Detecting Anomalies in Smart Buildings using Machine Learning

Introduction

A smart building is a structure that can provide users with an efficient, comfortable and convenient humanized building environment. Unfortunately, due to these benefits they have also become the next focus area for hackers with malicious intent. In this group project we wrote a program based on unsupervised algorithms to detect abnormal data gathered by smart buildings, specifically the Urban Sciences Building in Newcastle University. This program simulates what one form of attack could achieve, changing key data gathered 24 hours a day by the sensors encapsulated in the USB.

In the group project, we decided to use Python as the programming language of choice. Python could be described as an explanatory programming language and provides many libraries which can be invoked to achieve a lot of different goals. In this sense, Python made the most sense as it had many libraries corresponding to machine learning, and was easy to code in.

Following the specification, the project should implement data collection, model training, outliers detecting and attack injecting. Initially, six months of historic data should be collected from the given API provided very kindly by the USB. This API allows us to gather sensor information fairly far back in history, analyse these values and manipulate them. This data was then separated into 80% training data and 20% testing data, which is used to test and detect the outliers. Following on from this, we then manipulated this data by changing specific values, simulating an attack on the building.

Cyber and Physical Security Attacks on Smart Buildings

The following is data related to the security levels of smart buildings, and frequency of attacks.

***See Fig. 1***

Compared with similar statistics on industrial systems, it is obvious that smart buildings are more susceptible to attacks.

The next graph shows the different types of malware which could infect smart buildings, which gives an interesting insight into how hackers carry out their malicious intent.

***See Fig. 2***

From this data, we can analyse that the threat to a Smart Building is more than one aspect. The methods that an attacker can pick vary in complexity and function. In addition, it is worth noting the importance of monitoring network communications on the perimeter and inside the network of automation systems, as this can lead to attempts of a breach of security.

DDoS attacks could target any kind of system in the building, and flood with traffic, causing said system to malfunction. In addition to this, key information can be stored within smart buildings which in turn could be worth a lot of money, which is when ransomware can be used to control the system until a fee has been paid. [2] [3] [4]

**Digitization of Building Management Systems Can Reduce Lifecycle Costs** [1]

The digitization of building automation systems covers a varied and complex application space. These applications include, but aren’t limited to:

* HVAC
* Energy management systems
* Lighting control systems
* Video surveillance systems
* Access control systems
* Elevator control systems
* Sensors and devices (Cameras, Thermostats, Light sensors)

Each system and device, including its multiple versions and iterations, has its own level of cybersecurity risk. With this, a smart building is bound to be more susceptible to attacks, as there is much more data encapsulated in one system, and this data can be valuable to people with any sort of malicious intent.

Digitizing these systems presents a huge opportunity to reduce energy and operational costs for building and facility owner-operators. For example, commercial buildings consume over 70 percent of the electricity produced in the U.S. Many buildings are older and incorporate dated legacy technology and could significantly benefit from retrofitting the building control infrastructure to help reduce total cost of ownership and enhance security and safety.

According to the U.S. Department of Energy, both commercial and residential buildings produce about 38 percent of the greenhouse gas emissions, representing a significant opportunity for the new generation of IoT-enabled systems to reduce the sector’s carbon footprint. New smart, digital technologies for building monitoring and control can help improve occupant comfort and provide information that can be used to operate the building as efficiently as the physical structure and equipment allows.

**Current Instances of Cyber Attacks** [5]

Today, the functionality of many buildings – such as the ability to control room temperature, door locks, and alarms – is controlled by a Building Automation System (BAS).

Shodan is an internet search tool which can be used to look at these. To get a sense of scale, right now, a criminal searching for a range of different BAS systems to target probably has close to 30,000 to choose from in the US alone. These are IP addresses which, when entered into a web browser, may produce BAS login screens.

The next stage for the attacker is to try the default user name and password for that type of system (these can be found with basic Google skills). Unless the defaults have been changed, and rate limiting on password attempts has been implemented, a motivated criminal will probably gain access. The immediate goal of this search-and-guess-password process is to access a cooling system dashboard. This is a live dashboard accessible to anyone on the internet without a password.

**How to Prevent Cyber and Physical Attacks** [6]

In general, the three motives of the attacker have not changed as technology has evolved, in general these include financial gain, disruption of service and theft of personally identifiable information or intellectual property and within the smart building environment, control systems can present easy targets.

Operational Technology (OT) environments are easily overlooked without strict network regulations, which are often vulnerable to attack and, in certain situations, can easily be a weak link in the organization. Through analysing the possible problems, we can implement targeted OT security controls, and help professionals have corresponding training, as well as help the entire environment to have a strong cyber resilience.

Data Collection Methods

Python provides the ‘requests’ module, which sends request to the provided API. After receiving the response from the API, we can then decode the received text of the API to a Python object so that the gathered data can be collected and utilised from the Python object. There are some different data labels- in the provided USB API, including entity, feed and timeseries. In order to work with the collected data, the data meanings need to be understood.

**Entity –** Simply put, an entity represents a single room in the USB’s API. This could be room 1.003, the toilets of floor 2, the kitchen on the ground floor etc.

**Feed –** Feed is a regularly measured property of an entity, such as Co2 or Room Temperature. There is also a data label called ‘metric’ in the API, which has the same meaning as feed.

**Timeseries –** In order to request data from the past, the API provides a ‘timeseries’ data value which represents the historic data feed of an entity. By default in the timeseries API, only 3 days of historic data were provided. In order to receive 6 months of data required by the specification, a time range should be added onto the URL of said API. This time range contains the start and end time in order for the historic data to be received during these values.

The 6 months we used to follow the specification were within 2019-09-01T14:26:25.804Z and 2020-03-01T14:26:45.804Z.

***See Fig. 3***

As the USB API provides an independent API for every entity, feed and timeseries, the data from each of them needs to be received from their own API. In an entity API, the feed of the entity in question can be found. Similarly, in a feed API, every timeseries API can be found for said feed. This means we need to provide the specific entity and names of feeds required.

There are multiple functions provided in the data collection code, including ‘get\_text’, ‘get\_TS’, ‘metric\_list’, ‘create\_DF’, ‘get\_feedUrl’, and ‘main’. Each function has different uses, including examples like:

* Get json text from a given API,
* Get the URL of the timeseries API from a given entity URL
* Collect the data list from the timeseries API
* Collect the data list from the timeseries API and create a DataFrame by the data list
* Get the URL of the feed API from given entity

Detailed explanation including description, parameters return has been written in the notes of the code.

Data Collection Code

In the project, the feed ‘CO2’ and ‘Room Temperature’ were the metrics we decided on. This is because they would most likely be key targets for an attacker and are also good solid metrics for analysis as the outlier values are easier to spot. The time range of the historic data is from 2019-09-01T14:26:25.804Z to 2020-03-01T14:26:45.804Z.

1. We first set the global object CO2\_DF and Tem\_DF to represent the DataFrame of CO2 and Room Temperature. We also set the timeRange string for adding it to the URL of the timeseries.
2. Next we set the URL of 3 entities including the room 1.003, room 2.002, room 4.018 and then gathered the text from them to find the required feeds and the timeseries of the feeds.

***See Fig. 4***

1. In order to find the URL of the timeseries of CO2 and Room Temperature we set a for loop to locate them and gather the URL. After that, we added the time range to the end of the URL to get 6 months historic data.

***See Fig. 5***

1. Two DataFrames were then created for use with preprocessing, and these were combined.

***See Fig. 6***

1. This was then exported to the specific place it needed to be, with the specific name.

***See Fig. 7***

Data Preprocessing

In order to make the following process easier, the data preprocessing should be completed before moving on. As the historic data of CO2 and Room Temperature is collected, we need to format it into: ‘Time, CO2 value, Room Temperature’. The ‘Time’ column represents the time of measurement. Unfortunately, we encountered a few issues when combining the time column, as the measurement time of each feed is completely different. In order to proceed, we ignored the minute and second section of the time value, only showing down to the nearest hour. We also set up an index column for each DataFrame to avoid mixing up the data. By completing this, the process became a lot simpler for us, and we just had to remove the duplicate row of the combined DataFrame, and then export it all as a .csv file.

***See Fig. 8 and Fig. 9***

Model training

K-means

As the requirement of the paper, the first model used in the project to detect the outliers is the K-means. The main idea of K-means is that we divide every data point into different clusters. Each cluster has centroid which is the centre of the cluster.

Determine K

The first step of K-means is to determine the value K. This can be described as the most suitable number of clusters in the situation.

Firstly, we set a range of k (2 – 10) and then calculated the deviation square of distance of each point in each cluster. Using the deviation square to determine k, we can find the maximum decline of the deviation square in the range. Thus, the k after the maximum decline will be the most suitable k.

***See Fig. 10***

Determine clusters of points and centroid of each cluster

After we have determined k, the model needed training to determine the clusters which contains every point in the training dataset and centroid of each cluster.

Firstly, select k points in the testing dataset randomly and set them to the centroid of k different clusters. Then, use the distance between the other points in the dataset and the centroids to determine which cluster the points belong to. After that, calculate the mean of distance between each point and the centroid of the cluster which it belongs to, using the means of the clusters to determine new centroid of each cluster.

A loop is then carried out, finding a new centroid until the loop times reach a number set before, or the old centroids are as same as the new centroids. Finally, use the last centroids to determine which points belong to its cluster by comparing the distance between each point and different centroids.

***See Fig. 11 and Fig. 12***

Box Plot

In the project, the method to detect outliers is to find some points which do not belong to any clusters in the situation. In order to determine which point doesn’t belong to any cluster, the box plot can be used in this situation. The main idea behind the box plot is to calculate five lines of value which are known as the upper edge, upper quartile, median, lower quartile and lower edge.

***See Fig. 13***

After we have calculated these five lines, the lower edge and upper edge can be used to detect outliers in the provided values. In the project, the value will be the distance between points which will be detected and the centroids of each clusters.

The method to carry this out is as follows: Firstly, calculate these five lines for each cluster after having divided the training dataset points to different clusters and then determine the centroids of each cluster. Following on from that, determine which centroid is the closest to the points and use the centroid to calculate the distance between the centroid and point. If the distance is bigger than the upper edge of the cluster or lower than the lower edge, then the point will be detected to be an outlier.

Isolation Forest

In an Isolated Forest, outliers are defined as "more likely to be separated" which can be understood as points that are sparsely distributed and distant from a dense population. In the feature space, sparsely distributed regions indicate that the probability of an event occurring in this region is very low, so the data falling in these regions can be considered abnormal. Isolated forest is an unsupervised anomaly detection method suitable for continuous numerical data. That is, it does not require labelled samples for training, but its features need to be continuous. To find points that are easily isolated, iForest uses a very efficient strategy. In an isolated forest, the data set is randomly and recursively segmented until all sample points are isolated. With this random segmentation strategy, outliers usually have shorter paths.

Isolation Tree

In an Isolation Forest, there are many Isolation Trees. The number of Isolation Trees are equal to the number of loops. Every Isolation Tree represents an independent separation of nodes. First, we must construct the isolation tree and set its maximum height and subsample size. Then we select a cutting point between the data maximum and minimum points randomly, to divide the current node data space into two subspaces.

***See Fig. 14***

Data smaller than the split point data will be placed on the left child of the current node, and data bigger than or equal to the split point data will be placed on the right child of the current node. Construct new child nodes recursively until no child nodes can be generated or the child node has reached max depth which settled before.

***See Fig. 15***

Path Length

In the model of isolated forest, the path length of the sample point ‘xi’ is the number of edges that the root node of one single isolation tree passes to the leaf node, which is the depth of the node. So, a prediction function has been settled to calculate the path length of each sample ‘xi’.

***See Fig. 16***

Anomaly Score

The isolated forest algorithm uses an anomaly score to determine whether each sample is abnormal. For example, given a dataset containing n samples, the average path length of the tree is:



H (i) is the harmonic number, which can be estimated as ln(i)+0.5772156649. When c (n) is a given number of samples n, the average path length is used to normalize the path length h (xi) of the sample xi. So, the formula of the anomaly score calculated before is:



E (h (xi)) is the expected path length of sample xi in a batch of isolated trees.

After calculating the anomaly score of every point, the outliers can be determined by comparing the scores. Following the rules of Isolation Forest, the bigger scores which are closer to 1 will be judged to be the outliers. However, the scale of judgment might different between different datasets so the threshold of scores cannot be determined. In order to determine the threshold of scores, the box plot can also be used in the situation. Using the box plot to calculate the upper limit will be suitable to determine the threshold. After that, the points whose points bigger than the upper limit will be judged to be the outliers.

***See Fig. 17 and Fig. 18***

Model Testing and Comparison

Attack Injection

As the separation of dataset was done randomly, the result of testing will be difficult to compare. Because of this, some points which are obviously outliers have been added to the testing dataset. We simulated the attack by modifying the data of each dataset, such as adding some abnormal data, modify some correct data to be abnormal, etc.

The added points:

***See Fig. 19***

Testing

After adding the points to the testing dataset, the testing result is shown below (Room 1.003):

***See Fig. 20 and Fig. 21***

***See Fig. 22 and Fig. 23***

It is clear to see that the point with coordinate [300,24] will have been judged to be the outlier in K-means and have not been judged to be the outlier in Isolation Forest. This demonstrates that K-means is more sensitive than Isolation Forest in the project situation.

The reasons for this will be the following:

* The number of Isolation trees in Isolation Forest is not suitable for the project
* The isolation tree separates nodes randomly
* The dataset separates randomly and the testing dataset of them are not the same

After testing on room 1.003, we then test a dataset which contains much more data. The testing result of room 2.002 is shown below.

***See Fig. 24 and Fig. 25***

It is clear to see that Isolation Forest is much more sensitive now, as it has judged many more points to be the outliers than K-means. However, the method of judgement is not the same. K-means uses the distance and Isolation Forest uses the score of points. Because of this, we can reason that K-means will be better when the dimension of data is small (2-3) and number of data is large (At the same time, is the dimension of data is big, the distance will be hard to calculate), and Isolation Forest works better when the dimension of data is big (more than 3) and the number of data is relatively small.

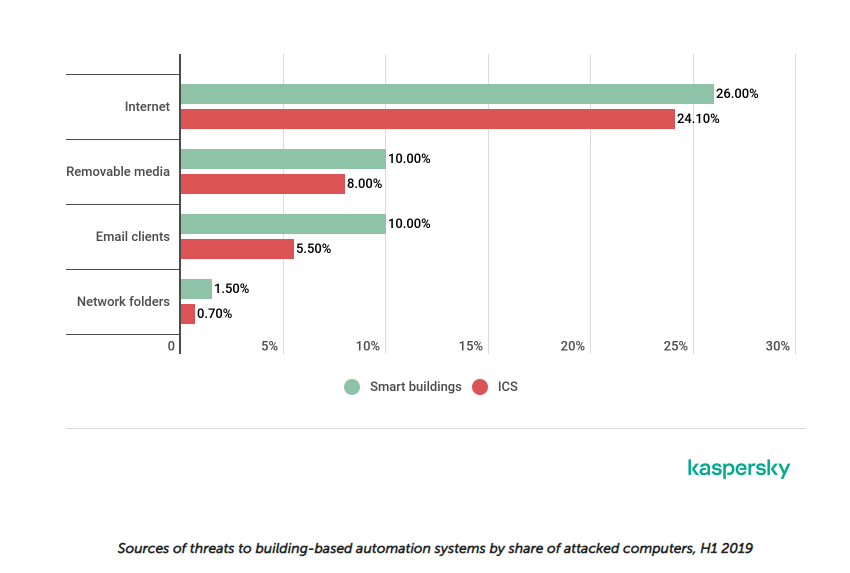
***See Fig. 26 and Fig. 27***

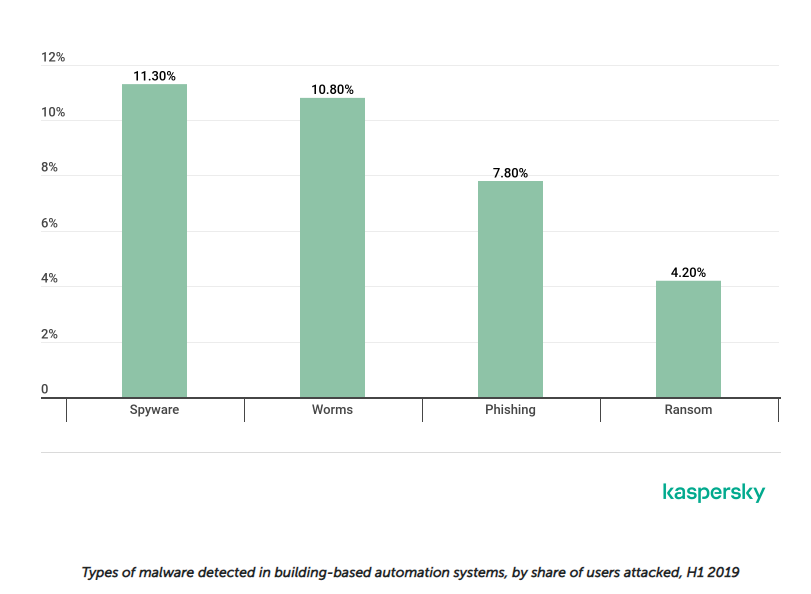
The data from room 4.018 is not easy to separate into different clusters. In K-means, the points are separated into 2 clusters. However, the results don’t seem accurate as the outliers seem to be not judged currently. Comparing this to the Isolation Forest, the result of Isolation Forest is much better than K-means in this situation.

So, we can reason that when the data are relatively dispersed and not easy to separate to be different clusters, the Isolation Forest works better than K-means.

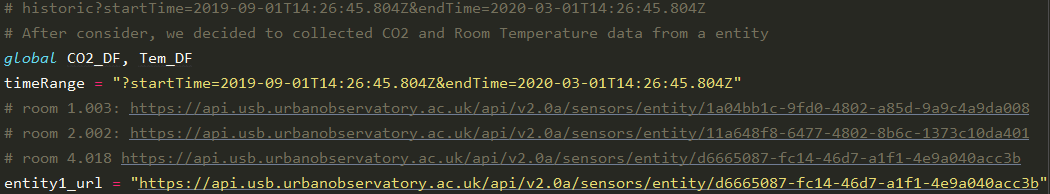
Appendix







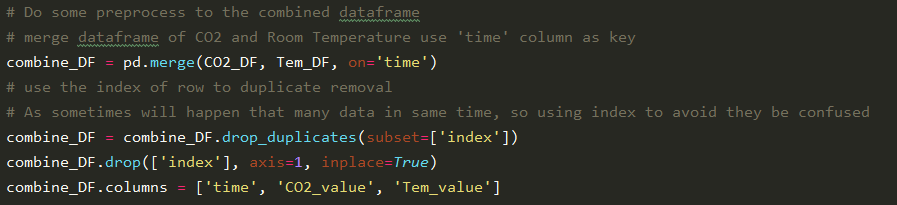
1. 



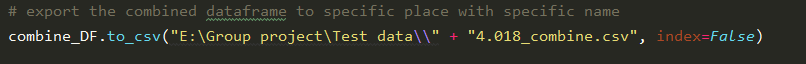




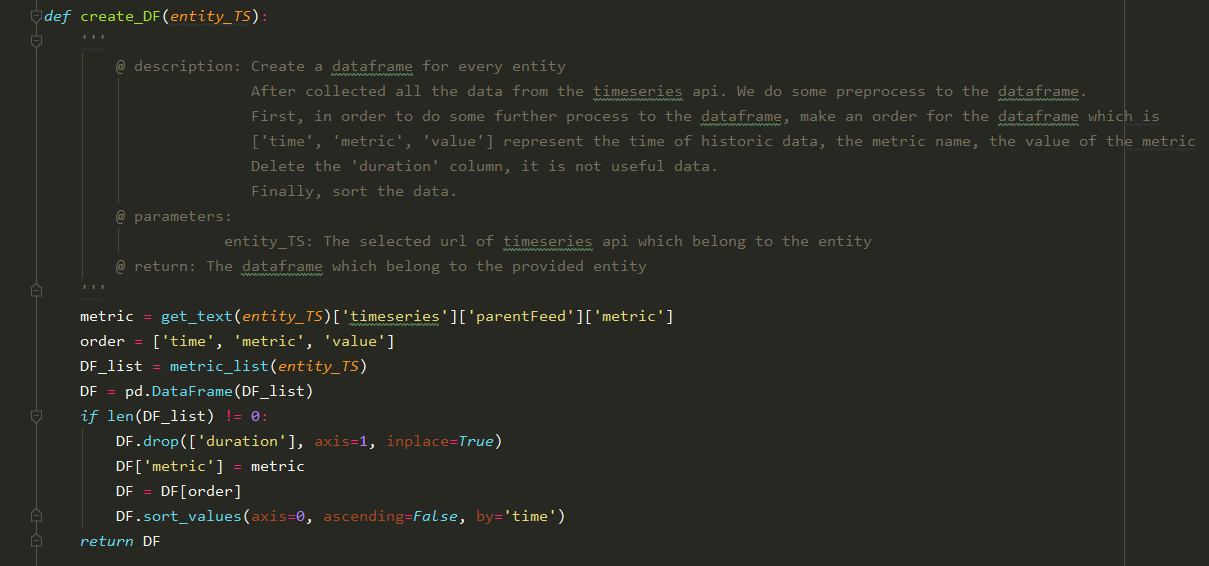




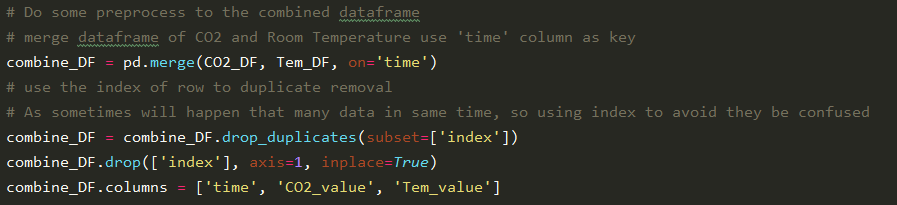




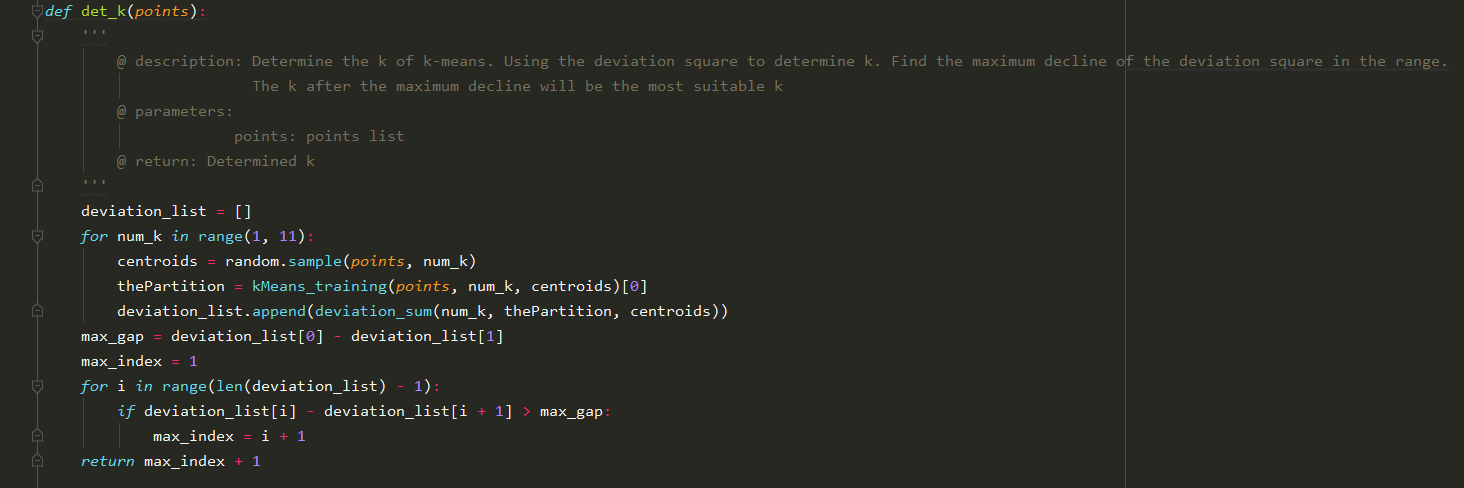




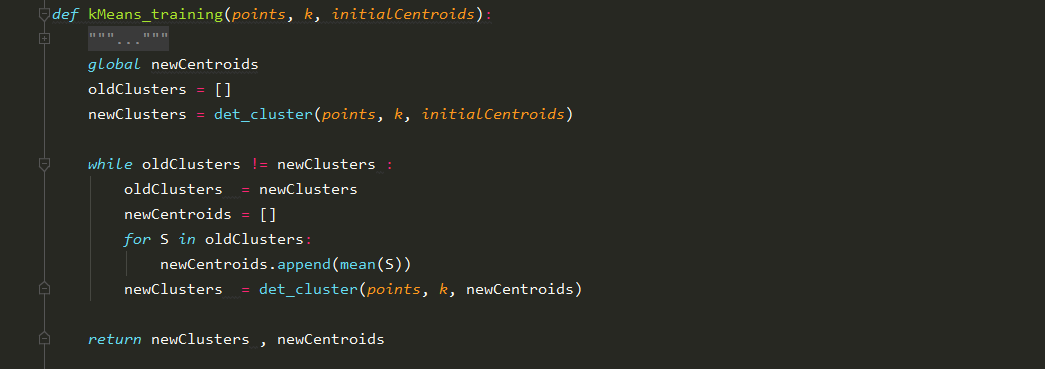




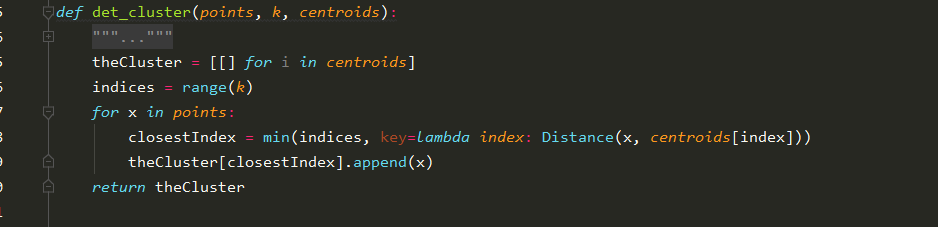




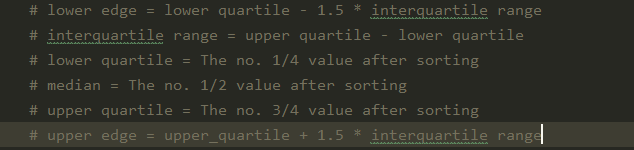




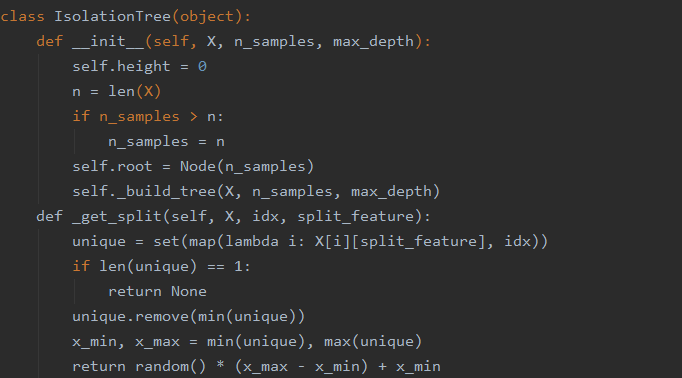




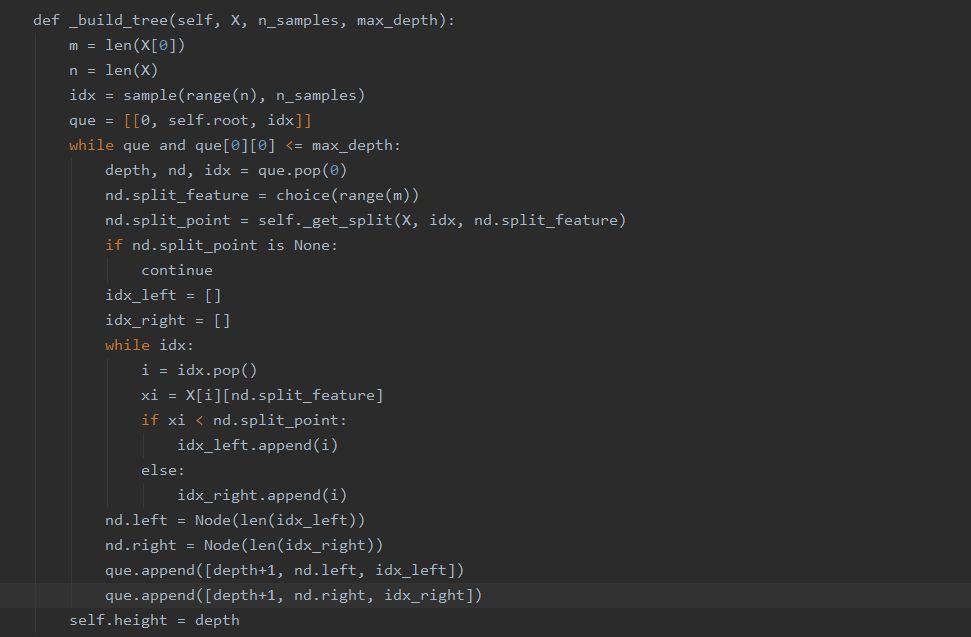




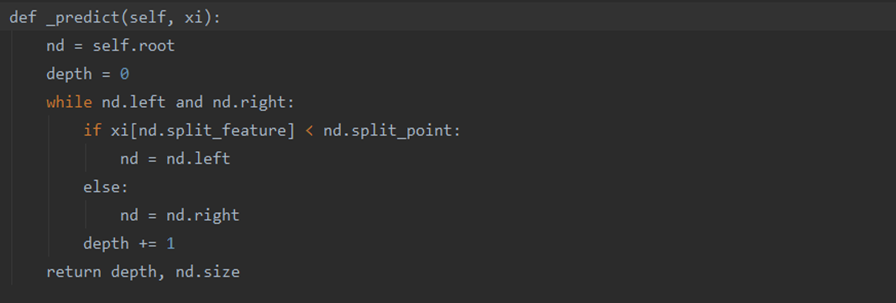




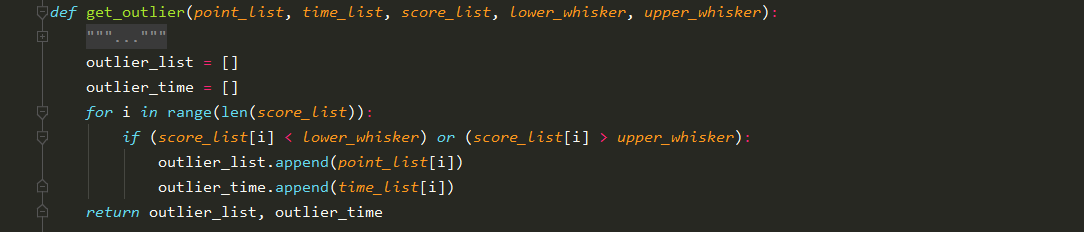




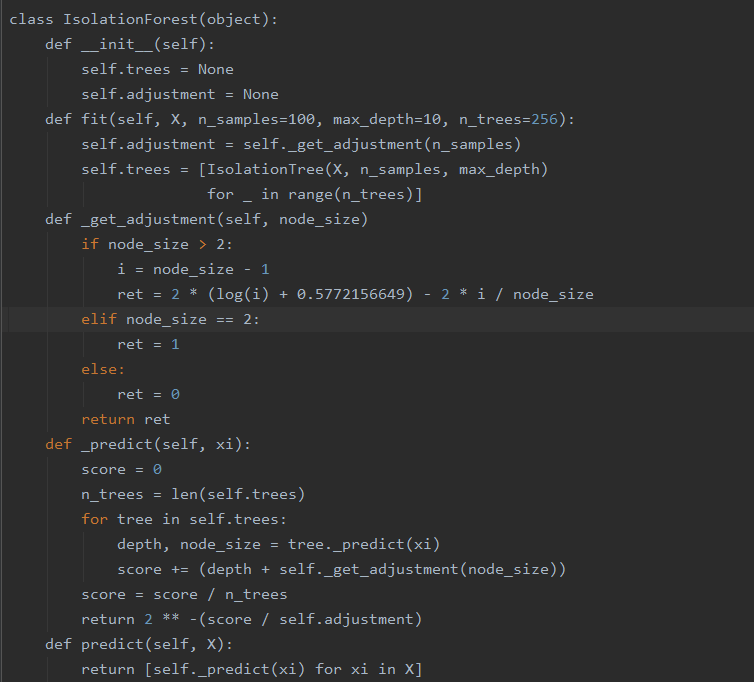




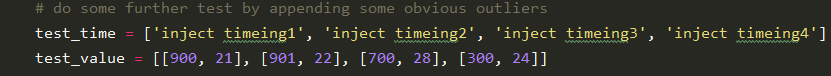




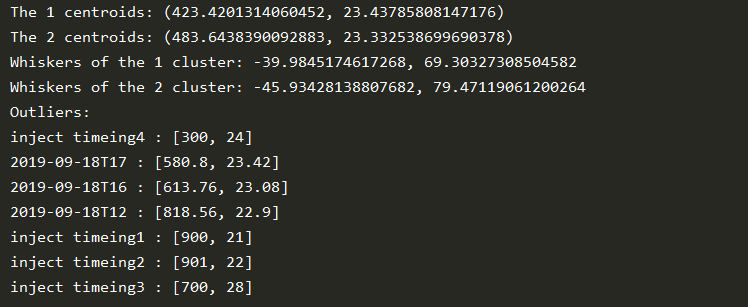






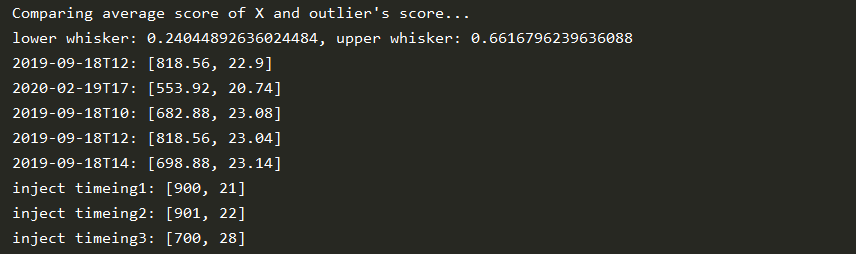






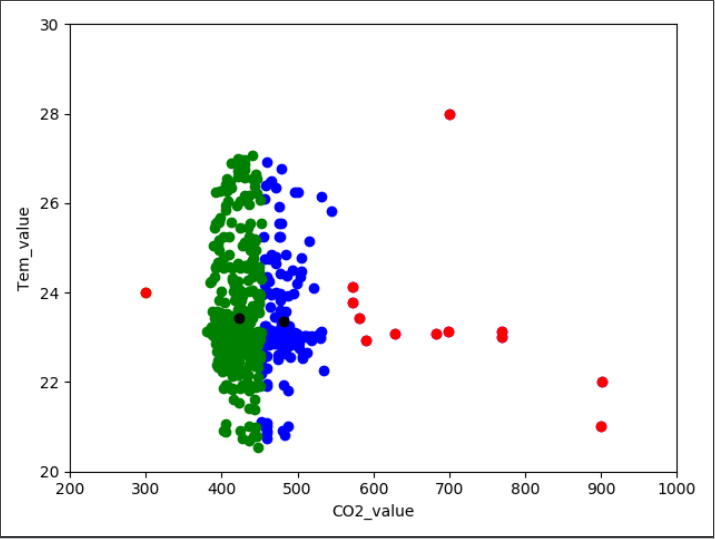
(K-Means)





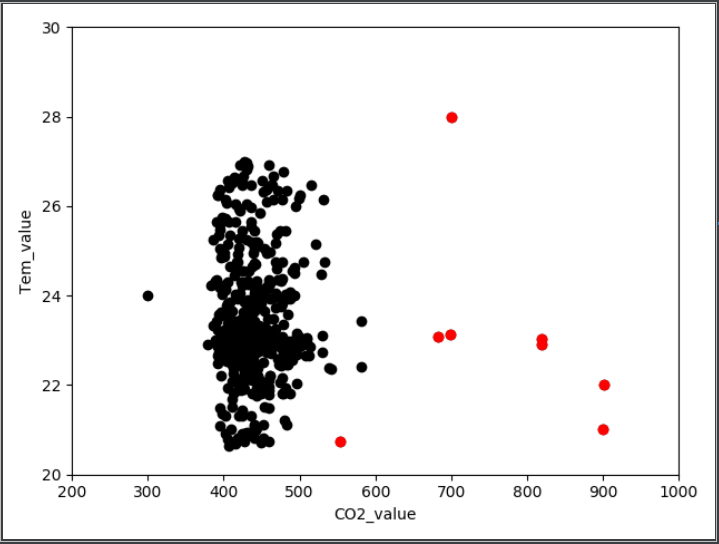
(Isolation Forest)



****

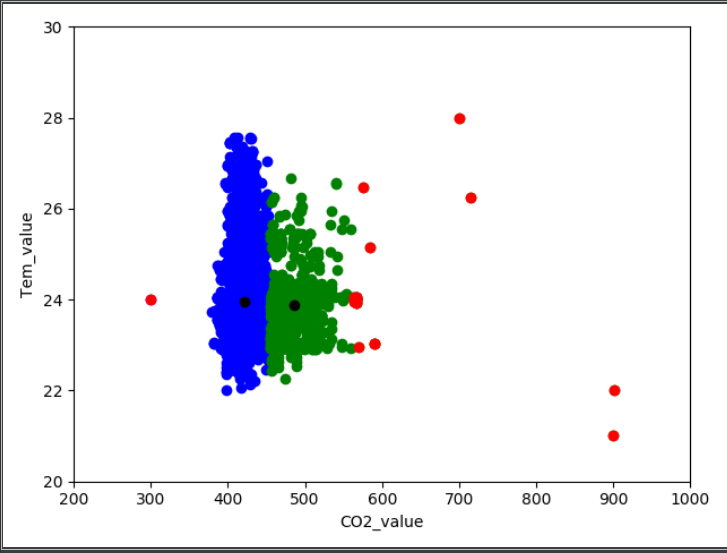
(K-Means)



****

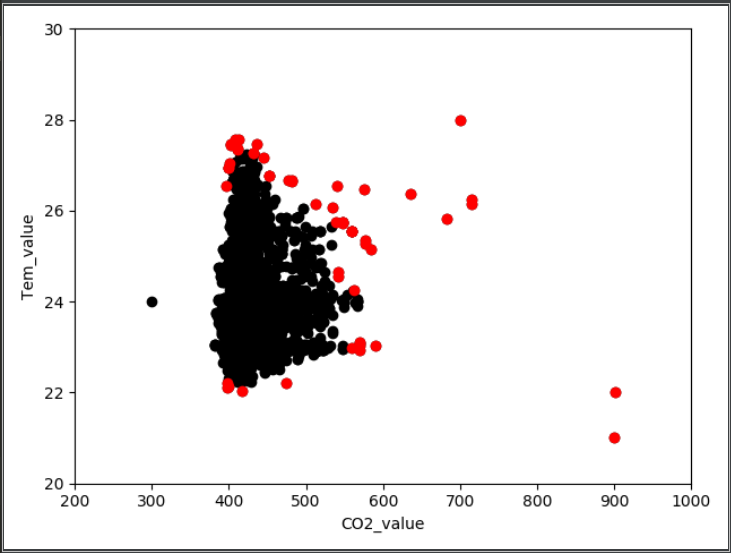
(Isolation Forest)





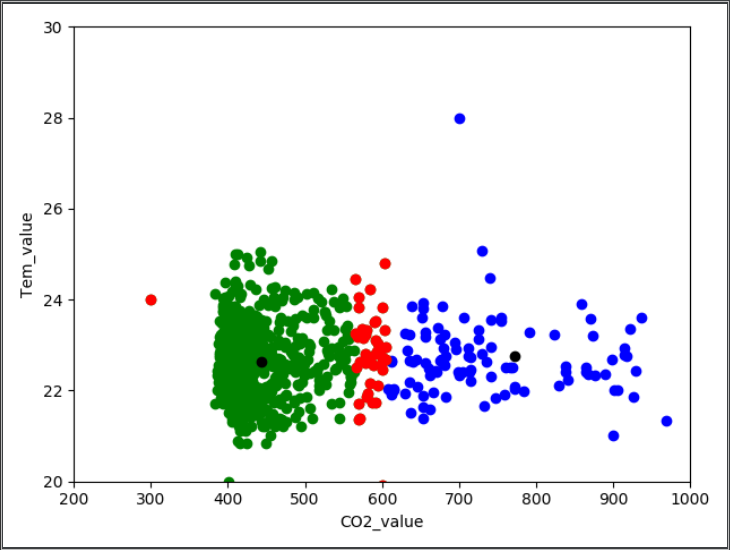
(K-Means)





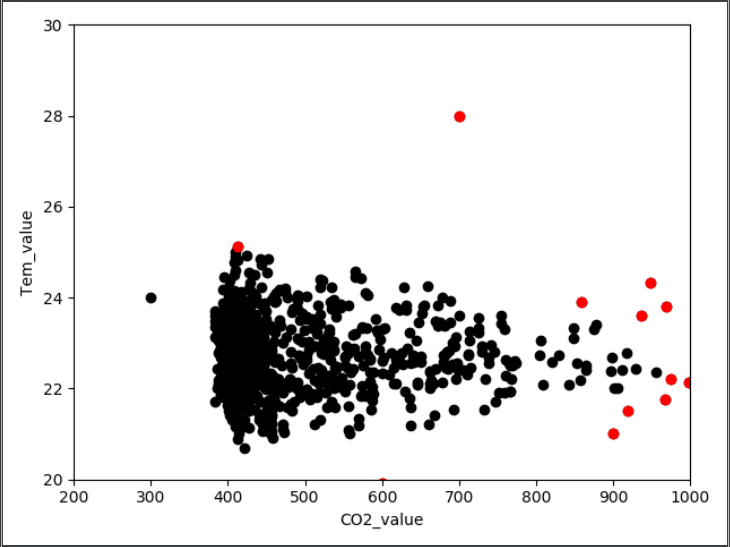
(Isolation Forest)





(K-Means)





(Isolation Forest)

References

1. <https://securelist.com/smart-buildings-threats/93322/>
2. Costante, E. (4/12/2017) *Smart Buildings: Trends and Challenges for a secure way.* Available at:<https://www.forescout.com/company/blog/smart-buildings-trends-and-challenges-for-a-secure-future/> (Accessed: 27/03/20)
3. Gavin, L. (2/8/2019) *The hidden risks of smart buildings.*  Available at: <https://www.ajg.com/uk/news-and-insights/2019/october/the-hidden-risks-of-smart-buildings/> (accessed: 27/03/20)
4. Kruglov, K. (19/6/2019) *Threat Landscape for smart buildings.* Available at: <https://securelist.com/smart-buildings-threats/93322/> (Accessed: 27/03/20)
5. https://www.welivesecurity.com/2019/02/20/siegeware-when-criminals-take-over-your-smart-building/
6. https://facilityexecutive.com/2019/06/operational-technology-cyber-smart-strategies-smart-buildings/