Window Alert Prediction using Environmental Sensor Data

Cheng Chun, Franklin Martinez, Kamal Pandala, Keshav Goel, Zac Turner
Department of Computer Science, University of Bristol
Email: {cc17534, fm17982, kp17860, kg17822, zac.turner.2014}@bristol.ac.uk
Github: https://github.com/ZacTurner/applied-data-science

1 Introduction

The extensive network of sensors in a smart home offers several opportunities for monitoring and assisting those living there. This project is based upon the foundation laid by the SPHERE project [1] at University of Bristol. It exploits the existing setup of environmental sensors to collect the temperature and humidity data and predict whether a window is left open in a room.

This project aims at helping those living in the SPHERE smart homes to maintain the air quality inside their homes. This can be achieved by alerting them about the prolonged activity of windows being left open in any of the available rooms. The primary reason behind opening the windows is for air circulation and ventilation. However, in most cases, the windows are left open for quite a long duration. This could be due to the forgetfulness of the people present in the homes, or leaving the home in a rush for work or other purposes. A window left open without any monitoring poses serious problems, especially in regions where health and safety is a concern. A major such concern is trespassing into the property by individuals if the windows are left open or unattended for long periods.

Another important problem is the degradation of air quality with prolonged exposure to the environment. Such prolonged exposure leads to the accumulation of disease causing pathogens and harmful suspended pollutants within the house. As soon as the windows are opened, we are able to perceive changes in temperature and humidity in the house. The humidity levels should be monitored, as it can cause dampness within the house. The relative temperature and humidity play a significant role in the influenza epidemiology and its evolution [2]. The elderly, new-borns and toddlers are sensitive to such abrupt shift in weather conditions and can easily face health problems [3]. The variation in temperature causes the heating systems or the air conditioner units to function such that the temperature is regulated at the optimal values. This results in severe wastage of energy and impacts the environment, undermining the practice of sustainability.

To address the issues highlighted above, in this project, we aim to build a data analysis pipeline that can predict if a window in any of the rooms is left open. This prediction can be used with other sensors and appliances such as the heating system to pause or resume the heating at optimal intervals, to set alarm systems for vigilance or to alert if the people are leaving the house with a window left open. To understand the dynamics of the temperature and humidity variation from data, we have performed experiments at the SPHERE house.

The following objectives have been fulfilled to achieve the aim of window alert prediction. A database has been set up in a similar manner to that of the SPHERE Project that can be used to query and explore the data at any time. However, the data is not clean, it might have missing and erroneous values. Therefore, the data was cleaned to ensure that the subsequent results are not affected. The data is then used for exploratory data analysis and is also used to provide informative visualisations with insights. Simultaneously, we performed the classification task, that comprises of feature engineering, model training and productionising the model.

2 Data Collection - Experiment Design

To build the data pipeline and train the machine learning model, we had to simulate the activity in the SPHERE house and fetch the recorded data. The SPHERE house is a fully furnished two-bedroom house in the University of Bristol campus. It was utilised by the SPHERE project to be used as an experimental facility.

It comprises of two bedrooms, lounge, hallway, a bathroom and a kitchen along with a dining room. The rooms are equipped with environmental sensors along with Passive Infra-Red (PIR) sensors. There are RGB-D cameras installed to take video recording in the living room, the hallway and the kitchen. There are also wearable sensors available in the house, that are equipped with tri-axial accelerometer. All sensors in the house are synchronised via Network Time Protocol (NTP) [4].

To simulate the activity, the windows in bedrooms and the living room were left open for about 90 minutes. To account for the routine activities such as shower and cooking in the kitchen that might affect the temperature and the humidity inside the house, we simulated these activities as well. Cooking in the kitchen was simulated by preparing tea for four with four people present in the kitchen for about 10 minutes. Then, the window in the kitchen was left open for an hour. To account for humidity in the bathroom from a hot shower, we left open the hot water tap in the bathroom for about ten minutes. Then the window in the bathroom was left open for about 25 minutes.

While performing the activities to simulate the real-world scenarios, the exact timestamps and the external weather conditions (available on the internet) were noted to be used later for annotation and comparison purposes.

3 Data Storage

The data recorded from the experiment performed at the SPHERE Smart Home is stored in a document-oriented database, MongoDB. However, due to the guidelines laid down by the SPHERE project to ensure privacy we are not allowed to access it directly. Instead, the data was provided to us in JSON format over the internet. The larger file (8 MB) contains the measurement readings of the environmental sensors fitted across various rooms in the house. The readings were taken on 21st March between 14:00 to 18:00 hours. The smaller file provides the metadata information, that is the information about the environmental sensors themselves.

In order to provide a data analysis environment similar to the SPHERE Smart Home project, we decided to set up a MongoDB database in the cloud. By provisioning a database in the cloud, it allows all the members of the group to query the initial data and its subsequent transformations. We have chosen the latest version of MongoDB database called Atlas, and the infrastructure was provisioned in Amazon Web Services (AWS) cloud. AWS cloud vendor was chosen because it provides a multitude of features that are relevant to our data pipeline in a single ecosystem which is not available with other cloud vendors. The table 1 provides a brief description of the implementation.

Table 1: MongoDB Atlas Cloud

Version	Version 3.4.14
Type	Replica Set:3 nodes
Region	N. Virginia (us-east-1)
Cloud Provider	Amazon AWS : free tier

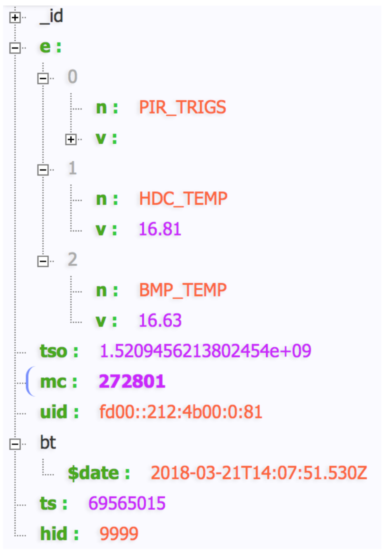

```
_id
e :
   0
      n : PIR_TRIGS
      v :
   1
      n : HDC_TEMP
      v : 16.81
   2
      n : BMP_TEMP
      v : 16.63
tso : 1.5209456213802454e+09
mc : 272801
uid : fd00::212:4b00:0:81
bt
   $date : 2018-03-21T14:07:51.530Z
ts : 69565015
hid : 9999
```

Figure 1: A sample of one element of the measurements in the JSON file

3.1 Data Import

The measurement readings in the JSON file were organized in a chronological order. The first step was to filter the data, to select the elements that contain only the readings about the temperature and humidity. The relevant sensor data was labelled as DC_TEMP, BMP_TEMP for temperature, and HDC_HUM for humidity. Having the data filtered, we import the JSON file into the MongoDB database.

A typical JSON element of the environmental sensor data is of the structure depicted in the figure 1, where the element "e" contains the set of the pairs: "n": type of the sensor and "v:" the measurement. Besides, the element "bt": provides the time of the measurement. Another important value is the "uid" which identifies the sensor in the house.

4 Data Cleansing

After the data is generated and sorted, it needs to be cleansed to ensure that it is usable. Data cleansing is essential for several reasons. Firstly, it ensures that all data is valid. This stops errors being propagated through the rest of the pipeline and potentially damaging the results. Secondly, it helps filter the data to remove unnecessary and excess data, so that this data doesn't have to be processed further into the pipeline. Finally, we used this part of the process to put the data into clearer and more usable formats, splitting it into different datasets for different uses.

4.1 Initial Cleansing

To access the database, we used PyMongo. PyMongo generates native Python dictionary objects from each collection, which allowed us to use Python and NumPy to cleanse the data.

The first step for all of the outputted datasets is to filter out the relevant datapoints. For our project, we firstly isolated only the relevant sensor data, using the given UIDs in the 'items' dataset. From these sensors, we extracted the temperature data (`BMP_TEMP` and `HDC_TEMP`), and the humidity data (`HDC_HUM`), where they exist, along with the time that these datapoints were recorded. For a final temperature value, the two temperature sensors' data were averaged if both were present. Otherwise, only the present value was used.

We used a dictionary of rooms with their sensor values to attach room names to the various sensor points. Further dictionaries contained times where the window in each room was opened and closed, if applicable. Using these times, we were able to determine whether the window was open for a given datapoint, and how long it had been open for. For datapoints where the window was not open, the '`window_open_time`' value was set to 0.

4.2 Dataset 1: temperature and humidity for all rooms

This dataset is the least modified from the base cleanse described above. Two CSV files were generated, one for temperatures and one for humidity, containing every data point with the relevant information, along with the time, room, whether the window was open and for how long.

`time`	`room`	`temperature`	`window_open`	`window_open_t`
15:22:59	hall	17	False	0
15:23:02	bedroom	19	True	0:06:02
15:23:05	stairs	16	False	0

Table 2: Extract from `dataset_1/temperature.csv`, values rounded to fit in this document

4.3 Dataset 2: split by rooms

Dataset 2 is very similar, but with two CSV files *per room*, split into subfolders, rather than having room as a column in the CSV files. Otherwise, the headers are identical.

`time`	`humidity`	`window_open`	`window_open_time`
15:40:52	73.2	False	0
15:41:17	73.2	True	0:00:17
15:41:42	73.1	True	0:00:42

Table 3: Extract from `dataset_2/kitchen/humidity.csv`, values rounded to fit in this document

4.4 Dataset 3: grouped by time

Dataset 3 again is very similar to Dataset 1, but rather than having every single individual data point, the datapoints were sorted and averaged by room for 30 second time periods. This meant that every data point can contain data both for temperature and humidity, despite these not always being recorded simultaneously in the original data. This is useful for machine learning. Furthermore, averaging also helps to reduce error, so these values are likely to be closer to the true values.

This grouping was achieved by iterating through the records, with a variable containing the current 30 second range. If the current record was within this range, the temperature and humidity were each added to arrays for the room's temperature or humidity for that 30-second range. If the record was outside of this range, the range would increase by 30 seconds until this was no longer the case, and the values in the previous arrays would be averaged to give the value for that previous 30-second range. Since the dataset is sorted by time, no records will be missed out through this process.

`time`	`room`	`temperature`	`humidity`	`window_open`
15:30:00	stairs	15.6	53.68	False
15:30:00	bedroom	18.4	33.03	True
15:30:30	hall	17.0	47.17	False

Table 4: Extract from `dataset_3/data.csv`, values rounded to fit in this document

5 Classification Task

In this section, we discuss about how we transformed the features for a better representation of the data and how we modelled the classification problem based on these features. We explain the rationale behind the resulting model and discuss the evaluation results of this model against test data.

5.1 Feature Engineering

ML algorithms are used to learn from data and arrive at a solution for a given task. The features that make up the data are the essential characteristics of the problem that we are trying to solve. Hence, we need to make sure that the utilised features are the best representation of the data that leads to a simple but robust model. To achieve this, we have explored the available features in the context of our problem, their relationships and their relevance. We have iteratively refined the feature set by going through the following steps:
Base feature set: time, room, temperature, humidity, window_open_time

Feature Selection: Windows in a home act as ventilation systems and, as an inlet for solar energy. By opening and closing widows, we alter certain physical characteristics within a room such as temperature and humidity [5]. However, how much these characteristics are influenced depend upon the inherent properties of a room such as initial temperature and humidity, the volume of a room, the number of people and light sources present within a room and so on. As the size of our dataset is small, we opted to choose a small subset of features that would likely help us distinguish whether a window is open or not. This led us to choose temperature and humidity as the first set of features. We did not select time as a feature as the dataset is relatively small and simple. Also, the experiment was simulated with an idealistic prior where all windows were closed. And, the resultant data has large contiguous blocks of the window being open and closed at the same times across all rooms which does not accurately represent the real-world. Hence, we did not opt for modelling based on the time series.
Feature set: temperature, humidity

Feature Construction: We explored the available inherent properties of a room and the house that we can use to better model our problem. In our existing dataset, we have the type of room as a feature. However, this does not generalise across homes as a label like 'living room' is different among them. Hence, we chose the volume of a room as a defining characteristic rather than the type of the room. This led to the addition of volume as the third feature in the feature set. The data we collected contains the absolute temperature and humidity on a day under a particular weather pattern in one season. Hence, the usage of absolute values would not lead to a model that would generalise across weather patterns and seasons. Hence, to counter this, we resorted to the usage of relative temperature and humidity difference. However, as our data was collected only in one instance, we were sceptical about practicality of this approach. Hence, we constructed two feature sets to model and evaluate.
1st feature set: absolute_temperature, absolute_humidity, volume
2nd feature set: relative_temperature, relative_humidity, volume

Feature Importance: We arrived at the two feature sets by exploring the data, reasoning based on the domain knowledge and from intuition. We wanted to reinforce these choices and quantify the importance of each feature to the model. This analysis would help us to better understand the problem.

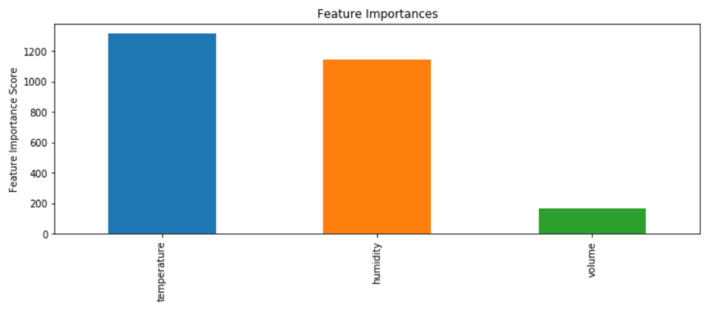


Figure 2: The feature importance of the 1st feature set (absolute values)

Figure 2 depicts the feature importance for the features present in the first feature set which contain the absolute values of temperature and humidity. We can infer from the figure that the features, temperature and humidity are highly relevant to the classification task.

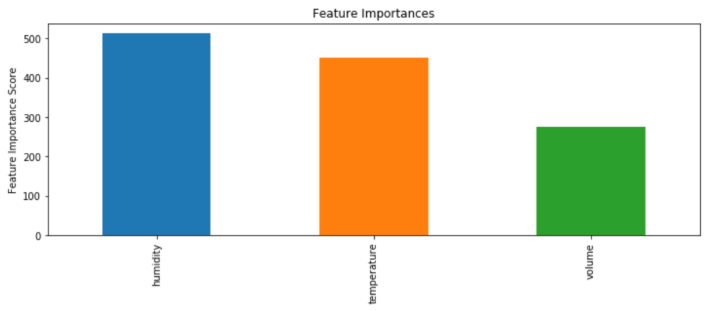


Figure 3: The feature importance of the 2nd feature set (relative values)

Figure 3 depicts the feature importance for the features present in the second feature set which contain the relative values of temperature and humidity. Unlike the feature importance of the first feature set, the importance of the different features are more spread out and no individual or a combination of features dominate the rest.

5.2 Model Training

Our problem consists of a dataset containing features that are of the numeric type. We also have associated labels of the binary type (0, 1) for each record in the dataset that represent whether the window is open or not. For a new data point consisting of the temperature and humidity readings along with the volume of the room, we would like to predict if the window is left open. This problem can be approached as a supervised learning problem and falls under the classification task. We have opted for using decision trees as they can represent visually and explicitly the decisions that are made in the decision-making process. This helps us better understand the problem and the logic involved in arriving at a solution. In our modelling task, we have opted for using Random Forest and Gradient Boosting algorithms with both the constructed feature sets, giving rise to 4 possible models. We pick the best model based on the evaluated metrics and productionise it on cloud for high availability and scalability.

5.2.1 Random Forest with scikit-learn

Random Forest is one of the most popular supervised Machine Learning (ML) algorithms in use. It falls under the ensemble ML methods category known as Bootstrap Aggregation, or bagging. An ensemble method approaches a problem by making use of multiple ML algorithms and combines their predictions to obtain a more accurate prediction than the usage of any single algorithm. Decision trees are easily affected by a small change in the input as the split points are altered as a result. This results in the decision trees having a high variance. Hence, they do not generalise well to new data. Bootstrap Aggregation is a means to reduce such variance and obtain a model that can generalise well by making use of votes by the different individual decision trees for the best prediction based on the maximum vote. However, these decision trees can still have correlation amongst themselves. Random Forest is an ensemble algorithm that reduces such correlation between the decision trees and gives better predictions than ordinary ensemble methods. We have used the Random Forest algorithm implementation of the scikit-learn library [7] with the 2 feature sets with 80% of the data used for training and the remaining for testing. The following tables show the obtained results from the training and testing of the model.

Table 5 shows the result of applying Random Forest algorithm with the first feature set. It produced a True Positive Rate (TPR) of 93.6% and a F1 score of 0.958. Table 6 shows the result of applying Random Forest algorithm with the second feature set. It produced a True Positive Rate (TPR) of 61.8% and a F1 score of 0.663.

Actual|Predicted	window_closed	window_open
window_closed	559	2
window_open	7	103

Table 5: Confusion Matrix for 1st feature set using Random Forest with 20% as test data

Actual|Predicted	window_closed	window_open
window_closed	534	27
window_open	42	68

Table 6: Confusion Matrix for 2nd feature set using Random Forest with 20% as test data

5.2.2 Gradient Boosting with py-xgboost

Gradient Boosting algorithm have recently become the most used algorithms in data science competitions such as those held by Kaggle. The basic idea behind boosting is to use a combination of weak learners according to some pattern that would result in a strong predictor. They differ from the previous bagging method as in the bagging approach, the classifiers are independent and are run in parallel. However, in boosting, a classifier depends upon the previous one and is sequential in nature. One of the first realisations of this idea is the Adaptive Boosting approach. The Gradient Boosting algorithm approached boosting as a numerical optimisation problem. We have used the py-xgboost library along with the scikit-learn library with the 2 feature sets with 80% of the data used for training and the remaining for testing. The following tables show the obtained results from the training and testing of the model.

Table 7 shows the result of applying XGBoost algorithm with the first feature set. It produced a True Positive Rate (TPR) of 91.8% and a F1 score: 0.957. Table 8 shows the result of applying XGBoost algorithm with the second feature set produced. It a True Positive Rate (TPR) of 62.7% and a F1 score of 0.670.

Actual|Predicted	window_closed	window_open
window_closed	561	0
window_open	9	101

Table 7: Confusion Matrix for 1st feature set using XGBoost with 20% as test data

Actual|Predicted	window_closed	window_open
window_closed	534	27
window_open	41	69

Table 8: Confusion Matrix for 2nd feature set using XGBoost with 20% as test data

From the above results, we can infer that the best results were achieved for the 1st feature set and both the algorithms performed nearly the same. We have also explored the resulting decision trees for the model and an example decision tree can be viewed at click here.

5.3 Productionization

We have trained the data by making use of the 2 constructed feature sets utilising the Random Forest and the Gradient Boosting algorithms. We had made use of libraries such as scikit-learn, py-xgboost, numpy and matplotlib on Python for training and prediction. At an infrastructure level, we had utilised our own personal laptops with data stored on their local storage for the modelling task. Although this approach is good enough for exploring, modelling and evaluating the data and the resulting models, it is not a best practice for production usage. The key requirements for production usage are high availability, scalability and robustness. A data pipeline is not complete if it is not customised for deployment. With this in mind, we have chosen to use AWS SageMaker [6], that allows users to build, train and deploy ML models at scale on the cloud. We uploaded the datasets onto Amazon S3 that offers durable, low-latency and high-throughput access. We provisioned a Jupyter notebook using SageMaker on a t2.medium instance (2 vCPU, 4GB RAM) which was used for exploring, visualising and modelling the data. SageMaker offers existing ML algorithms as templates and XGBoost is available due to its wide spread usage. Based on the results in previous section, we decided to use the 1st feature set with XGBoost algorithm for training. To achieve this, we provisioned via code a m4.xlarge instance (4 vCPU, 16GB RAM) for training of the model which exists only for the duration of the training. Once training is completed, the resultant model is stored on S3 automatically. We deployed a new m4.xlarge instance which would serve as the back-end for handling any prediction requests based on new data. SageMaker provides a comprehensive interface, both graphically and through code, that can be used to deploy large-scale ML applications for production usage. By deploying our data pipeline on SageMaker, we have geared it towards production usage.

6 Data Visualization

We have hosted 2 visualisation dashboards on cloud which can be accessed at `http://129.158.81.250:5006/ads-dv1` and `http://129.158.81.250:5007/ads-dv2`. In the visualization part, we use two Python libraries, Pandas and Bokeh. As a BSD-licensed open source library, Pandas library offers a set of functions to process rich types of data. It is based on

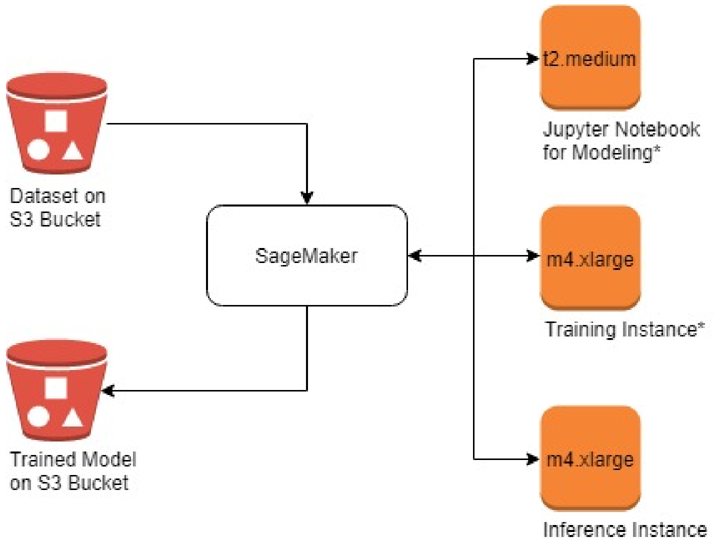


Figure 4: Production mode architecture

the matplotlib library, which makes it a very convenient tool for extracting and analyzing the data. Moreover, it provides flexible data manipulation techniques such as those available in spreadsheets and relational databases, and fast data processing as available in Numpy. Series and DataFrame are two commonly used data structures in Pandas. Note that the data format used in our project is CSV. After loading the CSV data into Python, we then use Bokeh to visualize it.

Bokeh is an interactive visualization library which specifically focuses on presentation on web browsers. Due to the requirements of our project, we collected temperature and humidity data from every room. Using other visualization tools, we obtain lacklustre visualisations that are not user friendly. However, using Bokeh, we are able to solve the problem by making use of templates that provide rich and interactive visualisations. Below are some plot results of the data we collected from the SPHERE house:

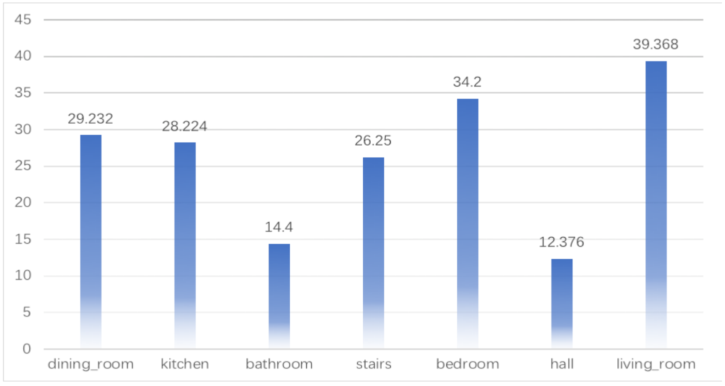


Figure 5: The volume (m^3) of each room in the SPHERE house.

Figure 5 is the volume histogram of various rooms in the SPHERE house. As shown in the picture, living room occupies the biggest space (39.368 m^3), followed by the master bedroom (34.2 m^3). Living room is on the ground floor and master bedroom is on the first floor. Both these rooms are in the south part of the house. Kitchen and dining room are adjacent and in the ground floor with bathroom located in the first floor. All of these rooms are in the northern part of the SPHERE house. Dining and living room are adjacent to each other.

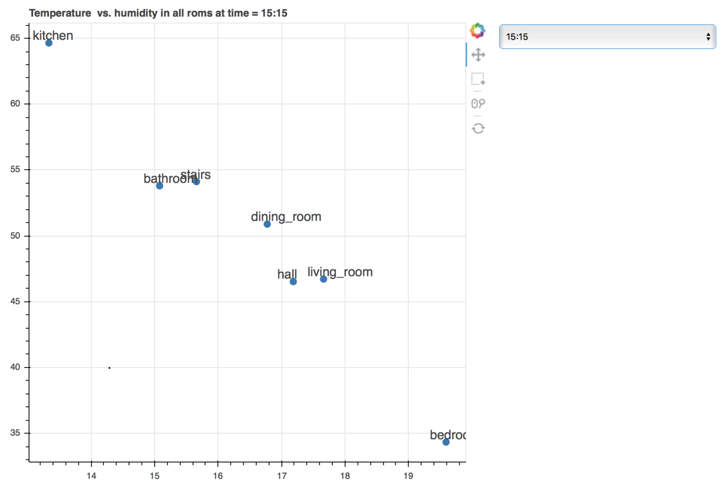


Figure 6: Temperature vs Humidity of all rooms at 15:15 hours

Figure 6 is the comparison between temperature and humidity in various rooms at 15:15 hours (before we opened the windows in the living room and the master bedroom). Bokeh tool box allows us to interactively choose different times to plot the result from the select box widget which is shown in the figure.

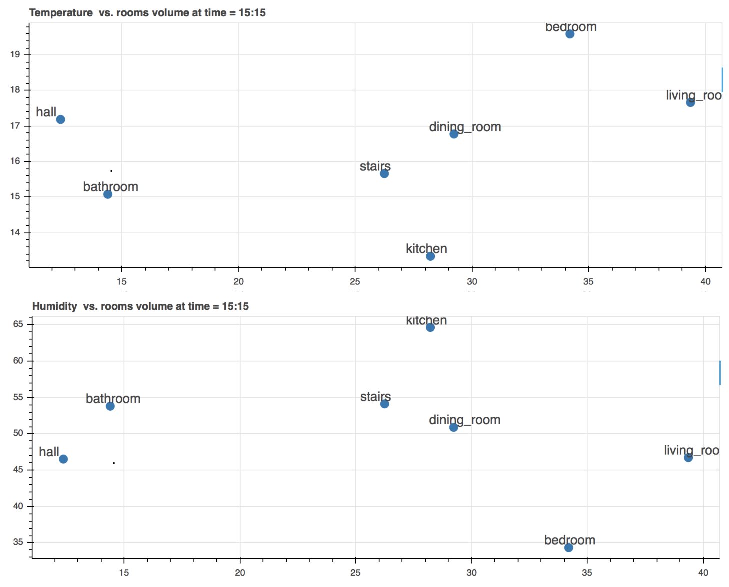


Figure 7: Temperature vs Volume (Top), Humidity vs Volume (Down) at 15:15 hours

Figure 7 is the comparison between temperature and volume in various room at 15:15 hours (before we opened the windows in the living room and master bedroom). Figure 7 is the comparison between humidity and volume as well. Windows were left open for about 90 minutes. Afterwards, the temperature and humidity in those rooms were observed to undergo changes.

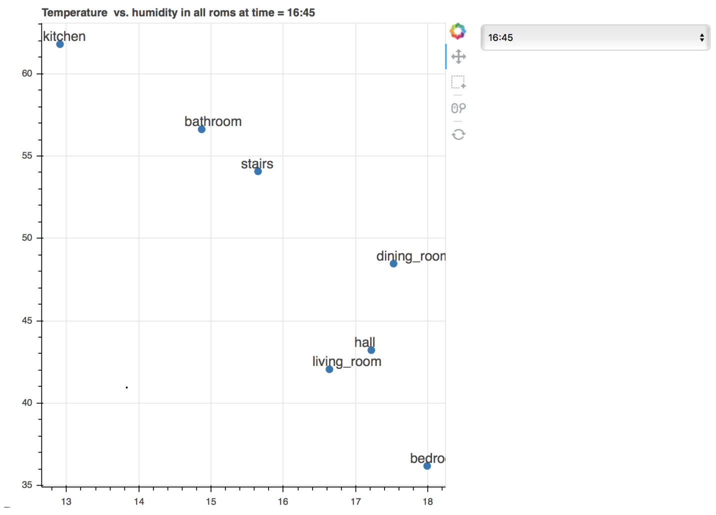


Figure 8: Temperature vs Humidity of all rooms at 16:45 hours

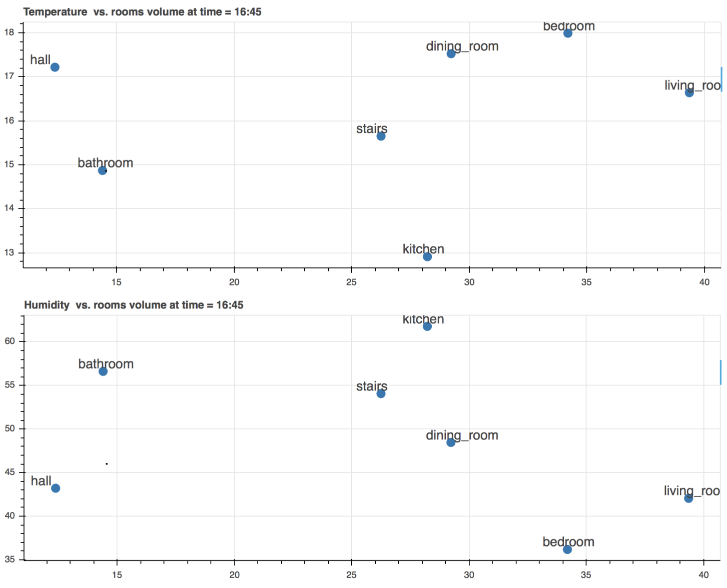


Figure 9: Temperature vs Volume (Top), Humidity vs Volume (Down) at 16:45 hours

Comparing the last two points(bedroom with volume 34.2 and living room with volume 39.368 m^3) in Figure 8 and Figure 9, we can observe that the temperature in the bedroom decreased from 19.6 °C to 18 °C and the humidity increased from 34 % to 36.2 %. The temperature in the living room decreased from 17.6 °C to 16.6 °C and the humidity decreased from 47% to 42%. The temperature in the bedroom decreased faster than that in the living room, this can be reasoned by the fact that the volume of the bedroom is smaller than the living room. However, the humidity change in both rooms is opposite.

One possible reason might be that the bedroom which is in the first floor is more exposed to sunlight than the living room which is in the ground floor.

Figure 10 are the plot results of the Temperature vs Time and Humidity vs Time in the kitchen:

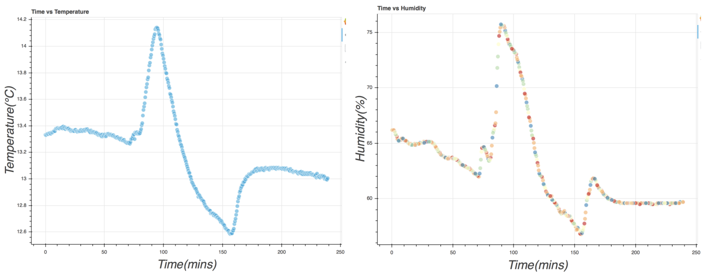


Figure 10: Temperature vs Time (left panel), Humidity vs Time (right panel) in Kitchen.

During the experiment, we made tea in the kitchen around 15:10 hours (at 70 minutes in Figure 10) which led to both the temperature and humidity to increase significantly. Then we opened the windows in the kitchen at 15:40 hours (at 100 minutes in Figure 10). This is the main reason that both the temperature and humidity decrease afterwards.

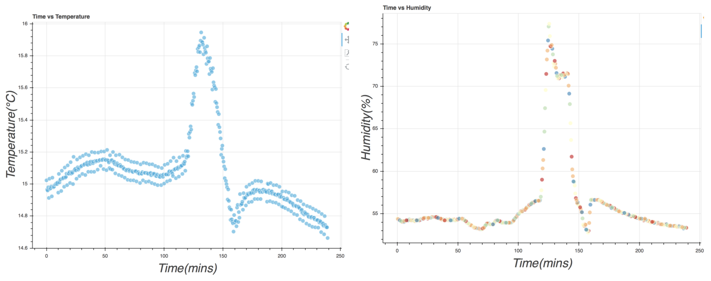


Figure 11: Temperature vs Time (left panel), Humidity vs Time (right panel) in Bathroom.

Similarly, we opened the hot water tap in the bathroom at 16:00 hours (at 120 minutes in Figure 11) to test the change that occurs in temperatures and humidity. We can infer from the figure that both the temperature and humidity have increased afterwards. Then, we opened the windows in the bathroom at 16:20 p.m (at 140 minutes in Figure 11), which was the reason behind both temperature and humidity decreasing afterwards.

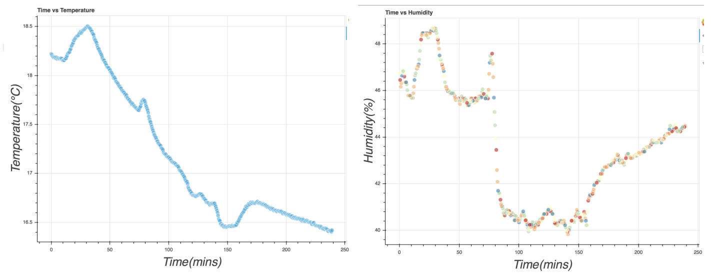


Figure 12: Temperature vs Time (left panel), Humidity vs Time (right panel) in the living room.

We opened the windows in the living room at 15:15 hours (at 75 minutes in Figure 12) and closed those at 16:45 hours (at 165 minutes in Figure 12). The Humidity vs Time graph experienced a decrease and an increase at corresponding times. Temperatures vs Time graph experienced a decrease after the windows were opened and an increase a bit after the windows were closed. Then, it went down again due to the external temperature decreasing as it was approaching night time and there was not enough sunlight in the living room.

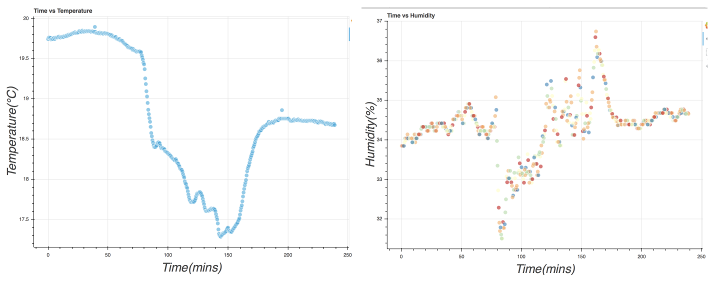


Figure 13: Temperature vs Time (left panel), Humidity vs Time (right panel) in the bedroom.

Figure 13 clearly detects the activities we have performed in the bedroom. The temperature and humidity decreased when we opened windows at 15:15 hours (at 75 minutes in the Figure 12) and increased after we closed those windows at 16:45 hours (at 165 minutes in the Figure 13).

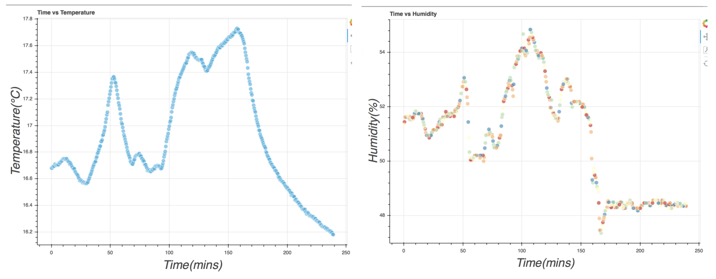
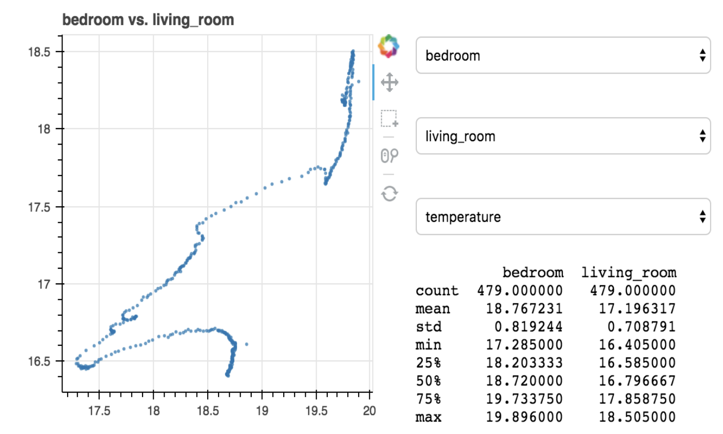


Figure 14: Temperature vs Time (left panel), Humidity vs Time (right panel) in the dining room.

Since all five of our group members and our supervisor stayed in the dining room, we could see from Figure 14 that the temperature and the humidity were spiking up and down throughout the duration of the experiment. Our own body temperature and the activities kept affecting the temperature and humidity of the dining room.

Figures 15 and 16 show the temperature comparison between the master bedroom and the living room, and the temperature vs time plots of both rooms. Due to the similar activities happening in these two rooms, their temperature vs time plots have similar patterns.



```
        bedroom  living_room
count  479.000000  479.000000
mean    18.767231   17.196317
std      0.819244    0.708791
min     17.285000   16.405000
25%     18.203333   16.585000
50%     18.720000   16.796667
75%     19.733750   17.858750
max     19.896000   18.505000
```

Figure 15: The temperature comparison between bedroom and living room.

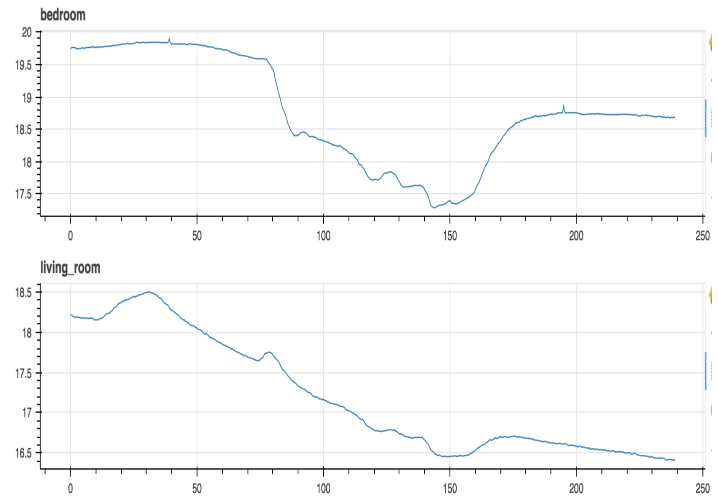
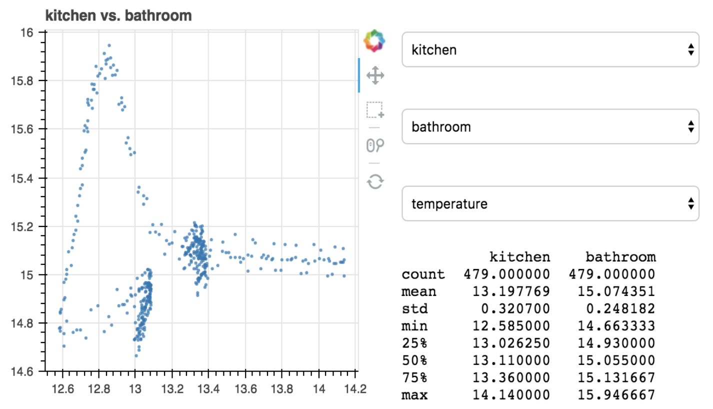


Figure 16: Temperature vs Time (top panel) in the bedroom, Temperature vs Time (down panel) in the living room.



```
        kitchen    bathroom
count  479.000000  479.000000
mean    13.197769   15.074351
std      0.320700    0.248182
min     12.585000   14.663333
25%     13.026250   14.930000
50%     13.110000   15.055000
75%     13.360000   15.131667
max     14.140000   15.946667
```

Figure 17: The temperature comparison between kitchen and bathroom.

Figure 17 is the temperature comparison between the kitchen and bathroom. Figure 18 presents the temperature vs time plots of the kitchen and bathroom. We made tea in the kitchen at 15:10 hours (at 70 minutes in Figure 18) and we turned on the hot water tap in the bathroom at 16:00 hours (at 120 minutes in Figure 18). It seems that both these activities had similar impact on the temperature which is indicated by the similar temperature change pattern.

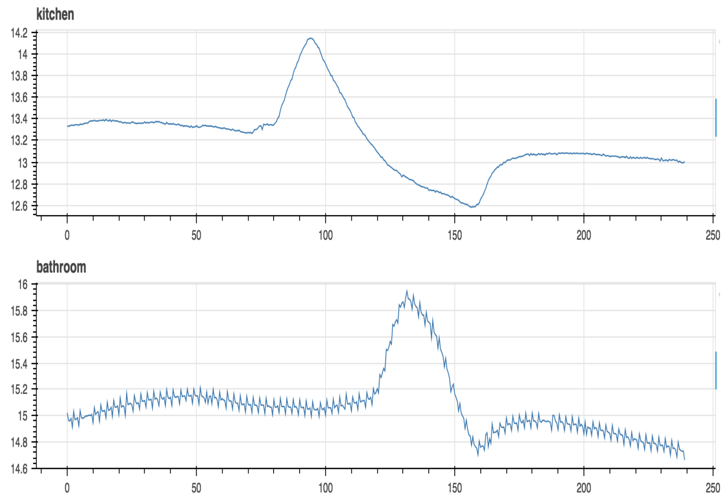


Figure 18: Temperature vs Time (top panel) in the kitchen, Temperature vs Time (down panel) in the bathroom.

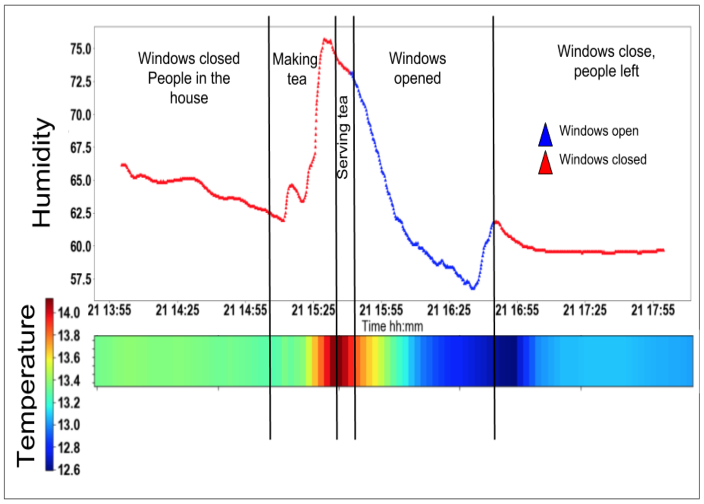


Figure 19: Kitchen: Making tea

Figures 19 and 20 show the relationship between the activities in the rooms and the measures of temperature and humidity. Figure 19 is segmented in five activities that happened in the kitchen. First, we were situated in the house moving about and occasionally in the kitchen. Second, there were four of us making tea. Third, the tea was being served. Forth, we left the kitchen and opened its windows. Finally, we closed the kitchen's windows. For all of these activities, there was a significant change in the temperature and the humidity. Especially, when the water was boiling in the kettle and the windows were open.

In the case of the bathroom, the main activities were taking a shower and leaving the room with the windows open. The figure 20 shows a clear increment of temperature and the percentage of humidity during the shower. Then, the it shows a sharp decrease in both measures when the window was open.

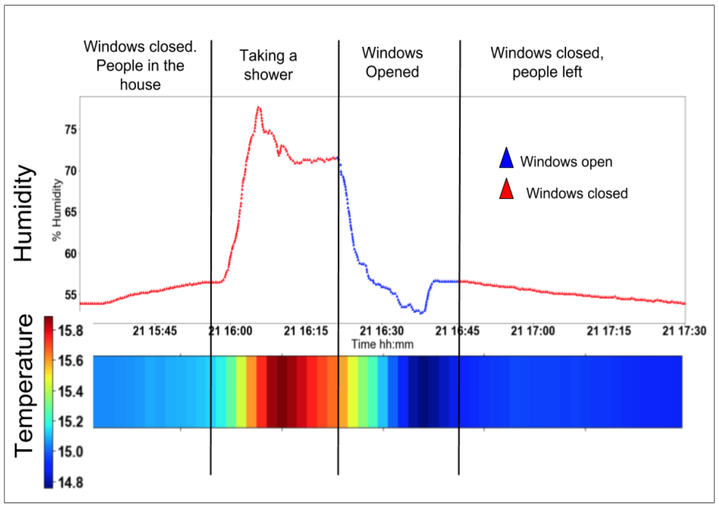


Figure 20: Bathroom: Taking a shower

7 Conclusion

In this project, we have successfully developed a production-scale data pipeline that can collect data from the environmental sensors deployed in smart homes and predict if a window is left open. All the components of the pipeline namely, data storage, data cleaning, data visualisation and prediction; have been built keeping in mind the scalability to perform well. To simulate the real-world scenarios, the data used here was collected with performing activities such as cooking and shower in the house. The data collected was put through the pipeline and the results obtained were good with the best model achieving a F1 score of 0.957.

In the case of deployment, the data collected over a longer range of time can be helpful in understanding the dynamics better and avoid overfitting of the model. It can also accommodate the outside temperature and account for routine activities of the people living in the house. This pipeline with certain modifications, can be employed in current setup of smart homes to help people maintain the air quality inside the house and live a healthier life. It can also be used to predict activities inside the house such as ironing clothes or drying hair. In the future, we would like to collect more data across weather patterns and seasons to better generalise the model. Also, we would like to utilise the data from other sensors such as the power draw sensor attached to heating or cooling systems as this would give us a new dimension to our data, and possibly a better predictor. It might also be possible to use the recording from the video camera to understand if the window is being attended and hence not raise an alarm. There are ample such opportunities of helping the residents of the smart house by utilising the data captured by various sensors installed in the house.

References

[1] *SPEHRE: Sensor Platform for Healthcare in a Residential Environment.* http://www.irc-sphere.ac.uk/about

[2] Lowen AC, Mubareka S, Steel J, Palese P, (2007) *Influenza Virus Transmission Is Dependent on Relative Humidity and Temperature*, PLoS Pathog 3(10): e151. https://doi.org/10.1371/journal.ppat.0030151.

[3] Garrett, Rayment, Hooper, Abramson and Hooper (1998) *Indoor airborne fungal spores, house dampness and associations with environmental factors and respiratory health in children.* Clinical & Experimental Allergy, 28: 459-467. https://doi.org/10.1046/j.1365-2222.1998.00255.x.

[4] *SPEHRE: Sensors.* http://www.irc-sphere.ac.uk/sphere-challenge/sensors

[5] Kunzel HM, Holm AZirkelbach D, Karagiozis AN (2005) *Simulation of indoor temperature and humidity conditions including hygrothermal interactions with the building envelope.* Solar Energy, 78(4): 554-561. https://doi.org/10.1016/j.solener.2004.03.002

[6] *Amazon SageMaker - Machine Learning Models & Algorithms.* https://aws.amazon.com/sagemaker/

[7] *scikit-learn: Machine Learning in Python.* http://scikit-learn.org/stable/