2020

Real-Time Detection of Demand Manipulation Attacks on a Power Grid

Srinidhi Madabhushi

Real-time Detection of Demand Manipulation Attacks on a Power Grid

---

A Thesis
Presented to
the Faculty of the
Daniel Felix Ritchie School of
Engineering and Computer Science
University of Denver

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

---

by
Srinidhi Madabhushi
December 2020
Advisor: Rinku Dewri

Author: Srinidhi Madabhushi
Title: Real-time Detection of Demand Manipulation Attacks on a Power Grid
Advisor: Rinku Dewri
Degree Date: December 2020

Abstract

An increased usage in IoT devices across the globe has posed a threat to the power grid. When an attacker has access to multiple IoT devices within the same geographical location, they can possibly disrupt the power grid by regulating a botnet of high-wattage IoT devices. Based on the time and situation of the attack, an adversary needs access to a fixed number of IoT devices to synchronously switch on/off all of them, resulting in an imbalance between the supply and demand. When the frequency of the power generators drops below a threshold value, it can lead to the generators tripping and potentially failing. Attacks such as these can cause an imbalance in the grid frequency, line failures and cascades, can disrupt a black start or increase the operating cost. The challenge lies in early detection of abnormal demand peaks in a large section of the power grid from the power operator's side, as it only takes seconds to cause a generator failure before any action could be taken.

Anomaly detection comes handy to flag the power operator of an anomalous behavior while such an attack is taking place. However, it is difficult to detect anomalies especially when such attacks are taking place obscurely and for prolonged time periods. With this motive, we compare different anomaly detection systems in terms of detecting these anomalies collectively. We generate attack data using real-world power consumption data across multiple apartments to assess the performance of various prediction-based detection techniques as well as commercial detection applications and observe the cases when the attacks were not detected. Using static thresholds for the detection process does not reliably detect attacks when they are performed in different times of the year and also lets the attacker exploit the system to create the attack obscurely. To combat the effects of using static thresholds, we propose a novel dynamic thresholding mechanism, which improves the attack detection reaching up to 100% detection rate, when used with prediction-based anomaly score techniques.

Acknowledgements

I would like to express my sincere gratitude towards my research advisor, Dr. Rinku Dewri, for providing me with the knowledge, and constantly guiding me in the right direction throughout my research. I am honored to have worked with him and gained a lot of experience in the field.

I would also like to thank my defense committee members, Dr. Anneliese Andrews and Dr. David Wenzhong Gao for reviewing my thesis and for being part of my oral defense committee.

My heartiest thanks to my parents Narayan and Srividya, my sister Srisowmya, and my friends, for their continuous support and encouragement throughout my thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet of Things (IoT) allows devices to connect and communicate to each other with the presence of the Internet. There has been an increase in the number of IoT devices connected to the internet every year. According to a forecast by the International Data Corporation, there will be upwards of 40 billion connected IoT devices, generating 79.4 zettabytes of data in 2025 [25]. This will include the fastest growth in industrial and automotive equipment, strong adoption in smart home and wearable devices, and a rapid growth in video surveillance. At the same time, although an individual IoT device has low power consumption, the sheer number of active devices at a time, and the necessary data centers to support them, will increase the rate of energy consumption [27]. In addition, IoT security, privacy, cost and regulation remains some of the most pressing concerns of enterprises [13].

The Open Web Application Security Project's (OWASP) Internet of Things project highlights the top ten threats looming around IoT devices[1]. These include weak, guessable or hardcoded passwords, insecure network services, insecure ecosystem interfaces, lack of secure update mechanisms, use of insecure or outdated components, insufficient privacy protection, insecure data transfers and storage, lack of device management, insecure default settings, and lack of physical hardening. The security risks in IoT devices have been converted over the years to many attacks, including the 2016 Mirai Botnet attack on the Dyn DNS provider that lead to a disruption of the Internet, the 2019 demonstration of hackable cardiac devices (deplete battery or administer incorrect pace), home

[1] https://owasp.org/www-project-internet-of-things/

surveillance camera hacks that allow anyone to view cameras, taking control of a Jeep SUV using the vehicle's controller area network bus, or the 2016 taking down of a central heating system in two housing blocks in Finland.

With the many security risks that IoT imposes, there is a threat to the power grid as well, when these devices are breached and manipulated by an adversary. This category of attacks where IoT devices are controlled by an attacker with the goal of affecting the power grid was discussed by Soltan et al. and termed as Manipulation of Demand via IoT devices (MadIoT) attacks [42]. When an attacker has access to high-wattage IoT devices, they can synchronously switch on/off these devices at the same time such that it causes sudden increase or decrease in electricity demand. Such attacks may have adverse effects on the power grid leading to line failures and damage of power generators. Real-time detection of IoT attacks targeted to disrupt the power grid is an area that has less research performed. This thesis aims to explore various anomaly detection methods to detect these attacks during their preliminary stages. Early detection of these attacks will help the power grid operator in taking necessary action to combat any resultant effects, one of which can be a black out.

Anomaly detection has gained lot of attention during the recent years because of the need to process big data in real time. It is humanly impossible to go through enormous amounts of data in search of anomalous situations. There has been many anomaly detection techniques proposed by researchers in different application domains. However, anomaly detection at a grid level is an area that has been less explored and with this thesis, we would like to compare different techniques as well as commercial methods and provide a list of situations that these methods are not able to detect in an attack. A specific class of detection techniques, referred to as threshold-based anomaly detection techniques, require setting a numeric threshold parameter that demarcates the boundary between anomalous and normal events in time. While a threshold based anomaly detection technique may be easier to set up, the primary research question we explore in this thesis is whether it leaves avenues for an attacker to inject higher usages without getting noticed. In fact, since statistical learning methods aim to not overfit a model to the training data, there remains the possibility for an attacker to exploit a model to work within its learned parameters and still inflict damage. In scenarios with time-series data where data points dynamically change over time, a method's tendency to adjust to changes in patterns can also create room for an attacker to change a model's understanding of normal behavior. By exploiting a threshold based anomaly detection system, we generate attacks that conforms to

the behavior of MadIoT attacks using real-world power consumption data consisting of more than hundred apartments' consumption at a minute level.

Based on the observations of the performance of different techniques, we propose a dynamic thresholding mechanism based on a spring system that aims to detect anomalies which other thresholding mechanisms fail to detect. This approach has provided encouraging results by detecting 100% of attack situations in some cases. This thresholding mechanism is aimed to detect situations that are similar to the behavior of attacks that involve manipulation of IoT devices.

1.1 Thesis Contributions

This thesis exposes the performance of various anomaly detection and commercial methods in detecting attacks that affect the power grid by manipulation of the demand from IoT devices. The principal contributions of this thesis are:

- Generation of a set of demand manipulation attacks to demonstrate gaps in detection techniques that rely on static anomaly thresholds, irrespective of their reliance on daily, weekly, monthly or seasonal patterns;
- Extensive comparative evaluation of multiple anomaly detection methods, including five commercially used applications, in terms of their ability to detect our demand manipulation attacks; and
- A novel dynamic thresholding mechanism to augment threshold-based detection methods and improve their detection capabilities.

1.2 Thesis Outline

The chapters in this thesis are organized as follows:

- Related Work: We discuss previous works in the field of anomaly detection in various application domains, including the power consumption domain. We also discuss research work performed in dynamic thresholding and collective anomaly detection.
- Background: This chapter covers description of the background to understand the working of a power grid, MadIoT attacks, anomaly detection and time series methods.

- Attack Generation: This chapter discusses the data set we use for attacks and the attack generation process.
- Attack Performance and Evaluation: This chapter discusses the results of performance of the attacks in various anomaly detection systems with different thresholds, as well as in commercial methods.
- Dynamic Thresholding: In this chapter, we provide insights on when an attack was not detected by a method and form a list of expectations of a thresholding mechanism. Then, we propose our dynamic threshold mechanism and show the results of using this mechanism.
- Conclusion and Future Work: This chapter summarizes the contributions of the work in the thesis and briefly discusses the different directions that can be explored for better detection of MadIoT attacks.

Chapter 2

Related Work

In this chapter, we will discuss the related work which is categorized into anomaly detection, anomaly detection in power grid specifically, anomaly detection using dynamic thresholding and collective anomaly detection.

2.1 Anomaly Detection in various Application Domains

Varun et al. provide a structured and comprehensive survey of the research on anomaly detection [8]. They have grouped anomaly detection techniques into various categories based on the underlying theory of each technique. Additionally, they also provide computational complexities, advantages and disadvantages for each of the existing techniques. A similar survey has been done by Hodge and Austin where they provide various anomaly detection techniques in machine learning and statistical domains [17]. A survey on real-time big data processing for anomaly detection was performed by Habeeb et al. in the domain of network security [5].

There has been many anomaly detection techniques proposed by researchers that apply for specific application domains using graph techniques. An extensive survey on graph based anomaly detection techniques was performed by Akoglu et al. highlighting the effectiveness, scalability, generality and robustness of various techniques [3]. They also present many real-world applications of graph-based anomaly detection encouraging the usage of graph-based tools for visualization, monitoring and detection of anomalies. Ranshous et al. also provide a comprehensive overview of anomaly detection in graphs, specifically for dynamic networks which are constantly changing

in terms of their structure and attributes [35]. For example, insertion and deletion of vertices or edges in a network causes a change in the structure. Similarly, Verdoja and Grangetto used novel graph-based solution for anomaly detection in images using a Laplacian model of the image's background [44]. They have claimed to have outperformed other benchmark methods in image anomaly detection by performing experiments on hyperspectral and medical images. GraphBad is a graph-based analysis tool proposed by Parkinson et al. which aims to analyze security configuration data, identify anomalies that could lead to potential security risks and suggest appropriate mitigation plans [32].

There are several anomaly detection methods that are based on neural networks. Zhou et al. use spacial-temporal Convolutional Neural Networks (CNN) for detection of anomalous behavior in video sequences of crowded scenes [50]. Using CNN, they are able to extract appearance and motion information encoded in continuous frames of the video. Murugan et al. also used Region based Scalable CNN (RS-CNN) for faster identification of different sizes of anomalies in pedestrian walkways [28]. They have compared the performance of RS-CNN with several state-of-the-art detection techniques and found RS-CNN to be faster and efficient in detection.

Zhao et al. provide a correlation-based anomaly detection method which detects anomalies based on the correlations between sensors [48]. This technique is used for predicting failures effectively earlier and reduce the cost of downtime and maintenance.

These are some of the various application domains explored by different researchers including networks and computer vision. In the next section, we will go through contributions done in the power grid domain.

2.2 Anomaly Detection in Power Grid

In this section, we will discuss related work in the same application domain as this thesis. Sisworahardjo and Saad have performed a spatio-temporal context anomaly detection, where they presented a contextual anomaly detection algorithm to detect irregular power consumption using a normalized anomaly score [40]. This score calculation is based on historical consumption data and is adjusted based on contextual variables like season or historical consumption patterns. Ouyang et al. proposed a three stage multi-view stacking ensemble (TMSE), which is a machine learning model based on hierarchical time series feature extraction (HTSF) methods [30]. This anomaly detection

algorithm is focused on accurate extraction of summary features, shift features, transform features and decompose features of the time series power consumption data, which aids in better detection of anomalies in consumption.

Anomaly detection in commercial buildings using a novel unsupervised anomaly detection algorithm and anomaly scores was provided by Janetzko et al., where they focus on visual analytics for anomaly detection in power consumption data [21]. Chahla et al. have explored the deep learning approach to anomaly detection and prediction of power consumption which allows offline and real-time detection of anomalies [7]. They proposed a novel unsupervised approach combining clustering-based methods with prediction-based methods to learn typical behavior scenarios and to predict consumption for the next hour. These scenarios are learned by using the K-means algorithm and the prediction of the next hour is given using Long Short-Term Memory (LSTM). The prediction value along with some recent historical values are compared as a group to the learned typical scenarios to detect the anomalies.

Shouyu et al. proposed a tree-based algorithm for computing anomaly scores and use K-means to cluster these anomaly scores and categorize them based on the alert type [39]. This research was conducted for power grid dispatching which involves guaranteed power production and providing end users with reliable power supply. Passerini et al. used the data from the underlying smart grid network to detect anomalies that occur in the power distribution grid [33]. They used the data coming from power line modems for monitoring and detecting the anomalies by estimating various grid parameters. A similar work was done by Jamei et al. where they monitor the distribution grid using microphasor measurement unit (μPMU) devices for detecting anomalous behavior in a controlled perimeter of the grid [20]. These devices are used for estimating the the magnitude and phase angle of an electrical phasor quantity like voltage or current in the electricity grid. They use the mean values of the quantities which ideally change smoothly over time, and if there are high variations in the mean, they categorize that as an anomaly. We use a similar technique in the dynamic thresholding section of our thesis where we observe the coefficients of variation of the anomaly score we calculate to find an anomalous point.

An anomaly detection algorithm to detect anomalous behavior in power grid substations was proposed by Hong et al. [18] where they used algorithms for detecting temporal anomalies in the substation facilities like intrusion attempts, change of file system, change of system's status, etc. and detecting the anomalies by observing network messages and filtering Generic Object Oriented

Substation Event (GOOSE) and Sampled Measured Value (SMV) protocol-based messages. Yang et al. proposed an anomaly detection algorithm with a similar goal of finding anomalies in Substation Communication Network (SCN) [46]. They use a fractional autoregressive integrated moving average (FARIMA) based threshold model to detect anomalies in the SCN traffic flow. A similar technique was used by Elbez et al. where they aim to detect Denial of Service (DoS) attacks using Autoregressive Fractionally Integrated Moving Average (ARFIMA) model where they aim to use GOOSE communication in the substation network to monitor and detect anomalies based on a statistical approach [12].

Moghaddass and Wang use large scale smart meter data collected at consumers' premises to build a real-time anomaly detection model for detecting abnormal conditions at both lateral and consumer levels [26]. They provide a parameter estimation method for model training and use that as the basis for anomaly detection. Karimipour et al. proposed an unsupervised anomaly detection scheme based on statistical correlation between various grid parameters which aims to recognize False Data Injection (FDI) attacks [23]. This recognizes the behavioral patterns using historical measurements of the data and captures the dependencies between variables.

Cui and Wang proposed an anomaly detection system which is a hybrid model that combines polynomial regression and Gaussian distribution to detect anomalies in school electricity consumption data [10]. Weng et al. provide a multi agent based unsupervised anomaly detection on smart campuses by using ensemble methods for labeling data and deep learning techniques for detecting anomalies in an unsupervised fashion [45].

Serrano-Guerrero et al. focused on time series treatment and they proposed a novel Seasonality Analysis of Electricity Consumption (SAEC) method for handing the time series components accurately [38]. Then, they use a novel SAICC (statistical assessment for identifying changes in consumption) methodology for presenting an index of change that quantifies and catalogs the anomalies in the consumption data profiles.

An anomaly detection technique for electricity consumption in smart grids was proposed by Li et al. where they collect consumption data continuously from smart meters and use a data mining algorithm to divide the incoming data into classes namely working day, holiday and outlier classes [24]. Then, it is checked whether the consumption falls in the normal range and flagged as an anomaly based on various contexts such as anomalous behavior of single sensor or multiple sensors, etc.

2.3 Anomaly Detection with Dynamic Thresholding

Haque et al. use dynamic thresholding for flagging anomalies in sensor behavior in wireless sensor network in healthcare [16]. They predict a sensor value from historic values and find a distance metric between the actual and predicted value. This metric is then compared to a threshold which is dynamically adjusted by finding the standard deviation of a certain window of historic data that allows an upper and lower bounds to be set for an anomaly score. A similar dynamic thresholding mechanism based on mean and standard deviation was proposed by Salem et al. aiming to detect anomalies in wireless sensor networks for reliable healthcare monitoring [37].

Bezerra and Wainer proposed a dynamic thresholding algorithm for anomaly detection in logs of process aware systems [6]. This thresholding technique is a formula that is based on the mean and standard deviation of the observed historic samples. David and Thomas used a sliding window approach to calculate the mean and variance which is used for determining the threshold at a point in time dynamically for detecting Distributed Denial of Service (DDoS) flood attacks [11]. Poornima and Paramasivan also use a mean and standard deviation based thresholding mechanism calculated in sliding windows of size 40 time units for detecting anomalies in wireless sensor networks [34].

Yeung and Ding provide an intrusion detection system using dynamic and static behavioral models [47]. They have modeled the dynamic approach based on hidden Markov models (HMM) and the principle of maximum likelihood. Park et al. also use a varying log likelihood mechanism for multimodal anomaly detection for assistive robots [31].

2.4 Collective Anomaly Detection

Ahmed and Mahmood provide a collective anomaly detection method which uses a partitional clustering technique to detect anomalies based on empirical analysis of any type of attack targeting networks [2]. They analyze network traffic patterns and detect anomalies based on a variation of K means algorithm.

Zeng et al. quantify the anomaly based on a scoring method and use stream anomaly score technique which is based on the variation of a data point when compared to its neighbors for finding contextual and collective anomalies in streaming data [22]. Araya et al. also aim on detecting contextual and collective anomalies based on a sliding window approach to determine the patterns

in the original data and comparing the pattern as a whole to a threshold value that is calculated using mean square errors in that window [4].

A collective anomaly detection technique based on Long Short-Term Memory Recurrent Neural Network (LSTM RNN) was proposed by Thi et al. where they detect collective anomalies based on observing whether the prediction errors of recent observations are above a threshold value [29]. Fisch et al. proposed a linear time method for detecting point and collective anomalies which is characterized by observing changes in mean, variance or both [14].

Zheng et al. proposed an anomaly detection algorithm for detecting collective anomalies in spatio-temporal data across different application domains [49]. They use a Spatio-Temporal Likelihood Ratio Test (ST_LRT) model for detection of collective anomalies which learns an underlying distribution for different datasets, and calculates an anomalous degree for each dataset based on a likelihood ratio test (LRT).

Rossi et al. used a frequent mining and categorical clustering with a goodness of fit thresholding mechanism to detect collective anomalies in streaming smart meter data [36].

2.5 Chapter Summary

In this chapter, we discussed related works in anomaly detection for various application domains and specifically discussed about the research in power grid. There has been research performed in the area of detecting anomalies at a grid level which is based on grid parameters like voltage, frequency and current. However, there are very few works that explored the area of detecting anomalies from a consumption perspective at a community scale by monitoring the wattage demand. There are very few works related to collective anomaly detection in power consumption which motivated us to explore this area. We conclude that there is a lack of research performed in detection of attack scenarios caused due to the manipulation of IoT devices. Through this thesis, we propose an anomaly detection model that can effectively detect point, contextual and collective anomalies along with a dynamic thresholding mechanism. We also explore static thresholds and provide detailed analysis and comparison of anomaly detection in power consumption data with static and dynamic thresholds, as well as commercial methods.

Chapter 3

Background

3.1 The Power Grid

The power grid is a complex and highly engineered network that coordinates between the generation and distribution of electricity to its customers. It comprises of four physical components- generation, transmission, distribution and storage as shown in figure 3.1. The electric power is produced at the generating station and is then transmitted through a high voltage transmission network. From there, it is then distributed to the end users that can be industrial or residential customers.

The rate at which the electricity is generated, transferred or consumed is measured with the unit called watts. The amount of electric energy which is generated, transmitted or used over time is measured as the number of watt-hours.

In the United States and other developed countries, electricity is generated instantaneously. Thus, the electricity is produced on demand[1]. The amount of electricity which is consumed varies and follows patterns throughout the day and year. The demand also changes regionally based on each season of the year. There could be situations where weekly seasonality can be seen. For example, commercial and industrial activities are usually less during weekends than weekdays. The non-commercial and activities of individuals also changes based on the day of the week. However, for the power grid to operate in a stable condition, it is required that there is a balance between the power supply and demand. In order to maintain this balance, the power operators use historical consumption data and weather data to predict the power demand in a timely manner. With these

predictions, they will be able to allocate adequate resources needed for the generation of the expected demand.

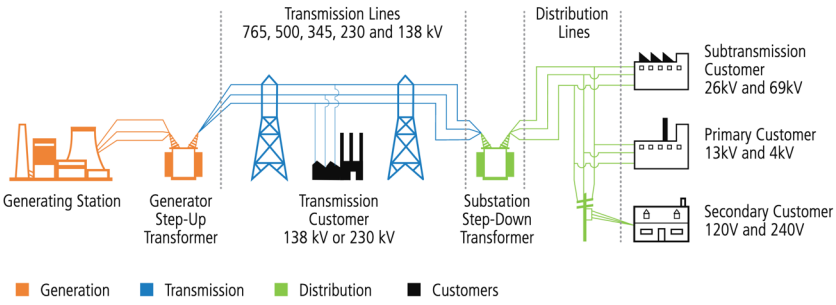

Figure 3.1: The power grid[1]

3.2 MadIoT Attacks

The MadIoT attack which is short for Manipulation of Demand via IoT is the term introduced by Soltan et al. [42] in their paper published in 2018. They have briefly described how different attacks can be possible in the power grid due to manipulation and control of various IoT devices by synchronously switching them on and off. All the attacks lead to the disruption of the power grid. They demonstrated attacks that can be broadly classified into one of the three types:

1. Attacks that result in frequency instability
2. Attacks that cause line failures and result in cascading failures
3. Attacks that increase operating costs

In order to simulate these attacks, MATPOWER and PowerWorld simulators were used by the authors. MATPOWER is an open-source MATLAB library which is widely used for computing the power flows in power grids. PowerWorld is an industrial-level software suite that is widely used by the industry for frequency stability analysis of power systems [42]. The detailed description of the attacks and the requirements for each successful attack is explained in the following sections.

[1]https://www.energy.gov/sites/prod/files/2015/09/f26/QER_AppendixC_Electricity.pdf

3.2.1 Attacks that result in Frequency Instability

Instability in a power grid can be caused by synchronously switching on or off many IoT devices within a geographical location. This drastic change causes an imbalance between the supply and demand. When the supply is greater than the demand, it causes a drop in the system frequency and the demand being greater than the supply causes a rise in the system frequency. This may result in the frequency going out of the nominal frequency range, which causes damage to the equipment. During such situations, the primary and secondary controllers are deployed to stabilize the system.

In order to demonstrate this type of attacks, the authors have used the WSCC 9-bus grid model. It represents a simple approximation of the Western System Coordinating Council (WSCC) with nine buses and three generators[2] and is widely used as a benchmark system [42].

The motives of this category of attacks may either be sudden generation tripping or disrupting a grid re-start. For successfully performing these attacks, the adversary needs access to a given number of IoT devices (bots) to perform a sudden increase or decrease in the demand. Assuming the attacker has access to the bots, the authors have simulated the attack for an increase in demand by 23MW and 30MW. This would require 200 to 300 bots per MW for causing the sudden generation tripping scenario.

The grid re-start is the process of restarting the grid after a blackout which is performed by the grid operator. The system's frequency is usually unstable during the black start and hence, the grid operator divides the grid into separate geographical islands and restarts the islands individually first. Once, the individual islands reach stability, the grid operator initiates the connections between the islands which may cause a small amount of instability, but the grid eventually reaches a stable state. The adversary would perform the attack during the reconnection of the different islands as the system is weak during that action. The attacker needs access to 100-200 bots per MW to disrupt a grid re-start.

3.2.2 Attacks that cause Line Failures and Cascading Failures

When there is a frequency instability in the power grid, the primary controller is deployed to stabilize the system. At this point, the operator does not have control over the power flows as it is now governed automatically by the grid based on Kirchoff's Law. In this situation, increasing the

[2]https://electricgrids.engr.tamu.edu/electric-grid-test-cases/wscc-9-bus-system/

demand by a small quantity causes failure of power lines due to overloads. This requires only 10 bots per MW to initiate a cascading failure resulting in 86% outage. The overall increase in the total demand is approximately 210MW.

These attacks may eventually lead to failure of other lines, thus causing cascading line failures. Another way of causing the cascading failure is by redistributing the demand keeping the total demand constant. This requires an absolute value of 80MW change in demand resulting in 4 bots per MW as the requirement to perform the attack.

The third type of attack in this category is the failure of a tie line which is used for carrying large amounts of power between two Independent System Operators (ISOs). The adversary requires 15 bots per MW in order to increase or decrease the demand in the tie line which results in the lines tripping.

These attacks were demonstrated using the Polish grid data Summer 2004 peak and Summer 2008 peak that are available through the MATPOWER library.

3.2.3 Attacks that Increase Operating Costs

These attacks are targeted to increase the operating costs of the grid without causing any line overloads. When the demand increases above the expected value, the ISO has to purchase additional power in the form of reserve generators. Using these generators results in the increase of power generation cost. The attacker will need access to 50 bots per MW to cause a 20% increase in operating cost. This attack was demonstrated on the Polish grid data Summer 2004 peak.

3.3 Time Series Analysis

A time series is data that is collected at various points in time. In a time series, there will be a temporal structure associated to it. In addition to the data points, the time at which each data point was recorded is crucial for modeling any time series. It also has the necessity to be updated periodically thus, leading to a large data size. There are several patterns that a time series exhibits that requires thorough analysis in recognizing them. This provides a better understanding of the data itself and aid in the choice of the forecast model. In the next few sections we will discuss about the different time series patterns and models.

3.3.1 Time Series Patterns

There are three types of patterns that we may see in a time series [19].

1. Trends: A trend occurs when there is a long term increase or decrease in the data. Figure 3.2 shows an example of an increasing trend in monthly shampoo sales over a three year period.

2. Seasonality: A seasonal pattern occurs in the data when it is affected by seasonal factors such as the time of the year or the day of the week. Seasonality is always of a fixed and known frequency. Figure 3.3 shows the minimum daily temperatures over ten years starting from 1981 to 1990 in the city of Melbourne, Australia. It can be clearly seen that the dataset exhibits strong yearly seasonality.

3. Cycle: A cycle occurs when the data exhibit rises and falls that are not of a fixed frequency. Figure 3.4 shows the monthly sale of single family homes in the US from the years 1966 to 1980, which exhibits cyclic behavior with a period of about 4 to 5 years.

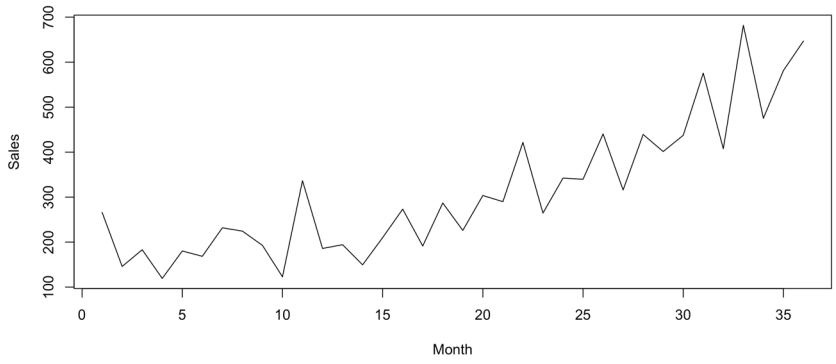

Figure 3.2: Increasing trend in monthly shampoo sales over a three year period

3.3.2 Time Series Models

There are many time series models which can be used for prediction of future values. We use six prediction methods discussed by Hyndman and Athanasopoulos [19] to evaluate the performance of our simulated attacks. The six methods can be broadly categorized into naïve and smoothing methods. Naïve, seasonal naïve and average methods fall under the naïve category and simple exponential smoothing, weighted average and Holt Winters' fall under the smoothing methods category.

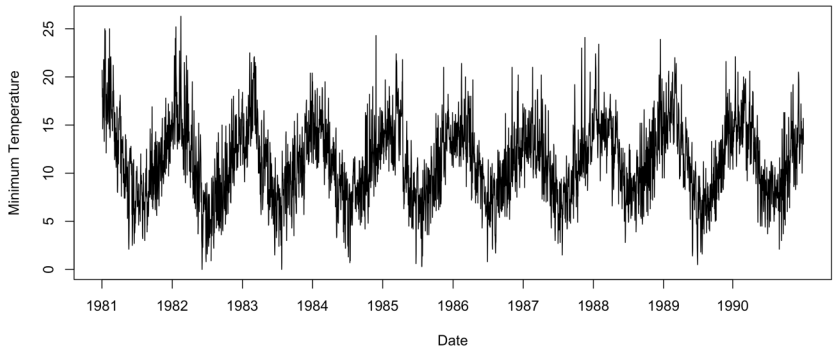

Figure 3.3: Yearly seasonality in minimum daily temperatures over ten years

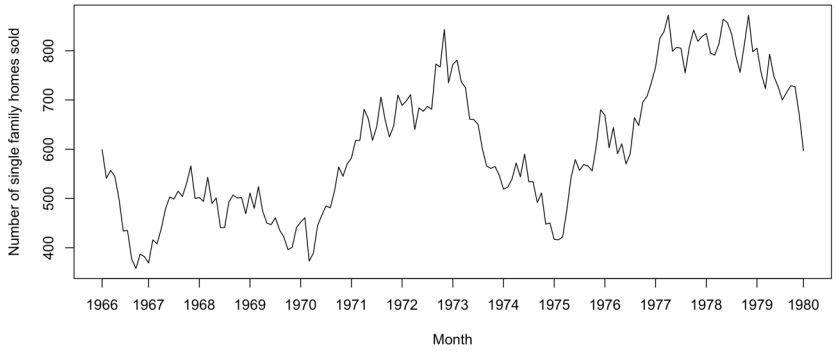

Figure 3.4: Cyclic behavior in the monthly sale of single family homes in the US

Naïve Method

For the naïve method, we set the prediction for each time unit to be the last observed value. Mathematically, the following equation describes the naive forecast.

$$y_{t+h|t} = x_t, \tag{3.1}$$

where $y_{t+h|t}$ is the prediction for the time unit $t+h$, $h\epsilon\{1, 2, ..., n\}$ is the number of units after the last observed value at t, and x_t is the observed value at time t.

Seasonal Naïve Method

The seasonal naïve method is similar to the naïve method, but it is used for seasonal data. For this method, we set the prediction for each time unit to be the same as the last observed value of the same season. For example, if we have yearly data with 12 observations per year denoting each month, then the prediction for January would be the value for January in the previous year. This method is denoted mathematically as follows.

$$y_{t+h|t} = x_{t+h-m(k+1)}, \tag{3.2}$$

where $y_{t+h|t}$ is the prediction for the time unit $t+h$, $h\epsilon\{1,2,...,n\}$ is the number of units after the observed value at t, m is the seasonal period and k is the floor function of $(h-1)/m$.

Average Method

As the name suggests, the average prediction method gives the mean of the previous observed values as the forecast for a particular time unit. The equation for the prediction is as follows.

$$y_{t+1|t} = \bar{x} = \frac{x_1 + x_2 + ... + x_t}{t}, \tag{3.3}$$

where $y_{t+1|t}$ is the prediction for the time unit $t+1$, $\{x_1, x_2, ...x_t\}$ are the observed values at time units $\{1,2,...,t\}$ respectively and x_t is the last observed value.

Simple Exponential Smoothing Method

The average method discussed previously assumes that all observations are equally important and hence, the weight allotted for each observation is the same. Whereas, the simple exponential smoothing method, which is one of the simplest smoothing methods, assumes a weight for each observation. It gives more importance to the recently observed values than the ones which are in the past. This weight is allotted exponentially and hence the name exponential smoothing. It gives the highest weight to the last observed value, which is the most recent observation, and decreases

the weights exponentially while moving further into the past. The mathematical representation is as follows.

$$y_{t+1|t} = \alpha x_t + \alpha(1-\alpha)x_{t-1} + \alpha(1-\alpha)^2 x_{t-2} + ..., \tag{3.4}$$

where $y_{t+1|t}$ is the prediction for the time unit $t+1$, $\{x_t, x_{t-1}, x_{t-2}, ...\}$ are the observed values at time units $\{t, t-1, t-2, ...\}$ respectively, x_t is the last observed value and α is the smoothing parameter with values $0 \leq \alpha \leq 1$ with $\{\alpha, \alpha(1-\alpha), \alpha(1-\alpha)^2, ...\}$ being the exponential weights.

Weighted Average Method

The weighted average method is similar to the simple exponential smoothing, but the weights are calculated differently. We have calculated the weights as done by Hao et al. [15]. Let n be the number of observations in the time series data and $c = \frac{2}{n(n+1)}$. The weights are then calculated to be $c, 2c, 3c, ..., nc$. Let $\{w_1, w_2, w_3, ..., w_n\}$ represent the weights $\{c, 2c, 3c, ..., nc\}$, then the mathematical notation for the weighted average is as follows.

$$y_t = w_1 x_{(t-s)} + w_2 x_{(t-2s)} + w_3 x_{(t-3s)} + ... + w_n x_{(t-ns)}, \tag{3.5}$$

where y_t is the prediction for the time unit t and s is the seasonal period.

Holt Winters' Seasonal Method

The Holt-Winters seasonal method comprises the forecast equation and three smoothing equations representing the trend, level and seasonal components with corresponding smoothing parameters. There are two varieties for this method that are used based on the seasonal component. The additive method is used when the seasonal variations are roughly constant through out the series and the multiplicative method is used when the seasonal component changes proportional to the level of the series.

The additive form of Holt Winters' is represented by the following equation.

$$y_{t+h|t} = l_t + hb_t + s_{t+h-m(k+1)}, \tag{3.6}$$

where $y_{t+h|t}$ is the prediction for the time unit $t+h$, l_t is the level component for the time unit t, b_t is the trend component for the time t and $s_{t+h-m(k+1)}$ is seasonal component with m as the frequency and k is the floor function of $(h-1)/m$.

The level, trend and seasonal components are represented as follows.

$$l_t = \alpha(x_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{3.7}$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{3.8}$$

$$s_t = \gamma(x_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}, \tag{3.9}$$

where α, β, γ are the smoothing parameters for the level, trend and seasonal components respectively, x_t is the observed value at time t and m is the frequency.

Similarly, the multiplicative form is represented by the following equation. The naming convention is the same as the additive method.

$$y_{T+h|T} = l_t + hb_t s_{t+h-m(k+1)} \tag{3.10}$$

The components are represented as follows.

$$l_t = \alpha(\frac{x_t}{s_{t-m}}) + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{3.11}$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{3.12}$$

$$s_t = \gamma(\frac{x_t}{l_{t-1} + b_{t-1}}) + (1-\gamma)s_{t-m} \tag{3.13}$$

3.4 Anomaly Detection

Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior [9]. Outlier is also a term that is used in the context of anomaly detection and sometimes interchangeably with the term anomaly. Anomaly detection is used in many application domains to find critical information regarding any abnormalities in the data. There are many anomaly detection techniques that have been developed for specific application domains. They may differ by the nature of the data, availability of labeled data and the type of anomaly to be detected. In the next few sections, we will go through the types of anomalies and the different anomaly detection mechanisms.

3.4.1 Types of Anomalies

The process of anomaly detection requires to identify the nature of the target anomaly which needs to be detected. Anomalies can be classified into the following three categories.

Point Anomalies

A point anomaly is an an individual data point that can be considered as anomalous with respect to the rest of data. This is the simplest type of anomaly and is the focus of majority of research on anomaly detection [9].

Contextual Anomalies

If a data instance is anomalous in a specific context, but not otherwise, then it is termed a contextual anomaly. The definition of the context must be specified and it can also be inferred from the structure of the data. Each data point is defined using the following two sets of attributes.

1. Contextual attributes: The contextual attributes are used to determine the context or the neighborhood for a data point. For example, spatial datasets have latitude and longitude as contextual attributes and time series data has time as the contextual attribute.

2. Behavioral attributes: The behavioral attributes define the noncontextual characteristics of a data point. For example, in a spatial dataset the temperature at any location is a behavioral attribute.

Collective Anomalies

If a collection of related data points is anomalous with respect to the entire data set, it is termed a collective anomaly. The individual data points in a collective anomaly may not be anomalies by themselves, but their occurrence together as a collection is anomalous [9].

3.4.2 Anomaly Detection Techniques

In a dataset, each data point will be categorized as a normal or an anomalous instance. This process is called data labeling. It can be difficult to obtain labeled data, especially for the cases where there are anomalous periods. Based on the availability of data labels, anomaly detection can be performed using any of the following techniques.

Supervised Anomaly Detection

Supervised anomaly detection is the technique of detecting anomalies with the use of a predictive model. This model is obtained by training a dataset consisting of the appropriate labels specifying

whether each data point is anomalous or not. Once the modeling is completed, new data points are fed to the model in order to classify them correctly. Usually, the anomalous instances are only a few when compared to the normal instances in the training data. This can lead to issues arising due to the imbalance in class distributions. It is also challenging to get accurate and representative labels for the anomalous class. There are several methods which have been proposed to inject artificial anomalies to obtain a training dataset [9].

Semisupervised Anomaly Detection

Semisupervised anomaly detection is the technique based on the assumption that the training data has labels only for the normal class. This makes it widely applicable than supervised techniques as the labels for the anomaly class are not required. The typical approach is to build a model using the data belonging to the normal class and identify the anomalous behavior in test data by using that model [9].

Unsupervised Anomaly Detection

Unsupervised anomaly detection is a technique where we do not require training data, thus making it most widely applicable. It is based on the assumption that the normal instances are frequent when compared to the anomalous ones. This assumption will prevent the technique from having a high false positive rate. Semisupervised techniques can be adapted to work in an unsupervised environment by providing a sample of unlabeled data during the training phase [9]. We use this category of detection techniques in the thesis to evaluate the performance of existing open source and commercial detection mechanisms.

3.4.3 Anomaly Score Computation

One way to represent anomalies is by using an anomaly score. The score gives the extent to which the data point should be considered as an anomaly. We use the anomaly score in order to label the data points as anomalous if they exceed a certain threshold. We use the method implemented by Janetzko et al. [21]. The anomaly score for a given time t is as follows.

$$a_t = \frac{|x_t - y_t|}{avg_{t' \epsilon T} |x_{t'} - y_{t'}|}, \tag{3.14}$$

where t is the time for which the score has to be calculated, y_t is the prediction obtained for the time t, x_t is the observed value for the time t, T is the set of all the time units starting from 1 until $(t-1)$

3.4.4 Anomaly Likelihood

The anomaly likelihood is built over an anomaly score function. We have used a modified version of the anomaly likelihood used by S. Ahmad et al. [1]. The anomaly likelihood gives a probabilistic measure defining how anomalous the current state is based on the scores computed in the past using a particular time series model. To compute the anomaly likelihood, we maintain a window of the last W score values. The distribution of the scores is modeled as a rolling normal distribution where the sample mean, μ_t, and variance, σ_t^2 , are continuously updated from previous scores.

$$\mu_t = \frac{\sum_{i=0}^{i=W-1} a_{t-i}}{W} \tag{3.15}$$

$$\sigma_t^2 = \frac{\sum_{i=0}^{i=W-1} (a_{t-i} - \mu_t)^2}{W-1} \tag{3.16}$$

A recent short term average of the anomaly scores is then computed as follows.

$$\widetilde{\mu_t} = \frac{\sum_{i=0}^{i=W'-1} a_{t-i}}{W'}, \tag{3.17}$$

where a_t is the anomaly score at time t and W' is the window for the short term moving average and $W' << W$.

This is followed by the application of a threshold to the Gaussian tail probability (Q-function) to decide whether or not to declare an anomaly. The anomaly likelihood is defined as the complement of the tail probability as follows.

$$L_t = 1 - Q(\frac{\widetilde{\mu_t} - \mu_t}{\sigma_t}) \tag{3.18}$$

The threshold for L_t is based on the user-defined parameter ϵ and is represented as follows.

$$anomaly\ detected_t \equiv L_t \geq 1 - \epsilon \tag{3.19}$$

There is an inherent upper limit on the number of anomalies detected and a corresponding upper bound on the number of false positives. With ϵ very close to 0, it would be unlikely to get alerts with probability much higher than ϵ. It is found that $\epsilon = 10^{-5}$ works well across a large range of domains and the user does not normally need to specify a domain-dependent threshold. However, we use $\epsilon = 10^{-3}$ in the thesis.

3.4.5 Anomaly Detection Packages

HTM Studio

HTM Studio is a tool developed by Numenta and is based on an online sequence memory algorithm called Hierarchical Temporal Memory (HTM). Hierarchical Temporal Memory is a biologically inspired machine intelligence technology that mimics the architecture and processes of the neocortex. Anomaly detection with HTM is a state-of-the-art, online and unsupervised method [1]. HTM networks continuously learn and model the spatiotemporal characteristics of their inputs. The output of an HTM system goes through two post-processing steps, which are the computation of the prediction error and the computation of an anomaly likelihood measure. Based on the user-defined threshold, each data point will be classified as an anomaly based on the degree of the anomalous behavior. The three classes used by HTM Studio are low, medium and high.

Arundo's ADTK

Anomaly Detection Toolkit (ADTK) is a Python package for unsupervised/rule-based time series anomaly detection. They provide a variety of detection algorithms (detectors), feature engineering methods (transformers), and ensemble methods (aggregators) that can be chosen and combined by the user for developing an anomaly detection model. Our main interest is in the detectors and ADTK provides fourteen detectors from which we have used the OutlierDetector. OutlierDetector performs multivariate time-independent outlier detection and identifies outliers as anomalies. The multivariate outlier detection algorithm could be those in Python Scikit-learn library or other packages following same API. The algorithm used here is the LocalOutlierFactor from the Python Scikit-learn library "sklearn.neighbors" which is an unsupervised algorithm. It measures the local deviation of density of a given sample with respect to its neighbors. The anomaly score depends on how isolated the object is with respect to the surrounding neighborhood and the locality is given by k-nearest neighbors. By comparing the local density of a sample to the local densities of its neighbors, the samples which have a substantially lower density than their neighbors are considered outliers.

Twitter's Anomaly Detection

AnomalyDetection is an open-source R package that automatically detects anomalies in big data [43]. The primary algorithm, Seasonal Hybrid ESD (S-H-ESD), builds upon the Generalized ESD (Extreme Studentized Deviate) test for detecting anomalies. It uses a variant of the Seasonal and Trend decomposition using Loess (STL) and then applies ESD to detect the anomalies. S-H-ESD can be used to detect both global and local anomalies. Global anomalies are the data points which occur above or below the seasonal pattern and thus be detected as it is does not represent the underlying seasonality or trend. Local anomalies are the data points which are masked by the seasonal patterns but are anomalous in their neighborhood. The detection is achieved by using time series decomposition and robust statistical metrics. The decomposition separates the seasonal component and then removes the median data. In addition, for long time series such as 6 months of minutely data, the algorithm uses piecewise approximation.

Anomalize

The Anomalize package enables a tidy workflow for detecting anomalies in data. Seasonal and Trend decomposition using Loess (STL) is used in this thesis for decomposition. It separates the season and trend components from the original values leaving the remainder for anomaly detection. For the anomaly detection, we have used the IQR method which uses an inner quartile range of 25% and 75% to establish a baseline distribution around the median[3]. The alpha value that is given to the detection function controls the ease of tagging a data point as an anomaly and it represents the width of the normal observations range. By increasing the alpha value, it increases the chance of a data point to be flagged as an anomaly. We have used the default alpha value, which is 0.05.

Facebook's Prophet

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. It is robust to missing data and shifts in the trend, and typically handles outliers well[4]. It uses a decomposable time series model with three main model components: trend, seasonality, and holidays. It is similar to the GAM (Generalized Additive Model) formulation, which has the advantage that it decomposes easily and accommodates new components as necessary, for instance when a new source of seasonality is identified [41]. This package is available in R and Python and we have used the Python package.

[3]https://cran.r-project.org/web/packages/anomalize/anomalize.pdf

[4]https://facebook.github.io/prophet/

Chapter 4

Attack Generation

4.1 Data Collection

The data was collected from the UMass Trace Repository[1], which is being used to optimize energy consumption in homes under a project named "Smart*".[2]

There are many datasets available out of which, we chose an apartment dataset that contains data for 114 single family apartments. This fits our requirement as we are trying to mimic the attacks at the grid level. From all the available datasets, this was the largest in terms of the number of apartments and hence, can be used to represent a subset of the power grid.

The dataset consists of daily consumption for each apartment for the years 2014, 2015 and 2016. The consumption data for the years 2014 and 2015 consists of missing values and the readings are taken at irregular intervals of one minute and fifteen minutes. Hence, we decided to go with the data recorded from the year 2016 which consists of minute-level consumption readings from January until mid December. For each apartment, a CSV file is provided which consists of two columns. The first column is the timestamp of when the reading was taken and the second column is the reading in kiloWatt(kW). The timestamp consists of the date and the time in 24 hour format. All these apartments belong to the same unknown area in West Massachusetts, United States of America.

[1] http://traces.cs.umass.edu/index.php/Smart/Smart

[2] http://lass.cs.umass.edu/projects/smart/

4.2 Data Cleaning

4.2.1 Missing Values

As there are 114 different CSV files, we had to combine them to get it in to a single CSV file. We performed a merge across all the different apartment files. The data post join consisted of 503,610 rows with the first column representing the time ranging from 2016-01-01 00:00:00 to 2016-12-15 18:29:00 and the remaining 114 columns representing all the apartment data with one column per apartment. This shows that we have missing values for the remainder of the year from 2016-12-15 18:30:00. As the daylight saving started on March 13, 2016 at 2:00:00 AM, we have missing values between 2016-03-13 2:00:00 and 2016-03-13 2:59:00. Also, as it ended on November 6, 2016 at 2:00:00, we have duplicate values from 2016-11-06 2:00:00 and 2016-11-06 2:59:00. We have not performed any imputation or estimation for the missing values.

4.2.2 Removing Duplicates

As discussed in the previous section, we detected duplicate rows which were due to the end of the daylight saving. We have dropped every first observation of the duplicate row and there were no other duplicate rows found in the dataset.

Timestamp	Apartment 1	Apartment 2	...	Apartment 114	Grid level
2016-01-01 00:00:00	0.68415000	0.7006667		0.85898333	176.3487
2016-01-01 00:01:00	0.68295000	0.7006833		0.85840000	158.4339
2016-01-01 00:02:00	0.68298333	0.7022333		0.85848333	162.1322
2016-01-01 00:03:00	0.68238333	0.7028000		0.85883333	154.2100
2016-01-01 00:04:00	0.68228333	0.7050667		0.55655000	156.9728
2016-01-01 00:05:00	0.68201667	0.7031167		0.47450000	165.4039
2016-01-01 00:06:00	0.17223333	0.6900833		0.43575000	175.4320
2016-01-01 00:07:00	0.16350000	0.5988833		0.37150000	179.2886
2016-01-01 00:08:00	0.16313333	0.5988000		0.31260000	182.1628
2016-01-01 00:09:00	0.16266667	0.5987167		0.18176667	187.8987

Table 4.1: Sample data showing the time stamp and consumption in kilowatt

4.3 Consumption Overview

We focus on the grid level analysis of the power consumption. We obtained the grid level data by adding the consumption values of all the 114 apartments. For every time unit, there will be an

aggregate value representing the grid level consumption at that point as shown in table 4.1. The grid level aggregate value is calculated by the following mathematical formula.

$$g_t = \sum_{i=1}^{114} x_{i,t}, \tag{4.1}$$

where g_t is the grid level consumption at time t, i is the apartment number and $x_{i,t}$ is the consumption of the i^{th} apartment at time t.

Figure 4.1 shows the consumption pattern with the demand recorded every six hours. It can be observed that the consumption is high for the five months with a decreasing trend. The consumption is low during the Summer and the beginning of Fall followed by an increasing trend while transitioning to Winter. Table 4.2 shows the average consumption with a seasonal breakdown. It can be observed that the consumption is highest during Winter followed by Spring, Fall and Summer.

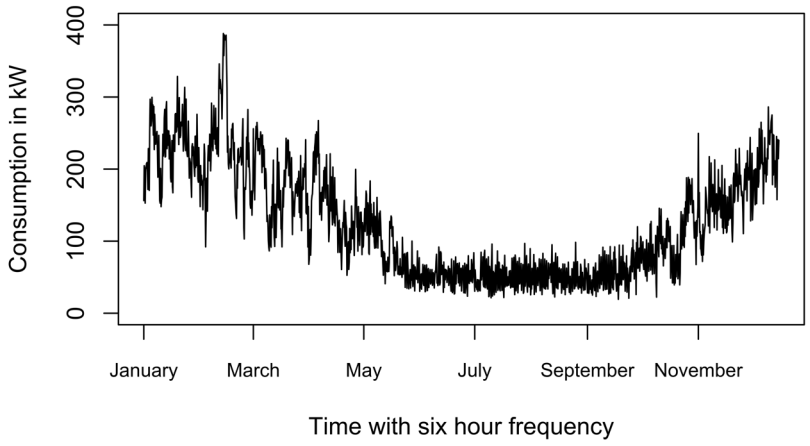

Figure 4.1: Power consumption with six hour frequency

Season	Average Consumption
Winter	227.88 kW
Spring	134.90 kW
Fall	102.29 kW
Summer	50.85 kW

Table 4.2: Average consumption by season

4.4 Common Terms

In this section, we will discuss some common terms used in our work.

4.4.1 Threshold

A threshold is defined as a value above which a point will be flagged as an anomaly. This is applied to an anomaly score or an anomaly likelihood score.

4.4.2 Adder

An adder is the consumption amount in kilowatts that is added to an existing consumption value at a time t. The adder represents the number of kilowatts that an attacker is able to increase for a particular time unit. For this thesis, we only use positive adders which means that we perform only a demand increase.

4.4.3 Attack Profile

A transformation is performed on the original dataset by adding adder values only to the two weeks of attack duration and the resulting dataset is called an attack profile. Attack profiles are created by adding adders to selected months of the year. The creation of an attack profile is represented mathematically as follows.

$$\begin{bmatrix} g_1 \\ g_2 \\ . \\ . \\ g_t \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \\ . \\ . \\ d_t \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ . \\ . \\ c_t \end{bmatrix}, \tag{4.2}$$

where g_t is the grid level consumption at time t obtained from the consumption dataset, d_t is the adder value added at time t and c_t is the resultant demand at time t, that is obtained by adding the adder to the consumption value. This resultant matrix is called an attack profile. It should be noted that the attacker might add demand values at only some time periods. We consider the adders to be zero when no demand increase is performed by the attacker.

4.4.4 Gain

Gain is the amount in kilowatt that the attacker was able to attain by increasing the demand values through out the attack period. The gain at any point in time is calculated by adding all the adders added previously. It is represented mathematically as follows.

$$gain_t = \sum_{i=1}^{t} d_i, \tag{4.3}$$

where $gain_t$ is the gain in kilowatt at minute t, i ranges from the first minute of attack until the current time t in minutes and d_i is the adder added at minute i.

4.4.5 Anomaly Detection System

An anomaly detection system consists of a prediction method to predict the consumption values followed by a scoring mechanism that uses anomaly score or likelihood score and a thresholding mechanism to flag anomalies. It can also be a commercial application that we evaluate the attack performance against.

4.5 Assumptions

Before we continue to performing the attacks, we will provide a few assumptions based on which the attacks were carried out.

1. The original dataset is assumed to have no anomalies.
2. The attacker has access to various IoT devices in all the 114 apartments.
3. The amount of demand per apartment can be of any value.
4. The attacker is aware of the anomaly detection technique used by the power grid operator to monitor the consumption.
5. The attacker is aware of all the threshold values used for the anomaly detection process.

4.6 Threshold Calculation

For the anomaly detection techniques we used, viz., anomaly score and anomaly likelihood, we had to decide on the thresholds. The anomaly score threshold is defined as the value beyond which the data point will be flagged as an anomaly. In order to generate a threshold, we should fully understand the consumption of the apartments. The following are the steps carried out to generate the thresholds.

1. First, we categorize the various thresholds based on how often we want to change the threshold value. We have 10 types of thresholds which change yearly, half yearly, quarterly, monthly, weekly, daily, every twelve hours, every six hours, every one hour and every fifteen minutes. Each type of threshold defines the number of threshold values we need to calculate for the entire data.
2. In order to find the values for each type of threshold, we find the 99^{th} percentile of the scores within the calculation range. The calculation range is the period for which the threshold is calculated. For example, for the yearly threshold, the calculation range will be the entire year which is in our case, the entire dataset and for half yearly threshold, we will have two calculation ranges which are the first half of the year and the second of the year. Refer to table 4.3 for a detailed breakdown.

3. For each calculation range, the 99^{th} percentile is stored as the threshold for that range. The score values for a calculation range will be flagged as an anomaly if they are above the threshold for that range. For the anomaly likelihood technique, we set the threshold to be 0.999 with the value of $\epsilon = 10^{-3}$ in anomaly likelihood calculation.

Threshold Type	Number of Thresholds
Yearly	1
Half yearly	2
Quarterly	4
Monthly	12
Weekly	50
Daily	350
Twelve hourly	700
Six hourly	1399
Hourly	8394
Quarter hourly	33574

Table 4.3: Number of thresholds by type[3]

4.7 Threshold Exploration

After generating the thresholds as described in the previous section, we explored these thresholds to understand the behavior of the anomaly detection technique using weighted average prediction and anomaly score along with each type of the listed thresholds. Figure 4.2 provides a summary of the false positive rates for every fifteen minute rolling window while using a particular threshold type.

The anomaly scores were calculated for the original dataset without modifying any of the consumption values. With the assumption that the dataset does not have any anomalous values, we expect it to have less false positives. It can be observed that the more generic the threshold is the less frequent we see false positive rates greater than zero. For example, the yearly threshold has a single value as the threshold for the entire year. We can see that initially, there are some instances which are detected as anomalies but, almost none of them are flagged as an anomaly for the rest of the year. When we look at the half yearly threshold, the number of false positives in the beginning

[3]The number of threshold values is calculated based on the size of our dataset which is 503610.

of the year has decreased when compared to the yearly threshold but, we see some of them being flagged in the second half of the year.

The more personalized the threshold becomes, we see more fifteen minute windows in that time period being detected. In other words, as the threshold is calculated for smaller windows, the number of anomalies decrease for that window but, we see them spread out across the neighboring windows, thus making more number of fifteen minute windows have at least one anomaly.

Ideally, there will be a tolerance level for the number of anomalies beyond which the system will alarm the grid operator for immediate attention. With a generic threshold, it becomes easy to increase the demand by some amount such that it is less than the tolerance level. For example, if we have an FPR tolerance level of 20%, we can see that for the yearly threshold, most of the fifteen minute windows will have no anomalies triggered. All those time periods are vulnerable to an attack as an adversary can take advantage of the tolerance level and increase the demand such that there are false positives, but the overall FPR for that window is less than 20%. However, as the thresholds are calculated for smaller windows, it becomes increasingly difficult to find such windows where the FPR is 0% thus, making it difficult to perform the attack.

This brings us to the idea of dynamic thresholding where the threshold changes for each point in time and is calculated dynamically based on the previous observations. We explore dynamic thresholding in this thesis after evaluating the performance of all the different threshold types discussed previously. Detailed analysis using dynamic thresholding approach is discussed in chapter 6.

4.8 Experiment Setup

In this section, we will give a summary of the setup and all the steps involved in generating and evaluating the attacks. Figure 4.3 gives a brief flow of the entire process.

First, we generate the attack profiles for each threshold type. The attack profile is generated in such a way that it is not detected while using an anomaly detection system which consists of weighted average as the prediction method and anomaly score to compute the severity of an anomaly. This will result in 96 different attack profiles as described in subsection 4.9.2.

This brings us to the second step where we select 10 attack profiles based on false positive rate and the gain achieved during the attack period. Detailed description can be found in section 4.10. Then, we evaluate the performance of these 10 profiles by using various anomaly detection systems

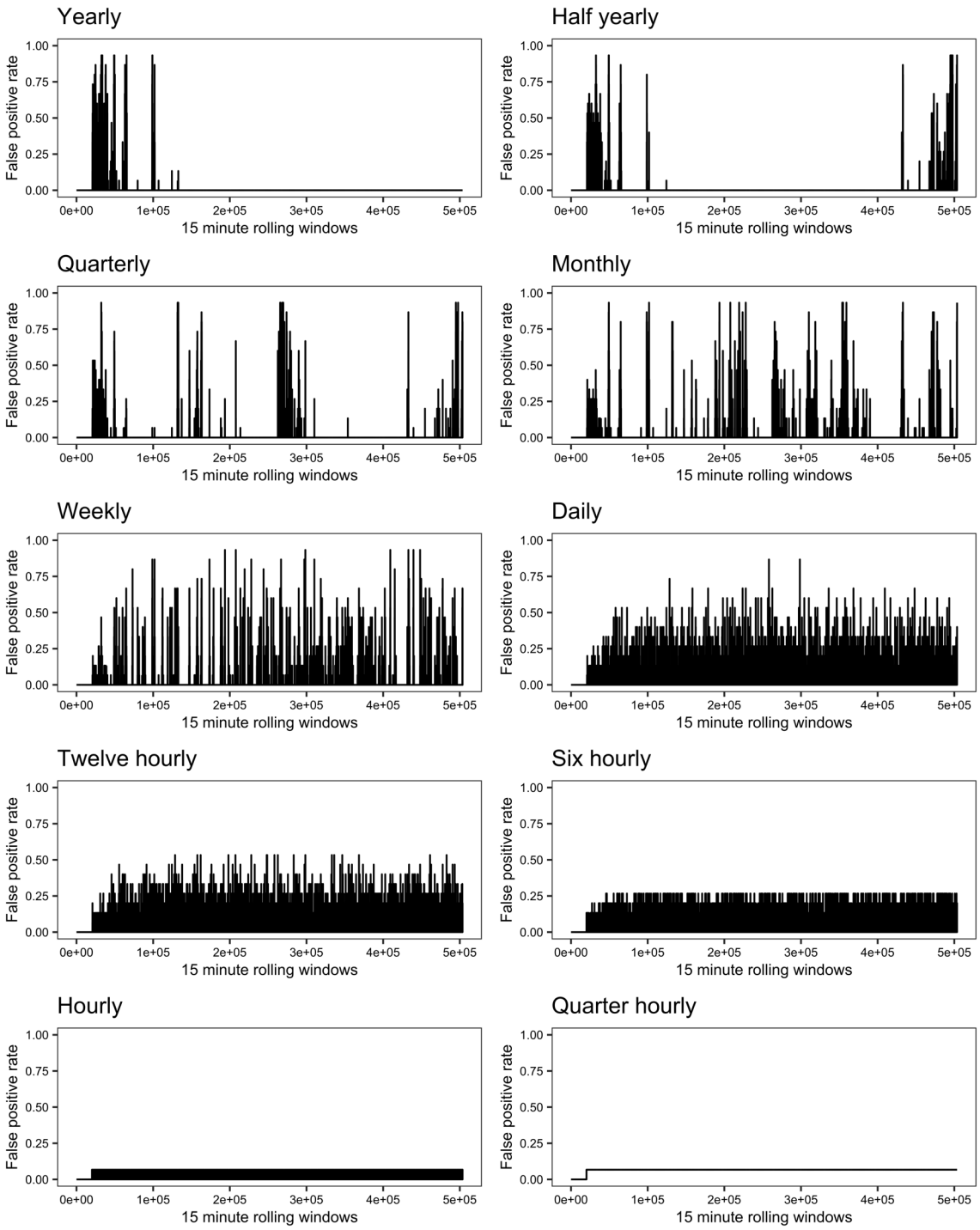

Figure 4.2: Threshold exploration

which were described in chapter 3. Table 4.4 gives the list of the anomaly detection systems we use for our evaluation.

We used R version 3.6.1 for implementing algorithms to generate the attacks and calculate evaluation metrics. Arundo's ADTK and Facebook's Prophet are the commercial methods available as Python libraries for which we used Python version 3.7.4 for running those methods against our attack profiles.

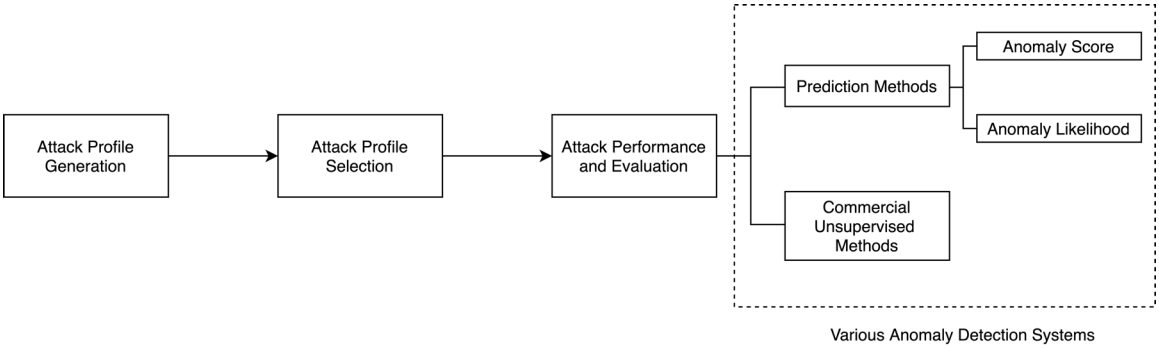

Figure 4.3: Overview of the experiment setup

Prediction/ Library	Detection technique
Naïve	Anomaly score and likelihood
Seasonal naïve	Anomaly score and likelihood
Average	Anomaly score and likelihood
Simple exponential smoothing	Anomaly score and likelihood
Weighted average	Anomaly score and likelihood
Holt Winters'	Anomaly score and likelihood
HTM Studio	Unsupervised
Arundo's ADTK	Unsupervised
Twitter's Anomaly Detection	Unsupervised
Anomalize	Unsupervised
80% confidence interval	Unsupervised
Total number of techniques	17

Table 4.4: List of anomaly detection systems used for performance evaluation

4.9 Attack Profile Generation

4.9.1 Overview

The attack profiles are generated by performing an obscure progressive attack on the power grid by controlling the overall demand of the grid. We perform one attack per threshold type and the anomaly detection system used is the weighted average prediction with an anomaly score method. These attacks will not be detected in this system while using the different thresholding techniques. All the attacks are performed for two weeks by adding some amount of adder with a minimum value of 0. This adder value is determined by reverse engineering the anomaly score computation.

4.9.2 Attack Generation Process

The attack generation process requires to precompute the adder value which will be added to each observation. This adder represents the amount of wattage contributed from the compromised IoT devices. In this subsection, we will discuss how this adder is computed and propose the algorithm for generating attack profiles.

Adder Computation

The adder specifies the value of the increase in demand to be carried out at the grid level. The adder is computed in such a way that the current anomaly detection system cannot detect the increase in demand. This is done by reverse-engineering the anomaly score formula and by giving the threshold value as the score. The adder is computed with the following formula.

$$x'_t = (a_t \ avg_{t'\epsilon T}|x_{t'} - y_{t'}|) + y_t, \tag{4.4}$$

$$d_t = x'_t - x_t, \tag{4.5}$$

where x_t is the existing consumption value, x'_t is the resultant demand to be created for time t which also consists of the adder, a_t is the anomaly score which is equal to the threshold at t, y_t is the

prediction of the consumption for time t and d_t is the adder for time t which is calculated by finding the difference between the resultant demand and the already existing consumption value.

At any point during the attack, the adversary will have access to the current threshold value and the prediction to calculate the adder at that time. The demand is increased by the adder amount. If the adder is negative or zero, the demand is not manipulated by the attacker. Figure 4.4 shows the summary of the adder calculation process.

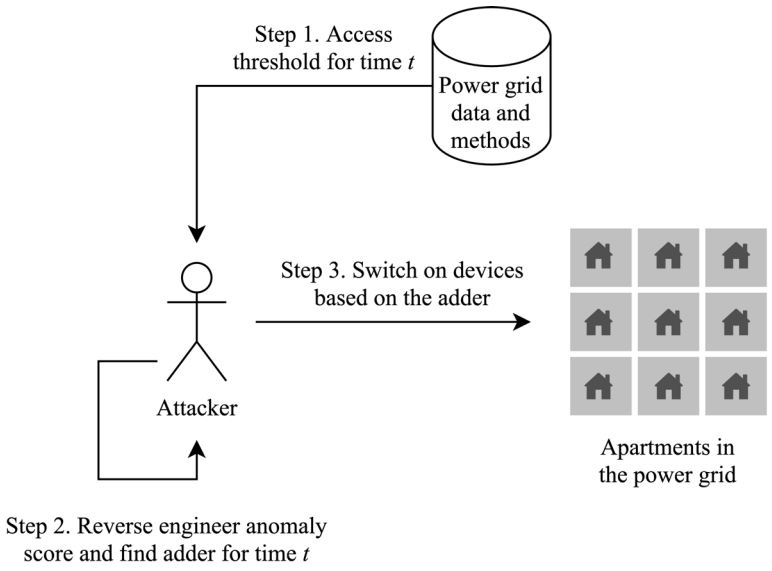

Figure 4.4: Adder calculation process

Attack Generation Algorithm

Algorithm 1 is used to generate the attack profiles for each type of threshold and the selected attack month. This algorithm takes the consumption dataset, pre-calculated predictions for the dataset, respective thresholds for each point in time in the consumption dataset, attack start and end times as input parameters. First, we create a copy of the original dataset which will be modified and returned as an attack profile. We go through each observation from the start time to the end time of the attack, get the resultant demand to be created for that observation by using equation 4.4 and find the prediction for the next observation based on all consumption values from the beginning until the current observation. As seen in the algorithm, FindPrediction is the method for finding the prediction of a data point using one of the six prediction methods we use. Once the end time is

reached, we find the predictions for the rest of the data after the end time of the attack and update the predictions. Lastly, the generated attack profile is returned as the output.

Algorithm 1 Attack Profile Generation

▷ D: Power consumption data, P: Predictions, R: Thresholds, t_1, t_n: Attack start & end times
1: **function** GENERATEATTACKPROFILE(D, P, R, t_1, t_n)
2: $AP \leftarrow D$
3: **for** $t \leftarrow t_1$ to t_n **do**
4: $AP[t] \leftarrow (R[t]\ avg_{t' \epsilon [1,t-1]} |AP[t'] - P[t']|) + P[t]$
5: $P[t+1] \leftarrow$ FINDPREDICTION($AP[1:t]$)
6: **end for**
7: $P[(t_n + 1) : size(P)] \leftarrow$ FINDPREDICTION(AP))
8: **return** AP
9: **end function**

Number of Attack Profiles

The attack profiles are created based on the the threshold type of the anomaly detection system and the time of the attack i.e. which quarter or month of the year the attack was performed. The attack duration is same for all profiles, which is of two weeks. For yearly, half yearly and quarterly thresholds, we create an attack profile for each quarter and for the others, an attack profile for each month is created. In total, we will be having 96 attack profiles created for each threshold type and the month of the attack. Table 4.5 has the breakdown of the count of profiles by threshold type.

Threshold Type	Number of Attack Profiles
Yearly	4
Half yearly	4
Quarterly	4
Monthly	12
Weekly	12
Daily	12
Twelve hours	12
Six hours	12
One hour	12
Fifteen minutes	12
Total	96

Table 4.5: Number of attack profiles by threshold type

4.10 Attack Selection

After generating the 96 attack profiles, we selected 10 profiles based on the gain in kW achieved during the attack period. The gain is defined as the total wattage the attacker could increase during the two weeks of the attack. First, we sort all the 96 profiles in decreasing order of the gain values. If the attacks were carried out during the first quarter or the the first two months, we ignore them to provide a buffer time to stabilize the anomaly scores. Additionally, these time periods are vulnerable to aggressive attacks where the adversary can reach a gain of up to 313 million kilowatt as there are less observations to normalize the anomaly score. Hence, we consider these attacks to be invalid.

Starting from the first valid attack with the highest gain, we choose every eighth attack going downward in descending order of the gain values. In an attempt to include all threshold types in this selection, we choose one attack per threshold type. If we hit a profile whose threshold type is already considered, we look at its neighbors on either side to search for a profile with a different threshold type. This technique is used to get attack profiles that are distributed between the minimum and maximum gain values.

Table 4.6 shows the list of selected attacks.

Threshold Type	Attack Month	Wattage Gain
Yearly	July	9,426 MW
Half yearly	July	5,225 MW
Quarterly	July	3,207 MW
Monthly	June	4,569 MW
Weekly	September	2,128 MW
Daily	December	1,235 MW
Twelve hours	May	4,063 MW
Six hours	April	2,742 MW
One hour	September	1,903 MW
Fifteen minutes	May	3,501 MW

Table 4.6: List of selected attack profiles

4.10.1 Time for MadIoT attack

As we performed the attacks obscurely, we have the limitation of not increasing the wattage all at once to reflect the effects of the attack within seconds. This could have been possible if we had the power consumption for each second. However, we analyzed the time it took to reach the gains required to carry out each of the variations of MadIoT attack. Figure 4.5 shows the time to

reach the gains required for each MadIoT attack. The gains were calculated by finding the amount of wattage needed from 114 apartments each contributing one IoT device. Table 4.7 shows the gain values for each attack.

Attack effects	Wattage from 114 IoT devices
Frequency Instability causing generators tripping	380kW
Frequency Instability during a black start	570kW
Cascading failure	11,400kW
Failure of tie lines	7,600kW
Increase in operating costs	2,280kW

Table 4.7: Wattage requirement from 114 devices for different MadIoT attacks

Except for daily threshold, we were able to reach the required gains for all attack scenarios within 2.5 hours. The smallest gain value which was 380kW was reached within 5 minutes and the gains 570kW and 2,280kW were attained within 10 minutes. The time to reach the larger gain values varies by threshold type, with yearly threshold achieving it the fastest within 27 minutes. The daily threshold seemed to be delaying the time to reach the required amounts of gain making it the best performing threshold to control aggressive attacks.

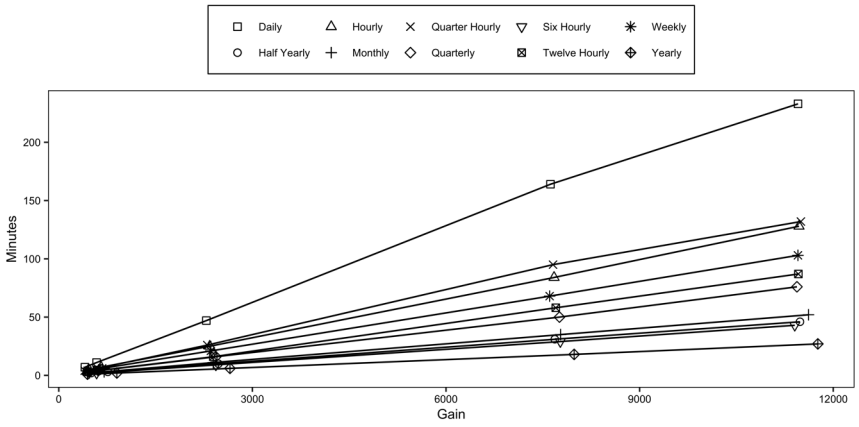

Figure 4.5: Time to reach gains required for each MadIoT attack

4.11 Chapter Summary

In this chapter, we discussed about the real-world power consumption dataset and how we obtained our final working dataset that we use for our thesis. We also discussed some common terms we use in the context of anomaly detection in power consumption. We described about the

threshold calculation process and some insights on the behavior of various static thresholds. We discussed how the attack profiles are generated and how the selection of ten attack profiles from 96 of them is done. With the ten selected attack profiles, we observe if the required gains for MadIoT attacks are attained and how long it takes to reach the gains. In the next chapter, we will evaluate the performance of these ten attack profiles against different anomaly detection systems.

Chapter 5

Attack Performance Evaluation

5.1 Overview

In this chapter, we will evaluate the attack performance using each anomaly detection system. We categorized the anomaly detection systems into anomaly detection using static thresholds and anomaly detection using commercial methods. We will evaluate each technique individually and also provide a comparision of the performance between them.

Table 5.1 shows the list of aliases for different prediction and thresholding methods which we have used in the plots.

Name	Aliases
Naïve	naive
Seasonal naïve	seasonal_naive
Average	average
Simple Exponential Smoothing	ses
Holt Winters'	holt_winters
Yearly threshold	yearly
Half yearly threshold	half_yearly
Quarterly threshold	quarterly
Monthly threshold	monthly
Weekly threshold	weekly
Daily threshold	daily
Twelve hourly threshold	12_hourly
Six hourly threshold	6_hourly
Hourly	hourly
Quarter hourly	quarter_hourly

Table 5.1: List of aliases used in plots

5.2 Anomaly Detection using Static Thresholds

In chapter 4, we selected ten attack profiles where the attacks were performed in different months and each profile assumes a unique threshold type. These threshold types are static as we have fixed the threshold values in advance and we do not change them regardless of the consumption. We evaluate the performance of two variations of detection:

1. The anomaly score is calculated for each point in time and the threshold is the deciding factor of whether to flag a point as an anomaly.

2. The anomaly likelihood is calculated for each point using the anomaly scores and the threshold is the deciding factor in this technique as well.

In the following subsections, we will walk through the attack performance using each of the above listed techniques.

5.2.1 Performance of Anomaly Score

The anomaly score is calculated by finding the difference between the predicted and actual usage values followed by normalizing it. We used five prediction methods and examined the performance in each combination of the prediction and anomaly score methods, that together make an anomaly detection system. As we used ten thresholding techniques, we also evaluate the performance of the thresholds individually.

Performance of each Prediction

Each prediction method was given the ten attack profiles to perform the detection using the anomaly score technique. Then, we use the thresholding mechanism that the profile represents to flag a point as an anomaly. First, we find the overall performance of the prediction method in detecting true positives across all the ten attack profiles. Figure 5.1 gives the percentage of true positive rate for each prediction method across all attacks. The true positive rate (TPR) measures the proportion of positives that are correctly identified. The following equation was used to calculate this metric for each prediction.

$$TPR_p = \frac{\sum_{i=1}^{i=10} TP_{(p,i)}}{\sum_{i=1}^{i=10} (TP_{(p,i)} + FN_{(p,i)})}, \tag{5.1}$$

where TPR_p is the true positive rate for prediction p, $p \in$ {naïve, seasonal_naïve, average, ses, holt_winters} represents each prediction method, i represents each attack profile, $TP_{(p,i)}$ is the true positives for the prediction p and the i^{th} attack profile, FN_i is the false negatives for the prediction p and the i^{th} attack profile. A true positive is an outcome where the model correctly predicts the positive class. Whereas, a false negative is an outcome where the model incorrectly predicts the negative class.

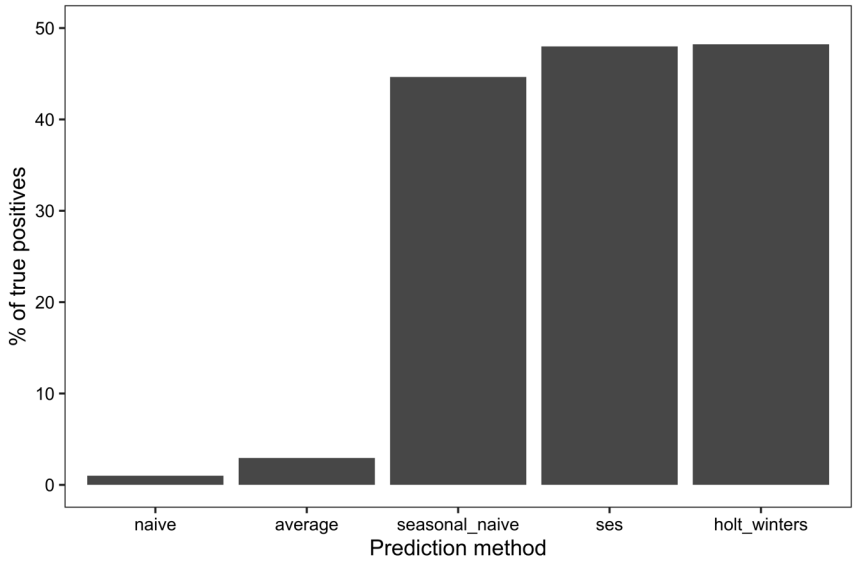

Figure 5.1: Overall true positive rate for each prediction method

It can be observed from the naïve prediction category, naïve and average prediction methods have detected less than 5% of the anomalies. Seasonal naïve, simple exponential smoothing and Holt Winters' prediction methods were able to detect $45\% - 50\%$ of the anomalies. This shows that when we have a seasonal element associated to the prediction method, the predictions are much accurate, thus contributing to a high anomaly score when there is an anomaly.

The false positive rate measures the proportion of negatives that were incorrectly classified as a positive. The false positive rate is calculated using the following formula.

$$FPR_p = \frac{\sum_{i=1}^{i=10} FP_{(p,i)}}{\sum_{i=1}^{i=10}(FP_{(p,i)} + TN_{(p,i)})}, \tag{5.2}$$

where FPR_p is the false positive rate for prediction p, $p \in$ {naïve, seasonal_naïve, average, ses, holt_winters} represents each prediction method, i represents each attack profile, $FP_{(p,i)}$ is the false positives for the prediction p and the i^{th} attack profile, TN_i is the true negatives for the prediction p and the i^{th} attack profile. A false positive is an outcome where the model incorrectly predicts the positive class. A true negative is the outcome where the model correctly predicts the negative class.

The false positive rates are ranging from 1% to 15%. As these values are calculated for the entire year, which consists of more than half a million rows, these values are still low. When we look at the false positive rates, we can see that naïve and average prediction methods have high false positive rates compared to others, again due to the low prediction accuracy. Though seasonal naïve and Holt Winters' performed well in detecting anomalies, the false positive rate is between 5% to 10%. Simple exponential smoothing has the least false positive rate of about 2%, thus this method with anomaly score would be the best option when looking for a detection technique with least false alerts.

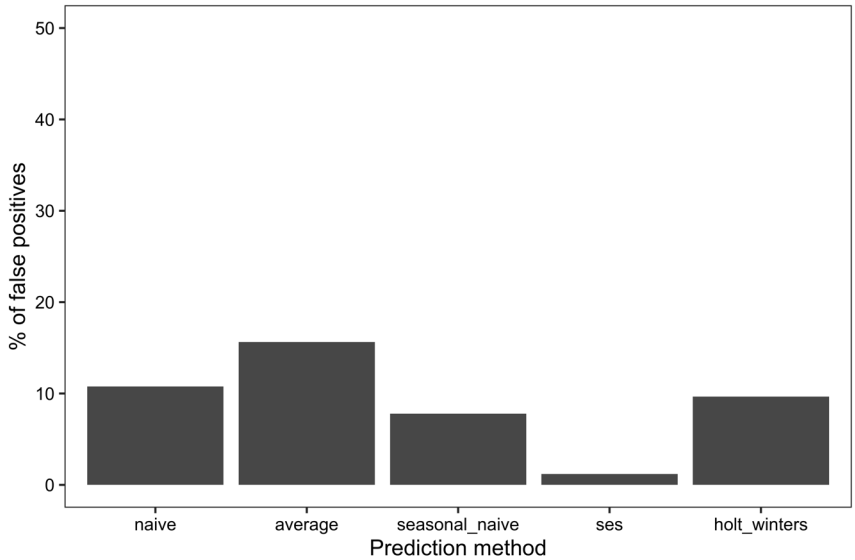

Figure 5.2: Overall false positive rate for each prediction method

Types of Anomalies Detected

As we performed the attack for two weeks with some points where we did not add any adder, we expect all observations where some amount of wattage was added by the adversary to be detected. There is some increase in wattage at most times during the attack due to which we categorize these type of anomalies to be collective. We will provide a brief analysis of the types of anomalies detected by each prediction method.

We could see from the true positive rate results that naïve prediction method performed poorly. Figure 5.3 shows the anomaly scores during the fourteen attack days when naïve prediction method is used with monthly thresholding mechanism. The solid dark like represents the threshold and is constant throughout the attack period as all the data points belong to the same month. Almost all anomaly scores are below the threshold line, resulting in a low true positive rate. However, there are some data points which have been detected which lie above the threshold line. As the naïve prediction method stores the last observed value as the prediction for the current time, it considers most of the anomalous wattage values to be normal. For example, minute one's observed value is stored as the prediction for minute two and minute two's observed value as prediction for minute three and so on. Hence, the detection using naïve prediction method performed poorly in detecting collective anomalies.

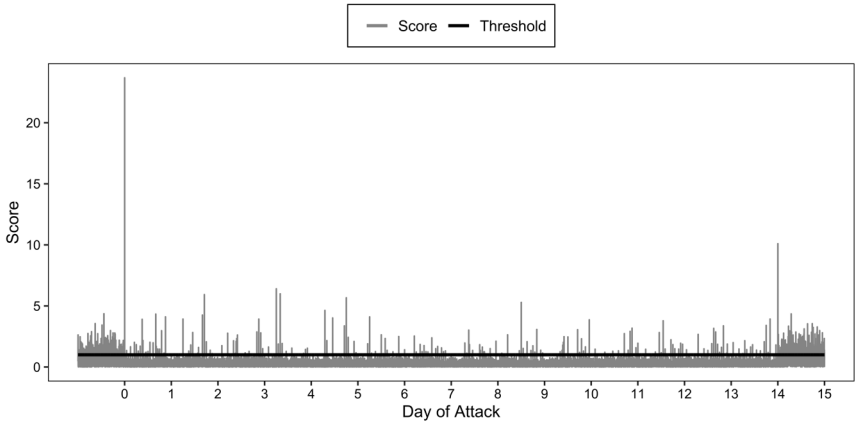

Figure 5.3: Anomaly scores when naïve prediction method is used with monthly threshold

The average prediction method also fails to detect collective anomalies as seen in figure 5.4. In this case, the average prediction method gives equal weights to all points in time considered for the prediction. As there may be highs and lows in the points considered for the prediction,

the average settles at a value close to the attack wattage, thus considering the attack points to be normal. Figure 5.4 shows that before the attack day i.e. day 0, some points have been flagged as anomalies, conveying that there might have been an increase in consumption genuinely during those minutes. This resulted in the detection system being in favor of the adversary by considering the attack periods to be normal as an increase was observed previously.

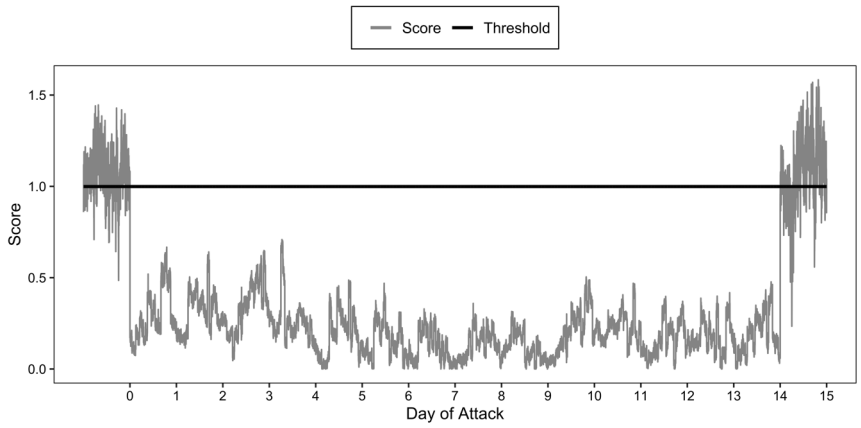

Figure 5.4: Anomaly scores when average prediction method is used with monthly threshold

As the name suggests, the seasonal naïve prediction is performed using a weekly seasonality. The seasonality is reflected in figure 5.5, where the first week of attack has high anomaly scores whereas the second week of attack has low anomaly scores. Although 50% of the anomalies are detected, the rest are considered to be normal due to this prediction method's inherent seasonal nature. The prediction values for the second week of attack are the observed values during the first week of attack, due to which the anomaly scores of the second week are less than 1. This is the case in all thresholding mechanisms due to the weekly seasonal nature. Being a simple prediction method, it was able to detect the first week of attack collectively, thus making it a good option for detecting anomalies within the first seasonal cycle.

Simple exponential smoothing also exhibits weekly seasonality (refer to figure 5.6) with the predictions calculated in a weighted average form with exponential weights. As seen in figure 5.9, simple exponential smoothing performed well in most of the thresholding mechanisms detecting 100% of the anomalies in some of them. It is interesting to see that with daily, twelve hourly, six hourly and quarter hourly thresholds, it has performed very poorly in detection. This brings us to a unique case where the true positive rate is low because of the month of attack. The attacks which

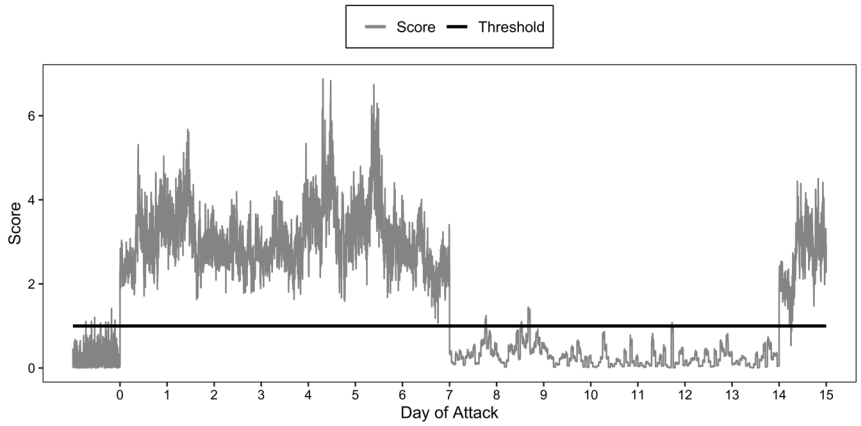

Figure 5.5: Anomaly scores when seasonal naïve prediction method is used with monthly threshold

were performed using twelve hourly, six hourly and quarter hourly were performed in the months of May, September and December. These months denote the season transition periods which simple exponential smoothing was unable to handle. These months also represent the possible times of performing a successful obscure attack, if this prediction method is used in the anomaly detection system.

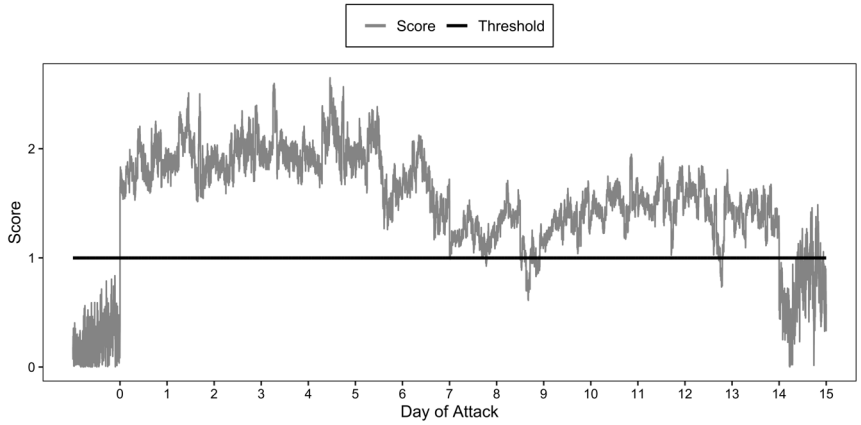

Figure 5.6: Anomaly scores when simple exponential smoothing prediction method is used with monthly threshold

Holt Winters' prediction method is one of the only methods which could consistently detect about 50% of the anomalies across all thresholds, except for the daily threshold. From figure 5.7, it is clear that Holt Winters' also shows the weekly seasonality in its anomaly scores. Due to this characteristic, it was able to successfully detect half of the total anomalies which existed in the first

half of the attack period. We can see that the scores are settling down within the first week itself rather than having the scores at a single level. The initial spike which gradually settles down is what makes this method to be consistent in detecting a proportion of anomalies in all attack profiles. This method is inherently good at detecting the collective anomalies during the initial stages of an attack, thus making it a good choice to alarm the system at the beginning of an attack.

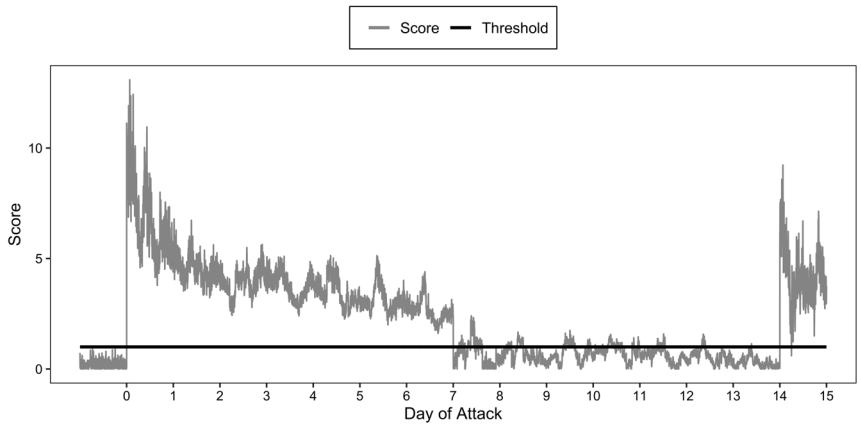

Figure 5.7: Anomaly scores when Holt Winters' prediction method is used with monthly threshold

Performance of each Threshold

While comparing the different thresholding mechanisms, the hourly thresholds have detected more than 40% of the anomalies across all prediction methods. The true positive rates do not follow a specific trend as the thresholds are calculated for smaller periods. However, the false positive rate shows an increasing trend as shown in figure 5.8. This interesting result comes from the threshold breakdown plot from chapter 4 figure 4.2. We could see that as the thresholds were calculated for smaller periods, the number of fifteen minute windows with at least one anomaly increases, finally leading to 10% anomalies in each window. Though 10% of alerts may not be significant when observed in each fifteen minute window, the sum of alerts through out the year increases from the previous thresholding mechanism where the thresholds are changed less often.

When comparing the performance of each threshold individually with prediction breakdown as shown in figure 5.9, it can be observed that the false positive rates have been increasing as the thresholds are calculated for smaller windows. Naïve and average prediction methods have performed

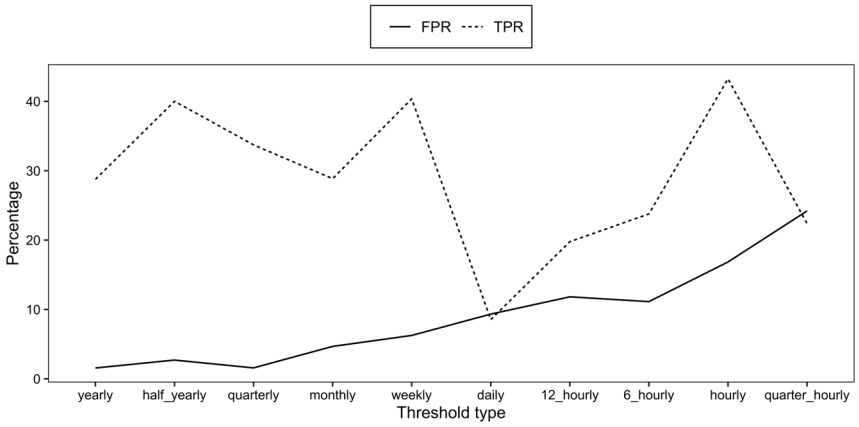

Figure 5.8: Performance of each threshold mechanism

poorly for all thresholding mechanisms except for six hourly thresholds where the average prediction has detected more than 25% of the anomalies.

The false positive rate for all threshold mechanisms is the least for simple exponential smoothing. Seasonal naïve and Holt Winters' methods performed well in detecting around 50% of the anomalies for all thresholding mechanisms except daily thresholds, when compared to the other prediction methods. In summary, the generic threshold mechanisms which are the yearly, half yearly, quarterly and monthly thresholds, weekly and hourly thresholds performed well in detection when combined with seasonal prediction methods and anomaly scores.

First Detected Time of an Attack

We could see that in most of the detection systems that we used, the first minute of attack is usually detected because of the sudden increase in demand. Table 5.2 gives when each attack was first detected. Majority of the attack profiles were detected within the first minute by their respective anomaly detection system. Six attacks were never detected and they belong to the anomaly detection system which uses average prediction method.

5.2.2 Performance of Anomaly Likelihood

The anomaly likelihood gives a measure of how anomalous the current point is based on the anomaly scores computed in the past. This can be considered as a layer over an anomaly score.

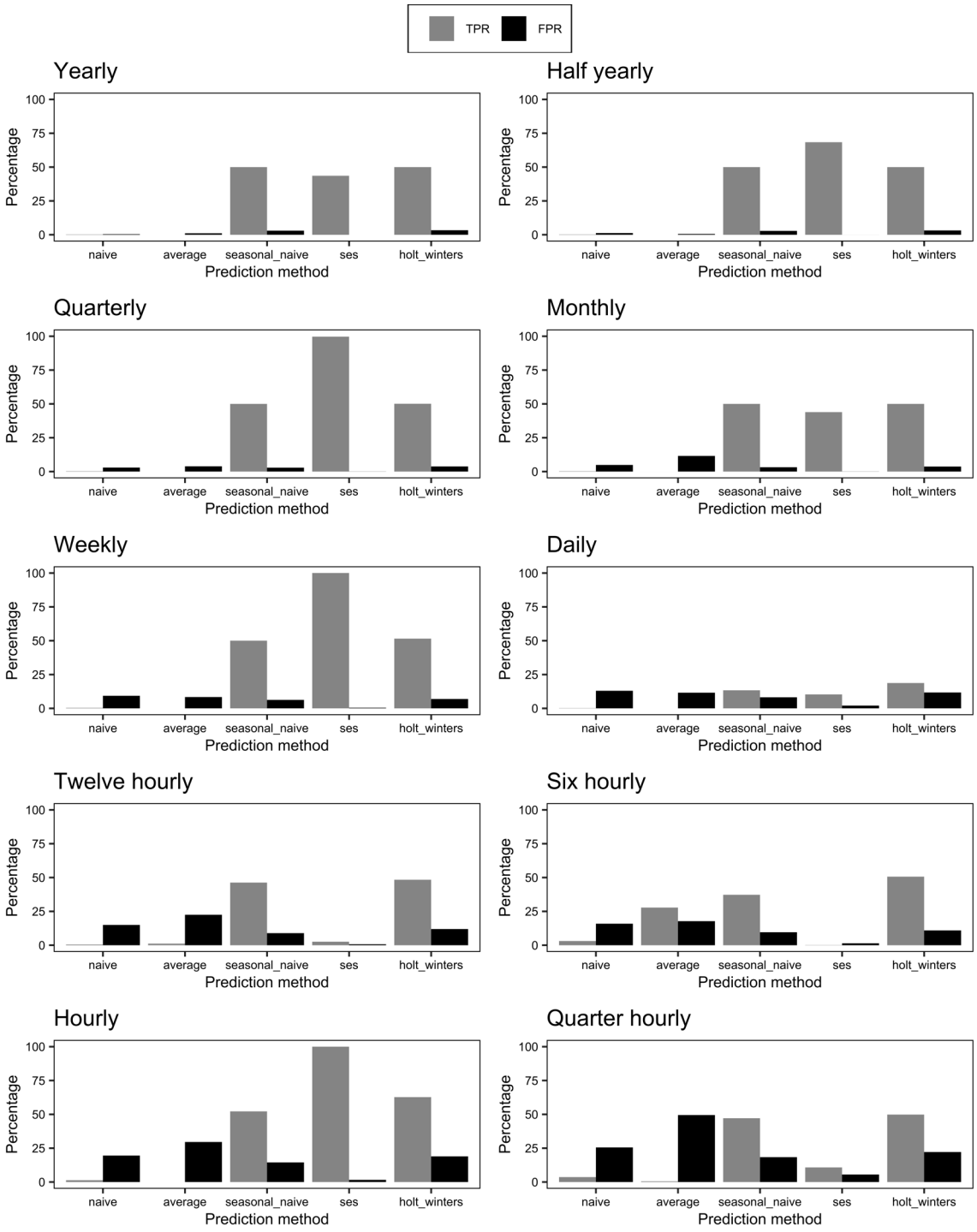

Figure 5.9: Performance of each thresholding mechanism using anomaly scores with prediction breakdown

First detected time	Number of attack profiles
Never detected	6
Minute 1 of attack	37
Between 2 to 5 minutes	1
Between 5 to 15 minutes	1
Between 6 to 24 hours	1
Between 2 to 7 days	3
Between 1 to 2 weeks	1

Table 5.2: Number of profiles by first detection time

Instead of using thresholds on the anomaly score itself, we first calculate the likelihood and then check if it is greater than a likelihood threshold of 0.999.

Performance of each Prediction

The true positive and false positive rates were calculated as shown in equations 5.1 and 5.2 repectively. It can be observed from figure 5.10 that the true positive rates for all methods are below 5%, thus performing poorly in detection when compared to anomaly score. However, the false positive rates for all predictions are below 1%, which is less when compared with the anomaly score method. Figure 5.11 shows the false positive rates for each prediction method using anomaly likelihood. As the anomaly likelihood is based on the scores in the past, it flags the current data point as an anomaly only when it is different from the scores calculated in the past window.

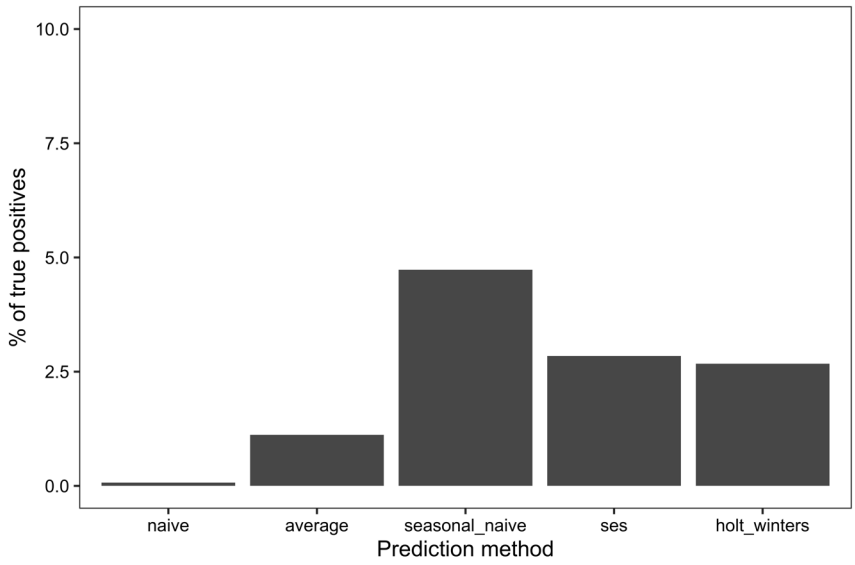

Figure 5.10: Overall true positive rate for each prediction using likelihood

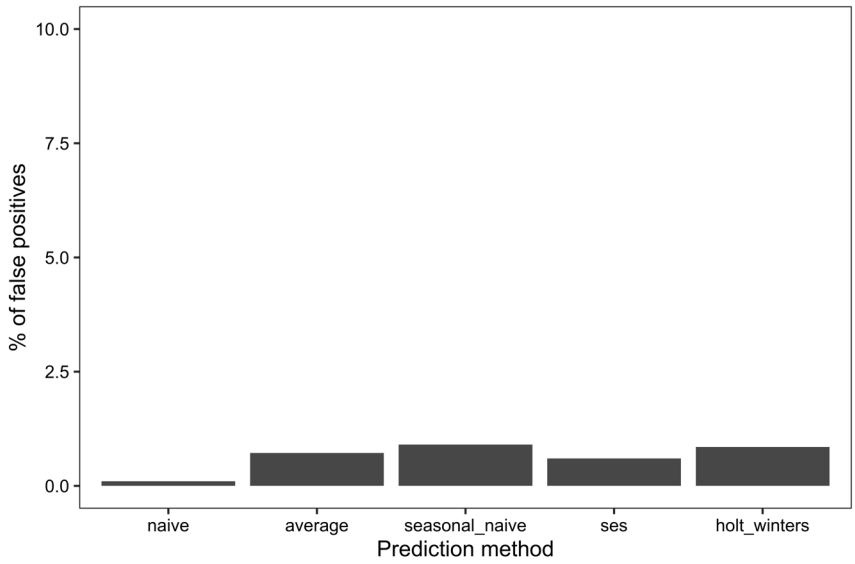

Figure 5.11: Overall false positive rate for each prediction using likelihood

Types of Anomalies Detected

The naïve and average methods show an increasing trend in the likelihood throughout the attack period, detecting anomalies only at the end of the attack. It is also interesting to see that though, naïve prediction gives a high likelihood to the first point of attack, average method does not detect it and gave a low likelihood value to the start of attack.

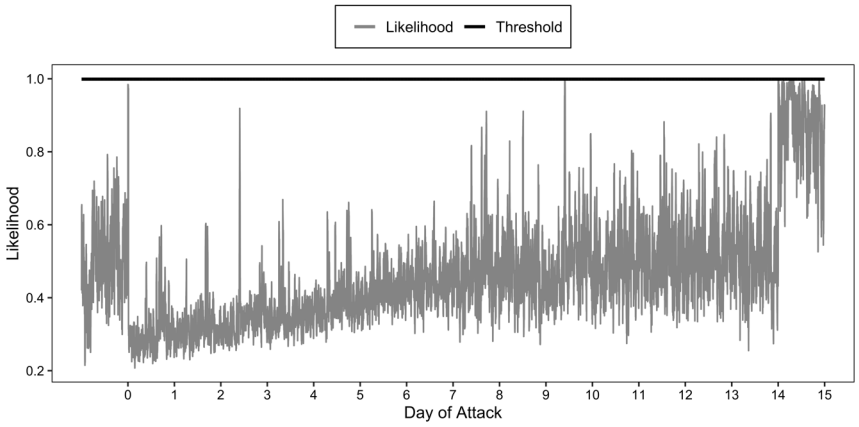

Figure 5.12: Anomaly likelihood when naive prediction method is used with hourly threshold

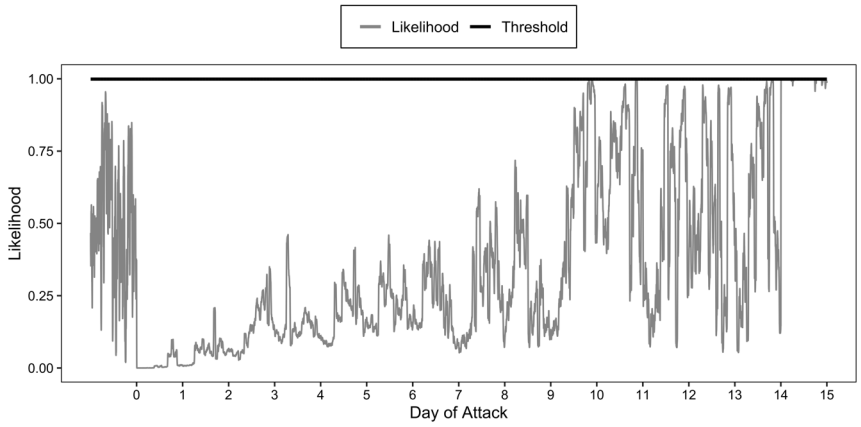

Figure 5.13: Anomaly likelihood when average prediction method is used with hourly threshold

Similar to the anomaly score performance results, anomaly likelihood also exhibits the weekly seasonality for seasonal naïve, simple exponential smoothing and Holt Winters' as seen in figures 5.14, 5.15 and 5.16. For the seasonal methods, we can see that the likelihood is high during the first two days of the attack and gradually reduces by the end of the week. We can then see an increasing trend in week two while staying below the threshold, as it belongs to a new cycle of seasonality. We can observe that the anomalies are detected at the beginning and at the end of the attack signifying that there were demand changes by a huge amount.

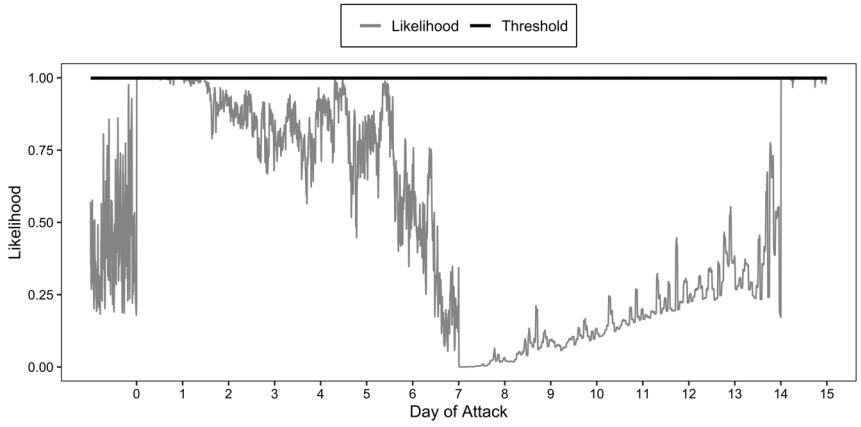

Figure 5.14: Anomaly likelihood when seasonal naive prediction method is used with hourly threshold

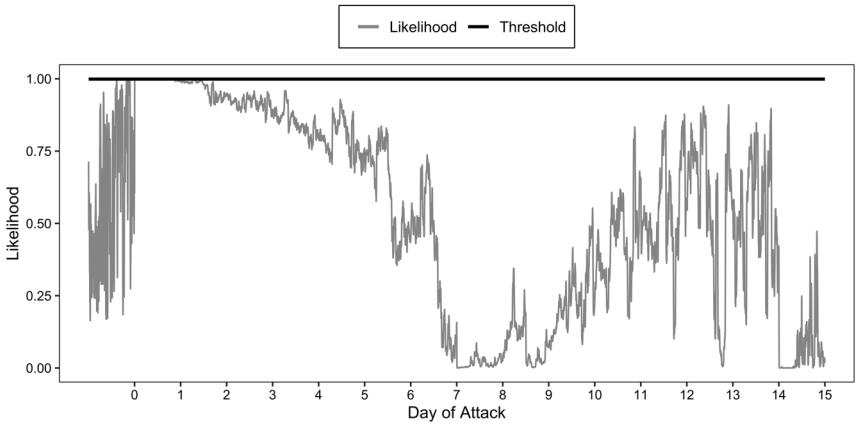

Figure 5.15: Anomaly likelihood when simple exponential smoothing prediction method is used with hourly threshold

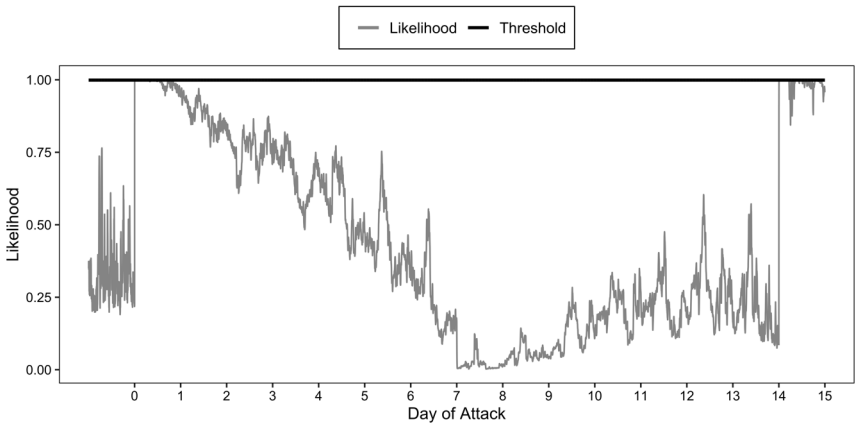

Figure 5.16: Anomaly likelihood when Holt Winters' prediction method is used with hourly threshold

Performance of each Threshold used for Attack Generation

With the likelihood function, we do not have many thresholds to compare and so, we compared the performance of the threshold used for attack generation. Figure 5.17 shows the prediction breakdown of true positive and false positive rates for each threshold type. When compared to the anomaly score method, the overall performance of likelihood in detecting anomalies is low. In all threshold types, the true positive rates are less that 8%. This is because the likelihood function always compares a point to its past neighbors to see if the state is anomalous. Hence, we saw the decreasing trend in seasonal methods as the likelihood values were settling down.

We can see that seasonal naïve performed consistently well when compared to the others. Similar, to the anomaly score, the likelihood did not perform well with the detection of attack profiles

generated using daily threshold. Simple exponential smoothing also shows similar characteristics of not being able to detect the attacks performed during seasonal transition months. Holt Winters' has the least true positive rate in the seasonal methods category. Unlike in anomaly score, the average method detected about 5% of the anomalies when yearly and six hourly thresholds were used for attack generation.

First Detected Time of an Attack

Eleven attack profiles were not detected by the likelihood technique and eight of them were detected in the first minute. Around fourteen attacks were detected between two to five minutes. Table 5.3 shows the summary of the times when the attack profiles were first detected.

First detected time	Number of attack profiles
Never detected	11
Minute 1 of attack	8
Between 2 to 5 minutes	14
Between 5 to 15 minutes	9
Between 6 to 24 hours	1
Between 2 to 7 days	1
Between 1 to 2 weeks	5

Table 5.3: Number of profiles by first detection time using likelihood

5.3 Anomaly Detection using Commercial Methods

In this section, we will evaluate the performance of each commercial method that we used for anomaly detection. We used five methods namely, Arundo's ADTK, Facebook's Prophet, HTM Studio, Anomalize and Twitter's Anomaly Detection. In the following sections, we will observe which methods performed the best in detection and what types of anomalies were detected.

Performance of each Method

The true positive and false positive rates were calculated as shown in equations 5.1 and 5.2 repectively. Figure 5.18 shows the overall true positive rate for each commercial method. Surprisingly, we can see that Facebook's Prophet performed the best in detecting the anomalies with about 40% true positive rate, though it is not an anomaly detection system as a whole, but produces accurate predictions. Next, we have Arundo's ADTK having about 5% true positive rate. HTM

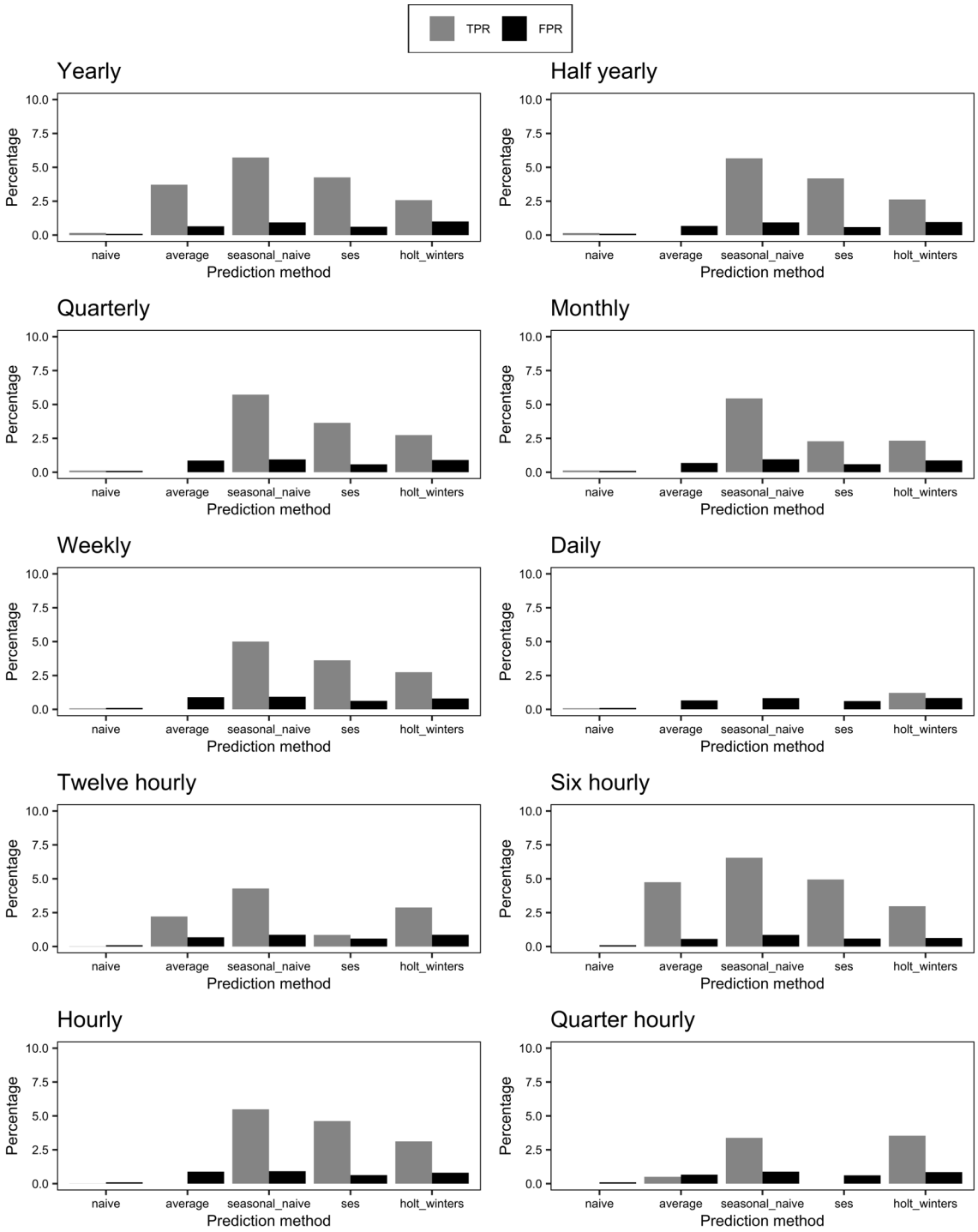

Figure 5.17: Performance of each thresholding mechanism using anomaly likelihood with prediction breakdown

Studio, Anomalize and Twitter's Anomaly Detection were able to capture less than 1% of the total anomalies.

Facebook's Prophet method has shown the highest false positive rate compared to the others with a value less than 20%. Arundo's ADTK performed similar to the true positive rate by having 5% as its false positive rate too. HTM Studio, Anomalize and Twitter's Anomaly Detection methods have low false positive rates too which shows that these methods depend on the recent observations for the anomaly detection which is why they start considering the attack consumption values to be normal as we progress during the attack.

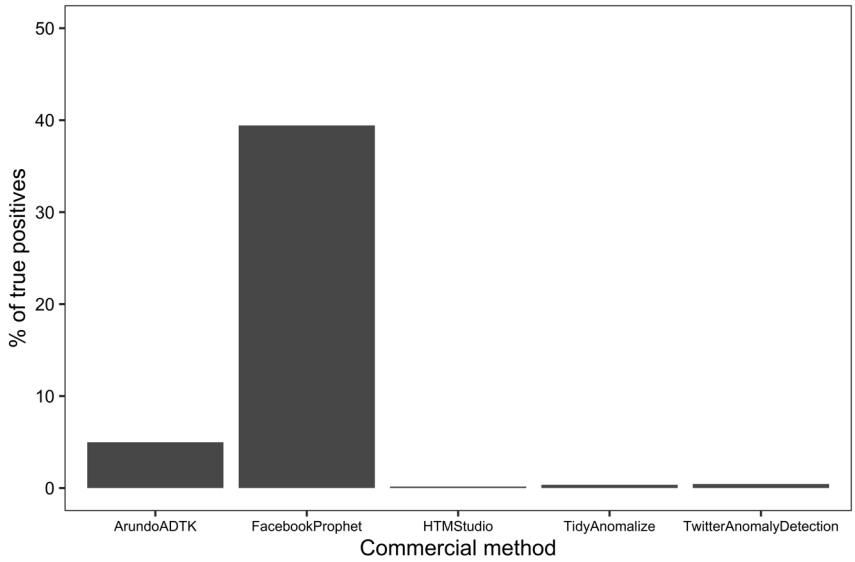

Figure 5.18: Overall true positive rate for each commercial method

Types of Anomalies Detected

Figure 5.20 shows the anomalies detected by Arundo's ADTK. This method has performed well in detecting anomalies distributed through out the time of the attack. Though a small fraction was detected, this method could also detect collective anomalies. However, all the point anomalies were not detected. There are some spikes which would be categorized as a point anomaly due to its abnormal nature when compared to its neighbors, but were not detected. Overall, this method could identify anomalies during the attack duration as opposed to detection only during the start or end of the attack.

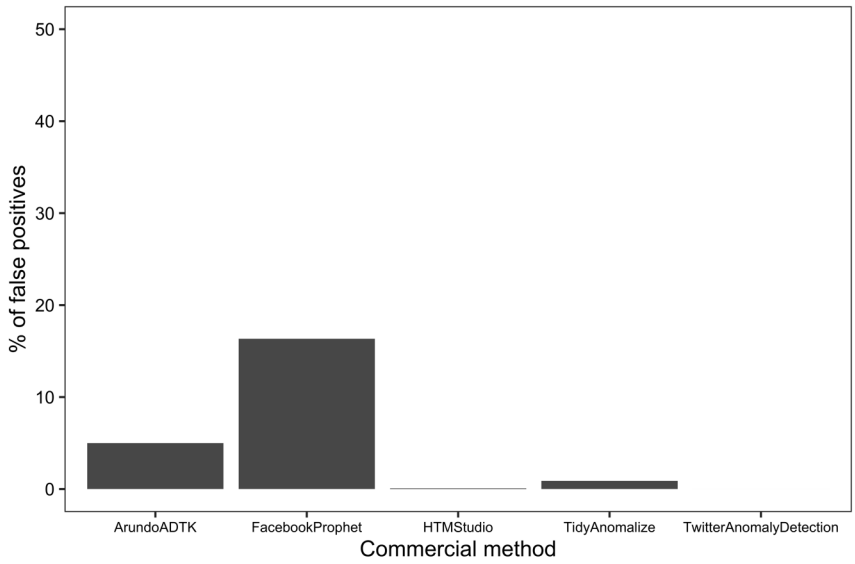

Figure 5.19: Overall false positive rate for each commercial method

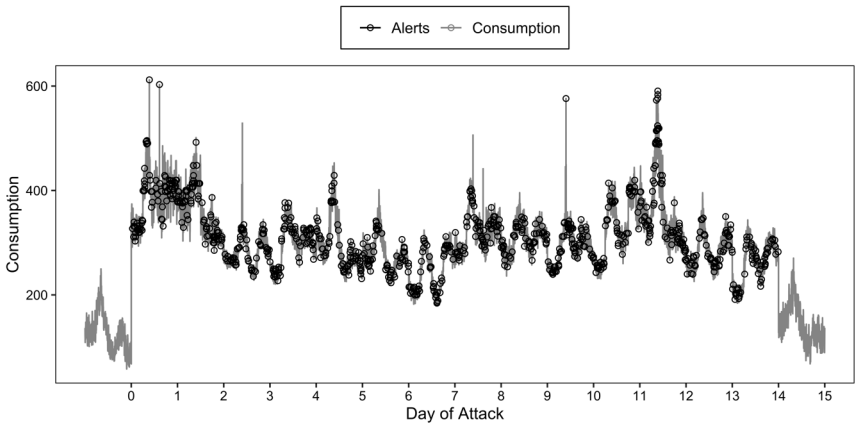

Figure 5.20: Anomaly detection using Arundo's ADTK

Facebook's Prophet has performed well in detecting spikes and troughs collectively as well as point anomalies represented as spikes. We can see in figure 5.21 that it could detect almost all the anomalies during the first two days of the attack which aids in detecting the attack in the initial stages itself.

HTM Studio could detect only a small amount of anomalies, some of them being detected collectively. It is interesting to see that none of the points at the end of the spikes were detected. Anomalize could detect the anomalies collectively at the start of the attack and gradually detected

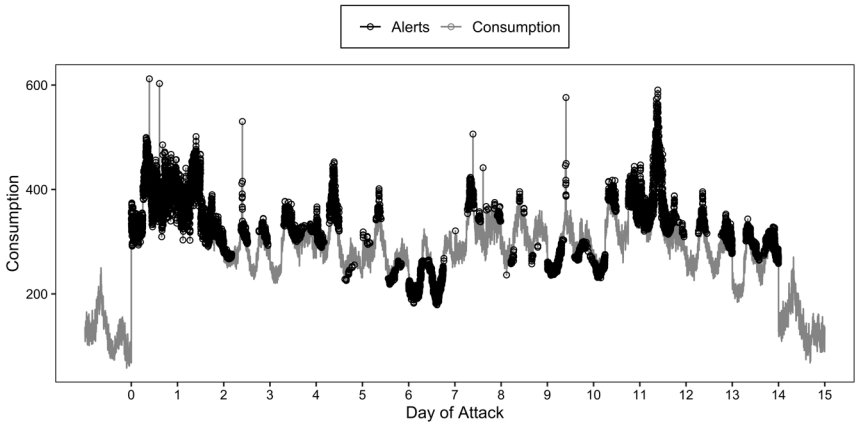

Figure 5.21: Anomaly detection using Facebook's Prophet

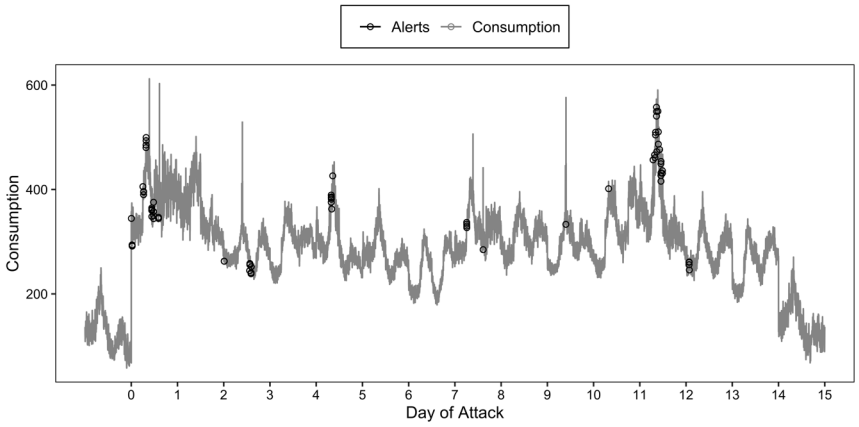

Figure 5.22: Anomaly detection using HTM Studio

only some increasing trends in consumption, whereas for the second week of the attack, only one anomaly was detected.

Twitter's Anomaly Detection could detect only some point anomalies with high consumption represented as spikes, but it could not detect any of the other consumption values.

In summary, most of the commercial methods are able to detect some of the point anomalies and the collective anomalies at the beginning of the attack. Facebook's Prophet had the best performance in detection when compared to the others as it could detect both point and collective anomalies when there is a change of direction in the trend.

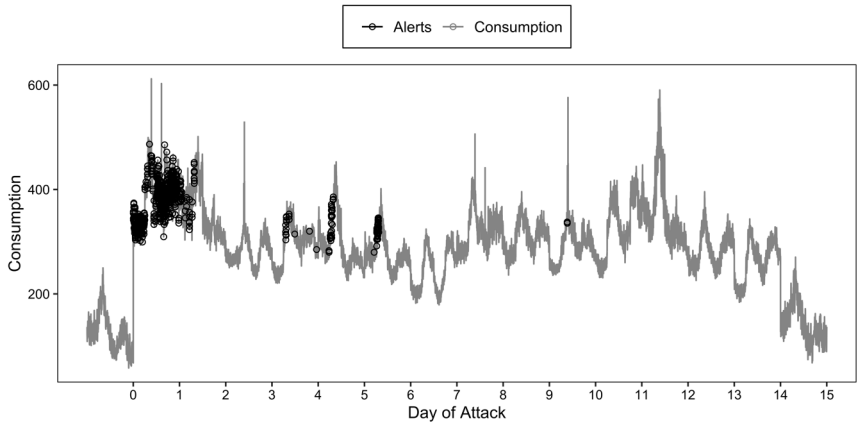

Figure 5.23: Anomaly detection using Anomalize

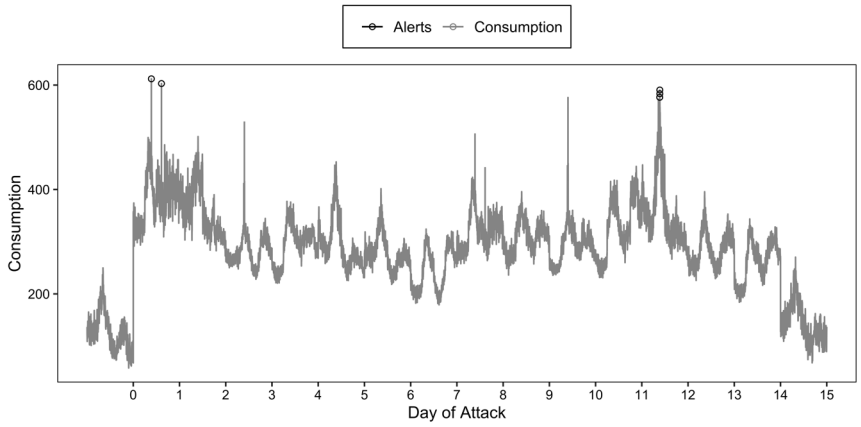

Figure 5.24: Anomaly detection using Twitter's Anomaly Detection

First Detected Time of an Attack

Thirteen attack profiles were not detected by the commercial methods in total. These methods which could not detect are Anomalize and Twitter's Anomaly Detection. Eleven attack profiles were detected during the first minute of the attack by Facebook's Prophet and Arundo's ADTK. Table 5.4 shows the summary of the times when the attack profiles were first detected.

5.4 Chapter Summary

In this chapter, we evaluated anomaly detection techniques which included anomaly score with thresholding, anomaly likelihood with thresholding and commercial unsupervised anomaly detection

First detected time	Number of attack profiles detected
Never detected	13
Minute 1 of attack	11
Between 2 to 5 minutes	10
Between 5 to 15 minutes	1
Between 6 to 24 hours	3
Between 2 to 7 days	2
Between 1 to 2 weeks	4

Table 5.4: Number of profiles by first detection time using commercial methods

methods. We observed that Holt Winters' prediction method along with anomaly score was able to detect some proportion of anomalies consistently across all thresholding mechanisms. Simple exponential smoothing with anomaly score detected 100% of the anomalies with quarterly, weekly and hourly thresholding while keeping its false positive rates low. Seasonal naïve prediction method with anomaly score performed well with generic thresholds which are yearly, half yearly quarterly, monthly and weekly in terms of detecting 50% of the anomalies. As the thresholds are calculated for smaller windows, the false positive rates started to increase for the seasonal naïmethod. Naïve and average methods performed poorly across all thresholds. From a thresholding view, daily thresholding failed to detected the anomalies across all prediction methods.

Anomaly likelihood technique detected less than 10% of the anomalies across all prediction methods. In this case, the attack generated using daily thresholding has again performed poorly in detection across all prediction methods. Seasonal naïve method with anomaly likelihood has performed well in detection across all attack profiles.

From the commercial anomaly detection methods, Facebook's Prophet performed better than the others detecting about 40% of the anomalies on an average. This method was able to detect point anomalies and some collective anomalies as well. Arundo ADTK detected about 5% of the anomalies on an average spread across the attack duration. This method also has a low false positive rate of 5%. Anomalize was able to detect a collection of anomalies in the beginning of the attack. HTM Studio and Twitter's Anomaly Detection were able to detect only some point anomalies.

In conclusion, we have seen anomaly detection methods which were good at detecting point anomalies, collective anomalies at the beginning or at the end of the attack duration and also collective anomalies in the first seasonal cycle of the attack. But, they were able to detect up to 50% of attacks on an average. Though simple exponential smoothing detected 100% of attacks when

used with certain thresholds, it was not able to consistently detect the attacks performed in different months. Hence, static thresholds are not always reliable. The commercial methods could also detect less that 50% of the anomalies. The nature of these attacks require an anomaly detection system that can handle these situations. In the next chapter, we will capture the cases where the anomaly detection system with static threshold was unable to detect the anomalies. We will also introduce dynamic thresholding to capture these situations as well as different types of anomalies that fall into different categories which are point, collective and contextual anomalies.

Chapter 6

Dynamic Thresholding

6.1 Introduction

We observed that static thresholding over anomaly scores and anomaly likelihood could detect only a proportion of anomalies. In some cases, the time when the attack was performed and the type of thresholding mechanism affected the performance of detection. There was a noticeable distinction in detection between naïve and seasonal prediction methods. Seasonal prediction methods aided in detecting the anomalies better, but when different thresholding mechanisms were used for detection or for attack generation, we could see that they were not able to perform consistently well across all thresholds. This shows that the choice of the thresholding mechanism significantly contributes to anomaly detection. However, a single thresholding mechanism might perform well in detection for a given time period during the year, say summers, but perform poorly during the winters. Also, if a thresholding mechanism is decided, it should be able to handle anomalous situations that may not be an attack, but were caused due to external factors. For example, a sudden increase in demand may happen if an unexpected storm enters an area, requiring the community members to stay home. This brings us to the necessity of having dynamic thresholding which can adapt to unexpected situations, but also identify attack periods. We will discuss the motivation for dynamic thresholding and our proposed method for detection in the following sections.

6.2 Motivation

While evaluating various thresholding mechanisms, we observed some common situations where an attack was not detected. We discuss each of these situations in the following subsections.

6.2.1 Detection using Static Thresholds

When static thresholds are used for anomaly detection, the threshold does not adapt to the scores based on the consumption. This might result in the threshold being too low or too high. Figure 6.1 shows that the first week of attack has the threshold low and for the second week of attack the threshold is too high. The thresholding mechanism used here is a yearly threshold, but as long as thresholds are generic this situation will occur.

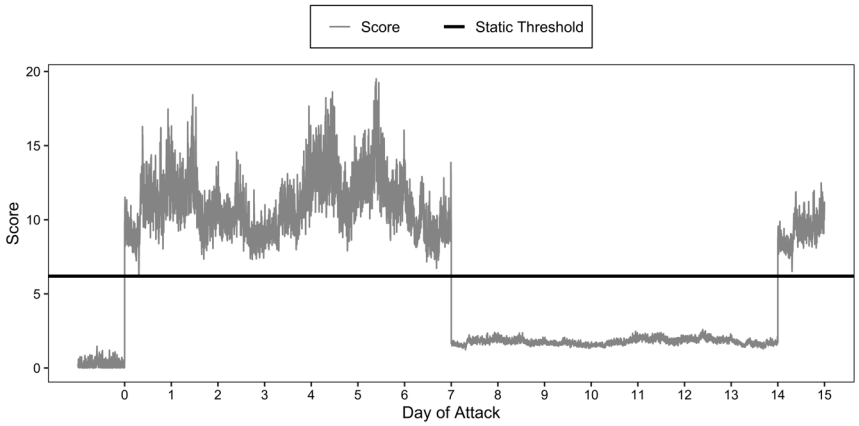

Figure 6.1: Anomaly detection using static threshold

With generic thresholds, due to the large gaps between the scores and the thresholds itself, the attacker could exploit that gap to perform the attack such that the demand is increased or decreased but the anomaly score lies below the threshold line. Figure 6.2 shows an attack that was carried out where demand was increased continuously for a duration of two weeks, but the anomaly scores are below the threshold. If the attacker has access to the threshold calculation, they can perform the attack by following the shape of the threshold, but staying undetected.

6.2.2 Early Detection of Attacks

It is necessary to detect the attack as early as possible to deploy the appropriate contingency plans for mitigating future effects of high increase or decrease in demand. Static thresholds do not

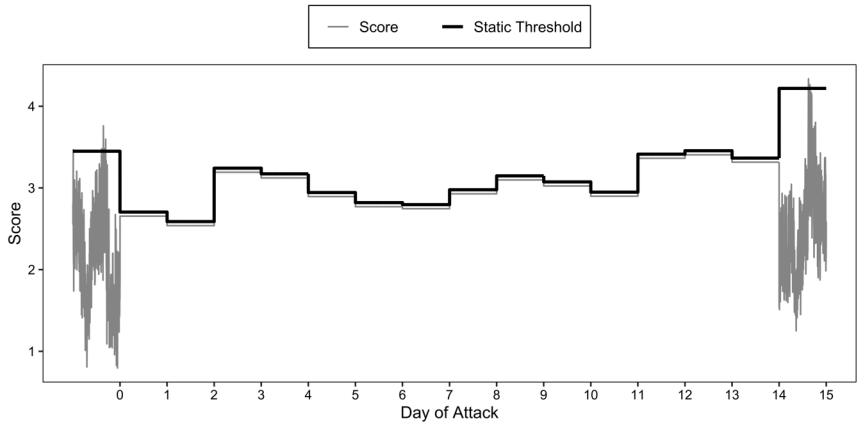

Figure 6.2: Example of an attack which exploits the gap between threshold and anomaly score

anticipate for such situations, which may result in attack periods going unnoticed. Figure 6.3 shows an example where the majority of the anomalies were not detected initially. In the first two days, there are less than 5% anomalies detected, and are considered normal when the anomaly tolerance level is 5% for each day. It is clear that after day six, except for a few points, the rest of the attack was not flagged as an anomaly.

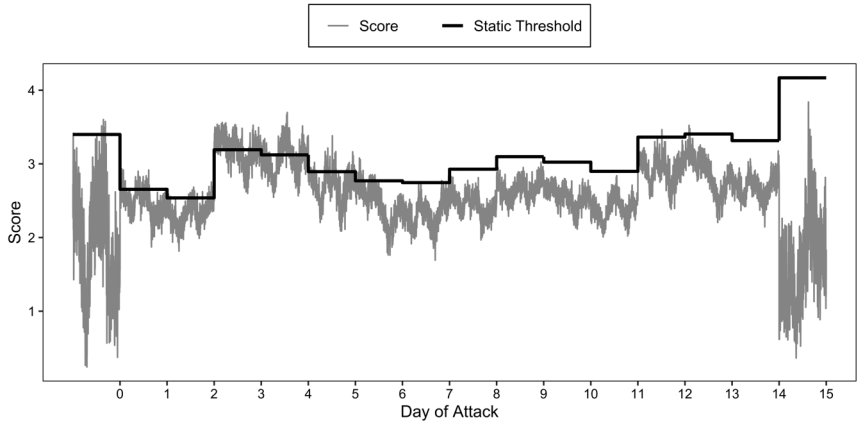

Figure 6.3: Example of partially detected attacks

6.2.3 Time of Attack

Figure 6.4 shows an example where a static threshold for the entire year was able to detect all the anomalies when the attack was performed in March, but was not able to detect any of them

when performed the June. The thresholding mechanism should be such that it can detect both the cases which requires the threshold to be personalized based on the season of the year.

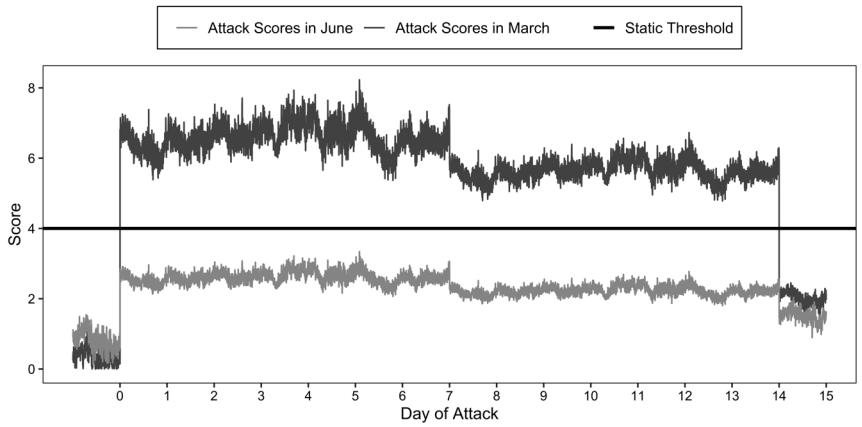

Figure 6.4: Example of attack detection based on the month of attack

6.2.4 False Alerts

It is also important to keep track of the number of false alerts that a thresholding mechanism may flag. The thresholding mechanism should be such that it detects most of the anomalies, but also has least false positives during normal consumption days. Figure 6.5 shows an example of fifteen minute thresholding that flags the normal consumption values as anomalies, resulting in a large number of false positives.

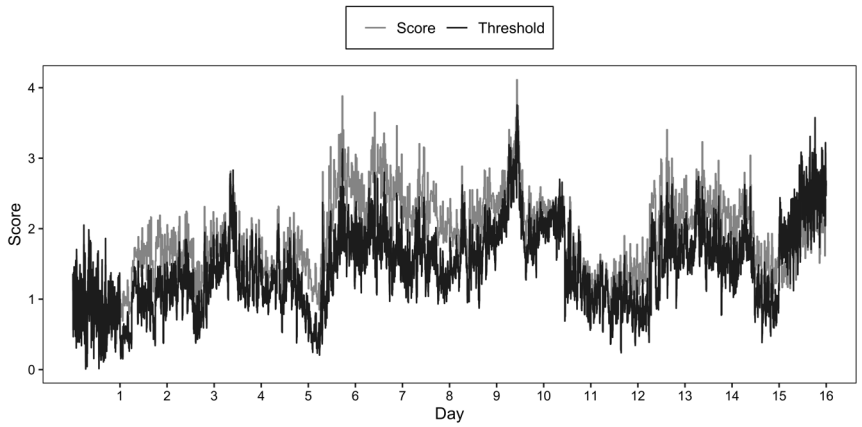

Figure 6.5: Example of normal consumption values being flagged as anomalies

6.2.5 Expectations from a Thresholding Mechanism

By analyzing these situations, we made a list of expectations from a thresholding mechanism when used for power consumption anomaly detection:

1. It should detect the start of any anomalous situation as this is a crucial indicator for mitigating any future effects of an ongoing situation.
2. It should detect when the anomaly scores are constant and are of the same value as this is very unlikely for power consumption data. Such anomalies are contextual and may not necessarily exceed the threshold.
3. It should detect collective anomalies effectively.
4. It should be able to detect anomalous situations which occur at any time of the year.
5. It should have low false positives when any prediction method with an anomaly score is used.

6.3 Methodology

To satisfy the list of expectations for a thresholding mechanism, we propose a dynamic thresholding technique which works analogous to a spring system. We will talk about the spring system and how we used it to fit our requirement.

6.3.1 Dynamic Thresholding using Spring System

Consider a traditional spring system consisting of a spring attached to a fixed surface. The spring is in its resting position when no force is applied. Every spring has a resistance, which is a reaction of the spring due to an external force. The resistance is also a force, but is applied by the spring in the opposite direction of the external force. In the following subsections, we will discuss about the properties of the spring system we used to design the dynamic thresholding model.

Non-linear Resistance

The first principle is that the spring system has a non-linear relationship between the force applied on the spring and the displacement of the spring. This principle enforces that the higher we

want to compress the spring, the more difficult it becomes. Figure 6.6 shows the non-linear property of the resistance. As the force increases, the displacement changes non-linearly.

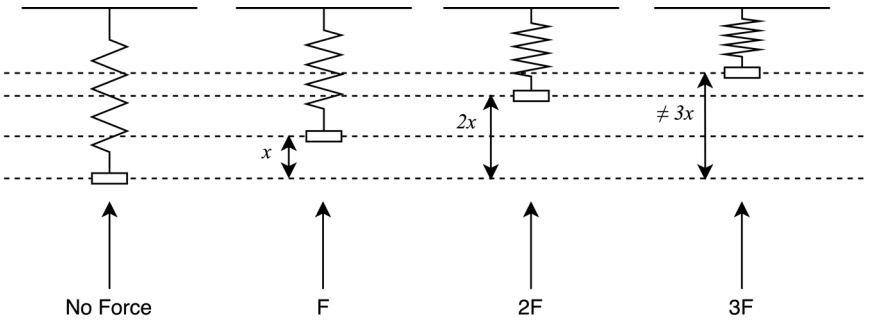

Figure 6.6: Non-linear resistance of the spring system

Varying Resistance

The second principle is that a constant force applied to a spring for a long period of time is unable to retain the same level of compression. In other words, the resistance of the spring can be viewed to increase such that the displacement occurs in the opposite direction of the force. This enforces that the longer a score is constant, the more difficult it becomes to sustain a threshold. Figure 6.7 shows the varying resistance property of the spring.

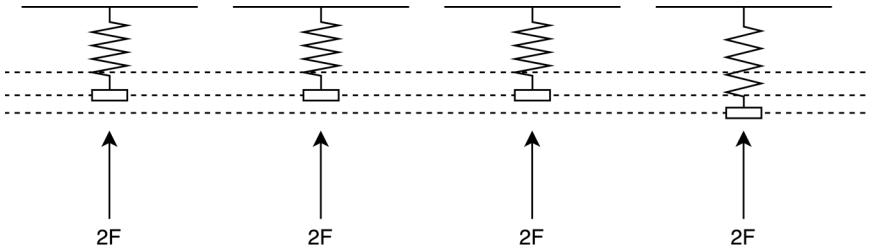

Figure 6.7: Varying resistance of the spring system

Dynamic Thresholding

Using the two principles that we explained, we will be designing the dynamic thresholding system. Figure 6.8 shows the high level working of our dynamic thresholding model. Initially, the spring is in its resting position and we want to apply some force to reach a target threshold. However, the spring is compressed only until a value less than the target threshold, which is considered to be the actual threshold due to the resistance model of the spring. The resistance model is the

deciding factor of the displacement that can occur for an applied force intended to reach a target displacement. The higher the resistance, the more difficult it will be to move towards the target threshold.

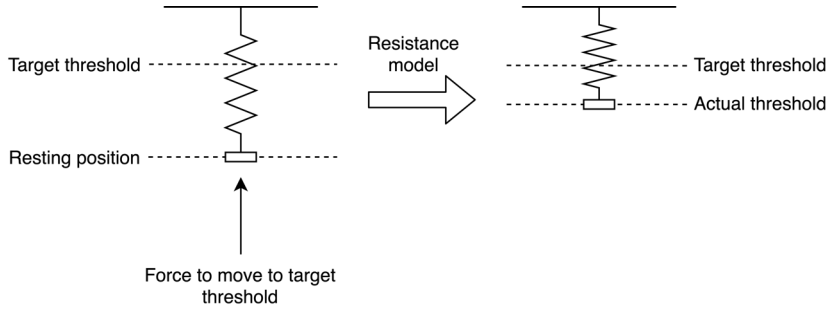

Figure 6.8: High level working of dynamic thresholding system

Resistance Model

The resistance should be modeled in such a way that it covers both the principles. We will define the resistance as a function of the displacement. Let $R(x)$ represent the resistance after compressing by a displacement of x. The resistance function gives a value between 0 and 1. It can be represented mathematically using the following equation.

$$R(x) : \mathbb{R}^+ \rightarrow [0, 1], \tag{6.1}$$

where $R(x)$ is the resistance function with x as displacement that takes a positive real number belonging to $\mathbb{R}^+$ as input and gives a value between 0 and 1 as output.

When we try to reach a target threshold, the resistance model is applied such that it results in the spring being displaced until a smaller actual threshold. Let $T_{target}(t)$ be the target threshold we wish to attain at time t. Then, the actual threshold is represented by the following equation.

$$T_{actual}(t) = \int_0^{T_{target}(t)} (1 - R(x))\, dx, \tag{6.2}$$

where $T_{target}(t)$ and $T_{actual}(t)$ are the target and actual thresholds at time t, x is the displacement and $R(x)$ is the resistance function.

We set the target threshold at each time t to be the following.

$$T_{target}(t) = 2\ Q(0.99|W_{as-recent}), \tag{6.3}$$

where $Q(.|W_{as-recent})$ is the quantile function of values in the time window $W_{as-recent}$ that is set to the most recent 15 minutes (i.e. $t-1$ to $t-15$) during the dynamic threshold calculation.

When a spring system does not have any resistance, then the resistance function will be $R(x) = 0$, thus being able to reach the target threshold. The resulting equation of actual threshold will be $T_{actual}(t) = T_{target}(t)$. Similarly, when the spring system has infinite resistance, $R(x) = 1$, we have $T_{actual}(t) = 0$. This means that there is no displacement at all.

We model the resistance function $R(x)$ as the cumulative distribution function of the anomaly scores in a time window. Let X be a random variable of anomaly scores, x represents some anomaly score and W_{as} be a window of anomaly score observations in the past one week. Then the resistance function is represented as follows.

$$R(x) = Pr(X < x|W_{as}), \tag{6.4}$$

where $R(x)$ is the resistance function and $Pr(X < x|W_{as})$ represents the probability of the variable X to be less than x in the learning window W_{as}. Figure 6.9 shows the cumulative distribution function of the anomaly scores.

We observed that the cumulative distribution function is very steep and 99 percent of the anomaly scores are less than 5 in the learning window. Hence, we introduce some slack, called resistance factor, and represented by r_f. This resistance factor is calculated by the following formula.

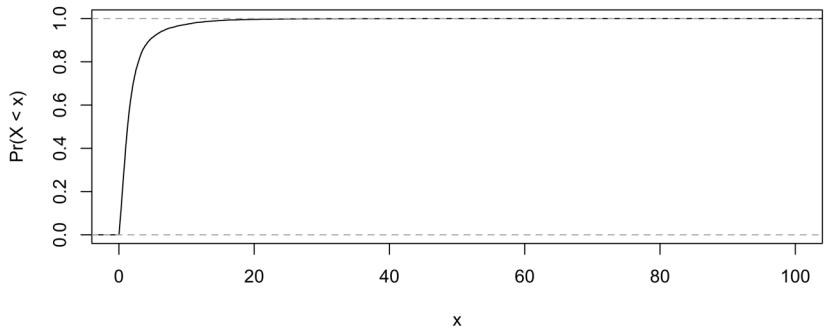

Figure 6.9: Cumulative distribution function of anomaly scores

$$r_f = \frac{Q(0.9|W_{as})}{Q(0.99|W_{as})}, \tag{6.5}$$

where $Q(.|W_{as})$ is the quantile function of values in the learning window W_{as}.

After introducing the resistance factor, we calculate the resistance as follows.

$$R(x) = Pr(X < x.r_f|W_{as}) \tag{6.6}$$

Figure 6.10 shows the additional line that has been drawn as the result of multiplying the resistance factor. We can see that for the same value of x which has been marked at 5, the resistance has now decreased from 0.99 to 0.9.

In order to accommodate the varying resistance property in the model, we will have to move in the area between the lines representing the resistance with $r_f = 1$ and $r_f = \frac{Q(0.9|W_{as})}{Q(0.99|W_{as})}$. We will represent the variable which will control the shift factor between these two lines as s_f, where $s_f = \frac{1}{r_f}$. The value of s_f ranges from $[1, \frac{1}{r_f}]$ and $s_f \geq 1$ Now, we will look at the sequence of coefficients of variation of anomaly scores over a single week.

Figure 6.11 shows the anomaly scores over a one week duration and figure 6.12 shows the coefficients of variation of the anomaly scores for the same week using a one day rolling window. It can be seen that the coefficients of variations are very unlikely to be zero. Figure 6.13 shows

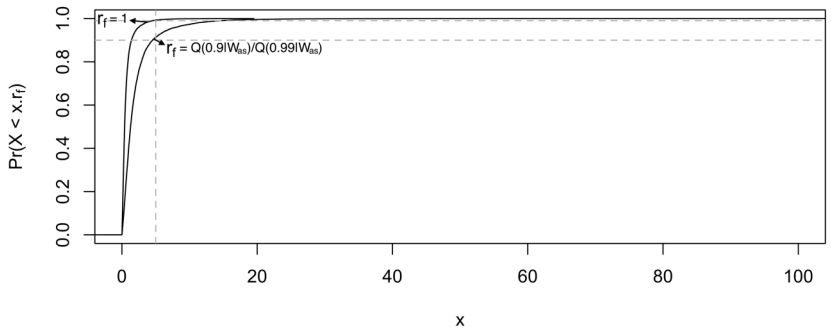

Figure 6.10: Allowing some slack for the resistance

that majority of values of the coefficient of variation lies between 0.5 and 1. We will be using this observation as the basis for finding if there is a constant anomaly score over a long period.

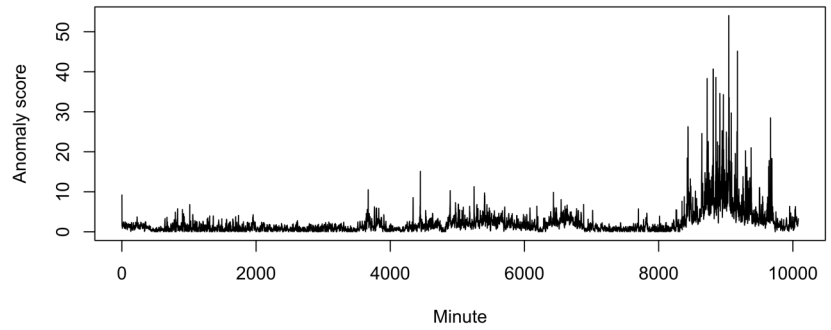

Figure 6.11: Anomaly scores in week 5

To find the shift factor, we will be using an exponential drop function which is calculated as follows.

$$s_f = C(c_t) = max(1, \alpha e^{\beta c_t}), \tag{6.7}$$

where c_t is the coefficient of variation at time t, $\alpha = \frac{1}{r_f}$, $\beta = \frac{ln(r_f)}{Q(0.001|W_{cv})}$ and W_{cv} is the sequence of coefficients of variation in a rolling windows of one day.

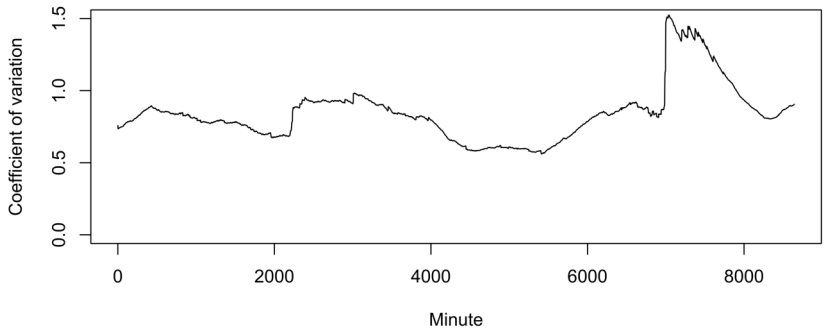

Figure 6.12: Coefficient of variation of anomaly scores in week 5

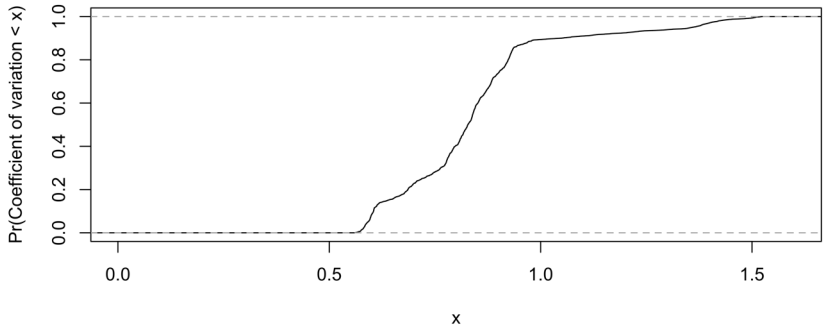

Figure 6.13: Cumulative distribution function of coefficients of variation of anomaly scores in week 5

After including the shift factor, equation 6.6 changes to the following.

$$R(x) = Pr(X < x.r_f.s_f | W_{as}, W_{cv}), \tag{6.8}$$

The final resistance model $R(x)$ described in equation 6.8 is then used as the resistance in equation 6.2 to find the value of the actual threshold at time t. The learning process, which updates r_f every week and controls the shift factor s_f by updating it every fifteen minutes is what makes this mechanism dynamic. We will discuss the overall working of this mechanism in detail in the next subsection.

6.3.2 Working

In this subsection, we will put all our concepts for dynamic thresholding together and explain the flow of generating a dynamic threshold.

Steps to Learn the Anomaly Scores

The learning process takes place every week, where the learning parameters α and β are updated. The steps for learning are as follows.

1. We use only non-anomalous past observations for the learning process and are identified by checking which time units have anomaly scores less than their respective dynamic thresholds.

2. We learn the cumulative distribution of these scores and also find the 99^{th} and 90^{th} percentiles of the scores. Using these percentile values, we calculate the resistance factor r_f as shown in equation 6.5. Now, we compute and store the value of the first learning parameter α. Note that, the value of α is the reciprocal of r_f.

3. To compute the second learning parameter β, we first create subsets of non-anomalous scores. Each subset consists of continuous non-anomalous observations over time. A new subset is created every time an anomaly occurs. In other words, we create a subset starting from the first non-anomalous observation until the next anomaly that is observed. This process is then repeated for every occurrence of an anomaly.

4. We find the coefficients of variation for each subset by using a rolling window of one week and find the 0.1^{th} percentile of all the computed coefficients of variation. This will give the value of the denominator of β, where $\beta = \frac{ln(r_f)}{Q(0.001|W_{cv})}$. Now, as we have the value of r_f, we compute the value of β and store it.

5. The result of the learning process is the computation of the learning parameters, which are α and β.

Steps to Generate the Threshold

We will discuss the steps involved in generating a thresholding dynamically for a point in time.

1. When the dynamic thresholding mechanism runs for the first time, it learns the scores from the previous week.

2. For every time unit in the current week, we set the target threshold to be twice the 99^{th} percentile of the anomaly scores in the most recent fifteen minute window. Equation 6.3 shows this step where $W_{as-recent}$ is set to the last fifteen minutes. It should be noted that if the dynamic thresholding mechanism is run for the first time, there is no target threshold set. Hence, the actual threshold itself is set to this quantity.

3. We calculate the coefficient of variation in the past one week and find the shift factor for the current time unit. The shift factor is calculated as shown in equation 6.7. Note that, r_f, α and β values are set during the learning process.

4. We find the actual threshold that can be reached for the current time unit by using equations 6.8 and 6.2.

5. This process is repeated for every time unit, where a new threshold is calculated and stored as the actual threshold for that time unit. An anomaly score will be flagged as an anomaly if it goes beyond the calculated threshold. The learning process is triggered every week, where the learning parameters are updated. Whereas, the shift factor controls the threshold during the current week based on the scores from the most recent fifteen minutes.

6.3.3 Alert Level

After a threshold is calculated for a time unit, we see whether the anomaly score for that time unit is greater than the computed threshold. Based on the number of times this occurs in the last 24 hours, we assign an alert type to each of these 24 hour rolling windows. There are four alert levels that we use and each alert is assigned based on the severity of the anomaly occurrence in the past 24 hours. The metric that decides the alert level for each time unit is the average number of alerts defined as the fraction of anomalies which occurred during the last 24 hours. Refer to table 6.1 for the breakdown of alerts.

Condition	Alert level
Mean alerts is between 0% and 25 %	LOW
Mean alerts is between 25% and 50 %	MEDIUM
Mean alerts is between 50% and 75 %	HIGH
Mean alerts is between 75% and 100 %	CRITICAL

Table 6.1: Condition for assigning each alert level

6.4 Results

Dynamic thresholding based on the spring system has shown encouraging results in terms of overall true positive rates as well as detecting anomalies in various contexts. In this section, we will evaluate the performance of dynamic thresholding and whether it satisfies all the expectations.

6.4.1 Overall Performance

Dynamic thresholding performed well in detecting anomalies overall across all seasonal prediction methods, when compared to static thresholding. Static thresholding was able to detect less than 50% of anomalies in all cases, whereas dynamic thresholding could detect more than 60% of anomalies in the case of simple exponential smoothing and Holt Winters' methods as seen in figure 6.14. Seasonal naïve with dynamic thresholding has detected an additional 3% of anomalies compared to the static thresholding. With naïve and average prediction methods, dynamic thresholding was able to detect more than 12% of the anomalies when compared to static thresholding that detected 1% and 3% respectively.

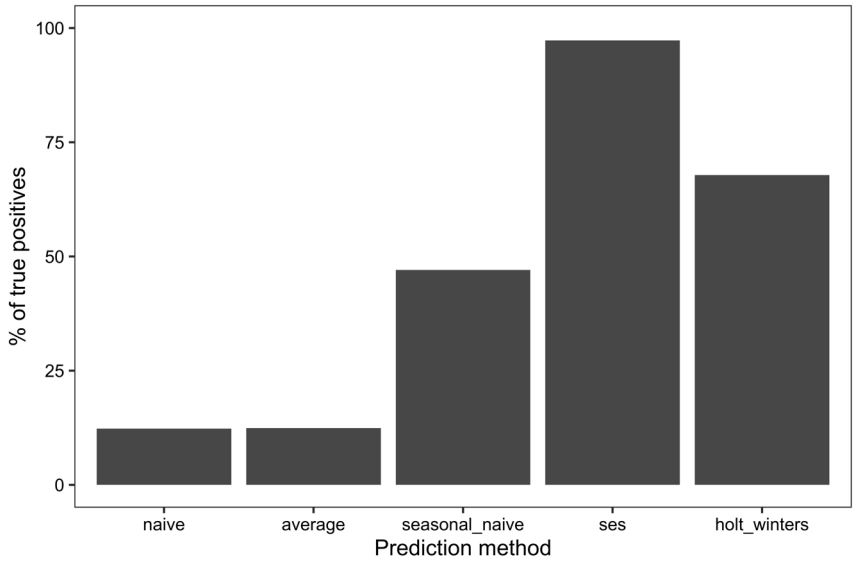

Figure 6.14: Overall true positive rate for each prediction method with dynamic thresholding

In terms of false positive rates, dynamic thresholding has about 27% for Holt Winters' which is the highest of all prediction methods. The average method has the least false positives of all prediction methods as opposed to its performance in static thresholding where it had the highest of

more than 15%. Figure 6.15 shows a comparision of overall false positive rates across all prediction methods.

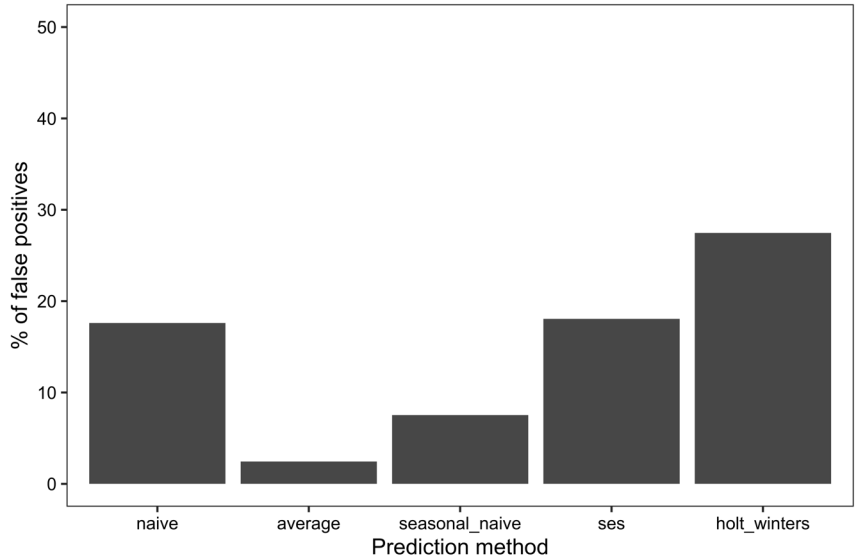

Figure 6.15: Overall false positive rate for each prediction method with dynamic thresholding

6.4.2 Performance of each Threshold used for Attack Generation

Simple exponential smoothing has performed consistently well across all thresholds used for attack generation and time of attack. When compared to static thresholding, dynamic thresholding with simple exponential smoothing adapted well during seasonal transitions. Dynamic thresholding has also detected attacks which were generated using daily thresholds. When compared to static thresholding, a maximum of 20% anomalies were detected. Holt Winters' prediction method has also performed consistently well across all thresholds used for attack generation, detecting at least 50% of anomalies on average in each case. Figure 6.16 shows the true positive and false positive rates for each thresholding mechanism used for attack generation with a prediction breakdown.

6.4.3 First Detected Time of an Attack

The first detected time of an attack has shown encouraging results where dynamic thresholding was able to detect 41 attack profiles in the first minute of the attack, which is the highest when compared to static thresholding, likelihood and commercial techniques. There were two attack

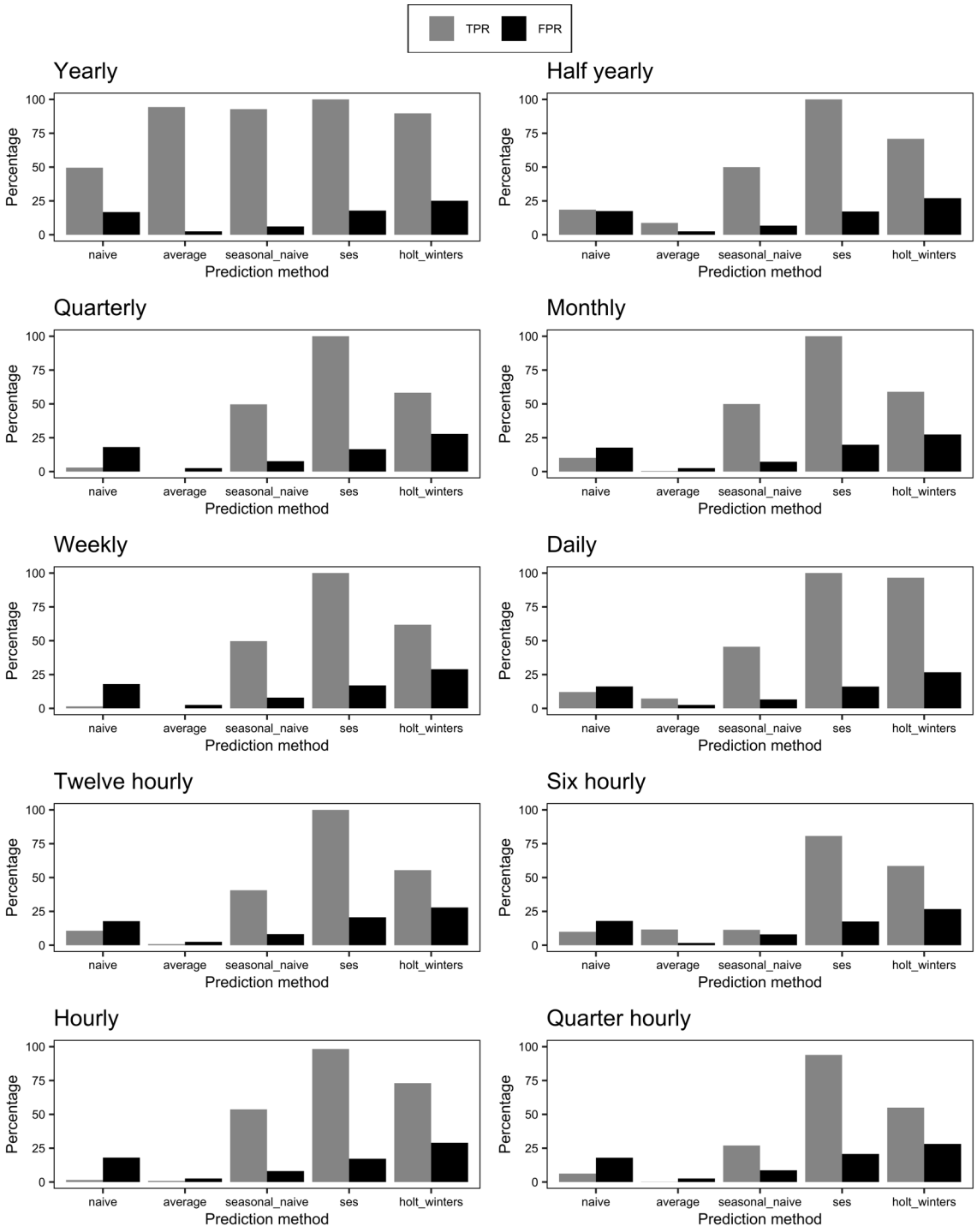

Figure 6.16: Performance of each thresholding mechanism used for attack generation with prediction breakdown

profiles that used average prediction method and were never detected. Table 6.2 shows the breakdown of number of attack profiles based on the first detected time. It can be observed that when a prediction method other than average prediction is used, the attack is detected during the first minute. This is because the dynamic thresholding mechanism does not tolerate unexpected increase in demand. It is difficult to stay undetected when there are sudden demand changes using this thresholding mechanism.

First detected time	Number of attack profiles
Never detected	2
Minute 1 of attack	41
Between 2 to 5 minutes	0
Between 5 to 15 minutes	0
Between 6 to 24 hours	3
Between 2 to 7 days	1
Between 1 to 2 weeks	1

Table 6.2: Number of profiles by first detection time

6.4.4 Evaluation of Thresholding Mechanism Expectations

In this subsection, we will go evaluate how far the dynamic thresholding satisfies the expectations that we put forward for a thresholding mechanism.

Detection of Start of Attack

Dynamic thresholding could detect all attack profiles in the first minute itself except the ones that used average prediction. It could detect only the attack that used yearly threshold for generation from the average prediction category. As shown in table 6.2, 41 out of the 50 attack profiles were detected in the first minute.

Detection of Constant Anomaly Scores

Figure 6.17 shows an example of an attack profile where an attack is performed based on a target anomaly score. We can see that there is an initial rise in the threshold in order to adapt to the increased consumption, but when it is detected to be constant for a day, there is a drop in the threshold, detecting the rest of the attack period.

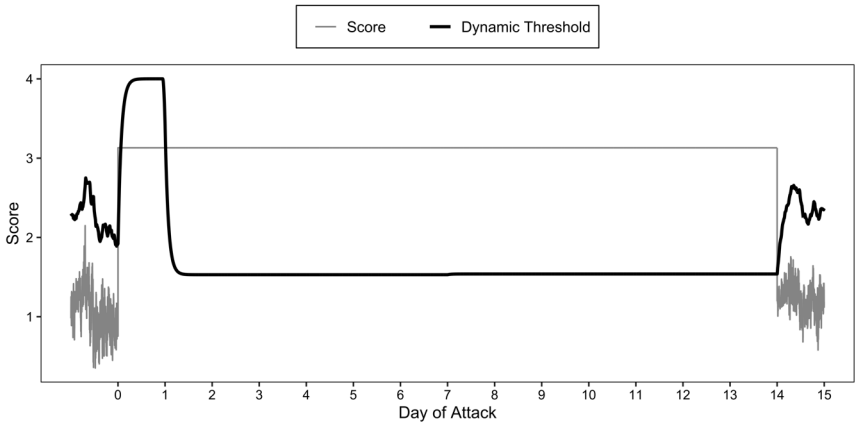

Figure 6.17: Detection of constant anomaly scores within a day

Detection of Collective Anomalies

Most of the commercial methods were good at detecting point anomalies and some areas of collective anomalies during the two weeks attack. Static thresholding was able to detect a proportion of collective anomalies based on the seasonality and the granularity of the threshold calculation range. Dynamic thresholding was able to detect the majority of the attack periods when compared to static thresholding. Figure 6.18 shows an example of an attack profile whose anomaly scores were calculated using Holt Winters' prediction method. As observed, dynamic thresholding does not let the threshold value to be increased easily at the beginning of the attack, thus detecting most attack periods collectively when compared to the static daily threshold that was unable to detect most attack periods in the second of the attack duration.

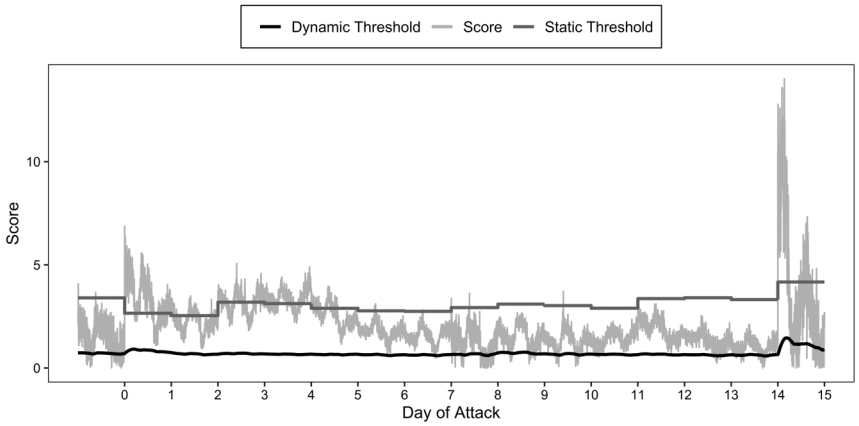

Figure 6.18: Detection of collective anomalies

Detection of Anomalies at any Time of the Year

This was observed in figure 6.16 that dynamic thresholding aided in detecting attacks during seasonal transition months. Static thresholding on anomaly score and likelihood scores as well as commercial methods, were unable to detect significant amount of anomalies during these times.

Low False Positives

Having detected majority of the anomalies in seasonal prediction methods, dynamic thresholding came with a price of having high false positives with a maximum of 27% across all prediction methods when compared to other anomaly detection methods. In this subsection, we will analyze the occurrence of false positives based on how long an alert level was sustained. As shown in table 6.1, we have assigned an alert level for each point in time.

Figure 6.19 shows the alert levels for each 24 hour rolling window during the year. We can see that the attack periods are highlighted and the alert levels for these attack times move gradually from LOW to CRITICAL. Most of the attack periods are at CRITICAL level until the end of the attack proving the ability of our dynamic thresholding method to sustain the CRITICAL alert level through out the attack.

We can see that there are some false alerts through out the year starting from spring and half way through summer and again from mid fall until the beginning of winter. Let us look into the average consumption in 24 hour rolling windows to analyze the distribution of these false alerts. Figure 6.20 shows the average consumption. When we compare figures 6.19 and 6.20, we can see the relation between the consumption and alerts. The consumption reached a peak value mid-February which is why we have a gradual increase in alert level during that time. It can also be observed that due to some peaks and troughs in the consumption until mid-summer, the alerts also move to different levels based on the consumption. We can see that the rest of the summer and the beginning fall has a stable consumption pattern except for the attack periods which is again reflected in the alert levels as well for those times. The steep increase in consumption trends while transitioning into winter along with significant peaks and troughs is reflected in the alert levels for those times too.

These peaks and troughs in consumption occurs due to highly varying temperatures and seasonal changes in the geographical area where the data was recorded. This consumption data was recorded

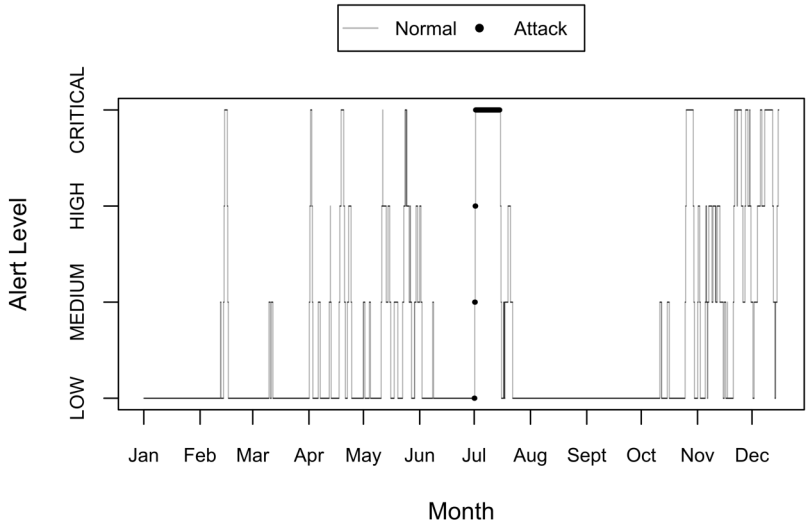

Figure 6.19: Alert level for each 24 hour rolling window during the year

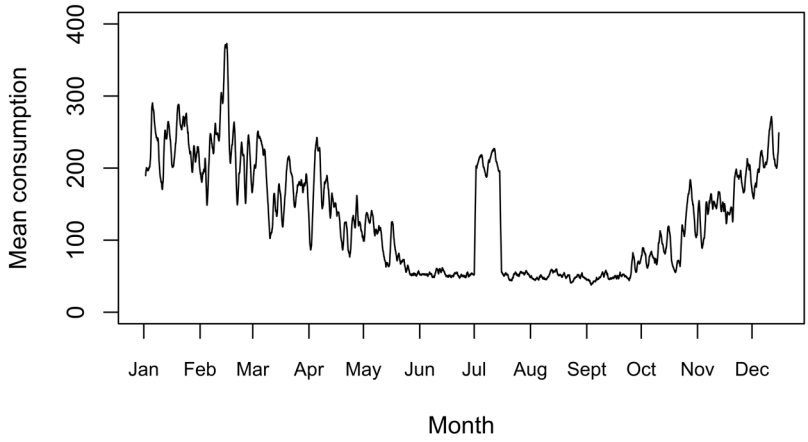

Figure 6.20: Average consumption in 24 hour rolling windows during the year

in West Massachusetts. From the consumption patterns, we analyzed that the consumption is more during colder seasons when compared to summers. Figure 6.21 shows the average monthly temperatures in West Massachusetts. We can see that there is an increase in temperatures from January until May. From May to September, the temperatures vary at a very slow rate. Again, from September to December there is a steep decrease in temperatures. Based on the minimum

temperatures seen every month, it is evident that high wattage appliances like room heaters are turned on showing the high consumption during these times.

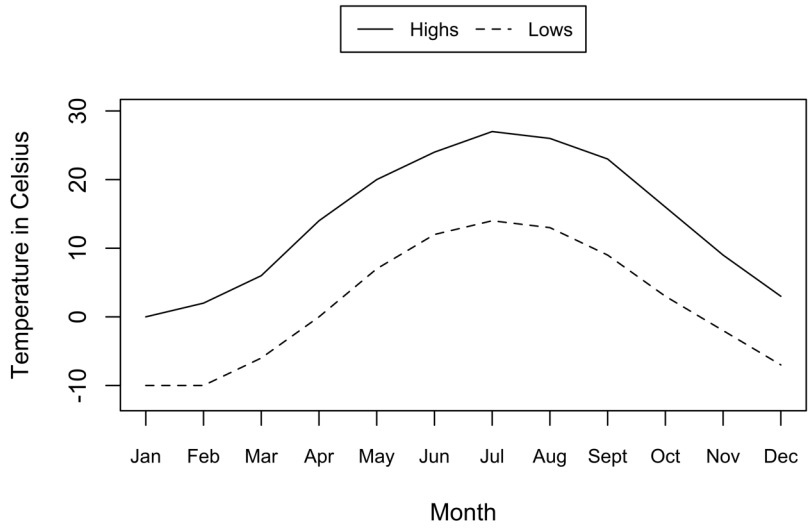

Figure 6.21: Average temperature highs and lows in West Massachusetts throughout the year[1]

From figures 6.22 and 6.23, we can observe that there are days with snowfall until the end of May. Though the frequency of snowfall decreases, the days or times when there is a snowfall affects the consumption patterns as well, which is why we see a sudden increase or decrease in consumption during these times.

Due to the high variability in consumption during each season, our dynamic thresholding approach assumes these unavoidable consumption changes to be anomalies. We will observe the top three sustained periods for each alert type to see how long an alert of type CRITICAL, HIGH and MEDIUM was continuously sustained. Table 6.3 shows the sustained time in minutes in decreasing order for each alert level. For CRITICAL level, the first entry shows how long this level was sustained during the attack period. The remaining two entries denote the times from the winter season approximately sustained for around 4 days. From figure 6.23, we can see that during the beginning of winter, there is snowfall for an average of 4 days. Though we do not know if these days are contiguous, after a single day of high snowfall, the temperature usually stays low during the following days. Similarly, it can be observed that for HIGH and MEDIUM alert level, the level

[1]Data accessed from National Centers for Environmental Information: https://www.ncdc.noaa.gov/

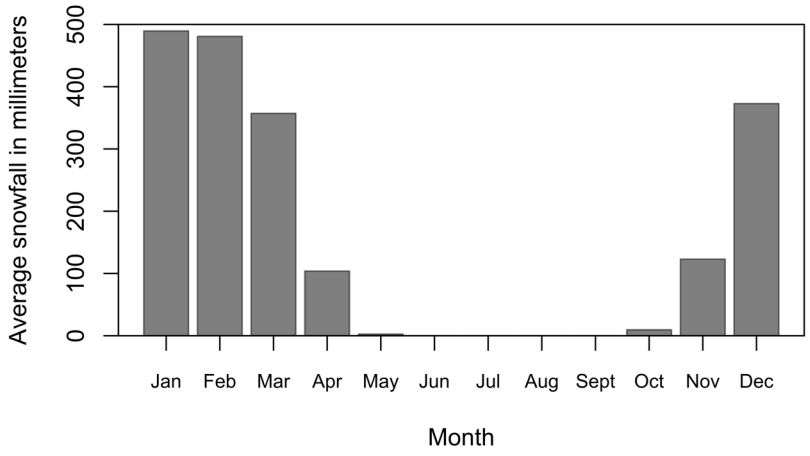

Figure 6.22: Average snowfall in millimeters in West Massachusetts throughout the year[1]

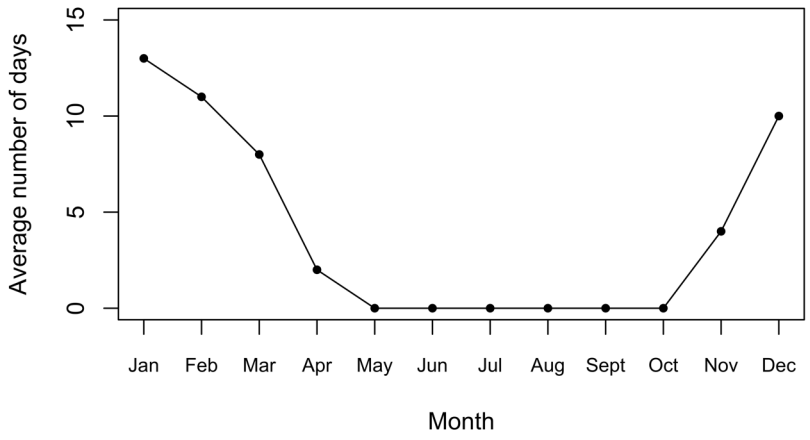

Figure 6.23: Average number of days of snowfall in West Massachusetts throughout the year[1]

is sustained for approximately two days which is observed to be occurring during spring. Again, in the figure 6.23, we can see that there are three days of snow on an average occurring during those months.

We can see a high dependence between the alerts and the seasonal variations. Due to the high seasonal variation of a particular geographic area, the consumption also shows high variation.

Alert level	#1	#2	#3
CRITICAL	19782	6244	5562
HIGH	2678	2517	2314
MEDIUM	2864	2667	2614

Table 6.3: Top three sustained times in minutes for each alert level

Because of this, the thresholding mechanism takes some time to adapt to these sudden variations and hence, flags these highly varying time periods as anomalies.

6.5 Chapter Summary

In this chapter, we identified various situations which were not detected accurately by the anomaly detection methods used in chapter 5. We then introduced dynamic thresholding based on a spring system, which is based on two principles of not being able to increase the threshold easily due to sudden increase in anomaly scores and being able to detect anomaly scores which are constant for long periods. We showed the results of the anomaly detection system that used a prediction method and anomaly score with a dynamic thresholding mechanism that could detect the anomalous situations better than static thresholds. We observed that simple exponential smoothing with anomaly score and a dynamic thresholding performed well compared to other prediction methods, being able to detect more than 70% of anomalies in all cases. Dynamic thresholding performed well in detecting the attack within the first minute when compared to static thresholding. It also performed well in detecting point, contextual and collective anomalies in power consumption data. Due to highly varying consumption data which is caused by the seasonal changes, the dynamic threshold flags these variations as anomalies which contributes to the false positives.

Chapter 7

Conclusion and Future Work

The increased usage in IoT devices across the globe poses a threat to the power grid. When an attacker has access to multiple IoT devices within the same geographical location, they can manipulate the electricity demand in the power grid which can lead to power line failures, cascading failures, increase in operating costs or even a blackout. This category of attacks were termed as MadIoT attacks by Soltan et al. [42]. By using an anomaly detection system, we can detect these attacks effectively which is the goal we try to achieve in our work.

We generated attack profiles such that the attack times were not detected when using an anomaly detection system consisting of weighted average prediction method and an anomaly score with static thresholds as the deciding factor whether to flag an observation as an anomaly. We provided an extensive evaluation of the performance of anomaly detection systems that use a combination of a prediction method and an anomaly score or likelihood score along with static thresholds for detecting anomalies. We observed that static thresholds performed well in some cases but, were unable to adjust to seasonal changes and contextual changes in demand. Additionally, they were able to detect at most 50% of the anomalies on an average across all attack scenarios due to the prediction method's seasonal nature. Commercial methods also performed poorly in detecting attacks, because these methods are focused on detecting only point anomalies. We concluded that a thresholding technique should be able to detect attacks taking place during these situations, and also allow gradual changes in the threshold such that it can adapt to seasonal transitions as well as to genuine increase or decrease in demands due to external factors. We proposed a novel dynamic thresholding mechanism which could also handle situations that static thresholds and commercial methods could not. This

mechanism showed significant improvement in terms of detecting prolonged attack periods. We were able to detect collective anomalies effectively when compared to commercial applications that are focused on detecting point anomalies distributed over time. We also observed that more than 80% of the attacks were detected within the first minute using our proposed dynamic thresholding approach.

In our experiments, we used grid level consumption data which is an aggregate of the consumption of all apartments. The work can be extended to apartment level data, providing the grid operator a detailed view of which apartments have high alerts and whether they belong to the same community or area. This allows the grid operator to take an individualized contingency action only to that location rather than imposing it to other non-anomalous areas as well. We observed that we were having false positives with highly varying consumption data during the non-attack periods due to the external factors like seasonal changes. This shows that we need to carry out the learning process more accurately such that it can accommodate these seasonal transitions. By including automated parameter tuning targeted to improve the learning of highly varying consumption could aid in reducing the false positives. Also, including a parameter for the grid operator to adjust the dynamic threshold based on situations like holiday seasons, unexpected storms, etc. will also improve the performance of the detection system.

It was observed that research using dynamic thresholding for various application domains is rare. The existing works use mean and standard deviation of the historic samples. This work can also be directed and applied in other application domains which are based on time series. We used a data set which had consumption recorded for every minute. With the availability of consumption data at a granular level, this approach can be evaluated based on its performance on high voluminous data. This will also improve the attack simulation and detection process, as most of the MadIoT attacks require less than a minute to show its effects.

Collective anomaly detection is also a possible future direction in terms of detecting collective anomalies as early as possible and sustaining the detection for long periods of anomalous situations. We also observed that attacks were possible by exploiting the detection model and staying obscure during the attack periods. Adversarial machine learning was performed in this work to generate attack vectors that exploit the flexibility retained by a detection technique by not overfitting to historical data. This area also needs future work to be performed in the application of the power grid with the aim of modeling consumption data effectively.

Bibliography

[1] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised Real-time Anomaly Detection for Streaming Data. In *Neurocomputing*, volume 262, pages 134–147, 2017.

[2] Mohiuddin Ahmed and Abdun Naser Mahmood. Novel Approach for Network Traffic Pattern Analysis using Clustering-based Collective Anomaly Detection. *Annals of Data Science*, 2(1):111–130, 2015.

[3] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based Anomaly Detection and Description: A Survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2014.

[4] Daniel B Araya, K Grolinger, Hany F ElYamany, Miriam A. M Capretz, and G Bitsuamlak. Collective Contextual Anomaly Detection Framework for Smart Buildings. In *International Joint Conference on Neural Networks (IJCNN)*, pages 511–518. IEEE, 2016.

[5] Riyaz Ahamed Ariyaluran Habeeb, Fariza Nasaruddin, Abdullah Gani, Ibrahim Abaker Targio Hashem, Ejaz Ahmed, and Muhammad Imran. Real-time Big Data Processing for Anomaly Detection: A Survey. *International Journal of Information Management*, 45:289–307, 2019.

[6] F. Bezerra and J. Wainer. A Dynamic Threshold Algorithm for Anomaly Detection in Logs of Process Aware Systems. *Journal of Information & Data Management*, 3(3):316–331, 2012.

[7] C Chahla, H Snoussi, L Merghem, and M Esseghir. A Deep Learning Approach for Anomaly Detection and Prediction in Power Consumption Data. *Energy Efficiency*, 13:1633 – 1651, 2020.

[8] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. *ACM Computing Surveys (CSUR)*, 41(3):1–58, 2009.

[9] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. In *ACM Computing Surveys*, volume 41, pages 1–72, July 2009.

[10] Wenqiang Cui and Hao Wang. A New Anomaly Detection System for School Electricity Consumption Data. *Information (Basel)*, 8(4):151, 2017.

[11] Jisa David and Ciza Thomas. Efficient DDoS Flood Attack Detection using Dynamic Thresholding on Flow-based Network Traffic. *Computers & Security*, 82:284–295, 2019.

[12] Ghada Elbez, Hubert B. Keller, Atul Bohara, Klara Nahrstedt, and Veit Hagenmeyer. Detection of DoS Attacks Using ARFIMA Modeling of GOOSE Communication in IEC 61850 Substations. *Energies*, 13(19):5176, 2020.

[13] Barbara Filkins and Doug Wylie. The 2018 SANS Industrial IoT Security Survey: Shaping IIoT Security Concerns. Corpus ID: 51842821, 2018.

[14] Alexander T. M. Fisch, Idris A. Eckley, and Paul Fearnhead. A Linear Time Method for the Detection of Point and Collective Anomalies, 2019.

[15] M. C. Hao, H. Janetzko, S. Mittelstädt, W. Hill, U. Dayal, D. A. Keim, M. Marwah, and R. K. Sharma. A Visual Analytics Approach for Peak-Preserving Prediction of Large Seasonal Time Series. *Computer Graphics Forum*, 30(3):691–700, 2011.

[16] Shah Haque, Mustafizur Rahman, and Syed Aziz. Sensor Anomaly Detection in Wireless Sensor Networks for Healthcare. *Sensors (Basel, Switzerland)*, 15(4):8764–8786, 2015.

[17] Victoria Hodge and Jim Austin. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22:85–126, 2004.

[18] Junho Hong, Chen-Ching Liu, and Manimaran Govindarasu. Integrated Anomaly Detection for Cyber Security of the Substations. *IEEE Transactions on Smart Grid*, 5(4):1643–1653, 2014.

[19] Rob J Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts: Melbourne, Australia, 2nd edition, 2018.

[20] Mahdi Jamei, Anna Scaglione, Ciaran Roberts, Emma Stewart, Sean Peisert, Chuck McParland, and Alex McEachern. Anomaly Detection Using Optimally-Placed ÎŒPMU Sensors in Distribution Grids. *IEEE Transactions on Power Systems*, 33(4):3611–3623, 2017.

[21] Halldor Janetzko, Florian Stoffel, Sebastian Mittelstädt, and Daniel Keim. Anomaly Detection for Visual Analytics of Power Consumption Data. *Computers and Graphics*, 38:27–37, 2014.

[22] Yexi Jiang, Chunqiu Zeng, J. Xu, and Tao Li. Real time contextual collective anomaly detection over multiple data streams. 2014.

[23] Hadis Karimipour, Sandra Geris, Ali Dehghantanha, and Henry Leung. Intelligent Anomaly Detection for Large-scale Smart Grids. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pages 1–4. IEEE, 2019.

[24] Meng Li, Keli Zhang, Jiamou Liu, Hanxiao Gong, and Zijian Zhang. Blockchain-based Anomaly Detection of Electricity Consumption in Smart Grids. *Pattern Recognition Letters*, 138:476–482, 2020.

[25] Carrie MacGillivray and David Reinsel. Worldwide Global DataSphere IoT Device and Data Forecast, 2019–2023. IDC Doc # US45066919, 2019.

[26] Ramin Moghaddass and Jianhui Wang. A Hierarchical Framework for Smart Grid Anomaly Detection Using Large-Scale Smart Meter Data. *IEEE Transactions on Smart Grid*, 9(6):5820–5830, 2018.

[27] Naser Hossein Motlagh, Mahsa Mohammadrezaei, Julian Hunt, and Behnam Zakeri. Internet of Things (IoT) and the Energy Sector. 13(2):494, 2020.

[28] B.S Murugan, Mohamed Elhoseny, K Shankar, and J Uthayakumar. Region-based Scalable Smart System for Anomaly Detection in Pedestrian Walkways. *Computers & Electrical Engineering*, 75:146–160, 2019.

[29] Nga Nguyen Thi, Van Loi Cao, and Nhien-An Le-Khac. One-Class Collective Anomaly Detection Based on LSTM-RNNs. In *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXVI*, volume 10720 of *Lecture Notes in Computer Science*, pages 73–85. 2017.

[30] Zhiyou Ouyang, Xiaokui Sun, Jingang Chen, Dong Yue, and Tengfei Zhang. Multi-View Stacking Ensemble for Power Consumption Anomaly Detection in the Context of Industrial Internet of Things. *IEEE Access*, 6:9623–9631, 2018.

[31] Daehyung Park, Hokeun Kim, and Charles C Kemp. Multimodal Anomaly Detection for Assistive Robots. *Autonomous Robots*, 43(3):611–629, 2018.

[32] Simon Parkinson, Mauro Vallati, Andrew Crampton, and Shirin Sohrabi. GraphBAD: A General Technique for Anomaly Detection in Security Information and Event Management. *Concurrency and Computation: Practice and Experience*, 30(16):e4433, 2018.

[33] Federico Passerini and Andrea M Tonello. Smart Grid Monitoring Using Power Line Modems: Anomaly Detection and Localization. *IEEE Transactions on Smart Grid*, 10(6):6178–6186, 2019.

[34] I. Gethzi Ahila Poornima and B Paramasivan. Anomaly Detection in Wireless Sensor Network using Machine Learning Algorithm. *Computer Communications*, 151:331–337, 2020.

[35] Stephen Ranshous, Shitian Shen, Danai Koutra, Steve Harenberg, Christos Faloutsos, and Nagiza F Samatova. Anomaly Detection in Dynamic Networks: A Survey. *Wiley Interdisciplinary Reviews. Computational statistics*, 7(3):223–247, 2015.

[36] B. Rossi, S. Chren, B. Buhnova, and T. Pitner. Anomaly Detection in Smart Grid data: An Experience Report. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 002313–002318, 2016.

[37] Osman Salem, Yaning Liu, Ahmed Mehaoua, and Raouf Boutaba. Online Anomaly Detection in Wireless Body Area Networks for Reliable Healthcare Monitoring. *IEEE Journal of Biomedical and Health Informatics*, 18(5):1541–1551, 2014.

[38] Xavier Serrano-Guerrero, Guillermo Escrivá-Escrivá, Santiago Luna-Romero, and Jean-Michel Clairand. A Time-Series Treatment Method to Obtain Electrical Consumption Patterns for

Anomalies Detection Improvement in Electrical Consumption Profiles. *Energies*, 13(5):1046, 2020.

[39] Liang Shouyu, Kun Zhang, Wenchong Fang, Zhifeng Zhou, Rong Hu, Wen Zhu, Yingchen Li, Yichang Wang, and Jian Hou. Anomaly Detection of Power Grid Dispatching Platform based on Isolation Forest and K-means Fusion Algorithm. *Journal of Physics: Conference Series*, 1601:22010, 2020.

[40] Nurhidayat Sisworahardjo and Akram A. Saad. Spatio - Temporal Context Anomaly Detection for Residential Power Consumption. *International Journal on Electrical Engineering and Informatics*, 9(4):776–785, 2017.

[41] Taylor SJ and Letham B. Forecasting at Scale. *PeerJ Preprints*, 2017.

[42] Saleh Soltan, Prateek Mittal, and H. Vincent Poor. BlackIoT: IoT Botnet of High Wattage Devices Can Disrupt the Power Grid. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 15–32. USENIX Association, 2018.

[43] Twitter. Introducing Practical and Robust Anomaly Detection in a Time Series. In *Engineering Blog*, 2015.

[44] Francesco Verdoja and Marco Grangetto. Graph Laplacian for Image Anomaly Detection. *Machine Vision and Applications*, 31(1):11, 2020.

[45] Y. Weng, N. Zhang, and C. Xia. Multi-Agent-Based Unsupervised Detection of Energy Consumption Anomalies on Smart Campus. *IEEE Access*, 7:2169–2178, 2019.

[46] Qiang Yang, Weijie Hao, Leijiao Ge, Wei Ruan, and Fujian Chi. FARIMA Model-based Communication Traffic Anomaly Detection in Intelligent Electric Power Substations. *IET Cyber-Physical Systems: Theory & Applications*, 4(1):22–29, 2019.

[47] Dit-Yan Yeung and Yuxin Ding. Host-based Intrusion Detection using Dynamic and Static Behavioral Models. *Pattern Recognition*, 36(1):229–243, 2003.

[48] Pushe Zhao, Masaru Kurihara, Junichi Tanaka, Tojiro Noda, Shigeyoshi Chikuma, and Tadashi Suzuki. Advanced Correlation-based Anomaly Detection Method for Predictive Maintenance. In *IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 78–83. IEEE, 2017.

[49] Yu Zheng, Huichu Zhang, and Yong Yu. Detecting collective anomalies from multiple spatio-temporal datasets across different domains. In *23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '15, pages 1–10. ACM, 2015.

[50] Shifu Zhou, Wei Shen, Dan Zeng, Mei Fang, Yuanwang Wei, and Zhijiang Zhang. Spatial-Temporal Convolutional Neural Networks for Anomaly Detection and Localization in Crowded Scenes. *Signal processing: Image communication*, 47:358–368, 2016.