raise Exception("unkonwn floor: " + row["room"])
20      vent["temperature"] = vent.apply(lambda row: temperature_deduction(row),
            axis=1)

        def humidity_deduction(row):
        if row["room"] == "up":
        return row["humidity_bathroom_upstairs"]
25      elif row["room"] == "mid" or row["room"] == "middle":
        return row["humidity_bathroom_middle"]
        else:
        raise Exception("unkonwn floor: " + row["room"])
        vent["humidity_goal"] = vent.apply(lambda row: humidity_deduction(row),
            axis=1)
30



        def config_code_to_size(row): # Converting the window code to the actual
            sizes
        code_mid = {1: 0.8*0.95, 2: 2*0.8*0.95, 3: 0.95*0.15 + 0.80*0.005, 4:
            2*(0.95*0.15 + 0.80*0.005)}
35      code_up = 0.6*0.9

        if row["room"] == "mid":
        return code_mid[row["Config"]]
        else:
```

```
    return code_up

vent["Config"] = vent.apply(lambda row: config_code_to_size(row), axis=1)

def room_code_to_size(row): # Converting the window code to the actual
    sizes
if row["room"] == "mid":
return 15*3
else:
return 10*3

vent["room"] = vent.apply(lambda row: room_code_to_size(row), axis=1)

def first_value(row):
first_values_local =  vent[(row["Id"] == vent["Id"]) & (vent["local time"
    ] == 0)]
try:
return first_values_local["humidity_goal"].values[0], first_values_local[
    "temperature"].values[0], first_values_local["humidity_outside"].
    values[0], first_values_local["temperature_outside"].values[0]
except:
return pd.NA, pd.NA, pd.NA, pd.NA

# Variables we only know from the first point in "production" so this is
    being simulated
first_values = vent.apply(
lambda row: pd.Series(first_value(row), index=["humidity", "temperature",
     "humidity_outside",
"temperature_outside"]), axis=1)

relevant_vent = vent[["room", "Config", "local time", "humidity_goal"]]
    # In the end this will be a minimum subset of data needed
# (column wise) so we do not drop too many lines removing NaNs

relevant_vent = pd.concat([first_values, relevant_vent], axis=1)
relevant_vent = relevant_vent.dropna(axis=0, how="any")
relevant_vent = relevant_vent.apply(pd.to_numeric)

relevant_vent = relevant_vent[relevant_vent["humidity"] > relevant_vent["
    humidity_goal"]]

def e_s(T): # Source https://journals.ametsoc.org/view/journals/bams
    /86/2/bams-86-2-225.xml?tab_body=pdf Equation 6 p.226
C_1 = 610.94 #Kp
A_1 =  17.625
B_1 =243.04 #C
return C_1*np.exp((A_1*T)/(B_1+T))

def h(h_0, h_out, F_größe, R_größe, t, k_1, k_2):  # The absolute
    humidity model
return (h_0 -h_out)/((t+1)**((k_1*F_größe)/(k_2*R_größe))) + h_out
```

```
def T(T_0, T_out, F_größe, R_größe, t, k_1, k_2): # The temperature model
return (T_0-T_out)/((t+1)**((k_1*F_größe)/(k_2*R_größe))) + T_out

def model_h(window, room, L_out, L_begin, T_out, T_begin,t, k1,k2,k3,k4,
    training=False):
L_abs_begin = (L_begin/100)*e_s(T_begin)
L_abs_out = (L_out/100)*e_s(T_out)
pred_T = T(T_begin, T_out, window, room, t, k3, k4)
pred_L = h(L_abs_begin, L_abs_out, window, room, t, k1, k2)
result= (pred_L/e_s(pred_T))*100

if training:
to_return = []
for res in result:
if res >100:
to_return.append(-np.inf)
else:
to_return.append(res)
return to_return
return result

def get_F(L_in, configs, func):
def F(parameters):
if [i for i in parameters if i <0]:
return np.inf
return sum((func(*configs, *parameters, training=True) - L_in)**2) # The
    sum of all errors through all measurements
return F



# Functions generating the square error for given constants
error_sum = get_F(relevant_vent["humidity_goal"], (relevant_vent["Config"
    ], relevant_vent["room"], relevant_vent["humidity_outside"],
relevant_vent["humidity"], relevant_vent["temperature_outside"],
    relevant_vent["temperature"], relevant_vent["local time"]), model_h)

start_values =   [50,1,50,1]

res = sco.least_squares(error_sum, start_values, max_nfev=10**6)#
res
```

A.3 WEBAPP CODE

```
function set_defaults() {
        localStorage["constants_vent"] = JSON.stringify([49.9737675 ,
            2.28452262, 49.3679433 ,  8.39747826]);
}
```

```
function toggle_element_vis(elem){
        var x = document.getElementById(elem);
        if (x.style.display === "none") {
                x.style.display = "block";
        } else {
                x.style.display = "none";
        }
}

function fetch_met_no_and_cache(){ //To fetch the weather data from the
    internet
        if (! localStorage.getItem("lat") || !localStorage.getItem("lon")
            ){
                alert("Bitte geben Sie ihren Standort in den
                    Einstellungen für die Wetterdaten ein.");
                throw new Error("Can not get weather data!")
                return
        }
        lat = localStorage["lat"];
        lon = localStorage["lon"];

        fetch('https://api.met.no/weatherapi/locationforecast/2.0/compact
            ?lat='+lat+'&lon='+lon).then(response => {return response.json
            ()})
        .then(response => {
                var result = response["properties"]["timeseries"]
                var last_difference = Date.now();
                var difference
                for (let i = 0; i < result.length; i++){
                        difference = Math.abs(Date.parse(result[i]["time"
                            ])-Date.now());
                        if (difference > last_difference){
                                localStorage["t_out"] = result[i-1]["data
                                    "]["instant"]["details"]["
                                    air_temperature"];
                                setWithExpiry("h_out", result[i-1]["data"
                                    ]["instant"]["details"]["
                                    relative_humidity"])
                                return;
                        }
                        last_difference = difference
                }
        })
        .catch((error) => {
                console.error('Error:', error);alert("Bitte stellen Sie
                    sicher das sie eine Internetverbindung haben, damit
                    die Wetterdaten abgerufen werden können")
        });
}
```

```
//To comply with the terms of met.no one has to cache the values to
    reduce load on their servers. The following two function do this while
     assuring to get new data if necessary
//The concept of the following two function comes from https://www.
    sohamkamani.com/blog/javascript-localstorage-with-ttl-expiry/
function setWithExpiry(key, value, ttl=3600000) {
        const now = new Date()

        // 'item' is an object which contains the original value
        // as well as the time when it's supposed to expire
        const item = {
                value: value,
                expiry: now.getTime() + ttl,
        }
        localStorage.setItem(key, JSON.stringify(item))
}

function get_weather_data() {
        const itemStr = localStorage.getItem("h_out")
        // if the item doesn't exist, return null
        if (!itemStr) {
                fetch_met_no_and_cache()
                return;
        }
        const item = JSON.parse(itemStr)
        const now = new Date()
        // compare the expiry time of the item with the current time
        if (now.getTime() > item.expiry) {
                // If the item is expired, delete the item from storage
                // and return null
                localStorage.removeItem("h_out")
                localStorage.removeItem("t_out")
                fetch_met_no_and_cache();
        }
        return parseFloat(item.value)
}

function import_settings(File) {
        read = new FileReader();
        read.readAsBinaryString(File);
        var json_data = JSON.parse(read.result);
        for (var key in json_data) {
                localstorage[key] = json_data[key];
        };
}

function e_s(temperature){//The maximum absolute humidity by temperature
        C_1 = 610.94 //Kp
        A_1 =  17.625
        B_1 =243.04 //C
        return C_1*Math.exp((A_1*temperature)/(B_1+temperature))
```

```
92      }

        function to_absolute(h, T) {//from relative
                return (h/100)*e_s(T);
        }
97
        function to_relative(absolute, T) {//from absolute
                return (absolute/e_s(T))*100;
        }

102     function temperature_model(T0, T_out, t, cons){
                return (T0-T_out)/((t+1)**((cons[2]*parseFloat(localStorage["
                    window-size"]))/(cons[3]*parseFloat(localStorage["room-size"])
                    )) ) + T_out
        }

        function humidity_model(h0, h_out0, t, cons){
107             return (h0 - h_out0)/((t+1)**((cons[0]*parseFloat(localStorage["
                    window-size"]))/(cons[1]*parseFloat(localStorage["room-size"])
                    ))) + h_out0
        }


        function humidity_over_time_vent(h0, t0, t_out0, h_out0, cons) {
112             var absolute = to_absolute(h0, t0);
                var absolute_out = to_absolute(h_out0, t_out0);

                function vent_humidity(t) {// function only dependend from time
                    for plotting etc.
                        return to_relative(humidity_model(absolute, absolute_out,
                            t, cons), temperature_model(t0, t_out0, t, cons));
117
                };
                return vent_humidity
        }

122     function x_and_y_values(func, start, end, steps) {
                var x = []
                var y = []
                for (let i = start; i <= end; i += steps) {
                        x.push(i)
127                     y.push(func(i))
                }

                return [x, y]}

132     function plot_graph(data, destination) {
                new Chartist.Line(destination, {
                        labels: data[0],
                        series: [
                        data[1]
```

```
                ]
        });
}

function vent() {
        var t0 = parseFloat(document.getElementById('t0').value);
        var h0 = parseFloat(document.getElementById('h0').value);

        if (h0 < 70){
                alert("Zielbereich bereits erreicht.")
                return
        }

        h_out = get_weather_data()

        //plotting the graph
        var func = humidity_over_time_vent(h0, t0, parseFloat(
            localStorage["t_out"]), h_out, JSON.parse(localStorage["
            constants_vent"]));

        if (!func(1)){
                alert("Etwas ist schief gegangen, wahrscheinlich müssen
                    Sie ihre Daten unter Einstellungen eingeben")
                throw new Error("Returned null")
        }
        plot_graph(x_and_y_values(func, 0, 40, 1), '.vent')

        //getting vent time
        var answer = "Die Luftfeuchtigkeit fällt auf absehbare Zeit nicht
             unter möglicherweise schimmelbildende Werte";
        if (h_out < 65){ //65 instead of 70 as a safety buffer
                for (i = 1; i<1000;i++){
                        if (func(i) < 65){
                                answer = "Sie sollten: " + i + " min lü
                                    ften.";
                                break;
                        }
        }}

        document.getElementById("vent_time").textContent = answer;
}

function insertAfter(referenceNode, newNode) {// by https://stackoverflow
    .com/questions/4793604/how-to-insert-an-element-after-another-element-
    in-javascript-without-using-a-lib
        referenceNode.parentNode.insertBefore(newNode, referenceNode.
            nextSibling);
}

var current_datapoint = 1;
```

```
function new_datapoint(){//function to add a new datapoint input to the
    train page
        var keep_data = document.getElementById("keep_data").checked
        var current_data = document.getElementById("current_data").
            checked

        var new_datapoint = document.getElementById("datapoint" + (
            current_datapoint-1)).cloneNode(true);

        new_datapoint.id = "datapoint" + current_datapoint

        for (var i = 1; i < new_datapoint.childNodes.length-2; i+=2){
                new_datapoint.childNodes[i].childNodes[0].childNodes[0].
                    id = new_datapoint.childNodes[i].childNodes[0].
                    childNodes[0].id.split("_")[0] + "_"+current_datapoint
                    ;

                if (!keep_data){
                        new_datapoint.childNodes[i].childNodes[0].
                            childNodes[0].value = "";
                }
        }

        insertAfter(document.getElementById("datapoint"+(
            current_datapoint-1)), new_datapoint);
        current_datapoint += 1;
}

function train(){
        //getting the data from the inputs
        temperature = document.getElementById("T_0").value;
        humidity = [];
        local_time = [];

        for (var i = 1; i < current_datapoint; i++){
                current_x = []
                var datapoint = document.getElementById("datapoint"+i);
                local_time.push(datapoint.childNodes[1].childNodes[0].
                    childNodes[0].value);
                humidity.push(datapoint.childNodes[3].childNodes[0].
                    childNodes[0].value);
        }

        fnc = function(cons){//sum of error function
                model_prediction = humidity_over_time_vent(humidity[0],
                    temperature, localStorage["t_out"], get_weather_data()
                    , cons);
                sum = 0
                for (var i = 0; i < humidity.length; i++){
                        sum += (model_prediction(local_time[i])-humidity[
                            i])**2;
```

```
                    };
                    return sum;
                };
                //optimizing and storing back
222             var solution = fmin.nelderMead(fnc, JSON.parse(localStorage["
                    constants_vent"]), {maxIterations:20});//Only change the
                    constants slightly
                localStorage["constants_vent"] = JSON.stringify(solution["x"]);

                alert("Das Modell wurde angepasst");
        };
```

A.4 AUSCHNITT: HOME-ASSITANT INTEGRATION

Der "wichtige" Teil des Codes für die Integration. Die Integration wurde von der Integration mold indicator abgeleitet, welche allerdings etwas völlig anderes macht, und nur als eine Art Vorlage verwendet wurde für die ganzen benötigten Klassen.

```
async def async_update(self) -> None:
        """Calculate latest state."""
        _LOGGER.debug("Update state for %s", self.entity_id)
4       # check all sensors
        if None in (self._indoor_temp, self._indoor_hum, self._outdoor_temp):
            self._available = False
            self._outdoor_absolute_humidity = None
            self._indoor_absolute_humidity = None
9           return

        # re-calculate e_s and vent time
        self._calc_indoor_absolute_humidity()
        self._calc_outdoor_absolute_humidity()
14      self._calc_time_to_vent()
        if self._state is None:
            self._available = False
            self._outdoor_absolute_humidity = None
            self._indoor_absolute_humidity = None
19      else:
            self._available = True

    def _calc_e_s(self, temp):
        """Calculate the maximum possible absolute humidity for the indoor
            temperature"""
24      # According to https://journals.ametsoc.org/view/journals/bams/86/2/bams
            -86-2-225.xml?tab_body=pdf Equation 6 p.226
        C_1 = 610.94 #Kp
        A_1 =  17.625
        B_1 =243.04 #C
        return C_1*math.exp((A_1*temp)/(B_1+temp))
29

    def _calc_indoor_absolute_humidity(self):
```

```
        self._indoor_absolute_humidity = (self._indoor_hum/100)*self._calc_e_s(
            self._indoor_temp)
        _LOGGER.debug("Indoor absolute humidity: %f", self.
            _indoor_absolute_humidity)

    def _calc_outdoor_absolute_humidity(self):
        self._outdoor_absolute_humidity = (self._outdoor_hum/100)*self._calc_e_s(
            self._outdoor_temp)
        _LOGGER.debug("Outdoor absolute humidity: %f", self.
            _outdoor_absolute_humidity)

    def _humidity_model(self, time):
        return (self._indoor_absolute_humidity - self._outdoor_absolute_humidity)
            /((time+1)**((k_1*self._window_size)/(k_2*self._room_volume))) + self.
            _outdoor_absolute_humidity

    def _temperature_model(self, time):
        return (self._indoor_temp - self._outdoor_temp)/((time+1)**((k_3*self.
            _window_size)/(k_4*self._room_volume))) + self._outdoor_temp

    def _calc_time_to_vent(self):
        """Calculate the time until the humidity is under max_allowed_hum"""

        if self._indoor_absolute_humidity <= self._outdoor_absolute_humidity:
            _LOGGER.debug("Venting has no point, the outside is to humid")
            self._state = "endless"
        else:
            # One could use a binary search here, but it is unecessary
            for i in range(301):
                if (self._humidity_model(i)/self._calc_e_s(self.
                    _temperature_model(i)))*100 <= self._max_hum_allowed:
                    self._state= f"{int(i):d}"
                    break
            else:
                self._state = ">5h"
                _LOGGER.debug("Venting would take longer than 5h")


        _LOGGER.debug("You have to vent %s minutes", self._state)
```

Habeck, Robert. "Habeck zur Gas-Frühwarnstufe". In: *tagesthemen* (März 2022). URL: https://www.tagesschau.de/wirtschaft/verbraucher/habeck-tagesthemen-gasversorgung-101.html.

Heseltine, Elisabeth und Jerome Rosen. *WHO guidelines for indoor air quality : dampness and mould*. Hrsg. von @ World Health Organization 2009. WHO Regional Office Europe, Jan. 2009. ISBN: 9789289041683.

Lawrence, Mark G. "The Relationship between Relative Humidity and the Dewpoint Temperature in Moist Air: A Simple Conversion and Applications". In: *Bulletin of the American Meteorological Society* 86.2 (2005), S. 225–234. DOI: 10.1175/BAMS-86-2-225. URL: https://journals.ametsoc.org/view/journals/bams/86/2/bams-86-2-225.xml.

Sedlbauer, Klaus und Martin Krus. "Schimmelpilz aus bauphysikalischer Sicht". In: *Fraunhofer-Institut fü r Bauphysik, Holzkirchen* (2003).

Weber, Dieter. *Technische Feuchtemessung: in Gasen und Festkörpern*. Nachdr. 1. Auflage. Vulkan-Verlag GmbH, 2002. ISBN: 978-3-8027-3201-0.

"Fachliche und politische Empfehlungen zur Verringerung von Gesundheitsrisiken durch Feuchtigkeit und Schimmel". In: (2010). Hrsg. von World Health Organization (WHO) Europe. URL: https://www.euro.who.int/__data/assets/pdf_file/0016/121426/E92998G.pdf (besucht am 02. 12. 2022).

ERKLÄRUNG

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als die angegebenen verwendet habe.

Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Mutterstadt, Dezember 2022

Jonathan Gärtner

KOLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Der Code zu der Arbeit kann hier eingesehen werden:

https://github.com/HrGaertner/vent-optimization