https://afrjcict.net

A Framework for End-Point Input Validation against Faults and Attacks in Big Data

Aliyu Omeiza[1], Oluyemi E. Amujo[2] and M. B. Hammawa[3]
Department of Computer Science,
University of Abuja,
Nigeria.
Email: [1]aomeiza@me.com,
[2]oluyemiamujo@gmail.com,
[3]mohammed.hammawa@uniabuja.edu.ng

ABSTRACT

A key challenge in big data collection process is input validation/filtering which is daunting due to untrustworthy input sources, especially with the "bring your own device" (BYOD) model. This paper presents a model and algorithms for input validation/filtering of large data sets using algorithms such as heuristic rule used for detecting out-of-range faults, temporal correlation method used for detecting struck-at faults, and modified z-score method used for detecting outliers and spike faults. The research work contributes in two ways; first, a novel algorithm for identification and validation/filtering of endpoint node against spoofing attack was proposed. Second, those existing algorithms for sensor data validation were extended and integrated into a single and robust algorithm for endpoint input validation that we sought. The implementation of our algorithms using Python programming language with open data from weather sensors shows that the proposed data validation algorithm is effective in detecting different types of data faults and reports high fault detection rate by eliminating false positives.

Keywords: *Big data, Filtering, Identity, Analytics, Internet of Things (IoT), Validation.*

African Journal of Computing & ICT Reference Format:

Aliyu Omeiza, Oluyemi E. Amujo and M. B. Hammawa (2019),
A Framework for End-Point Input Validation against Faults and Attacks in Big Data,
Afr. J. Comp. & ICT, Vol.12, No. 4, pp. 37 - 58 .

I. INTRODUCTION

Big data is similar to small data, but bigger. Big data is the outcome of various data collected from various devices through various means. It is unavoidable due to high rate of manufacturing of IoT devices [1]. Sensors that are so small and efficient that they can power themselves with ambient radio waves are becoming a reality, making the growth of data unstoppable [2]. As a result, data is becoming cheaper to collect and keep and our ability to analyze all of this data is constantly improving.

It is obvious that big data and the Internet of Things have potentially tremendous benefits. Cities can better maintain their infrastructures by developing sophisticated early warning systems for dangerous situation. Medical researchers can enrol patients in large-scale research

projects and collect streams of useful data instead of relying on data from surveys and patients' own reports [3]. Connected and online courses are making it possible for people all over the world to learn and earn degrees from the world's leading experts and most prestigious institutions. Data is helping governments to better plan and deliver their services [4]. Big data technologies are designed to economically extract value from very large volumes of a wide variety of data by enabling high-velocity capture, discovery and/or analysis [2]. Big Data is not just about the technology; it is also about the people and business processes that help companies gain competitive advantage.

However, security of data collection, among other security challenges pointed out in [5], is very important since it is the first aspect in big data management. Often times, big data collected are faulty due to internal and external influences, such as environment effects, limitations of resources, power problems, hardware malfunctions, software problems, network problems and security attacks. This challenge becomes more severe with the utilization of the "bring your own device" (BYOD) model [6]. Also, aside data validation issue as it is common to classical data, the integrity of data source (node) must be ascertained to ensure that data only come from credible sources.

This paper is aimed at developing a framework and algorithm for big data endpoint input validation/filtering using data collected from weather sensor made available online for research purpose. In order to fulfil the aim, two strategies for endpoint input data filtering/validation were compared and the existing algorithms for data validation were extended and integrated into a single and robust algorithm for endpoint input validation that we sought. Furthermore, our extended algorithm ensures that data that are filtered in are from only credible sources.

The motivation for the work comes from the paper released by [6]. The proposed paper outlined the current challenges faced by big data. The challenges comprise top ten issues that need urgent solutions. Among the outlined challenges, the one that motivates the work is the need for input validation/filtering techniques for big data noting that it is the first line of defense to big data system and security. Furthermore, in a paper by [11], characteristics of big data and security issues were discussed in which various solutions were suggested for endpoint input validation and filtering, among are tampered-proof software, cryptographic protocols, analytics to detect outliers, trust certificate and trusted devices. The one that interest us among them is analytics to detect outliers and other faults of input data, meaning algorithm and techniques are required to detect faults in data collected from endpoint device and also detect if the device itself has been spoofed.

II. LITERATURE REVIEW

2.1 Big Data Security Challenges

Some significant risks go along with the potential benefits of connected devices and big data. As we add devices to our homes, classrooms, and clothes, much more sensitive data will be collected. User interfaces on devices will shrink or disappear, making it more difficult for consumers to know when data is being collected, or to exercise any control [7]. These developments pose difficult challenges for privacy, security, and fairness in our society. The data from connected devices will be deeply personal, and big data analytic will make the data more readily actionable. Some of these devices will handle deeply sensitive information about our health, our homes, and our families. Some will be linked to our financial accounts; some to our email accounts. And devices themselves will be more closely connected with our actions in the physical world, making data security and device security critically important.

2.1.1 Data Accuracy Concern

The systematic approach toward data collection in order to enhance randomness in data sampling and reduce bias is not apparent in the collection of big data sets. The data collected can still be incomplete and distorted which, in turn, can lead to skewed conclusions. Consider the case of Twitter which is commonly scrutinised for insights about user sentiments. There is an inherent problem with using Twitter as a data source as only 40% of Twitter's active users are merely listening and not contributing [8]. This may suggest that the tweets come from a certain type of people (probably people who are more vocal and participative in social media) than from a true random sample.

2.1.2 People and Business Needs Concern

Big Data is not just about the technology; it is also about the people and business processes. There is a danger that Big Data adopters are missing the bigger picture by excluding the discussions around the people and business processes. Before adoption, companies must first evaluate the business case and specific outcomes of their proposed Big Data initiatives [9]. They would need to know what to

ask of the data, assess how the business will react to it and be able to offer actionable operational measures. Developing a roadmap of how to achieve the desired business outcomes will give the organisation the understanding of what is required and enable it to be prepared financially and organizationally [18].

2.1.3 Data Privacy Concern

Big data can also raise privacy concerns and reveal unintended information. A research described in [9] has found that anonymized data from a social network platform such as Flickr, combined with readily available data from other online sources such as Twitter, can render the data de-anonymized and reveal sensitive information about a person. Security and privacy issues are magnified by the volume, variety, and velocity of big data [19]. Large-scale cloud infrastructures, diversity of data sources and formats, the streaming nature of data acquisition and high-volume, inter-cloud migration all play a role in the creation of unique security vulnerabilities

2.2 Big Data Security Mechanism

Information is emerging at volatile rate, coming into organization from diverse areas and in numerous formats. Big Data validation is not only around validation of just what is different; it's also about validation of new integrated components to what you already have. In order to sustain growth, it was suggested in [10] that enterprises adopt next generation data integration platforms and techniques that fuel the demand for quality assurance mechanisms around the new data perspectives. Also, enterprises should consider their big data validation approach and lay it in a strategic manner [10]. Furthermore, it was argued in [11] that many big data scenario in enterprise setting require information gathering from a few sources like end-point devices. Thus it becomes fundamental to confirm the data itself along with the basis of data such as the level of conviction of the data, the mechanism to utilize to confirm the foundation of the data is not spiteful, and how to eliminate spiteful data as of the accumulated data. Thier study only specifies the mitigation to threats and challenges based on data poisoning scenario, the framework is not detailed for instance to give the algorithms for the outlined mitigation. Therefore, to apply this framework requires 'more tasks of developing the algorithms.

A novel algorithmic framework for clustering massive numbers of high-dimensional data with sketching and validation (SkeVa) family were introduced by [12]. The first two members, a batch and a sequential one tailored to streaming modes of operation, required means clustering in low-dimensional spaces and/or a small number of data-points. To enable clustering of even non-linearly separable data, a third member of the family leveraged the kernel trick to cluster linearly separable mapped data. A divergence metric was utilized to develop the fourth member of SkeVa K-means that bypasses intermediate means clustering to trade-off accuracy for reduced complexity. However, it is essential to note that the algorithms are only applicable to data at rest or in a repository. In other words, the algorithms are applicable to sketch and validate big data clusters. The work fails to develop algorithm for end-point data validation/filtering. The problem with [12] is that incoming data can contain malicious data or the end-point devices may be spoofed.

Furthermore, data quality is a thorny issue in most Big Data projects. It's been reported that more than half of the time spent in Big Data projects goes towards data cleansing and preparation [13]. A close examination of the data validation functions typically used reveal that they can broadly be classified under two groups: a broad set of commonly used functions: A highly nuanced customs set of functions, very closely tied with specific business rules in the domain of the application; common out of the box validators which are available under different categories. The following validators are generic in nature and do not depend on the field data type;

- notMissing - Ensures field is not missing
- ensureInt - Ensures field value is integer
- ensureLong - Ensures field value is long
- ensureDouble - Ensures field value is double
- ensureDate - Ensures field value is a date
- minLength - Ensures field length is greater than or equal to specified length
- maxLength - Ensures field length is less than or equal to specified length
- exactLength - Ensures field length is equal to specified length
- min - Ensures that field is lexicographic-ally greater than or equal specified string
- max - Ensures that field is lexicographic-ally less than or equal to specified string
- pattern - Ensures that field matches given pattern
- zscoreBasedRange - Validates range based on mean and std deviation
- robustZscoreBasedRange - Validates range based on median and median average deviation
- membership - Ensures field value is member of the categorical value set

2.3 Endpoint Input Data Faults

i. Spoofing and Sybil attack: In Sybil, malicious node consist one or more fake identity and try to establish confidence among linked nodes [14] while a spoofing attack is a one in which one person or program successfully masquerades as another by falsifying data and thereby gaining an illegitimate advantage [15]. It arises when the attacker is capable to root a user or a device on a system to believe that a piece of information came from a source from which it actually did not originate.

ii. Out-of-range faults: sensor data samples that deviate significantly from expected range of values are called out-of-range faults. Out-of-range faults represent sensor values that are physically not possible in the deployed region.

ii. Struck-at fault: series of data samples with little or no variation for a period of time greater than expected are called struck-at faults. Data is frozen or remains to a given value. It can be within or outside the expected range of values. Struck-at fault is also called constant fault [14].

iii. Outliers: outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism [16].

iv. Spike fault: with spike fault, the rate of change in gradient of data samples over period of time is much greater than expected. It occurs in combination of at least few successive data samples [17].

III. RESEARCH METHODOLOGY

3.1 Notations

Here, we present the lists of the notations used in algorithms in the work.

x - Sensed data
[] - Array/set of sensed data
n - Number of sensed data in array x
fault location [] - Array for fault locations
status [] - Array for status of sensed data
δ - Threshold value
$\bar{x}$ - Sample mean
$\tilde{x}$ - Sample median
σ - Sample standard deviation
α - Confidence coefficient
$\sigma_{critical} = \sigma \times \alpha$ - Critical deviation
$\sigma_{observed} = |x[i] - \bar{x}|$ - Observed deviation
M - Modified z-score
MAD - Median absolute deviation

3.2 Description of Existing Algorithms

The existing algorithm for endpoint (sensor) input validation of big data [14] comprised heuristic, temporal and spatial algorithms.

Heuristic rule is used to check data samples for faults. It is a commonsense rule (or set of rules) intended to increase the probability of solving some problems such as data filtering. Heuristic algorithm makes use of no statistical model rather it filters data element against a certain threshold. If sensed data x is within threshold limit ($\delta_{\min}$ and $\delta_{\max}$) then data x is likely good and it is stored in a location meant for good data else likely fault and sent to another location meant for faulty data. Threshold limit is based on domain knowledge.

Temporal correlation, which is a statistical model, is used to filter data samples collected by same sensor node. If the difference among successive data samples remains zero for multiple instances then the data samples are in struck-at faults. Consider a node N that sends data sample x_1, x_2, x_3… if $(x_1 - x_2) = 0$, $(x_2 - x_3) = 0$, $(x_3 - x_4) = 0$ and so on then there exists a struck-at faults in the data sets.

Lastly, spatial correlation also makes use of statistical method to filter data samples of neighbor sensor nodes by checking for faults. Let us consider sensor node N_i and N_j as neighbors. Let x_i and x_j be values reported by N_i and N_j at a time t. Let $\hat{x}$ be expected value at N_i based on x_j reported by N_j. If ($|x[i] - \hat{x}_i| < \delta$) then data x is likely good else likely faulty.

Having reviewed the existing algorithms, we noted that they are not robust enough to cater for certain data input attacks such as spoofing and Sybil attacks. The reason is because data sources are not being put into consideration. All the algorithms focus on data value as the only parameter for filtering and validation. There should be a mapping, as in our case, between the sensor identity and its collected/transmitted data. This helps to identified if data is sent from unknown source and/or if the sensor is hijacked by adversary.

Also, the existing algorithms are not well integrated which may make the implementation difficult. We solved the problem in a holistic manner by developing an integrated algorithm. Accurate filtering of data is achieved by allowing data to flow from one validation

section to another in a perfectly consistent and coherent manner.

3.1 The Proposed Big Data End Point Validation/Filtering Model

We present the proposed model in Figure 4.1 with focus on validation/filtering engine which receive data from node $N_1 \ldots N_m$. the data received contain the node id and the main data to validate.

The first aspect is the application of id_filtering rule that makes sure that the data only comes from genuine sources while other rules validate the main data once the source identity has been certified. The model ensures that only validated and trusted data are transferred to the next part such as data analytical engine.

3.2 Proposed Algorithms

Our proposed id_filtering algorithm presented in Algorithm 4.1 uses Heuristic rule to validate source identity with the assumption that deployed nodes are identified and the identity value are stored as part of the procedure. The data collected from sensor node comprises of data and metadata. The metadata is the information about the data which tells the identity number, date and time delivered etc. We then validate node_id by comparing with each element in the array of stored node id. If it is the same with any id in the stored array then it can be trusted otherwise not.

Since this is the first point of validation, therefore it is only data from trusted sources that will be passed on for data validation.

Input: array of incoming node_id (n), array of known node_id (m)
Output: status for node_id (n)
$\text{count}_{\text{likely fake}} = 0$;
for $i \leftarrow 0$ to j do $i \leftarrow i + 1$:
    if (n[i] = m[i]):
        status [i] ←likely trusted
        trusted location [$\text{count}_{\text{likely trusted}}$] ← i
    else:
        status [i] ←likely fake
        fake location [$\text{count}_{\text{likely faults}}$] ← i
        $\text{count}_{\text{likely faults}} \leftarrow \text{count}_{\text{likely faults}} + 1$
    end if
end

Algorithm 4.1: Heuristic rule for source validation/filtering

Furthermore, we extend validation/filtering algorithms in such a way that data from only trusted sources are passed in for validation. This is achieved by modifying the data source. Rather than specifying node as source we specified that the data come from the outcome of the id_filtering in Algorithm 4.1. We present an integrated algorithm which comprises multiple methods as demonstrated in Algorithm 4.2. The algorithm allows data to pass through different logic stages and the faulty ones will be filtered off and stored in separate files based on the type of fault they belong to. The first aspect validates data based on source identity and filtered off the data if from unknown source. The rest of the data proceed to the next stage and so on, until faultless data is obtained.

Input: array of incoming node_id (n), array of known node_id (m)
Output: status for node_id (n), status for sensed data (x)
$\text{count}_{\text{likely fake}} = 0$; $\text{count}_{\text{out-of-bound faults}} = 0$;
$\text{count}_{\text{outliers}} = 0$; $\text{count}_{\text{spike faults}} = 0$;
$\text{count}_{\text{likely faults}} = 0$; $\text{count}_{\text{struck-at faults}} = 0$;
//node id filtering using heuristic rule
for $i \leftarrow 0$ to j do $i \leftarrow i + 1$:
    if (n[i] = m[i]):
        status [i] ←likely trusted
        trusted location [$\text{count}_{\text{likely trusted}}$] ← i
    else:
        status [i] ←likely fake
        fake location [$\text{count}_{\text{likely faults}}$] ← i
        $\text{count}_{\text{likely faults}} \leftarrow \text{count}_{\text{likely faults}} + 1$
    end if
end
//out-of-range faults are detected using heuristic rule
for $i \leftarrow 0$ to n do $i \leftarrow i + 1$:
    if ($x[i] \geq \delta_{\text{minimum}}$ and $x[i] \leq \delta_{\text{maximum}}$):
        status [i] ←likely good
    else:
        status [i] ←likely fault
        fault location [countlikely faults] ← i
        countlikely faults ← countlikely faults + 1
    end if
end
//struck-at faults are detected using temporal correlation

for $i \leftarrow 0$ to n do $i \leftarrow i + 1$:
    if ($|x[i] - x[i+1]| = 0$):
        fault location [$\text{count}_{\text{likely faults}}$] $\leftarrow i + 1$
        $\text{count}_{\text{likely faults}} \leftarrow \text{count}_{\text{likely faults}} + 1$
    else:
        status [i] ←likely good
        status [$i + 1$] ←likely good
    end if
end
for $j \leftarrow 0$ to $\text{count}_{\text{likely faults}}$:
    $j \leftarrow j + 1$
    $i \leftarrow$ fault location [j]
    $i + 1 \leftarrow$fault location [$j + 1$]
    if ($|x[i] - x[i+1]| = 0$):
        status [$i + 1$] ←struck-at fault
        fault $\text{location}_{\text{struck-at faults}}$ [$\text{count}_{\text{struck-at faults}}$] $\leftarrow i + 1$
        $\text{count}_{\text{struck-at faults}} \leftarrow \text{count}_{\text{struck-at faults}} + 1$
    else:
        status [i] ←likely good
        status [$i + 1$] ←likely good
    end if
end
//outliers & spike faults are detected using modified z-score
for $i \leftarrow 0$ to n:
    $i \leftarrow i + 1$
    $\mathrm{M_i} = \mathrm{real_{val}} \times (|\mathrm{x[i]} - \bar{\mathrm{x}}|)/\mathrm{MAD}$
    If ($|M_i| > \delta$):
        status [i] ←likely fault
        fault location [$\text{count}_{\text{likely faults}}$] $\leftarrow i$
        $\text{count}_{\text{likely faults}} \leftarrow \text{count}_{\text{likely faults}} + 1$
    else:
        status [i] ←likely good
    end if
end
for $j \leftarrow 0$ to $\text{count}_{\text{likely faults}}$:
    $j \leftarrow j + 1$
    $i \leftarrow$ fault location [j]
    $i + 1 \leftarrow$fault location [$j + 1$]
    if (|fault location [i] − fault location [$i + 1$]| = 1) :
        status [$i + 1$] ←spike fault
        fault $\text{location}_{\text{spike faults}}$ [$\text{count}_{\text{spike faults}}$] $\leftarrow i + 1$
        $\text{count}_{\text{spike faults}} \leftarrow \text{count}_{\text{spike faults}} + 1$
    else:
        status [$i + 1$] ←likely good
    end if
end

Algorithm 4.2: Novel data validation algorithm.

IV. IMPLEMENTATION AND EVALUATION

4.1 Development

The data flow diagram presented in Appendix I shows the logical design of the algorithms. The algorithms were implemented in Python programming language. The reason for choosing Python is because it is one of the favorite languages for data analysis. Also, it is the favorite language for IoT development platform such as Raspberry Pi.

4.2 File Structure

The Python codes used for the implementation of our big data validation/filtering algorithms is structured in modules for maintainability. The modules representing the algorithms are coded in separate files which are then imported into the main module. Some of the modules we care about are:

- **heuristic_val.py:** The file that contains the Python codes that implements heuristic algorithm that filters out data from unknown sources and validates data against out-of-range error. The program then direct faulty data to a file named fake_id_send_data.txt for data from fake sensor identity and out_of_range_err_data_file.txt for data that contains out-of-range error.

```
#Heuristic Algorithm
sensed_fileObj = open('sensed_data_file.txt', 'r')
good_fileObj = open('good_data_file.txt', 'w')
faulty_fileObj = open('faulty_data_file.txt', 'w')
count_likely_fault = 0
sensed_reclst = sensed_fileObj.readlines()
for data in sensed_reclst :
   if float(data) >= -8.3 and float(data) <= 16.1:
      #print(data, "is likely good")
      good_fileObj.write(data)
   else:
      #print(data, "is likely faulty")
      good_fileObj.write(data)
sensed_fileObj.close()
good_fileObj.close()
     faulty_fileObj.close()
```

- **temporal _cor_val.py:** The file that contains the Python codes that implements temporal algorithm for filtering data against struck-at faults. The program

then direct faulty data to a file named temporal_cor_err.txt.

- **modified_z_core_val.py:** The file that contains the Python codes that implement modified z core for filtering data against outliers and spike faults. The program then direct faulty data to a file named outliers_err.txt and spike-Err.txt respectively.

4.3 Data Source

Climate sensor dataset is used for algorithm evaluation. The data is sourced from National Oceanic and Atmospheric Administration (NOAA)'s Archives site (size 10 to 1000 samples). The dataset, stored in a file separate from the program code that implements the algorithm. At run time, the code fetches the datasets for validation. For simplicity, we only make use of temperature data set injected with different data faults already described in Sub-section 2.3. The threshold (δminimum and δmaximum) is found out to be 23.90^{o}C and 28.60^{o}C respectively. Also for simplicity, we would assume the data was collected from five (5) sensors with ID01, ID02, ID03, ID04 and ID05. The program therefore separates data into separate files depending on the status of data.

4.4 Evaluation of Proposed Algorithm

We consider fault detection rate and false positive rate as the criteria for evaluation of the algorithms. Fault detection rate (FDR) is the ratio between numbers of correctly detected data faults and total number of data faults. The result of data volition process can also be false positive or true positive. Faulty data misclassified as normal data is called false positives otherwise it is true positive.

The receiving data from sensor nodes and data is stored in the form of tables with time stamps. Data validation algorithm discussed in Section 3.6 is used in sensor nodes and base station to check sensed data for fault and only valid data is forwarded to the big data analytical engine. Algorithms are evaluated using temperature sensor data set (size 10 to 1000 samples) of WSNs prototype for environment monitoring injected with different types of data faults such as out-of-range faults, struck-at faults, outliers, and spike faults. Minimum and maximum possible temperatures in the deployed region are set as threshold limit for validating temperature sensor data.

Let the threshold limit (δminimum and δmaximum) be 18 to 45∘C.
Let us consider the following cases for evaluation.

Case 1. Data set with 30% out-of-range faults.
Case 2. Data set with 30% struck-at faults.
Case 3. Data set with 10% outliers and spike faults.
Case 4. Data set with 30% data faults which include 10% out-of-range faults and 20% outliers and spike faults.
Case 5. Data set with 40% data faults which include 20% out-of-range faults and 20% outliers and spike faults.

4.5 Discussion

Figure 4.2 illustrates results of Algorithm 1 in Cases 1 to 5. FDR of Algorithm 1 is 100% in Case 1, 0% in Case 2, 0% in Case 3, 33% in Case 4 and 50% in Case 5. Result in Case 1 shows that Algorithm 1 is effective in detecting out-of-range faults and the results in other cases show; presence of struck-at faults; outliers and spike faults are not detected by Algorithm 1.

Figure 4.3 illustrates results of Algorithm 2 in Cases 1 to 5. FDR of Algorithm 2 is 0% in Case 1, 100% in Case 2, 0% in Case 3, 0% in Case 4 and 0% in Case 5. Result in Case 2 shows that Algorithm 2 is effective in detecting struck-at fault and the results in other cases show presence of out-of-range faults; outliers and spike faults are not detected by Algorithm 2.

Figure 4.4 illustrates results of Algorithm 3 in Cases 1 to 5. FDR of Algorithm 3 is 100% in Case 1, 0% in Case 2, 100% in Case 3, 100% in Case 4 and 100% in Case 5. Result in Case 2 shows that presence of struck-at faults is not detected by Algorithm 3 and the results in other cases show that Algorithm 3 is effective in detecting out-of-range faults and outliers and spike faults. Communication and computation cost for Algorithm 3 will be high compared to other algorithms because it depends on neighboring node data samples for detecting faults.

V. CONCLUSION

Having discovered that no single method is perfect in detecting different types of data faults and reports false positives when data set contains different types of data faults, the research therefore presents a novel model for end-point validation for big data. This model contributes to the existing algorithms in two ways. First, as shown in Figure 4.1 and demonstrated in Algorithm 4.1, the ID of each end-point is incorporated and considered the starting point of validation and filtering. Second, the existing algorithms and the one just developed in Algorithm 4.1 are combined into a single and robust solution with various modules that filters data from various faults as shown in Algorithm 2.

Compared to other methods, the proposed novel data validation algorithm is effective in detecting different types of data faults and reports high fault detection rate by eliminating false positives. Therefore the proposed novel data validation algorithm is desirable to apply at sensor nodes and base station to effectively eliminate different types of data faults.

Furthermore, mitigating the security challenges requires new techniques, the old rules no longer apply and because of nature and characteristics of the concept; the existing algorithms for data validation and filtering are not capable enough to handle the challenges. In addition to that, the existing cyber security frameworks, models and policies, all have to be extended and improved to cater for big data security which has eventually open new grounds for research. In view of this, we present model and algorithms for validation/filtering of endpoint node against faults and attacks.

REFERENCES

[1] E. Dave, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," 2011 CISCO White Paper. [Online]. Available: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf. [Accessed Aug. 10, 2017].

[2] P. Carter, "Big Data Analytics: Future Architectures, Skills and Roadmaps for the CIO". A White Paper by IDC Asia Pacific, 2011.

[3] E. Whitman, "Is New Open-Source Software for Sales or the Greater Good of Health Care?" Apple ResearchKit Intl. Business Times 16 March, 2015. [Online]. Available: http://www.ibtimes.com/appleresearchkit-new-open-source-software-sales-or-greater-good-health-care-1848612. [Accessed 10 Aug., 2017].

[4] B. Casselman, "Big Government is getting in the way of Big Data," FiveThirtyEight Economic Statistics, Mar. 9, 2015. [Online]. Available: http://fivethirtyeight.com/features /big-government-is-getting-in-the-way-of-big-data/. [Accessed on August 10, 2017]

[5] B. Ryan, "100 Best Practices in Big Data Security and Privacy," [Online]. Available: https://blog.cloudsecurityalliance.org/2016/08/26/100-best-practices-big-data-security-privacy/ [Accessed 20 Nov., 2017].

[6] Cloud Security Alliance (CSA) Top ten Big Data security and Privacy challenges - Service Implementation Guidance Version 1.0., 2012.

[7] J. Brill, "Privacy and Data Security in the Age of Big Data and the Internet of Things" - Keynote Address at Washington Governor Jay Inslee's Cyber Security and Privacy Summit, Jan. 5 2016, [Online]. Available: https://www.ftc.gov/system/files/document/public_statement/904973/160107wagovprivacysummit.pdf [Accessed 10 Nov., 2018].

[8] R. Matt, "Twitter Has 100 Million Active Users – And 40% Are Just Watching," [Online]. Available: http://articles.businessinsider.com/2011-09-08/tech/30127585_1_ ceo-dick-costolo-twitter-users. [Accessed 9 Jul., 2016].

[9] E. Naone, "Unmasking Social-Network Users," MIT Technology review, May 2009. [Online]. Available: http://www.technologyreview.com/web/22593/. [Accessed 9 July, 2016].

[10] A. Mittal, "Trustworthiness of Big Data," International Journal of Computer Applications. Volume 80 – No.9, Oct. 2013. pp. 0975 – 8887.

[11] S. H. Bindu, O. Gireesha, A. N. Sahithi, and A. Mounicama, "Security Aspects in Big Data," International Journal of Innovative Research in Computer and Communication Engineering - An ISO 3297: 2007 Certified Organization, Vol. 4, Issue 4, 2016, pp. 1111 – 1118.

[12] P.A. Traganitis, K. Slavakis, and G.B. Giannakis, "Sketch and Validate for Big Data Clustering," IEEE Journal Of Selected Topics In Signal Processing, VOL. 9, No. 4., pp. 678 – 690, Jun 2015.+

[13] G. Pranab, "Validating Big Data". Mawazo, 28 July, 2015. [Online]. Available: https://pkghosh.wordpress.com

/2015/07/28/validating-big-data. [Accessed 16 June, 2017].

[14] R. Jaichandran and I.A. Anthony, "Data Validation Algorithm for Wireless Sensor Networks". International Journal of Distributed Sensor Networks. Vol.13, Article ID 634278, http://dx.doi.org/10.1155/2013/624278, pp. 100 - 111, Dec 2013.

[15] J. Keshav, D. Surjeet, and K.S. Kamal, "Analyzing Spoofing Attacks in Wireless Networks," Paper presented at International Conference on Advanced Computing & Communication Technologies, Rohtak, India. 978-1-4799-4910-6/14. DOI 10.1109/ACCT.2014.46.

[16] K. Hans-Peter, P. Kröger, A. Zimek, "Outlier Detection Techniques". The 2010 SIAM International Conference on Data Mining, Ludwig-Maximilians-Universität, Munich, Germany, 2010, pp. 101-117.

[17] K. Ni, N. Ramanathan, M.N.H. Chehade, "Sensor network data fault types," ACM Transactions on Sensor Networks, vol. 5, No. 3, article 25, 2009.

[18] M. Rosoff, "Twitter Has 100 Million Active Users – And 40% Are Just Watching," [Online] Available from: http://articles.businessinsider.com/2011-09-08/tech/30127585_1_ceo-dick-costolo-twitter-users [Accessed 9th July 2012].

[19] G.B. Tarekegn and Y.Y. Munaye, "Big Data: Security Issues, Challenges and Future Scope," International Journal of Computer Engineering & Technology (IJCET), Volume 7, Issue 4, pp. 12–24, 2016.

Authors' Biodata

OMEIZA, Aliyu Omeiza holds Master of Science (M.Sc.) degree in Computer Science (Information Security as a Major) from University of Abuja, Nigeria. He received a BSc (Hons) Degree in Computer Science from the same University. His research interests include; Information Security, Machine Learning, Software Engineering. He can be reached via aomeiza@me.com or +2347039781808

AMUJO, Oluyemi E. holds Master of Science (M.Sc.) degree in Computer Science (Information Security as a Major) from University of Abuja, Nigeria. In 2007, he obtained his National Diploma (ND) in Computer Science from Rufus Giwa Polytechnic, Owo Ondo State and later obtained his B.Sc. (Hons) degree in Computer Science from Adekunle Ajasin University, Akungba-Akoko, Ondo State, Nigeria in 2014. He is a Practitioner who sees the widening gap between slow classroom and fast paced industrial halls and decided to bridge the gap. His research interests include; Information Security, Machine Learning, Software Engineering, IoT Development & Security. He can be reached via oluyemiamujo@gmail.com or oluyemi.amujo@hotmail.com or +2347063717176.

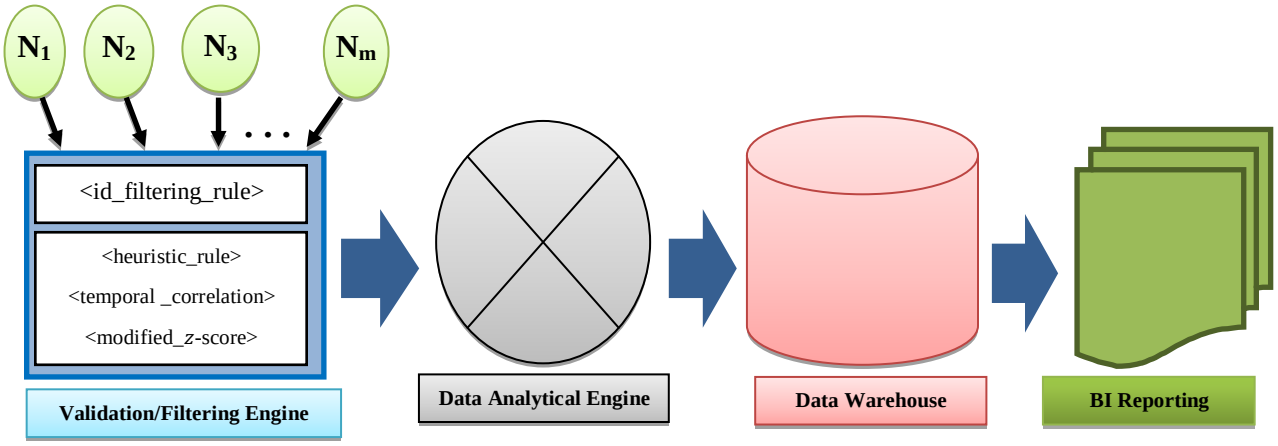
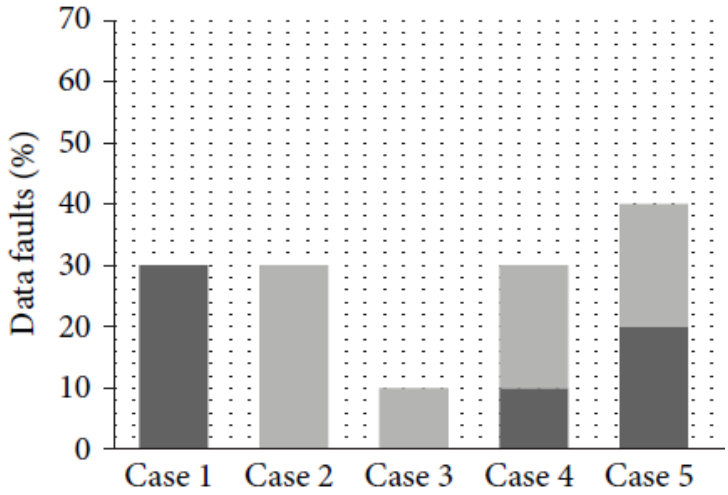


Figure 4.1: Proposed big data end -point validation/filtering model



■ Data faults detected

■ Data faults not detected (false positives)

Figure 4.2: Results of Algorithm 1 in Cases 1 to 5.

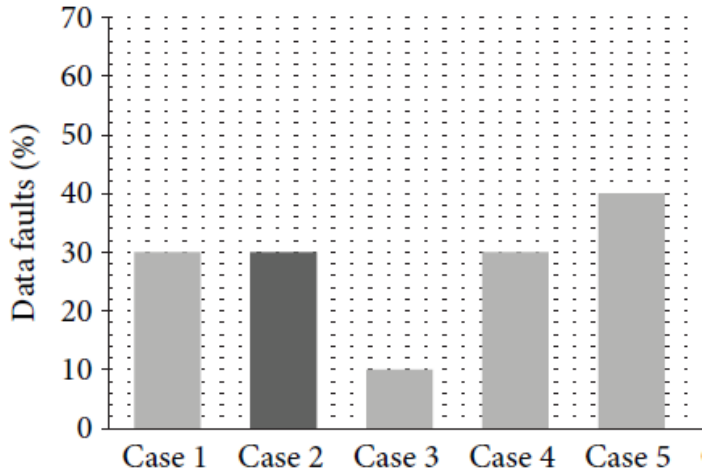


Figure 4.3: Results of Algorithm 2 in Cases 1 to 5.

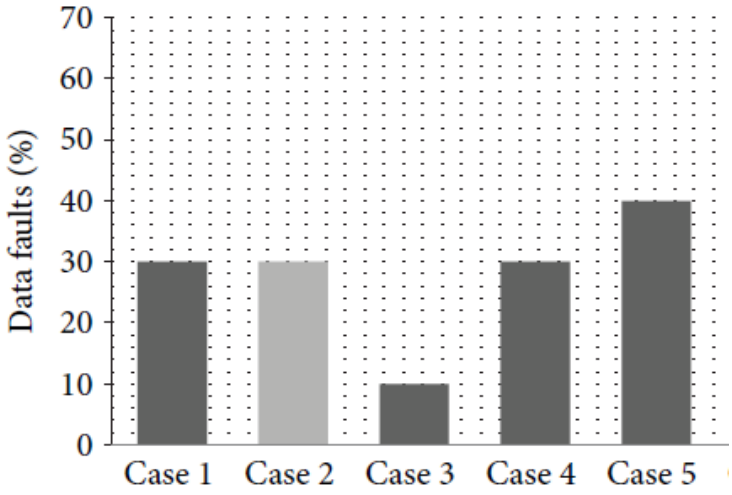
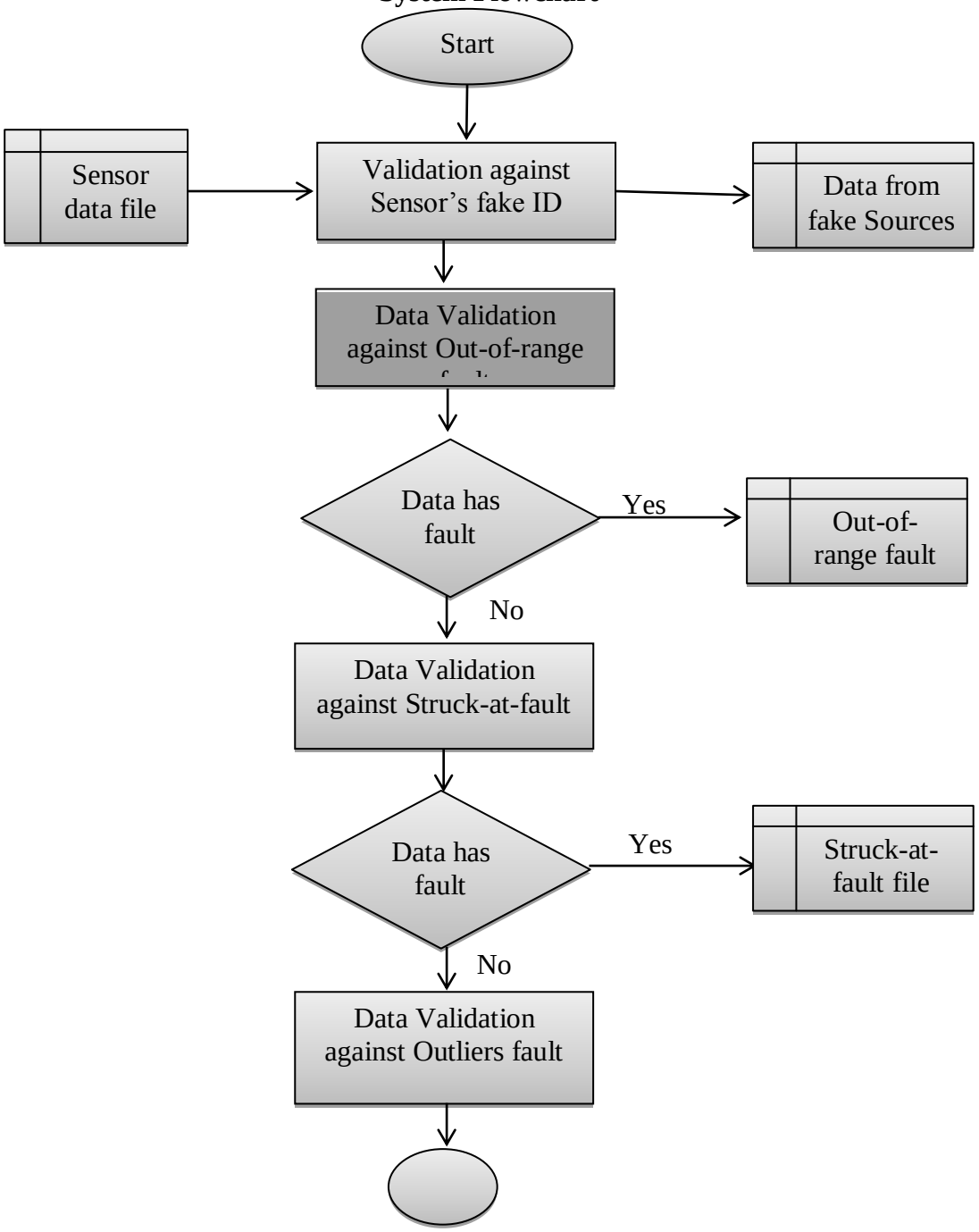
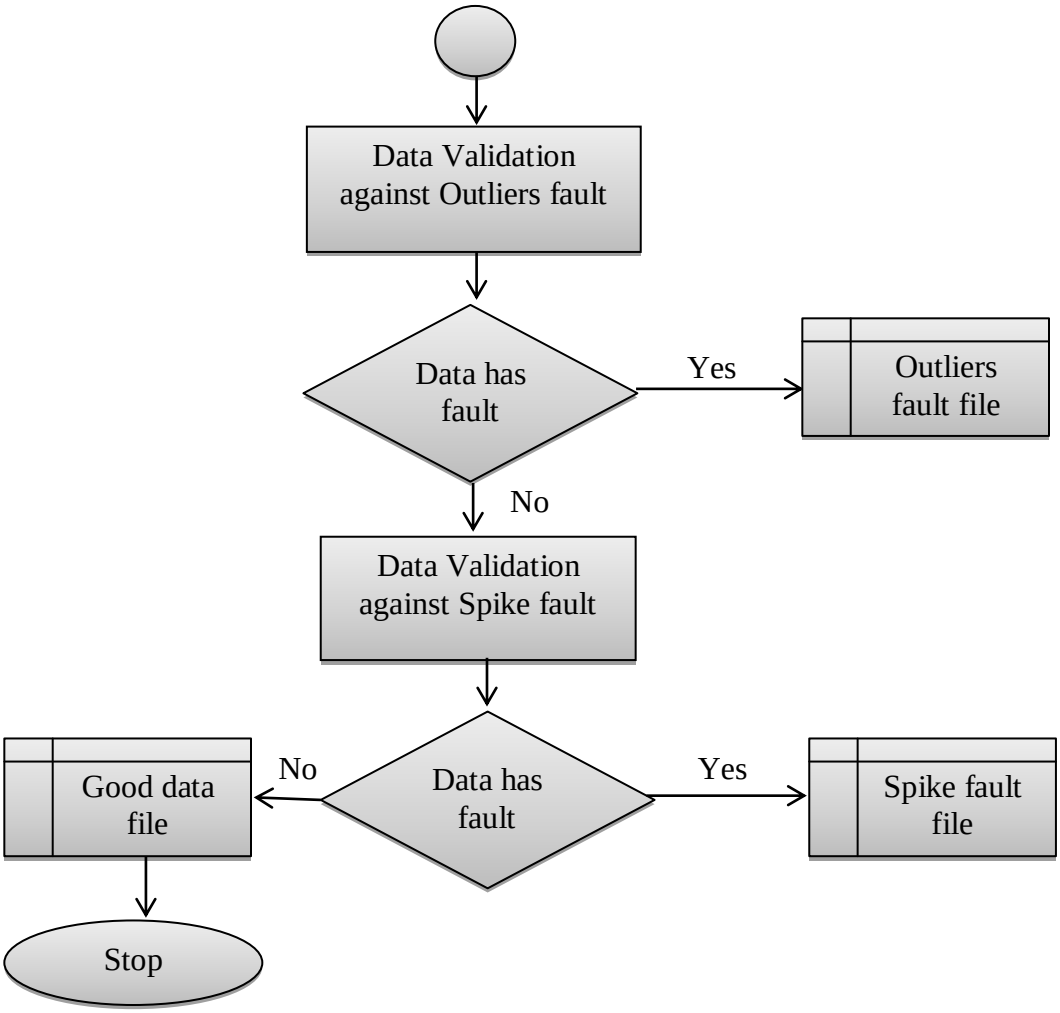


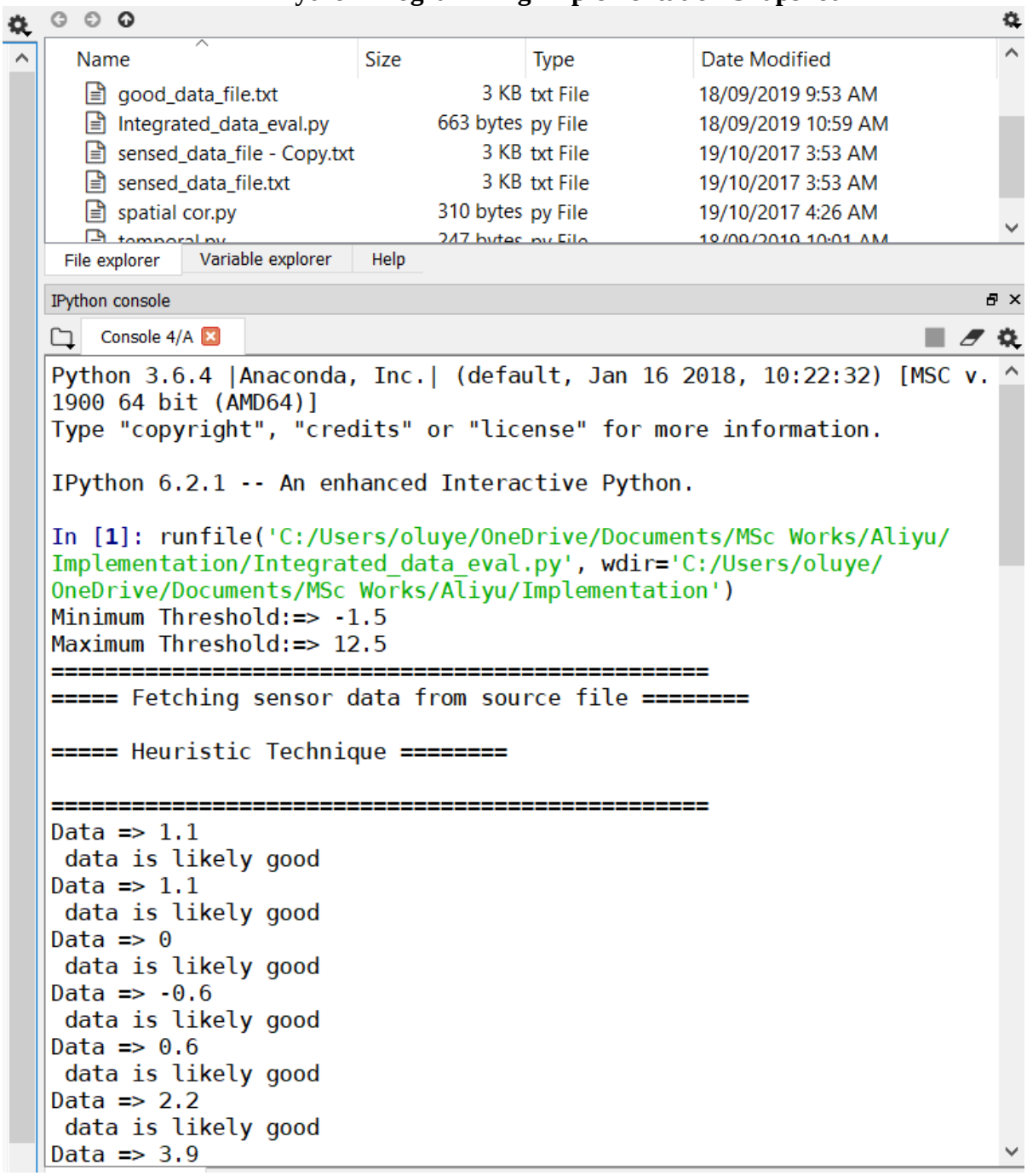
Figure 4.4: Results of Algorithm 3 in Cases 1 to 5.

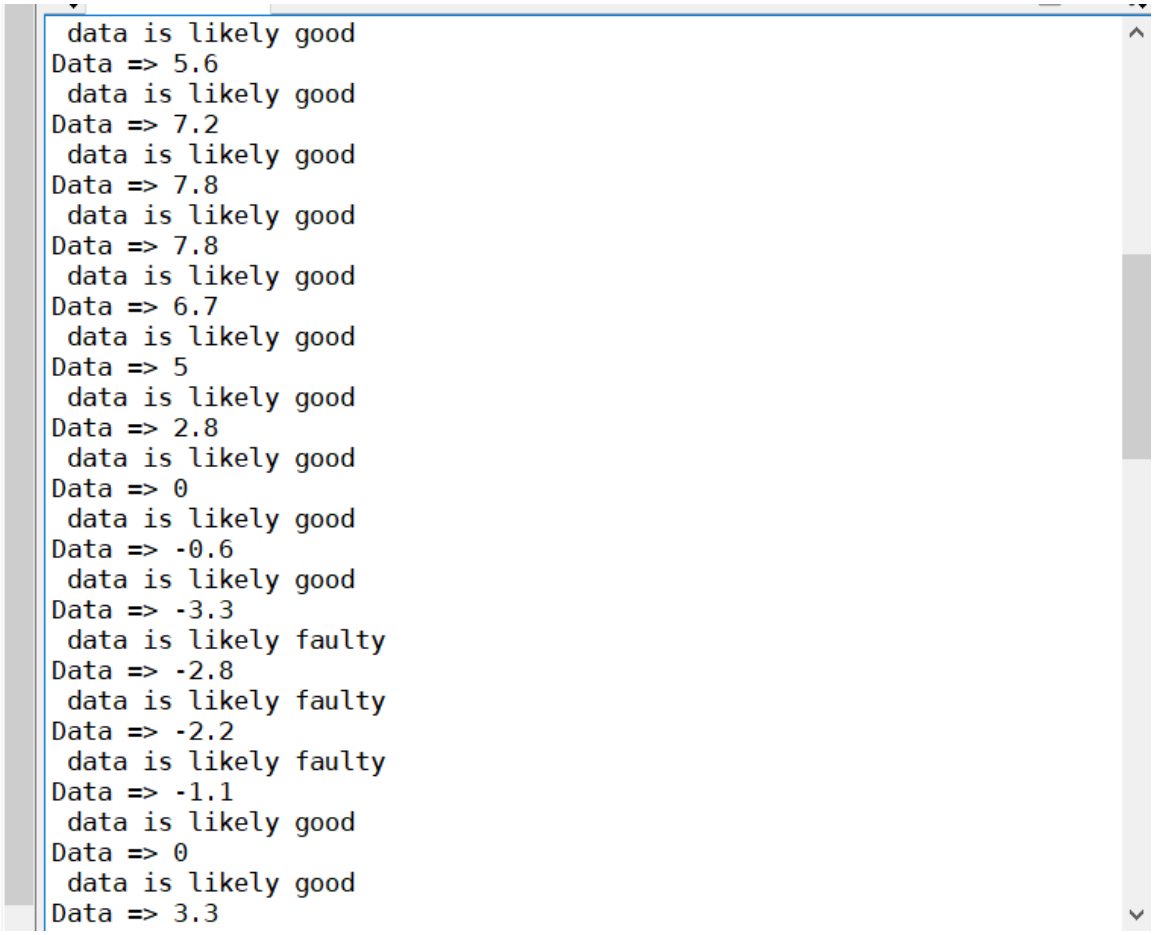
Appendix I
System Flowchart

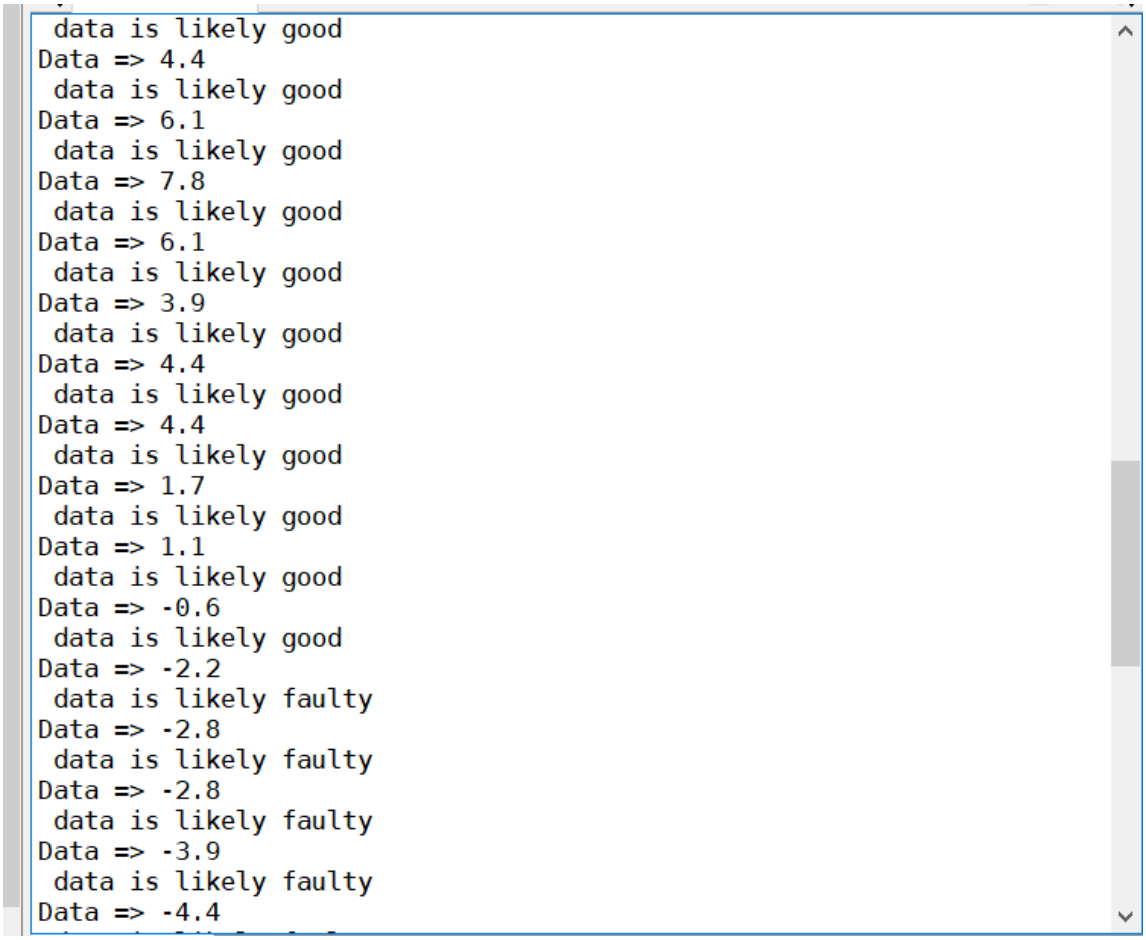




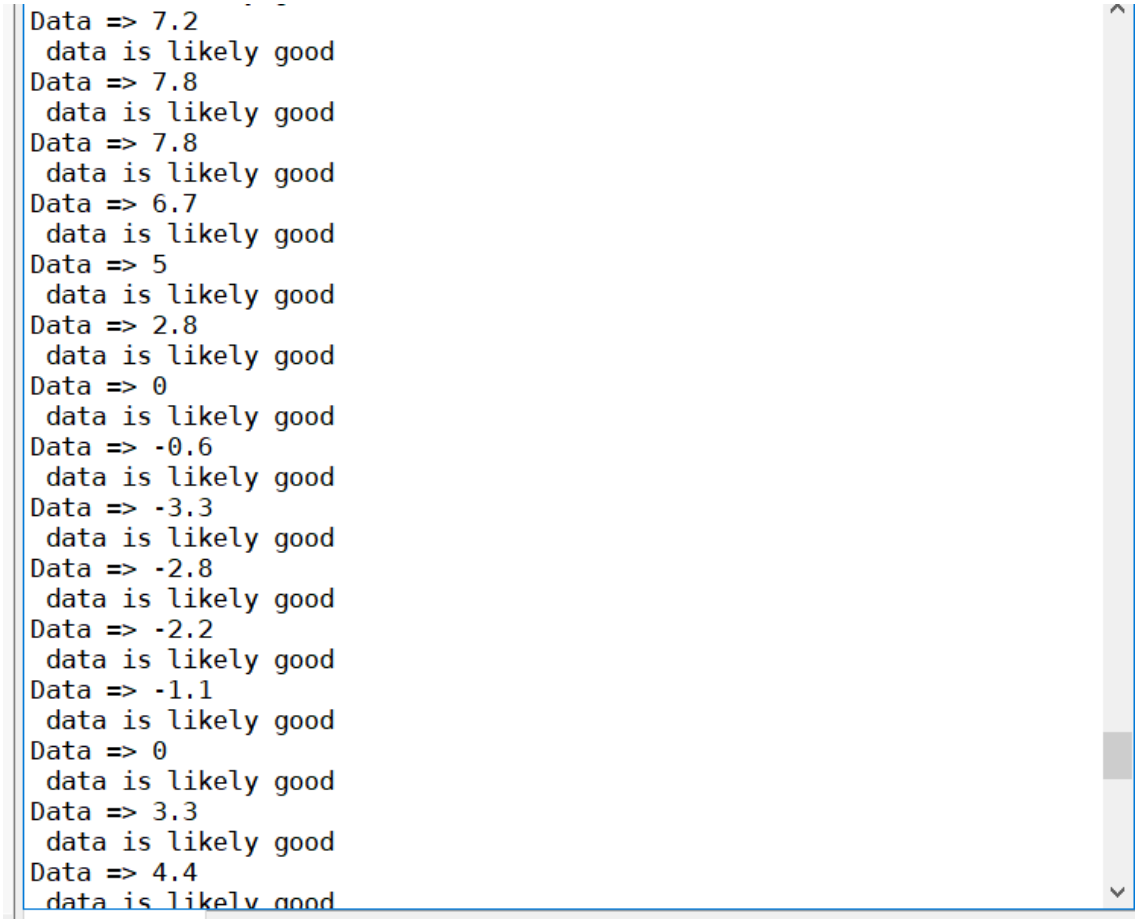
Appendix II
Big Data Validation
Python Programming Implementation Snapshot





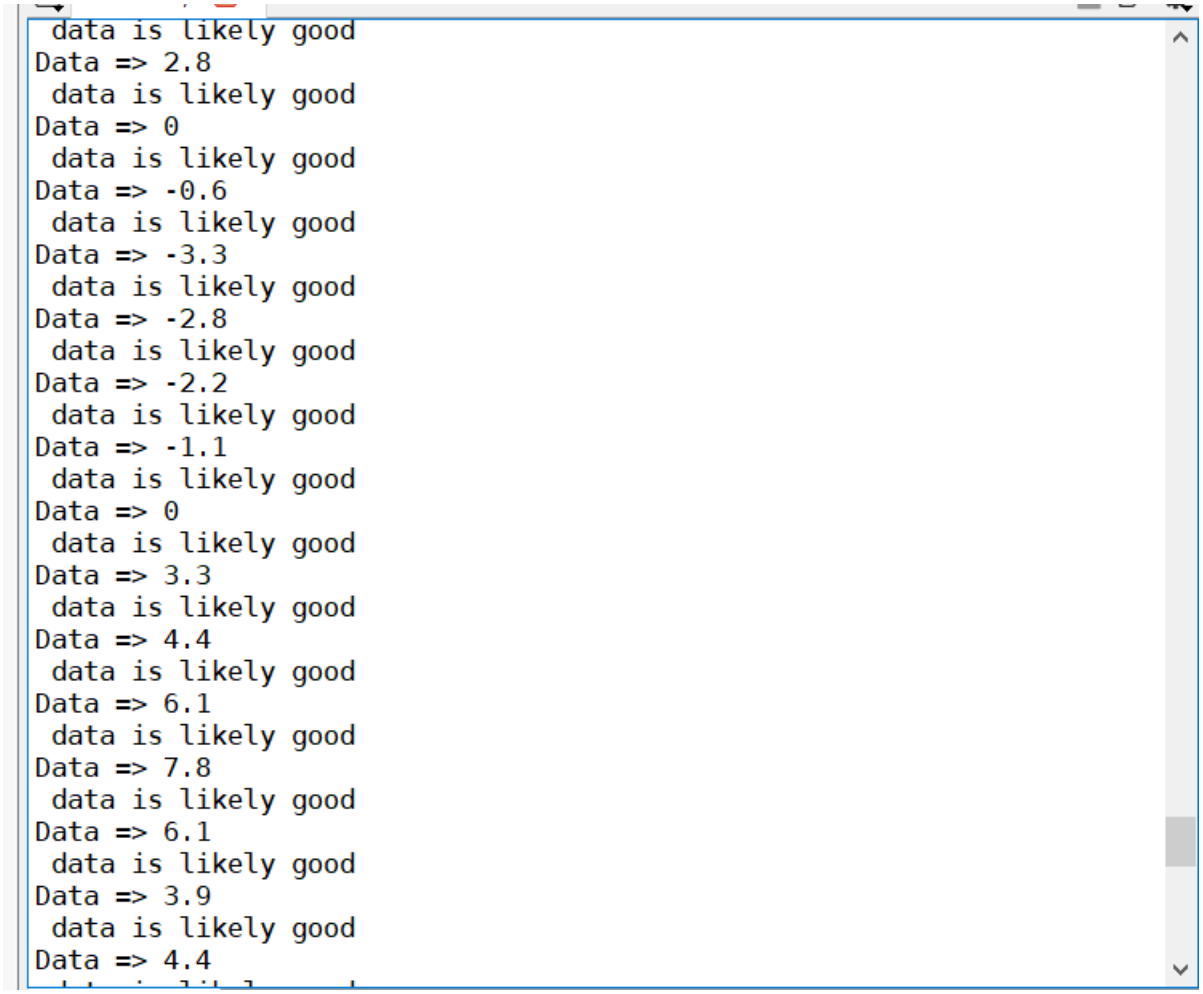


```
Minimum Threshold:=> -1.5
Maximum Threshold:=> 12.5
================================================
===== Fetching sensor data from source file ========

===== Spatial Correlation Technique ========

================================================
List Sum => 63.800000000000004
List Mean => 1.6789473684210527
List Length => 38
================================================
Data => 1.1
 data is likely good
Data => 1.1
 data is likely good
Data => 0
 data is likely good
Data => -0.6
 data is likely good
Data => 0.6
 data is likely good
Data => 2.2
 data is likely good
Data => 3.9
 data is likely good
Data => 5.6
 data is likely good
Data => 7.2
 data is likely good
```



```
Data => 7.8
 data is likely good
Data => 6.1
 data is likely good
Data => 3.9
 data is likely good
Data => 4.4
 data is likely good
Data => 4.4
 data is likely good
Data => 1.7
 data is likely good
Data => 1.1
 data is likely good
Data => -0.6
 data is likely good
Data => -2.2
 data is likely good
Data => -2.8
 data is likely good
Data => -2.8
 data is likely good
Data => -3.9
 data is likely good
Data => -4.4
 data is likely good
Data => -3.9 data is likely good
```

```
================================================
===== Fetching sensor data from source file ========

===== Temporal  Correlation Technique ========
================================================
Data => 1.1
 data is likely good
Data => 1.1
 data is likely faulty
Data => 0
 data is likely good
Data => -0.6
 data is likely good
Data => 0.6
 data is likely good
Data => 2.2
 data is likely good
Data => 3.9
 data is likely good
Data => 5.6
 data is likely good
Data => 7.2
 data is likely good
Data => 7.8
 data is likely good
Data => 7.8
 data is likely faulty
Data => 6.7
 data is likely good
```



```
Data => 4.4
 data is likely good
Data => 4.4
 data is likely faulty
Data => 1.7
 data is likely good
Data => 1.1
 data is likely good
Data => -0.6
 data is likely good
Data => -2.2
 data is likely good
Data => -2.8
 data is likely good
Data => -2.8
 data is likely faulty
Data => -3.9
 data is likely good
Data => -4.4
 data is likely good
Data => -3.9 data is likely good
```