Anomaly Detection Using IoT Sensors

Chandana Dasari

Submitted in Partial Fulfilment of the Requirements for the Degree of
Masters of Science in Computer Science (Data Analytics)

College of Engineering & Informatics
National University of Ireland, Galway
August 2019

Supervisor: Professor Mathieu d'Aquin

Declaration

I hereby certify that this material, which I now submit for assessment on the program of study leading to the award of Masters in Science is entirely my own work, that I have exercised with reasonable care to ensure the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Acknowledgement

First and foremost, I would like to thank my supervisor Professor Mathieu d'Aquin for all of his support, guidance, and knowledge throughout this project.

I would also like to thank my fellow Masters students for all their camaraderie throughout the year. You made work seem a lot less like work. Finally, I would like to thank all my friends and family, and in particular my husband Krishna Sandeep, for their constant support and encouragement in undertaking and completing this Masters.

Abstract

This project investigates an application of machine learning in detecting anomalies on real time sensor data. In this work, I focused on identifying the unusual behavior in the Insight Centre for Data Analytics building (Galway) based on the multiple environmental sensors data. The main idea is identifying "Anomalous data points which do not fit well with the rest of the data". There are wide spread techniques for anomaly detection. This work mainly focuses on statistical methods. The idea is to learn a generative model fitting the given multiple sensors data, and then identify the data points in low probability regions of the model as anomalies. The data is a mixture of various environmental variables such as temperature, humidity, pressure, CO2, noise, and WiFi sensor to assess the number of people in the building. As data is real time and huge, there is a significant need for automatic monitoring and analysis with better visualization. This work aims to develop a prototype for a tool which can be used to monitor the sensors data and anomalous behaviour in the data. My algorithm for identifying anomalies is based on the error distribution between the actual data and resulting model data. The threshold for detecting anomalies varies for each sensor data, and hence a confidence score is considered to reduce the false positives in the detection. The final prototype shows the actual sensor data, predicted sensor data and anomalies identified so far and it's confidence information in a web user interface.

Keywords : Anomaly Detection , Time Series data, Data Distribution, Error distribution, Outliers, Machine Learning, Real-time data

Contents

List of Figures

Chapter 1

Introduction

The work carried out in this project involves the task of classifying the sensor data points as usual and unusual by the machine learning model. The data points are marked as unusual if they don't fit well with the other data points by our model.

1.1 Background

Sensors are a significant part of safety and security monitoring Systems. As the number of sensors in the buildings is increasing day by day, there is a need to detect any rare, suspicious events automatically, also called as anomalies. These anomalies differ significantly from standard data and are rare, unique circumstances. Both statistical techniques and machine learning techniques prove to be efficient in detecting anomalies. In analytical approaches, statistical properties of the distribution such as mean, median, mode, quantiles, and moving averages are used to detect anomalies [4]. Machine learning techniques such as clustering, classification, and regression are used to identify anomalies [9].

There are many challenges faced in anomaly detection. The first challenge being the quality of the data. If the data used in anomaly detection is noisy or missing, which may be similar to abnormal behaviour, then the model will not perform well. Along with this, if there are any seasonal changes in the data, the complexity of the model will increase. The next challenge comes in defining the anomaly. A lot of domain knowledge is required to establish an anomaly which varies according to the use case. All these factors should be considered in designing an anomaly detection model. And finally, the model created should be robust to the false positives.

1.2 Problem Statement

This project aims to model data from a set of sensors located at the Insight Center for Data Analytics and detect anomalies. As part of this experiment, the research questions I addressed are

- Can we model this as a machine learning problem?
- How confident can we be in the anomalies detected?
- How to deal with real-time anomaly detection ?
- How to deal with missing data?

1.3 Methodology

The overall work in the project can be divided into two parts, data pre-processing, and modelling.

In data pre-processing, I have carried out the below activities :

- Getting the raw data from a sensor.
- Joined all the individual sensors data to a data frame based on the "time" field.
- Unified all the entries based on the "time" field (By grouping the observations for every 15 minutes)
- Delete the missing values if any (which doesn't contribute much to the modelling)

In modelling, below activities are involved:

- As most of the features are very correlated. With the help of Linear regression, we derived the value of one feature using the rest.

- From the Error distribution of each feature. we experimented with the threshold value to classify the anomalies
- Assigned anomaly score to each data point to mitigate the False alarms.

1.4 Research Aims

The research aims of this work as follows:

- Demonstrating this problem as Machine Learning problem
- Explore the use of statistical approaches in identifying anomalies in time series data.
- Successful application and validation of our modelling technique.
- Exploring other effective techniques in the area of anomaly detection.

1.5 Structure of Thesis

This thesis divided into five chapters. Chapter 1 provides background information and an overview of the problem domain. It also contains an outline of the research aims addressed in this work. Chapter 2 reviews the relevant literature in the fields of anomaly detection. Which covers wide rage of techniques in the supervised, semi-supervsed and unsupervised modes. Chapter 3 describes further data and preprocessing techniques. Chapters 4 establish the principal methodology and result in sections of the thesis. Finally, Chapter 5 provides a final review of the results in the context of the stated research aims and also discusses possible future work extending from this project's research.

Chapter 2

Literature Review

This chapter covers existing literature regarding the core topics of this thesis.

2.1 Introduction

Anomaly detection is a significant problem that reads in diverse research areas and application domains. Several anomaly detection techniques have been specifically developed for specific application domains, where as others are a lot more generic. Anomaly detection has studied in a large variety of records, including high-dimensional records, stream records, network records, time-series records and spatial records,[7]. Based on the type of records, the modelling techniques conjointly vary. Historically, anomaly detection has been dispensed manually with the help of visual information [15]. However manual methods are inadequate for real-time detection of streaming data. Therefore, In this chapter we emphasis on all the existing techniques in the field of anomaly detection.

2.1.1 What are Anomalies?

Consonant with [8], we describe an anomaly as a point in time where the behaviour of the system is unusual and significantly differ from previous normal functioning. An anomaly may signify an adverse change in the system, like a sudden increase in the noise levels around midnight, possibly indicating an anomaly. An anomaly can also be positive, like an abnormally high number of selling/buying orders of a XYZ company, implying stronger than regular demand. Anomalies can be pigeonholed by type as follows:

- **Point Anomalies :** "A single instance of data considered as anomalous concerning to rest of the data" referred to Point anomalies[7].

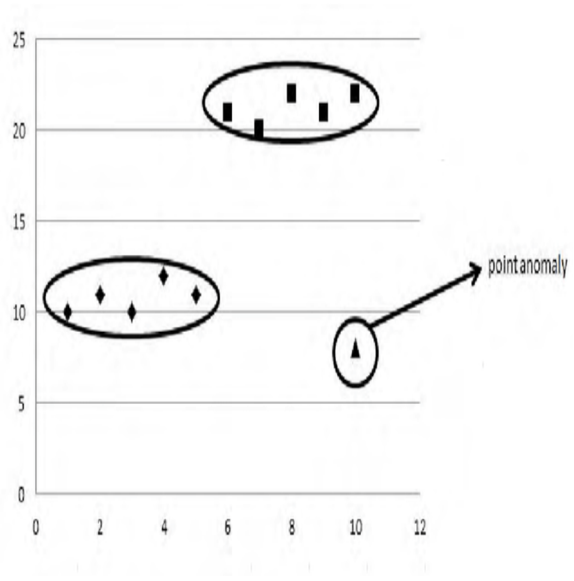


Figure 2.1: Example for Point Anomaly

- **Contextual Anomalies :** Most typical in time-series data, a contextual anomaly is "context specific"[7].

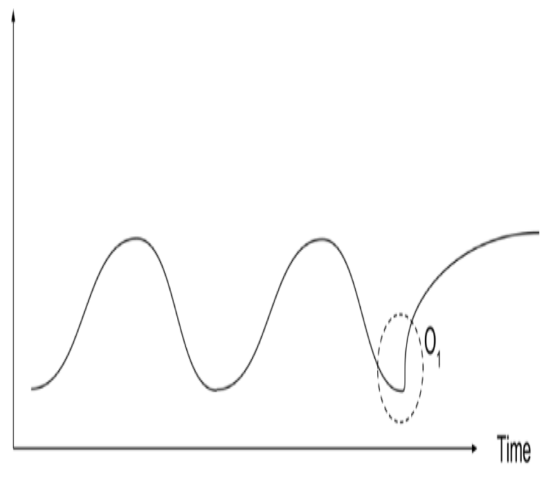


Figure 2.2: Example for Contextual Anomaly

- **Collective Anomalies :** A group of data instances collectively assisting in detecting anomalies[7].

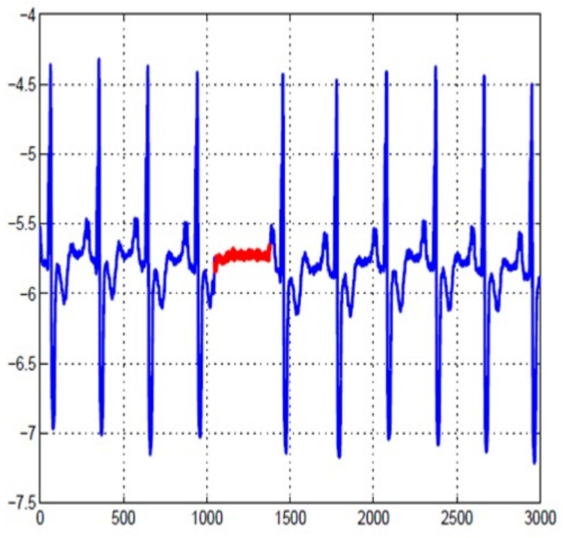


Figure 2.3: Example for Collective Anomaly

2.1.2 Anomaly Detection

Anomaly detection is about finding data points that deviate from the rest of the data. Anomaly detection can be practised for a variety of different problems, such as intrusion detection system, fraud detection, detecting a safety-critical system which is running abnormally before any significant infliction is done and many more.

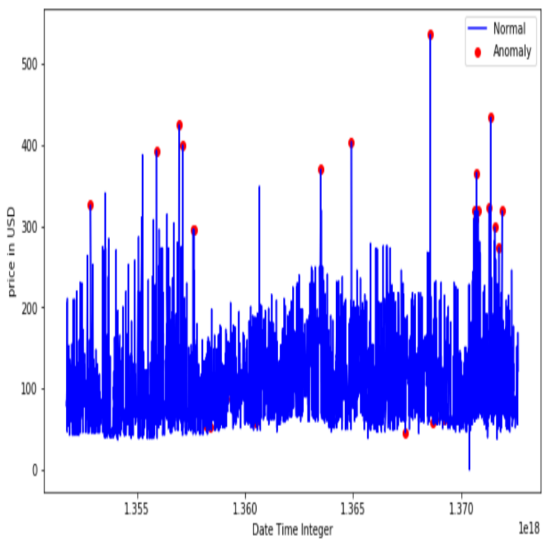


Figure 2.4: Example for Anomaly detection

2.4 shows a typical example of anomaly detection. In this figure, the data has some pattern. Most of the data spikes are in between 100 - 200 dollars and whereas the spikes above 300 dollars are sporadic. But not all the points above 300 are anomalies, since it depends on other correlated factors as well. In this case, the type of the room, the type of the hotel, etc.

2.2 Anomaly Detection Methods

There are many anomaly detection methods covered in the literature which are used in practice. Firstly the anomaly detection methods differ according to whether the sample of data for analysis given with labels that can be used to build an anomaly detection model. Secondly, methods divided into groups according to their assumptions regarding conventional versus anomalous data samples. Based on the availability of the labels to the data instances. The methods used can be divided into Supervised, Semi-Supervised and Unsupervised methods.

2.2.1 Supervised Methods

Techniques trained in supervised mode assume the availability of a training data set which has labelled instances for standard as well as anomaly class. The conventional approach in such cases is to build a predictive model for standard vs anomaly classes. Any unseen data instance is compared against the model to determine the category it belongs to.

The challenges involved in supervised method are as follows:

- Providing labels is very expensive.
- Imbalanced class distribution in the data. That is, the population of anomalies is generally much smaller than that of normal objects. Techniques such as oversampling cab be used handle the unbalanced classes.
- Catch as many anomalies as possible, i.e, recall is more important than accuracy.
- Obtaining accurate labels, especially for the anomaly class, is usually challenging. Several techniques proposed that inject artificial anomalies in a normal data set to obtain a labelled training data set[23].

Other than the above challenges, the supervised anomaly detection problem is similar to building predictive models.

2.2.2 Unsupervised methods

In most of the applications, labelled data is not available. They are thus leading the necessity to use Unsupervised techniques. The methods in this category make the implicit assumption that normal instances are far more frequent than anomalies in the data. and the normal objects are generally "clustered." In other words, an unsupervised anomaly detection method expects that regular objects follow a pattern far more frequently than anomalies.

- Normal objects may not share any strong patterns, but the collective anomalies may share high similarity in a small area.
- In case when normal data instances are diverse and do not fall into high-quality clusters, unsupervised methods may have a high false-positive rate and may let many actual anomalies be undetected.

The latest unsupervised methods developed smart ideas to tackle anomalies directly without explicitly detecting clusters.

2.2.3 Semi-Supervised Methods

Techniques that operate in a semi-supervised model assume that the training data has labelled instances for only the normal class. Since they do not require labels for the anomaly class, they are more widely applicable than supervised techniques. For example, in spacecraft fault detection [12], an anomaly scenario would signify an accident, which is not easy to model. The typical approach used in such techniques is to build a model for the class corresponding to normal behaviour, and use the model to identify anomalies in the test data. A limited set of anomaly detection techniques exist that assume the availability of only the anomaly instances for training. Such methods are not commonly used, Because it is difficult to obtain training data which covers every possible anomalous behaviour that can occur in the data.

In many applications, although obtaining some labelled examples is feasible, the number of such labelled cases is often small. If some labelled natural objects are available: Use the labelled examples and the proximate unlabeled objects to train a model for normal objects Those not fitting the model of normal objects detected as anomalies if only some labelled anomalies are available, a small number of labelled anomalies may not cover the possible anomalies well.

2.3 Statistical Approaches

Statistical methods assume that normal data follow some mathematical model (a stochastic model). Statistical anomaly detection techniques are on the following key assumption - "Normal data instances occur in high probability regions of a stochastic model, while anomalies occur in the low probability regions of the stochastic model". Statistical techniques fit a mathematical model to the given data and then apply a statistical inference test to determine if an unseen instance belongs to this model or not. Examples that have a low probability of being generated from the learnt model, based on the applied test statistic, are declared as anomalies. Both parametric, as well as non-parametric techniques, have been used to fit a statistical model. While parametric methods assume the knowledge of underlying distribution and estimate the parameters from the given data, non-parametric methods do not generally consider the knowledge of underlying distribution. Lets briefly look into the both methods.

2.3.1 Parametric Methods

A parametric model captures all its information about the data within its parameters. With this parameters we can predict the future data. For example, in case of a linear regression with one variable, we have two parameters (the coefficient and the intercept). Knowing these two parameters will enable us to predict a new value. Let us assumes that the data generated with parameter θ. The probability density function of the parametric distribution $f(x, \theta)$. The smaller this value, the more likely x is an anomaly. Normal data occurs in the region of high probability and anomalous data lies in low probability region of stochastic model. The effectiveness of statistical methods highly depends on the assumption for distribution[21].

$$y = \beta_0 + \beta_1 x_i + \epsilon \tag{2.1}$$

In the above parametric model, you know how the regression follows a linear line.

2.3.2 Non-Parametric Methods

On the other hand, a non parametric model makes no assumption on the population distribution to generate a model. It allows more information to pass from the current set of data that is attached to the model at the current state, to be able to predict any future data. The parameters are usually said to be infinite in dimensions and so can express the characteristics in the data much better than parametric models. It has more degrees of freedom and is more flexible. A Gaussian mixture model for example has more flexibility to express the data in form of multiple gaussian distributions. Having observed more data will help you make an even better prediction about the future data[21].

$$y = f(x_i) + e_i \tag{2.2}$$

where $f(.)$ scan be any function. This depends on the underlying data.

2.3.3 Gaussian Model

Gaussian model techniques assume that the data is from a Gaussian distribution. The parameters are estimated to be using Maximum Likelihood Estimates (MLE). The distance of a data instance to the estimated mean is the anomaly score for that instance. The threshold is applied to the anomaly scores to determine the anomalies. Different techniques in this category calculate the distance to the mean and the limit in different ways. A simple outlier detection technique, often used in the process of quality control domain [20], is to declare all data instances that are more than 3σ distance away from the distribution mean μ, where σ is the standard deviation for the data distribution. The $\mu \pm 3\sigma$ region contains 99.7 percent of the data instances. More sophisticated statistical tests have also been used to detect anomalies, such as Grubb's test which is used to detect anomalies in a uni-variate data set [13] and [22] under the assumption that the data is generated by a Gaussian distribution.

For each test instance x, its z score is computed as follows:

$$z = \frac{|x - \mu|}{s} \tag{2.3}$$

where μ and s are the mean and standard deviation of the data sample, respectively.

A test instance is declared to be anomalous if

$$z > \frac{N-1}{\sqrt{N}} \sqrt{\frac{(t^2)}{N - 2 + t^2}} \tag{2.4}$$

where N is the sample size and t - t-statistic.

The below are the mathematical expressions to calculate mean and standard deviation:

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} (x_j^i) \tag{2.5}$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} (x_j^i - \mu)^2 \tag{2.6}$$

2.3.4 Regression Based Model

Anomaly detection using regression has been extensively investigated for time-series data [1]. The fundamental regression-based anomaly detection includes two steps. In the first step, a regression model is fitted on the data. In the second step, for each test instance, the residual for the test instance is used to determine the anomaly score. The residual is the part of the instance which is not explained by the regression model. The magnitude of the residual can be used as the anomaly score for the test instance, though statistical tests have been proposed to determine anomalies with certain confidence [24]. Presence of anomalies in the training data can influence the regression parameters and hence the regression model might not produce accurate results. A conventional procedure to wield such anomalies while fitting regression models is named as robust regression. This techniques detect the anomalies based on the large residual. As the anomalies tend to have larger residuals from the robust fit. A similar robust anomaly detection approach has been applied in Autoregressive Integrated Moving Average (ARIMA) models [16].

2.3.5 Multivariate Linear Regression

For identifying anomalies, when dealing with 1 or 2 attributes data visualization is a good starting point. However when scaling this upto high-dimensional data, this approach becomes increasingly difficult. This is where multivariate statistics comes to help [17]. When dealing with a set of data points, they will typically have a particular distribution. To identify anomalies quantitatively, we compute the probability distribution $p(x)$ from the data points. Then when a new example, x, comes in, we compare $p(x)$ with a threshold τ. If $p(x) < \tau$, it is considered as an anomaly. The reason behind this is that standard samples tend to have a significant probability distribution, while anomalous samples tend to have a small probability distribution. In several applications, there is more than one factor that affects the output. Multiple regression models thus describe how a single response variable Y depends linearly on a number of predictor variables. A multiple linear regression model with k variables $X_1, X_2, ..., X_k$ and a single output variable Y , can be formulated as

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + ... + \beta_k x_k + \epsilon \tag{2.7}$$

As before, the ϵ are the residual terms of the model and the distribution assumption we place on the residuals will allow us later to do inference on the remaining parameters in the model. $\beta_0, \beta_1, \beta_2, ..., \beta_k$ are the associated weights for each variable in this model. This method is used to find the probability of each example and based on some threshold value we decide whether to flag an anomaly or not.

In this work we are primarily focusing on applicability of statistical techniques to this problem.

2.4 Advantages of statistical techniques

- If the assumptions regarding the underlying data distribution hold true, statistical techniques provide a statistically justifiable solution for anomaly detection.

- The anomaly score provided statistically is associated with support information, which can be used for additional learning while labelling test samples.
- If the distribution estimation step is robust to anomalies in data, statistical techniques can operate in a unsupervised setting without any need for labeled training data.

2.5 Deep Learning Methods

Of course, the literature survey would not be complete without discussing deep learning techniques. Fast advancements in recent years in neural networks methodologies did not skip the anomaly detection field. There is a lot of continuing work in the industry in this area, similarly to what deep learning has done in other machine learning areas. The most recent works have shown promising results, mostly using LSTMs, Auto-Encoders, and Variational Auto-Encoder.

2.5.1 One Class SVM

Our goal is to find a large-scale, high-dimensional anomaly detection algorithm that is robust, i.e. generates an accurate model for data from a wide range of probability distributions. In addition, it is desirable that the algorithm be efficient in terms of time complexity, memory complexity and the required number of labelled records.

One-class Support Vector Machines (1SVMs) are a a popular technique for unsupervised anomaly detection. Generally, they intend to model the underlying distribution of data while being indifferent to noise or anomalies in the training records. A kernel method essentially maps the input space to a higher dimensional to create a precise separation between normal and anomalous data. When applied, in system a kernel-based method can model any non-linear behaviour of data. [10].

2.5.2 Deep Belief Nets(DBNs)

Deep Belief Nets (DBNs) has emerged as a multi-class classifier and dimensionality reduction technique. DBNs are multi-layer generative models. It learns a layer of features at a time from unlabelled data. The extracted features are input for training the next layer. This greedy learning can be followed by finetuning the weights to improve the performance of the network. DBNs have a deep architecture, composed of multiple layers of parameterized non-linear modules. DBNs learn higher-level features that yield good classification accuracy; they are parametric models whose training time scales linearly with the number of records; they can use unlabelled data to learn from complex and high-dimensional datasets.

[10] has proposed a novel unsupervised anomaly detection approach, which combines the advantages of deep-belief nets with one class SVMs. In this model, an unsupervised DBN is trained to extract features that are reasonably insensitive to irrelevant variations in the input, and a 1SVM is trained on the feature vectors produced by the DBN. This hybrid model yields significant performance over standalone systems, which extract insensitive features.

2.5.3 Autoencoder Networks

The auto encoder is similar to the statistical analysis. It is an artificial neural network used to learn the data encodings(representation) in an unsupervised manner, and also focuses on feature reduction. Autoencoder tries to generate from the reduced encoding representation close to the original data.

The simplest autoencoder is a feedforward. It is a non-recurrent neural network very similar to many single layer perceptrons which makes a multilayer perceptron. It consists of an input layer, and an output layer has the same amount of input layer to reconstruct its inputs. In the context of anomaly detection, the basic idea is to use the autoencoder network to compress the sensor readings to a lower-dimensional representation, which captures the correlations and interactions between other sensors.

The autoencoder network is trained on normal data to reconstruct the input. During the dimensionality reduction phase, the system learns the interactions between the other sensors and can restore them back. The main idea is that if one variable suddenly changes, and t significantly, this should affect other sensors, as well as there, are correlated with each other (e.g. changes in temperature, co_2, humidity, etc.).

As this happens, we will start to see an increased reconstruction loss in the network. We here use the probability distribution of the reconstruction error to identify if a data point is normal or anomalous.

2.6 Output of Anomaly Detection

An essential aspect of anomaly detection techniques is how the anomalies notified. Typically, the outputs delivered by anomaly detection methods are one of the following two types:

- **Anomaly scores** Scoring techniques assign scores to each data instance depending on the degree to which the instance is considered an anomaly. In other words, the score represents the likelihood of the time series value being anomalous.

- **Alerts** Alert assigns a label (normal or anomalous) to each data instance. Scores allow us to use a domain-specific threshold to select the most relevant anomalies. The Alert column contains a flag with a value of 0 or 1, where one means that an anomaly was detected.

2.7 Time Series Data

Time series analysis is diverse, depending on the application. Time series analysis deals with the records collected over time. Depending on the application, data may be collected minute wise, hourly, daily, weekly, monthly, yearly, and so on. We use the notation such as X_t and Y_t for $(t = 1, 2, 3, ..., T)$ to denote the time series of length T. The main objective of the time series is to unveil the probability law to understand underlying dynamics, forecast future events and control future events via intervention[11],[6]. To draw inferences from such time series data,it is necessary to setup a hypothetical probability model to represent the data. Then it is then possible to estimate parameters, check for goodness of fit to the data, and possibly to use the fitted model to enhance our understanding of the mechanism generating the series. Once a satisfactory model has been developed depending on the domain knowledge we need to understand the trend, seasonality and random terms[6]. it is important to recognize the presence of seasonal components and not to confuse them with anomalies. Plot the time series data and examine the main features of the graph, checking in particular whether there is a

- a trend
- a seasonal component
- any sharp changes in the behaviour
- any outlying observations

Linear Time Series models consist of autoregressive moving averages (ARMA). ARMA models are used to capture linear relationships and are useful for linear forecasting. The Autoregressive Integrated Moving Average Model (ARIMA) is a standard statistical model for time series forecasts and analysis which uses stationary ARMA as subclass.

2.7.1 AutoRegressive Integrated Moving Average (ARIMA)

ARIMA is a generalization of the simpler AutoRegressive Moving Average and with the new notion of integration. This acronym captures the key aspects of the model itself. Briefly, they are as follows:

- **Autoregression:** A prototype that observes the relationship between the data sample and some lagged data samples
- **Integration:** difference of raw data instances to make the time series stationary.
- **Moving Average:** A model that uses the dependency between observation and residual errors from a moving average model applied to lagged observations.

Each of these components is explicitly specified in the model as a parameter. Standard notation is used for ARIMA(P, d, Q) where the parameters are substituted with integer values to indicate the specific ARIMA model. The parameters of the ARIMA model are defined as follows: P - The number of lag observations included in the model is also called the lag order. d - The difference in the data points is also called the degree of difference. Q - The size of the window. [5] also proposed the Box-Jenkins method for time series analysis and forecasting. The approach starts with the assumption that the process that generated the time series can be approximated using an ARMA model if it is stationary or non-stationary. The Box Jenkins method is a stochastic model building, and it is an iterative approach that consists of the following three steps:

- Identification: Use the data and all related information to help select a subclass of the model that may best summarize the data.
- Estimation: Apply the data to train the parameters of the model.
- Diagnostic Checking: Evaluate the model on the data and check for improvements.

It is a repetitive process so that as new learning comes during diagnostics, loop back to step 1 and incorporate that into the model.

Chapter 3

Data Analysis

In this chapter, we will discuss about how we regrouped and formatted our data before applying machine learning methods to it. We will also discuss about the different features we added or removed from the dataset and get more insights about the data.

3.1 Sensor Networks

Sensors have become more and more popular, due to their potential to be used in various applications of many different fields such as monitoring systems to environmental forecasting. All applications that make use of IoT sensors, strongly rely on their accurate performance, however, is considerably challenging to ensure. These sensors in fact, are typically prone to malfunction. Additionally, for many tasks (e.g. monitoring, forecasting, etc.), sensors stationed under conceivably harsh weather condition, making their breakage even more likely. TThe high plausibility of unreliable readings or data exploitation during transmission, brings up the problem of ensuring quality of the data collected by sensors. Since sensors have to operate continuously and therefore generate very large volumes of data every day, the quality-control process has to be automated, highly scalable and fast enough to apply to the data stream. The most common approach to ensure the quality of sensors' data, consists of automatic discovery of inaccurate readings or abnormal behaviours of sensors.

3.2 The Nature of Input Data

A vital aspect of any anomaly detection procedure is the essence of the input data. Input is generally a collection of data instances. All input records can be described using a collection of attributes. The attributes can be of distinct types such as binary, categorical or continuous. Each data record consists of only one attribute or multiple attributes. In the event of multidimensional data occurrences, all attributes might be homogenous or heterogeneous. These kinds of attributes determine the applicability of anomaly detection techniques. In our case the data instances belong to the sequence data. The instances are linearly ordered and belong to different data types.

3.3 Pre-Processing

3.3.1 Gathering Data

The first step to consider is the retrieval of sensor data. For this we use 24 sensors in the insight building. 8 sensors on each floor. Sensor generates the data for every 15 minutes. Therefore we receive 4 data instances per hour from each sensor. Other than having the environmental sensors, to assess the number of people in the building we configured the wifi sensor which give the number of devices logged into the wifi. The next step is to send this sensor data to the platform where we can handle the data efficiently (json format).

3.3.2 Cleaning the Data

The sensors we configured measure the following variables : For each floor (ground, first and second floors)

- *Time*(created while inserting the data on the server)
- $C02$ level in ppm
- *Humidity* level in percentage
- *Temperature* level in degrees celcius

- *Noise* in db
- *Pressure* in bar
- *people count*

Each sensor data is generated for unique time and its generated for every 15 minutes. However, all the sensors may not generate data at the same time. Thus there is a need for unifying the time for each data instances.

- Aggregate the sensor data on 15 minute basis (00,15,30,45) and converted the date to unique id format to differentiate each instance.
- Typically sensors are prone to go down(connection failure), and we have many missing values in data set. We therefore dropped the missing values as they don't contribute to the modelling.
- All the 24 sensors data are aggregated into a single dataframe based on the unique time identifier.
- Remove the columns which are redundant.

Before finding useful correlations and extracting knowledge, we need to retrieve the data that will be used to train the machine learning model to predict the anomalies.

3.3.3 Time Series Data

The data we deal with in this work is of type sequence. Time plays an important role. It is a sequence of observations taken sequentially in time. It is a series of observations obtained sequentially over time. Predictions will be given for new data when the actual outcome is not known until some future date. The future prediction is based on treating all prior observations equally. Time series adds an explicit order dependence between observations: a time dimension. This extra dimension is both a compulsion and a composition that provides a source of additional knowledge.

3.3.4 Data Visualization

- As the data is huge, and its cumbersome to read, understand and get insights from. As a first step, we observed the data distribution graph with respect to each sensor. These graphs are based on the Sensor variable Vs Time.

- Visualization allows us to build a monitoring system, in which we can easily observe the sensor failures, and other unusual behaviours.

- In the final prototype, We used the High Charts library which can make graphs directly and dynamically for a given variable using a GUI.

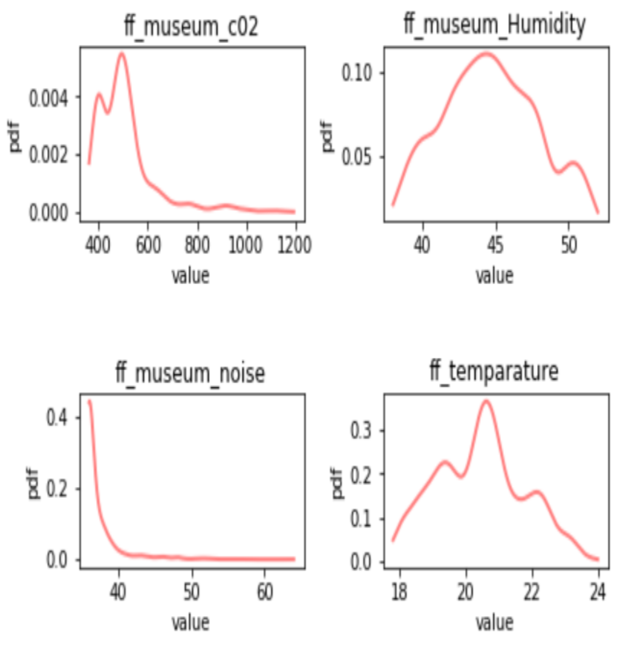


Figure 3.1: Sensor data distribution - 1

As shown in figure 3.1 plots describe how the sensor data is distributed for first floor CO_2, first floor humidity, first floor noise and first floor temperature. With the help of this graph, we can notice, for each sensor, the high probable values and low probable values.

As shown in figure 3.2 plots describe how the sensor data is distributed for first floor pressure, second floor humidity, second floor CO_2 and first floor temperature. With the help of this graph, we can notice, for each sensor, the high probable values and low probable values.

As shown in figure 3.3 plots describe how the sensor data is distributed for first floor humidity, second floor noise, second floor pressure and second floor temperature. With the help of this graph, we can notice, for each sensor, the high probable values and low probable values.

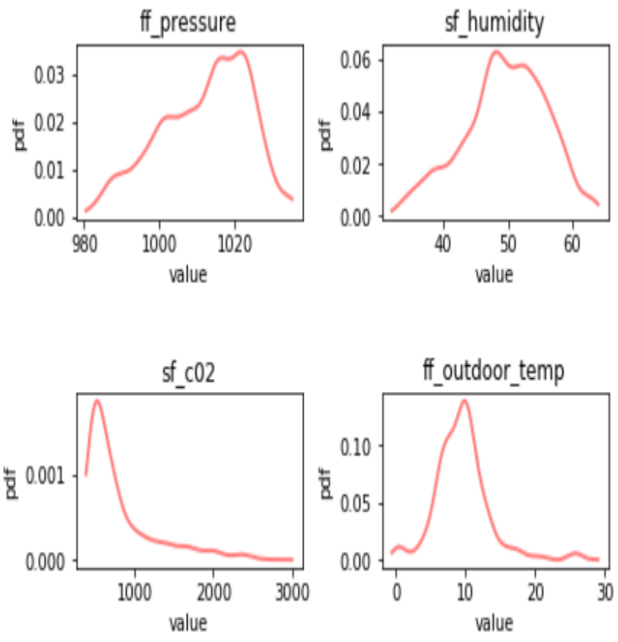


Figure 3.2: Sensor data distribution - 2

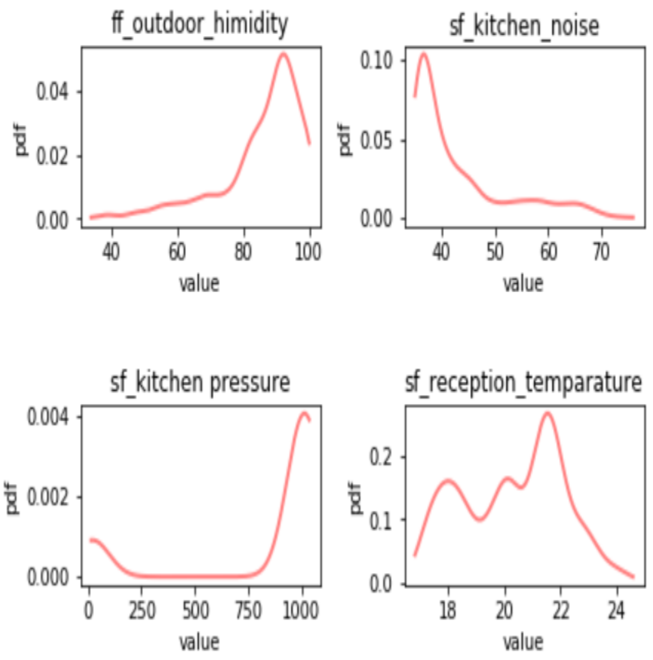


Figure 3.3: Sensor data distribution - 3

As shown in figure 3.4 plots describe how the sensor data is distributed for second floor humidity, second floor temperature, ground floor temperature and ground floor c02. With the help of this graph, we can notice, for each sensor, the high probable values and low probable values.

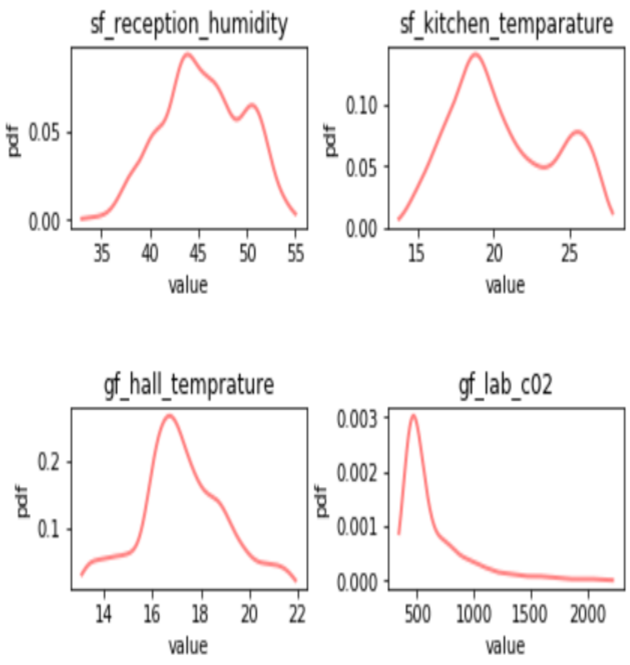


Figure 3.4: Sensor data distribution - 4

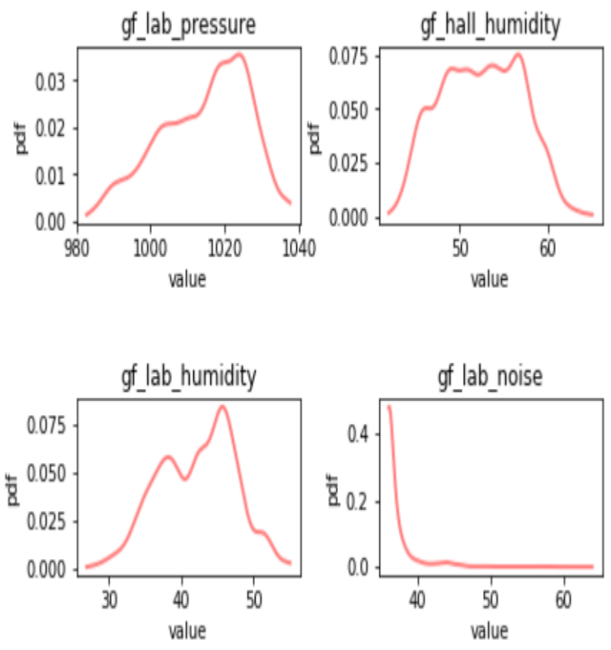


Figure 3.5: Sensor data distribution - 5

As shown in figure 3.5 plots describe how the sensor data is distributed for ground floor pressure, ground floor humidity , ground floor lab humidity and ground floor noise. With the help of this graph, we can notice, for each sensor, the high probable values and low probable values.

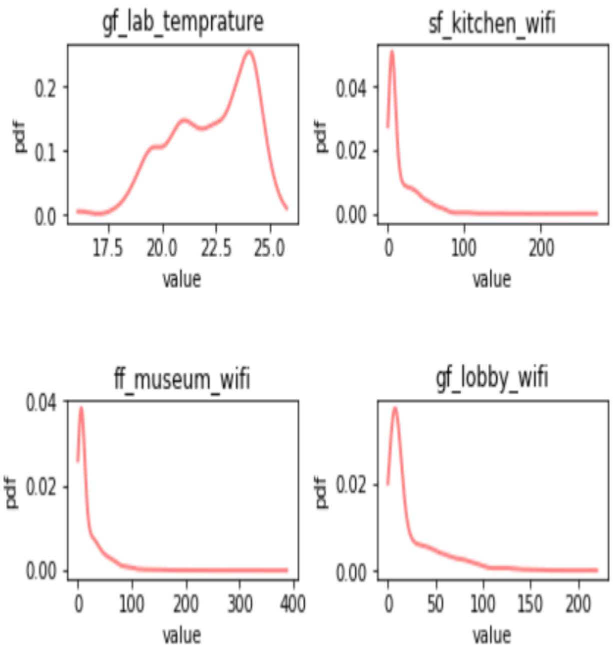


Figure 3.6: Sensor data distribution - 5

As shown in figure 3.6 plots describe how the sensor data is distributed for ground floor temperature, second floor wifi , first floor wifi and ground floor wifi. With the help of this graph, we can notice, for each sensor, the high probable values and low probable values.

3.4 Issues in the Time Series Data

- Time lags.
- Correlation over time.
- Forecasting models built on regression methods: autoregressive (AR) models autoregressive distributed lag (ADL) models need not (typically do not) have a causal interpretation.
- Conditions under which dynamic effects can be estimated, and how to estimate them.
- Calculation of standard errors when the errors are serially correlated
- It is very hard to get the desired data without noise.

Chapter 4

Data Modelling

Now it is time to manipulate our data and attempt to extract knowledge from it using supervised learning technique.

4.1 Early ML Experiments

In this section, we will detail our methodology, techniques, heuristics used to predict the anomaly in a building based on the CO2 level, pressure, Noise, humidity, temperature, people Count etc. We decided to use predefined ML models. To this end, we decided to use the sci-kit python library [19]. Because it is simple, reusable, built on top of other famous libraries and open-source, it is a popular choice for machine learning. It includes many different and well known machine learning techniques. However, if one is more interested in neural networks or similar techniques, then it's better to consider using other libraries, such as tensorflow or keras, since sci-kit does not provide many of those and does not support GPU acceleration or other optimizations.

Figure 4.1 explains the day wise readings of the CO_2., and we can observe that in the x-axis we have the time index, and in the y-axis we have the CO_2 readings of that day. . Trend : a general systematic linear or (most often) nonlinear component that changes over time and does not repeat seasonality, a general systematic linear or (most often) nonlinear component that changes over time and does repeat. Noise: a non-systematic component that is neither trend nor seasonality within the data

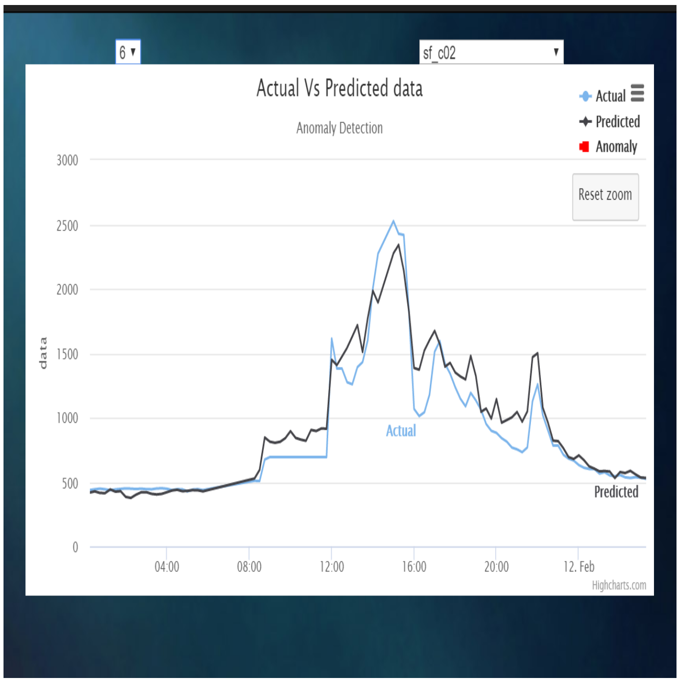


Figure 4.1: Day wise - Sensor data

4.2 Multivariate Linear Regression

In many applications, there is more than one factor that influences the response. Multiple regression models thus describe how a single response variable Y depends linearly on a number of predictor variables.

A multiple linear regression model with k predictor variables $X_1, X_2, ..., X_k$ and a response Y , can be written as

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + ... + \beta_k x_k + \epsilon$$

As before, the ϵ are the residual terms of the model and the distribution assumption we place on the residuals will allow us later to do inference on the remaining model parameters. Interpret the meaning of the regression coefficients $\beta_0, \beta_1, \beta_2, ..., \beta_k$ in this model.

4.2.1 Algorithm

Algorithm 1: Linear Regression model to predict the Sensor value based on other sensors

```
Fit the Linear Regression model to the each Sensor using other sensors
for sensor in 1 to 24(sensors) do
    Fit the Linear Regression model
    $y$ = Sensors[column]
    $X$ = Sensors.drop(column,axis=1)
    $lm$ = LinearRegression()
    model = $lm.fit(X, y)$
    predictions = lm.predict(X)
    error = abs(y - predictions)
    ErrorDist[column] = error
Plot the Error Distribution for Each Sensor.
```

Based on Model's error probability distribution for each sensor, the corresponding plots for each sensor is as follows

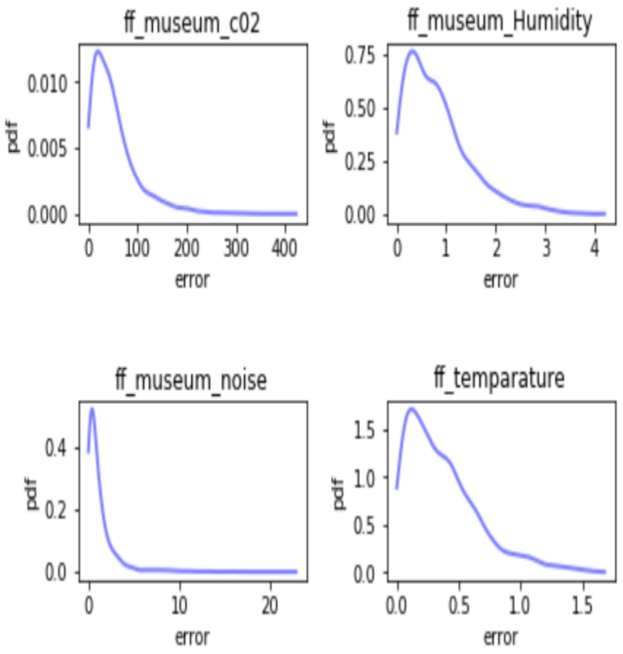


Figure 4.2: Error Distribution - 1

Observation

From the above sensor error distributions, we can clearly see that, usual sensor values lies with in the range of the curve, and the rare values lies towards the extreme

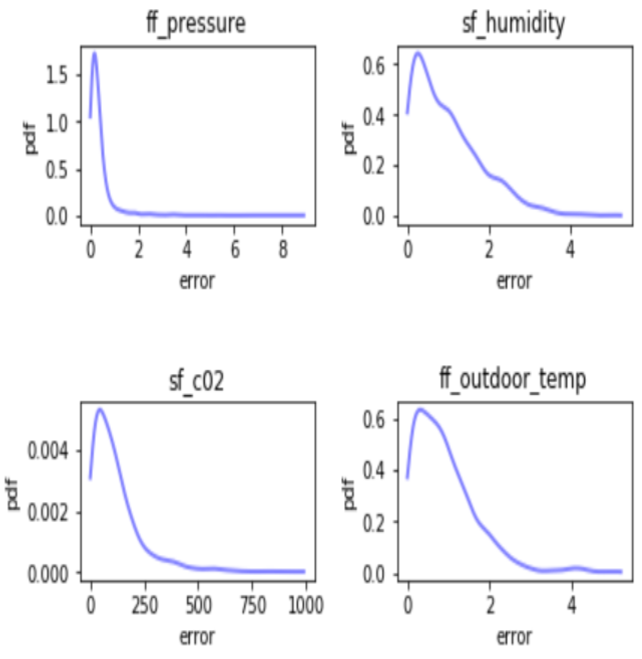


Figure 4.3: Error Distribution - 2

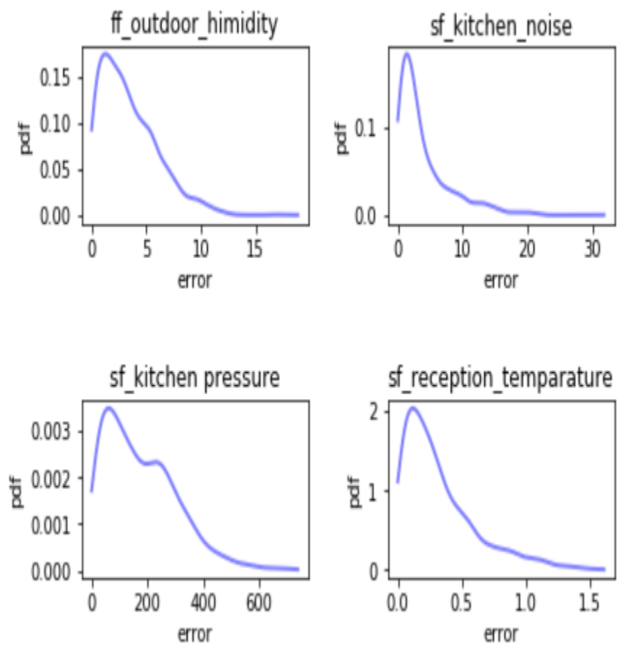


Figure 4.4: Error Distribution - 3

tail of the curve. Therefore our focus of interest is to identify the extreme events.

4.2.2 Error Distribution Parameters

We experimented with the parameters to find out the right set.

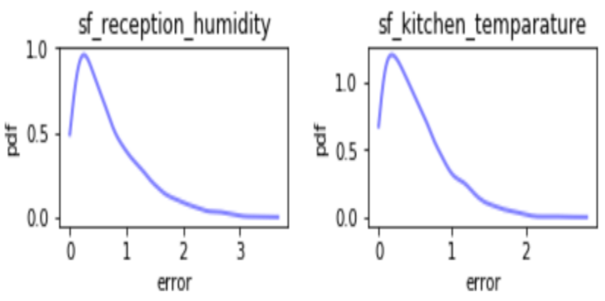
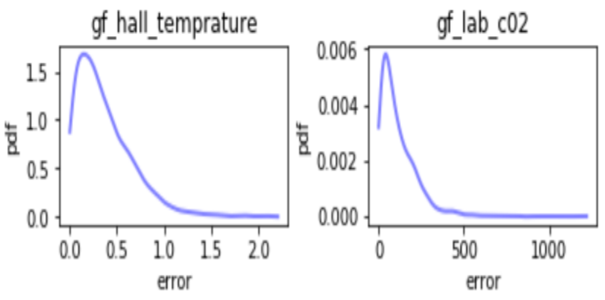



Figure 4.5: Error Distribution - 4

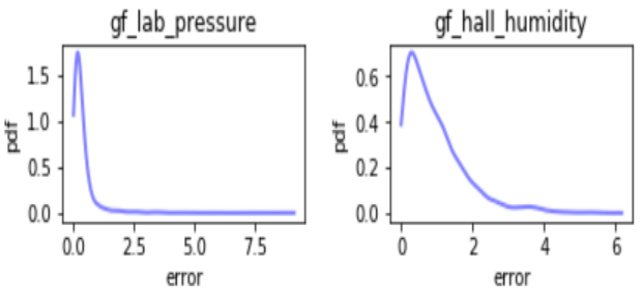
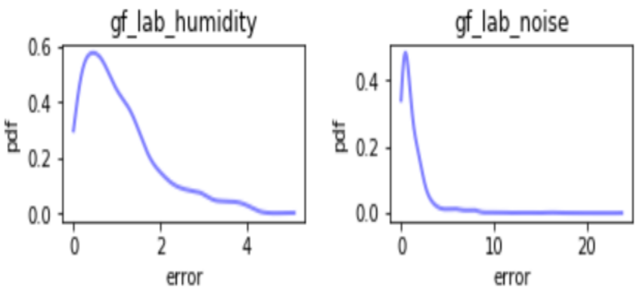



Figure 4.6: Error Distribution - 5

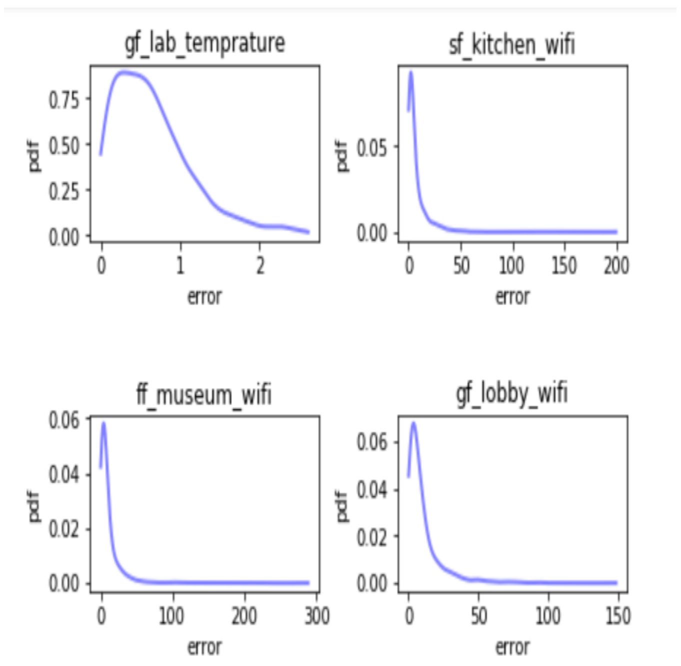


Figure 4.7: Error Distribution - 6

```
Algorithm 2: Parameters to find the rare values of Sensors
Find the Parameters for each Sensor
for sensor in 1 to 24(sensors) do
    Find the mean for each sensor $\mu_1, \mu_2, ...., \mu_24$
    Find the standard deviation for each sensor $\sigma_1, \sigma_2, ...., \sigma_24$
    Find the Threshold for each sensor $\tau_1, \tau_2, ....., \tau_24$
    $\tau_i = \mu_i + \alpha\sigma_i$
Plot the Error Distribution for Each Sensor.
```

In the above algorithm we experimented with the α value, and considered 2 in this work. After Finding the thresholds for each sensor, we filtered the time stamps based on these.

From the above figures [4.8,4.9,4.10] it is clearly showed that with the increase of support the anomaly count decreases. Therefore with the right threshold support we can eliminate the False positives.

4.2.3 False Positives Mitigation

From the above derived thresholds, we deduce the Anomalies for each sensor. We observed that, Using these thresholds we get a lot more anomalies for each sensors. i.e. too many False positives for the Anomaly detection using these Thresholds

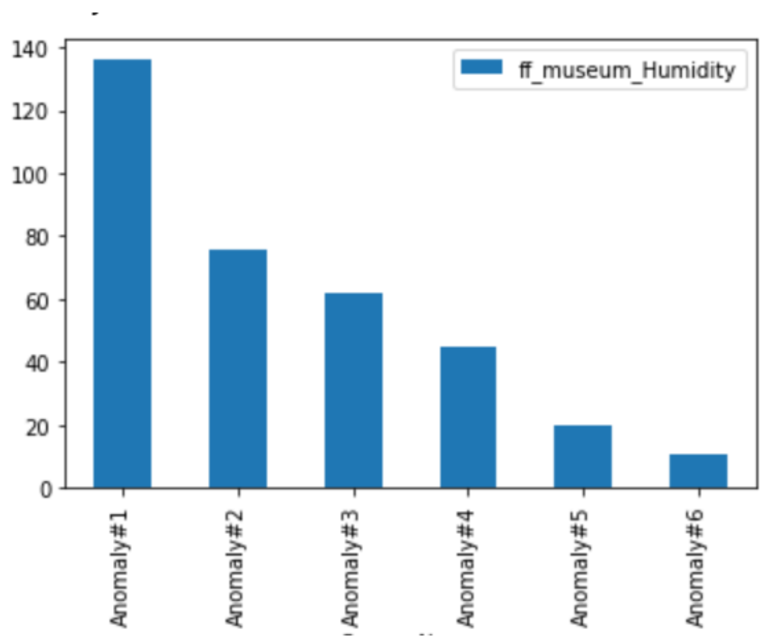


Figure 4.8: Sensor Anomaly Count - 1

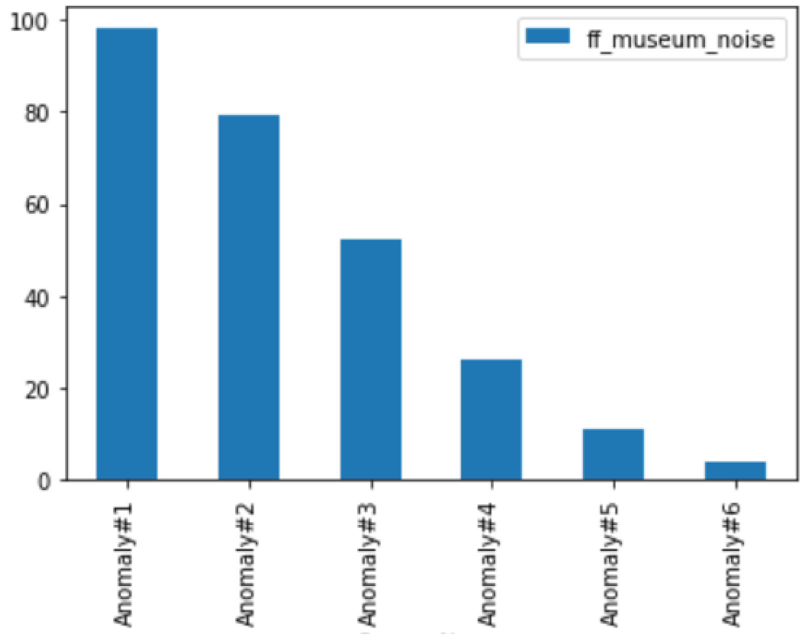


Figure 4.9: Sensor Anomaly Count - 2

straight away.

From the Figure 4.11, if we observe the first row, those are the number of anomalies reported by considering the Thresholds.

If we only consider these thresholds, There would be many False positives. To mitigate this, we include other aspects of data to support the Anomaly reported such as Scores. Therefore for each anomaly we are giving a score, which indicates

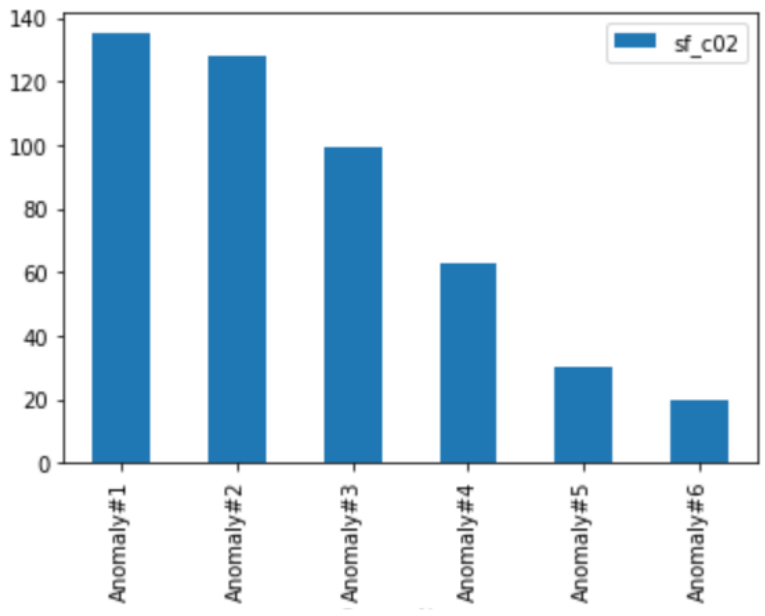
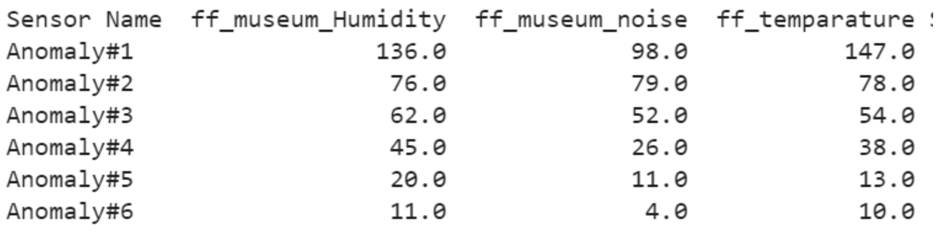


Figure 4.10: Sensor Anomaly Count - 3



```
Sensor Name  ff_museum_Humidity  ff_museum_noise  ff_temparature :
Anomaly#1                 136.0             98.0            147.0
Anomaly#2                  76.0             79.0             78.0
Anomaly#3                  62.0             52.0             54.0
Anomaly#4                  45.0             26.0             38.0
Anomaly#5                  20.0             11.0             13.0
Anomaly#6                  11.0              4.0             10.0
```

Figure 4.11: Anomaly Count - Sensor data

how strongly it is reported as anomaly with respect to all the other sensors.

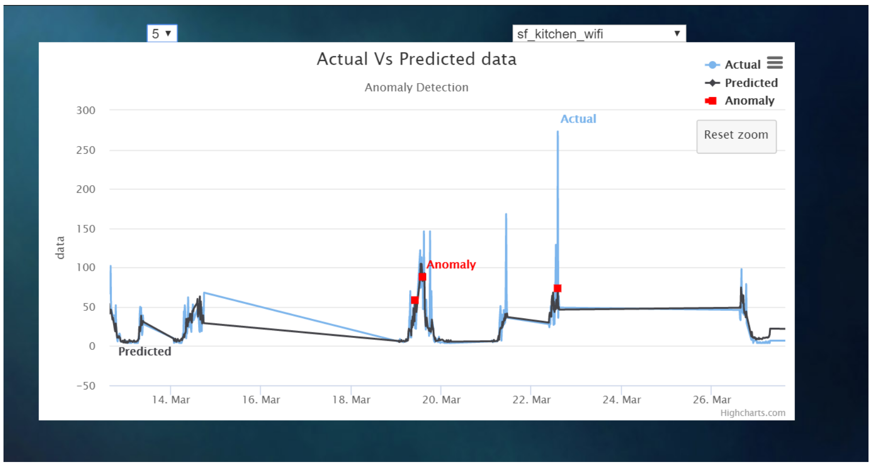


Figure 4.12: Anomaly detection

As shown in figure 4.12 shows the data from second floor wifi sensor, anomalies are labelled with red circles. If the difference between actual and predicted is more than the corresponding sensor threshold then we consider the score information to reduce the False Positives. The red dots shown in the plots are identified as Anomalies with 5 more other Sensors as well.

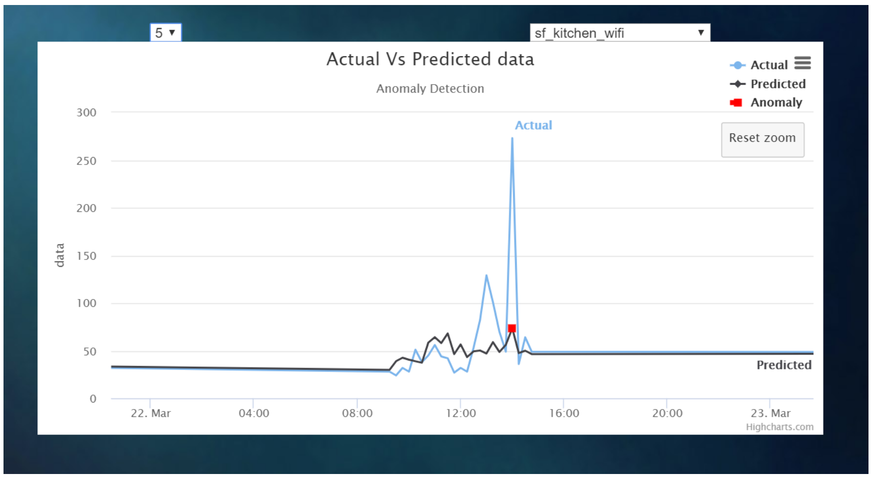


Figure 4.13: Web Interface

Figure 4.13 shows the people count from second floor kitchen wifi. Anomalies are labeled with small red circles. This Anomaly corresponds to March 22nd i.e on Friday. There is a sudden increase in the number of people in the second floor kitchen around 2PM. Which indicates unusual event may be gathering or some event.

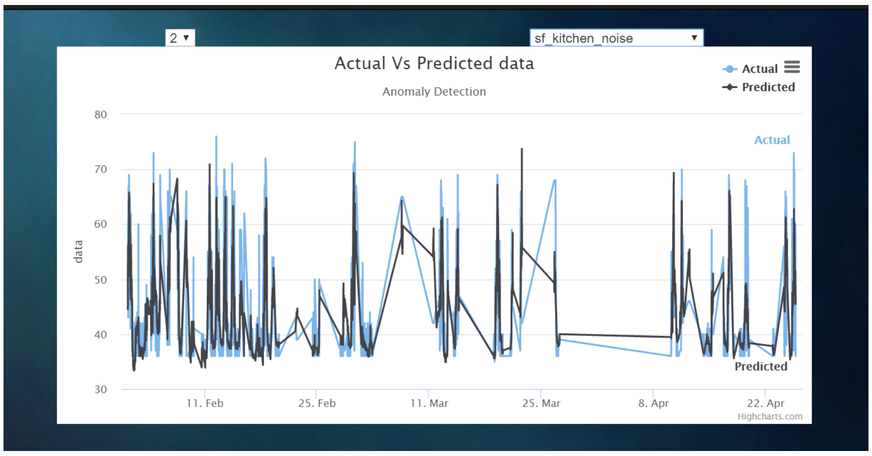


Figure 4.14: Linear Regression Model Correlation-1

Figure 4.14 and 4.15 explains the model correlation. This model is very much correlated with the actual data. This proves that our model is behaving well with this data.

There are some errors with respect to predicted data modelling. But they do

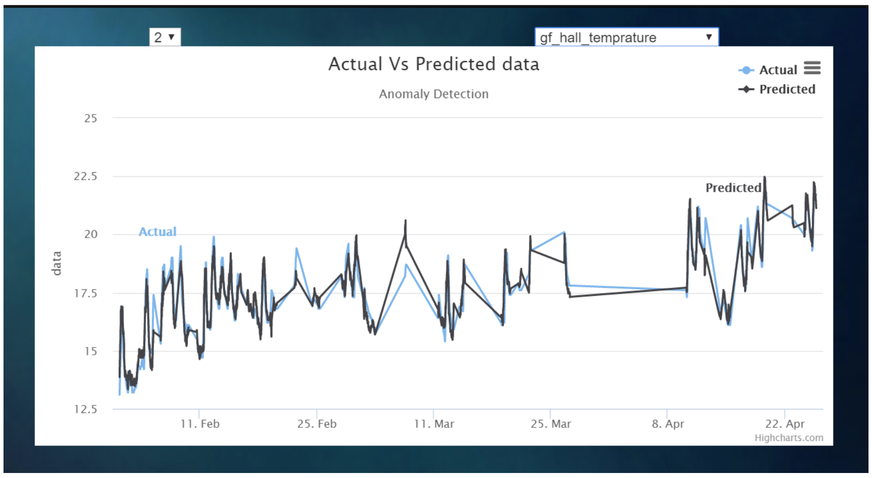


Figure 4.15: Linear Regression Model Correlation-2

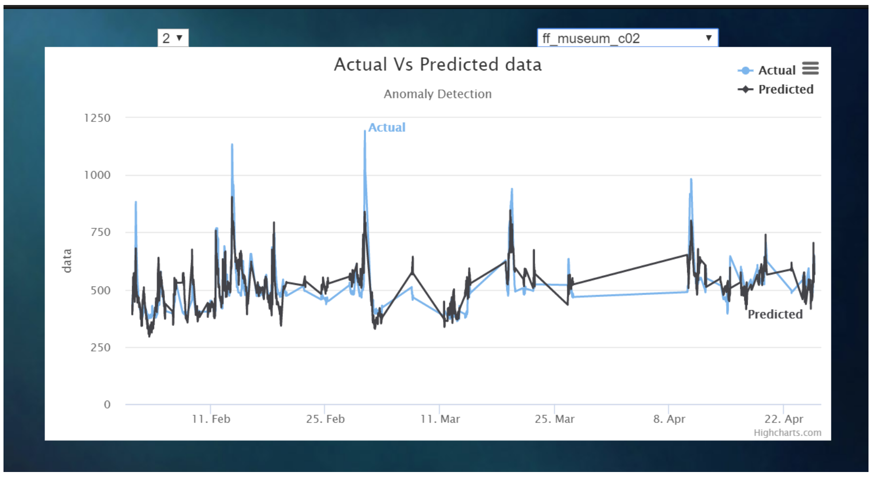


Figure 4.16: Linear Regression Model Correlation-3

not appear to be getting bigger – the model is not deteriorating

4.3 Web User Interface

As the part of final prototype, I have developed a web user interface, to visualize the values of each sensor in real time. The framework developed is best for monitoring and visualization. we used the below

- AngularJS[3]
- node.js[18]
- highcharts[14]

for the development of UI.

AngularJS is a JavaScript-based open-source front-end web framework mainly maintained by Google. Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Used HighCharts for the data view. It's a

lightweight and cross platform compatible. Easy and quick to setup. Implementation of high chart functionality is very simple and comes with bundle of other user friendly options.

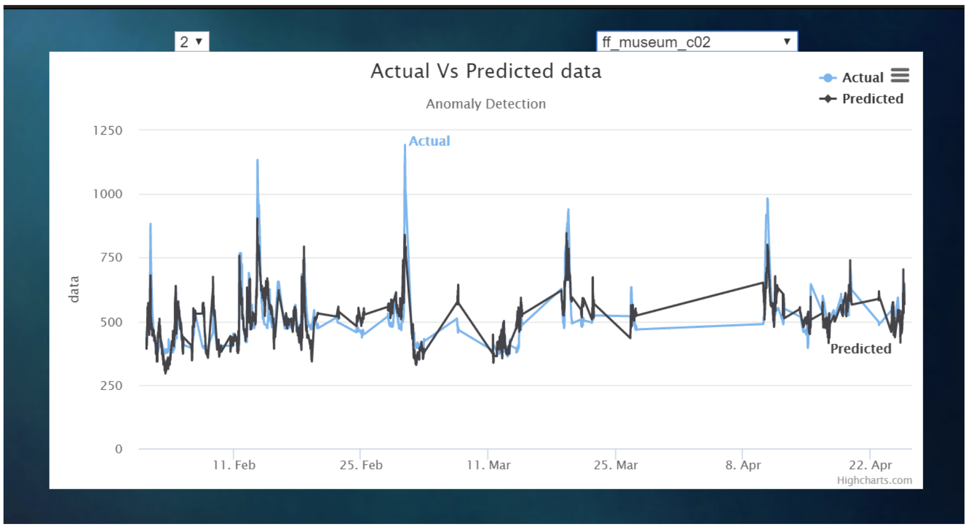


Figure 4.17: Anomaly Detection - User Interface

Figure 4.17 shows the User interface for the anomaly detection. Which has 2 drop down menus. The right side drop down has list of all sensors. The left side drop down menu indicates the score of the anomaly.

Figure 4.18 shows the drop down menu, which consists of list of all sensors.

Figure 4.19 shows the drop down menu, which shows the score or support of anomaly. For Example : If we select 5, the graph shows all the anomalies which are supported by 5 other sensors. This can be observed in Figure 4.20

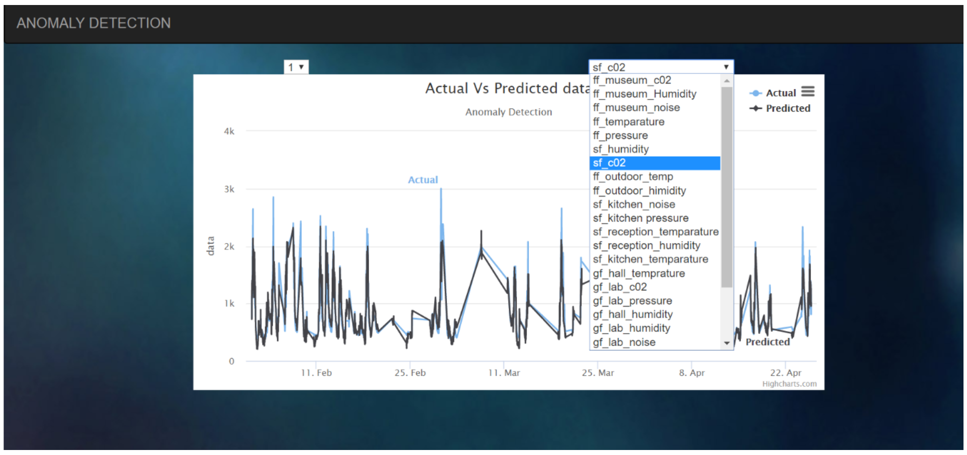


Figure 4.18: User Interface - Right DropDown

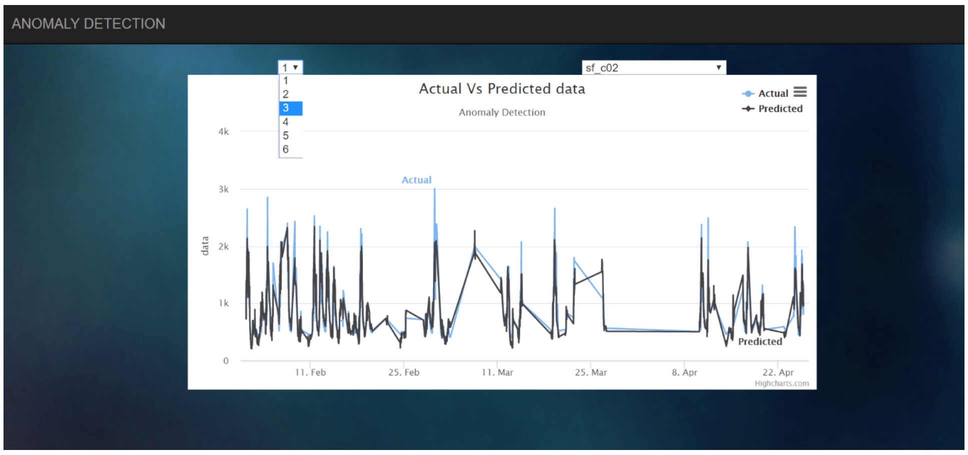
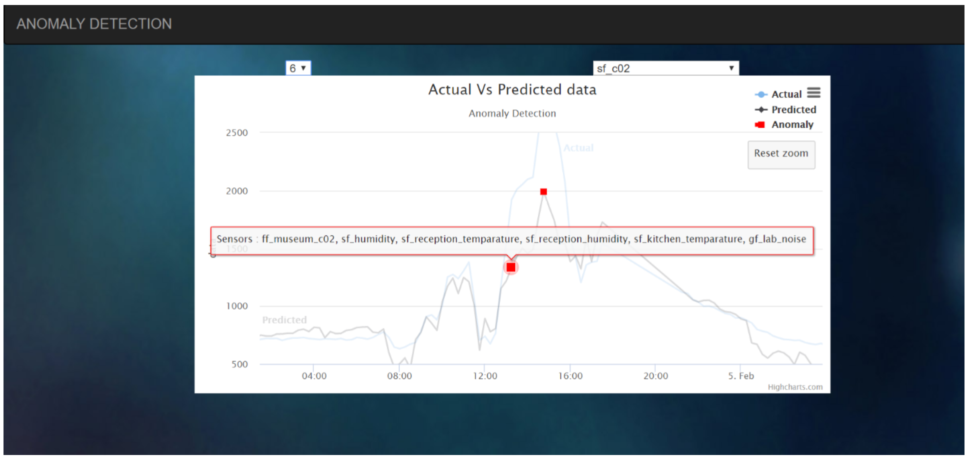


Figure 4.19: User Interface - Left DropDown



Sensors : ff_museum_c02, sf_humidity, sf_reception_temparature, sf_reception_humidity, sf_kitchen_temparature, gf_lab_noise

Figure 4.20: User Interface-4

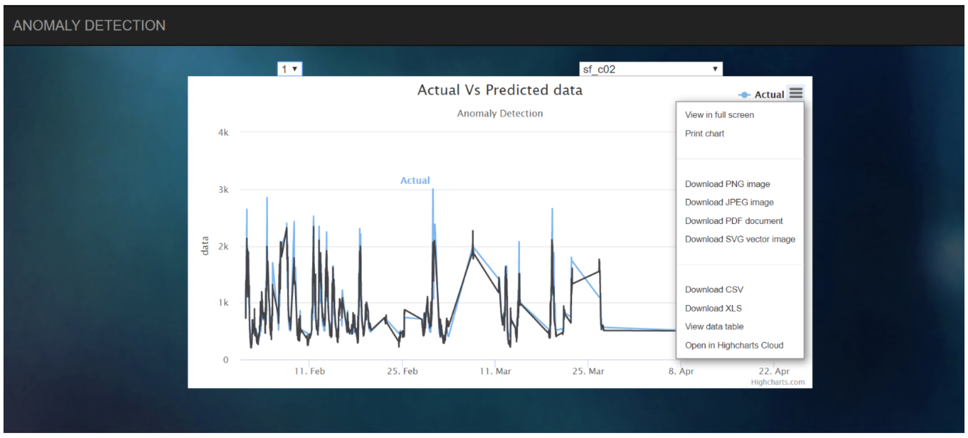


Figure 4.21: User Interface-3

Chapter 5

Conclusion and Further Work

5.1 Conclusion

Anomalous activity can happen anywhere. Detecting it at the earliest will be very effective in terms of resources. Due to the increasing demand for identifying anomalies faster. we progress towards wielding connected real-time sensors, and the detection of anomalies in data streams [2].

The contributions made in this work are as follows:

- Demonstrating this problem as Machine Learning problem .
- Explored the use of statistical approaches in identifying anomalies in time series data.
- Successful application and validation of our Modelling technique.
- Exploring other effective techniques in the area of Anomaly detection.

5.2 Further work

There is a big room for improvements in this work.

- Design End to End applications by automatizing every step.
- Designing the best strategy to handle the missing values, such as efficiently imputing the missing values.

- Managing the temporal dependencies well.
- Capturing the seasonality well to reduce the false positive alarms.
- Capturing the trend, seasonality and bias well in time series data.

Below is the final prototype to be implemented as a further step in this analysis. Our goal is to generate a framework that can scale to the streaming data, which can be applied using a pipeline-based API. In the pipelines, we model the data processing as a sequence of transmutations implemented over input data. The different steps in the process are as follows:

- Extract the statistics (mean and standard deviation) in analyzing for anomalies from the raw data stream.
- As the input data is an infinite stream, group events by the time window.
- Transform the activities in each window to be analyzed by the anomaly detection model.
- Apply the anomaly detection model to the data and evaluate if it is an anomaly or not, then produce an output report.
- The output is a stream of records that are ready for post-processing, activating an alerting system or plug into a real-time dashboard.

Window referencing mechanisms enable the system to represent the expected behaviour of the data dynamically. For example, if a data has seasonality which correlates with a day of the week hour of the day, when analyzing a current time window, e.g. 10:00 – 11:00 on a Monday, we might want to look at previous Mondays at 10:00-11:00 to understand what to expect. After the detection model evaluates a windowed data model, it remains in the system pool. This pool can be stored in memory as a key-value pair. We calculate which old windows are relevant to the detection model when analyzing a specific real-time window and retrieve them from the pool. We can supply a reducer function that merges the historical data retrieved and generates a source of expected behaviour. Thus allowing the system to operate on real-time series data efficiently and detect the anomalies effectively.

Appendix A

Code

The code written as part of this project is uploaded in this gitgub link `https://github.com/chandanadasarii/AnomalyDetection_IOT`

The code in the github link has two parts

- The iPython notebook 'AnomalyDetection_FinalProject.ipynb' contains the code written for modelling.
- 'index.html' and other supporting files of nodejs contains the code for the web interface.

The link to the web interface developed as part of this project is `https://chandanadasarii.github.io/AnomalyDetection_IOT/#`

Bibliography

[1] Bovas Abraham and Alice Chuang. "Outlier detection and time series modeling". In: *Technometrics* 31.2 (1989), pp. 241–248.

[2] Subutai Ahmad et al. "Unsupervised real-time anomaly detection for streaming data". In: *Neurocomputing* 262 (2017), pp. 134–147.

[3] *AngularJS*. URL: `https://angularjs.org/`.

[4] *Anomaly Detection Techniques*. URL: `https://www.datascience.com/blog/python-anomaly-detection`.

[5] George EP Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[6] Peter J Brockwell, Richard A Davis, and Matthew V Calder. *Introduction to time series and forecasting*. Vol. 2. Springer, 2002.

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), p. 15.

[8] Varun Chandola, Varun Mithal, and Vipin Kumar. "Comparative evaluation of anomaly detection techniques for sequence data". In: *2008 Eighth IEEE international conference on data mining*. IEEE. 2008, pp. 743–748.

[9] *Chapter 13 - Anomaly Detection*. URL: `https://www.sciencedirect.com/science/article/pii/B9780128147610000137`.

[10] Sarah M Erfani et al. "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning". In: *Pattern Recognition* 58 (2016), pp. 121–134.

[11] Jianqing Fan and Qiwei Yao. *Nonlinear time series: nonparametric and parametric methods*. Springer Science & Business Media, 2008.

[12] Ryohei Fujimaki, Takehisa Yairi, and Kazuo Machida. “An approach to spacecraft anomaly detection problem using kernel feature space”. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM. 2005, pp. 401–410.

[13] Frank E Grubbs. “Procedures for detecting outlying observations in samples”. In: *Technometrics* 11.1 (1969), pp. 1–21.

[14] *highcharts*. URL: `https://www.highcharts.com/`.

[15] David J Hill and Barbara S Minsker. “Anomaly detection in streaming environmental sensor data: A data-driven modeling approach”. In: *Environmental Modelling & Software* 25.9 (2010), pp. 1014–1022.

[16] Cheng-Ming Lee and Chia-Nan Ko. “Short-term load forecasting using lifting scheme and ARIMA models”. In: *Expert Systems with Applications* 38.5 (2011), pp. 5902–5911.

[17] *MultivariateStatistics*. URL: `https://en.wikipedia.org/wiki/Multivariate_statistics`.

[18] *nodejs*. URL: `https://nodejs.org/en/`.

[19] *scikit-learn*. URL: `https://scikit-learn.org/stable/`.

[20] Walter Andrew Shewhart. *Economic control of quality of manufactured product*. ASQ Quality Press, 1931.

[21] Brian L Smith, Billy M Williams, and R Keith Oswald. “Comparison of parametric and nonparametric models for traffic flow forecasting”. In: *Transportation Research Part C: Emerging Technologies* 10.4 (2002), pp. 303–321.

[22] Wilhelmine Stefansky. “Rejecting outliers in factorial designs”. In: *Technometrics* 14.2 (1972), pp. 469–479.

[23] James P Theiler and D Michael Cai. “Resampling approach for anomaly detection in multispectral images”. In: *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery IX*. Vol. 5093. International Society for Optics and Photonics. 2003, pp. 230–240.

[24] Philip HS Torr and David W Murray. “Outlier detection and motion segmentation”. In: *Sensor Fusion VI*. Vol. 2059. International Society for Optics and Photonics. 1993, pp. 432–443.