COMPARISON BETWEEN DEEP LEARNING AND STATISTICAL LEARNING APPROACHES FOR ANOMALY DETECTION APPLICATIONS IN A MICROGRID

by

Chaitanya Bhure

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Electrical Engineering

Charlotte

2020

Approved by:

Dr. Madhav Manjrekar

Dr. Fareena Saqib

Dr. Chen Chen

ABSTRACT

CHAITANYA BHURE. Comparison between deep learning and statistical learning approaches for anomaly detection applications in a microgrid. (Under the direction of DR. MADHAV MANJREKAR)

A microgrid is a localized group of electricity sources and loads that normally operates connected to and synchronous with traditional wide area synchronous grid (macrogrid), but can also disconnect to island mode and then function autonomously as physical or economic conditions dictate. In this way, a microgrid can effectively integrate various sources of distributed generation (DG), especially Renewable Energy Sources (RES) and can supply emergency power, changing between island and connected modes. With the various devices connected on the microgrid, there is a lot of useful data which can be used to analyze their individual behaviours as well as for understanding how they operate with each other. Data Analytics has made tremendous progress in this decade and is only going to grow. It is natural that the utilities industry will want to make use of this technology of Machine Learning available at their disposal. This thesis investigates the novel approach of Deep Learning over traditional or statistical data analysis by comparing their performance in the analyzing the behaviour of the microgrid. Deep Learning architectures like Recurrent Neural Networks (RNN), Long Short Term Memory Networks (LSTM) have been proved to be effective in understanding time series data or temporal sequences in previous research in comparison to statistical algorithms like Auto Regressive Integrated Moving Average (ARIMA) or Vector Auto Regression (VAR). A new deep learning architecture of a ConvolutionalLSTM (Convolutional Long Short Term Memory Networks) is developed and tested. Finally the results are compared based on time required to train the models to their best performance and their accuracy in detecting the anomalies in the microgrid.

DEDICATION

I would like to dedicate this thesis to my parents who have raised me to the person I am today. Thank you for all the unconditional love, guidance and support you have given me and instilling in me the confidence that I am capable to doing anything I put my mind to. I could not have imagined accomplishing what I have without you by my side. Thank you for everything.

ACKNOWLEDGEMENTS

There are several people I would like to thank for their support and contributions to the project. First I would like to thank my advisor Dr. Madhav Manjrekar and my supervisor at Open Energy Solutions Inc. Mr. Dave Mulder for giving me the opportunity to be involved in this cutting edge research. I feel lucky to have had the chance to work with Dr. Manjrekar who is at the forefront of research in the cyber vulnerability in electrical infrastructure. He enjoys teaching as well as research and I have learnt a lot from working with him as a Research Assistant. Dave Mulder is a very supportive supervisor who is flexible and understanding of the balance between work and studies which is important for a student. He has provided a lot of useful feedback and I hope the results of this research effort are promising for the incorporation of Machine Learning Analytics at Open Energy Solutions Inc. I would also like to thank Dr. Fareena Saqib and Dr. Chen Chen for being part of my graduate committee. Dr. Chen is an incredible instructor and has, what seems to be an infinite amount of knowledge in the Deep Learning domain. He has always provided valuable feedback on the results of this research and interesting ideas to tackle problems that came along the way. Although I have not had the pleasure of taking a course with Dr. Saqib, she has background in the research regarding machine intelligence and IoT security which this research effort deals with on a broader scope. She has been very helpful with insights on the deep learning models used in this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADF	Augmented Dickey Fuller.
AIC	Akaike information criterion.
ANN	Artificial Neural Network.
AR	Auto Regression.
CEC	Constant error carousel.
CNN	Convolutional Neural Network.
DER	Distributed Energy Resources.
DOE	Department of Energy.
ECE	An acronym for Electrical and Computer Engineering.
ESTCP	Environmental Security Technology Certification Program.
GRU	Gated Recurrent Unit.
IDS	Intrusion Detection Systems.
LSTM	Long short term memory.
R&D	Research and Development.
RDSI	Renewable and distributed systems integration.
RES	Renewable Energy Sources.
RNN	Recurrent Neural Network.
SGDP	Smart Grid Demonstration project.
VAR	Vector Auto Regression.

CHAPTER 1: INTRODUCTION

The U.S. Department of Energy (DOE) Smart Grid R&D Program considers microgrids as a key building block for a Smart Grid and has established microgrid R&D as a key focus area [1]. Microgrids have been identified as a key component of the Smart Grid for improving power reliability and quality, increasing system energy efficiency, and providing the possibility of grid independence to individual end-user sites. The DOE defines the microgrid as "a group of interconnected loads and distributed energy resources within clearly defined electrical boundaries that acts as a single controllable entity with respect to the grid. A microgrid can connect and disconnect from the grid to enable it to operate in both grid-connected or island-mode [2]".

The benefits of microgrids include:

1. Integration of renewable energy sources to help reduce peak load and losses by locating generation near demand

2. Meeting end-user needs by ensuring energy supply for critical loads, controlling power quality and reliability at local level

3. Supporting the macrogrid by handling sensitive loads and variability of renewables locally and supplying ancillary services to the bulk power system

The bulk of DOE microgrid R&D efforts have been focused on demonstration activities to meet niche application needs such as needs for meeting peak load reduction, renewable energy mandates and directives and reliability at some critical facilities including military installations [1]. The ongoing microgrid demonstration projects consist of lab-and field-scale R&D test beds, renewable and distributed systems integration [3] (RDSI) projects for peak load reduction, select Smart Grid Demonstration

Program (SGDP) projects as part of DOE's implementation of grid modernization under the American Recovery and Reinvestment Act of 2009 [4] (ARPA), and assessment and demonstration projects jointly supported by the Department of Defense (DoD) and DOE. These and other microgrid development and deployment projects are shown in 1.1 including those projects funded under the DoD Environmental Security Technology Certification Program [5] (ESTCP) Installation Energy Test Bed Initiative. The DOE projects shown in 1.1 are summarized below.

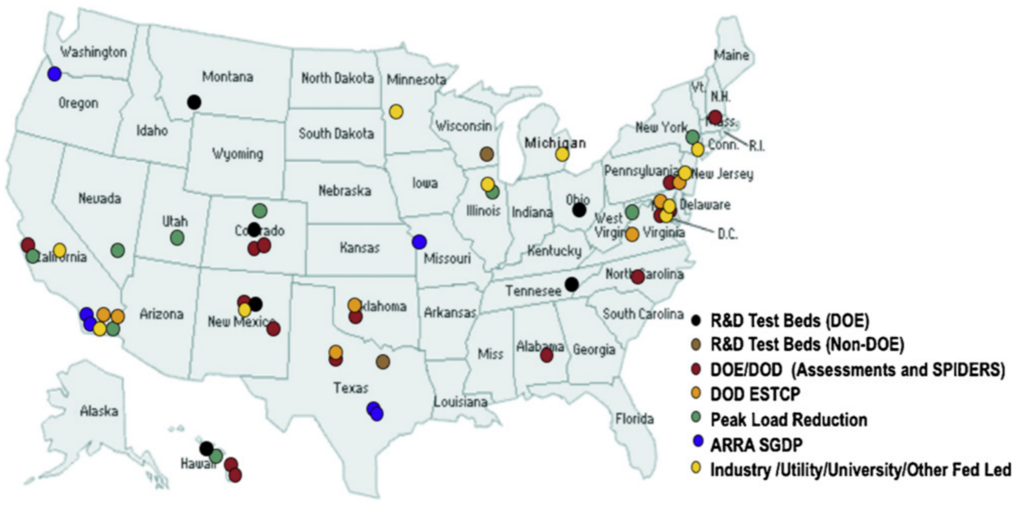
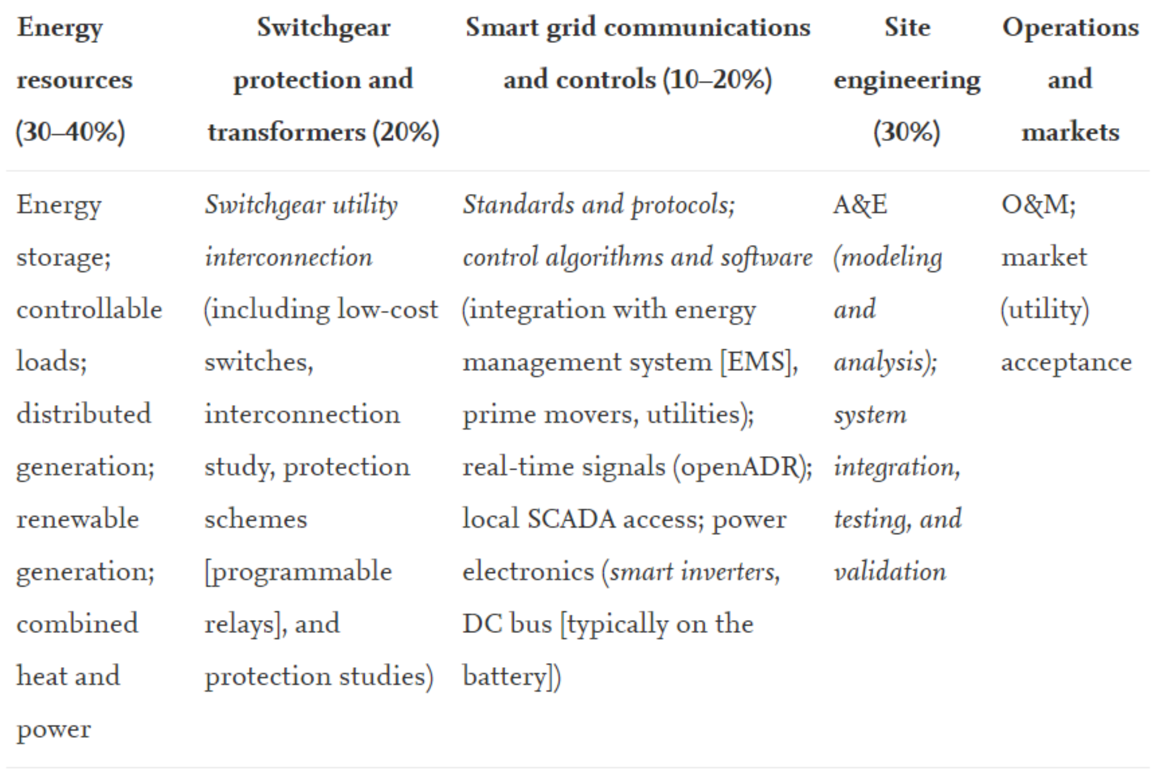


Figure 1.1: Microgrid Assessment and Demonstration Projects in the U.S.

Nine RDSI projects were selected in 2008 via a competitive DOE solicitation. The primary goals of these projects were to demonstrate at least 15 percent peak demand reduction on the distribution feeder or substation level through integrating distributed energy resources (DER) and to demonstrate that could operate in both grid parallel and islanded modes [6]. The application of technologies in an integrated fashion has the potential to allow more power to be delivered through existing infrastructure, thereby deferring transmission and distribution investment and to increase the reliability of the grid by adding elements that make it more stable and re-configurable [7]. Other potential benefits include addressing vulnerabilities in critical infrastructure, managing peak loads, lowering emissions, using fuel resources more efficiently and helping customers manage energy costs. These RDSI projects progressed toward achieving the goal of at least 15 percent peak reduction and some already demonstrated 15 percent or more in reductions. The total value of the RDSI program exceeded $100 million, with approximately $55 million from the DOE over five years and the rest through participant cost share [3].



Energy resources (30–40%)	**Switchgear protection and transformers (20%)**	**Smart grid communications and controls (10–20%)**	**Site engineering (30%)**	**Operations and markets**
Energy storage; controllable loads; distributed generation; renewable generation; combined heat and power	*Switchgear utility interconnection* (including low-cost switches, interconnection study, protection schemes [programmable relays], and protection studies)	*Standards and protocols; control algorithms and software* (integration with energy management system [EMS], prime movers, utilities); real-time signals (openADR); local SCADA access; power electronics (*smart inverters*, DC bus [typically on the battery])	A&E *(modeling and analysis); system integration, testing, and validation*	O&M; market (utility) acceptance

Figure 1.2: Major cost components of the microgrid

1.1 Microgrids: Components and Working

A microgrid [8] can effectively integrate various sources of distributed generation (DG) especially Renewable Energy Sources (RES) and can supply emergency power, changing between island and grid-connected modes as shown in 1.3. Control and protection are challenges to the microgrid [9]. A very important feature is also to provide multiple end-use needs such as heating, cooling and electricity at the same time since this allows energy carrier substitution and increased efficiency due to waste heat utilization for heating, domestic hot water and cooling purposes.

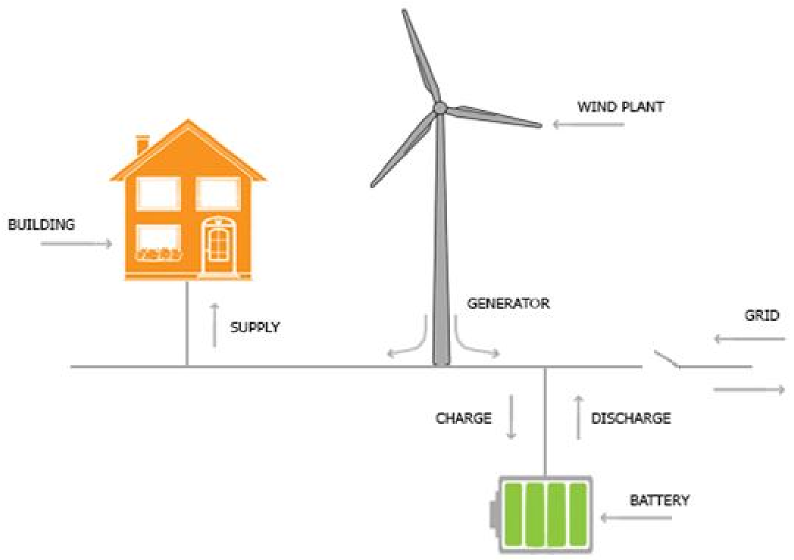


Figure 1.3: A typical scheme of an electric based microgrid with renewable energy resources in grid-connected mode

The types of microgrids are :

1. Campus environment/Institutional microgrids whose focus is aggregating on site generation with multiple loads in tight geography where the owner can easily manage them [10].

2. Community Microgrids which can serve up to few thousand customers and support penetration of local energy. In a community microgrid, some houses may have some renewable sources that can supply their demand as well as that of their neighbors within the same community [11]. The community microgrid may also have a centralized or several distributed energy storage. Such microgrids can be in the form of an ac and dc microgrid coupled together through a bi-directional power electronic converter.

3. Remote-Off grid microgrids never connect to the Macrogrid and instead operate in an island mode at all times because of economic issues or geographical position. Typically, an "off-grid" microgrid is built in areas that are far distant from any transmission and distribution infrastructure and, therefore, have

no connection to the utility grid. Studies have demonstrated that operating a remote area or islands' off-grid microgrids, that are dominated by renewable sources, will reduce the levelized cost of electricity production over the life of such microgrid projects [12].

4. Military microgrids are being actively deployed with focus on both physical and cyber security for military facilities in order to assure reliable power without relying on the Macrogrid [13].

5. Main reasons for the installation of an industrial microgrid are power supply security and its reliability. There are many manufacturing processes in which an interruption of the power supply may cause high revenue losses and long start-up time.Industrial microgrids can be designed to supply circular economy (near-)zero-emission industrial processes, and can integrate combined heat and power (CHP) generation, being fed by both renewable sources and waste processing; energy storage can be additionally used to optimize the operations of these sub-systems.

There are a few basic components of a microgrid as described below:

1. Local Generation: A microgrid presents various types of generation sources that feed electricity, heating, and cooling to user. These sources are divided into two major groups namely; thermal energy sources (e.g, natural gas or biogas generators or micro combined heat and power) and renewable generation sources (e.g. wind turbines, solar).

2. Consumption: In a microgrid, consumption simply refers to elements that consume electricity, heat, and cooling which range from single devices to lighting, heating system of buildings, commercial centers, etc. In the case of controllable loads, the electricity consumption can be modified in demand of the network.

3. Energy Storage: In microgrid, energy storage is able to perform multiple functions, such as ensuring power quality, including frequency and voltage regulation, smoothing the output of renewable energy sources, providing backup power for the system and playing crucial role in cost optimization. It includes all of chemical, electrical, pressure, gravitational, flywheel, and heat storage technologies.

4. Point of Common Coupling (PCC): It is the point in the electric circuit where a microgrid is connected to a main grid. Microgrids that do not have a PCC are called isolated microgrids which are usually presented in the case of remote sites

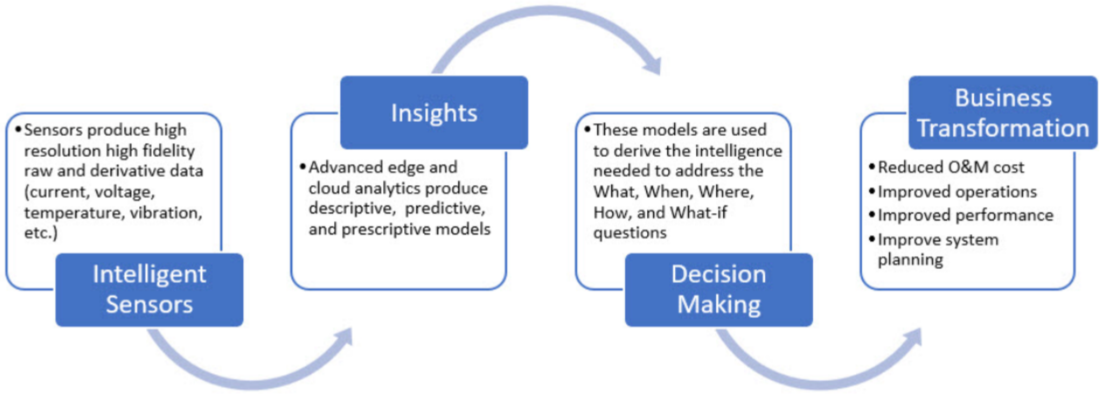
Figure 1.4: Sustainable Housing Community at Freiburg, Germany

A microgrid is capable of operating in grid-connected and stand-alone modes and of handling the transition between the two. In the grid-connected mode, ancillary services can be provided by trading activity between the microgrid and the main grid. Other possible revenue streams exist. In the the islanded mode, the real and reactive power generated within the microgrid, including that provided by the energy storage system, should be in balance with the local loads [8]. Microgrids offer an option to balancing the need to reduce carbon emissions while continuing to provide reliable electric energy in periods of time that renewable sources of power are not available. Microgrids also offer the security of being hardened from severe weather and natural disasters by not having large assets and miles of above-ground wires and other electric infrastructure that needs to be maintained or repaired following these events [14]. A microgrid may transition between these two modes because of scheduled maintenance, degraded power quality or a shortage in the host grid, faults in the local grid or for economical reasons. By means of modifying energy flow through microgrid components, microgrids facilitate the integration of renewable energy generation such as photo voltaic, wind and fuel cell generations without requiring re-design of the national distribution system. Modern optimization methods can also be incorporated into the microgrid energy management system to improve efficiency, economics and resiliency.

Microgrids and the integration of DER units in general, introduce a number of operational challenges that need to be addressed in the design of control and protection systems [9], in order to ensure that the present levels of reliability are not significantly affected and the potential benefits of Distributed Generation (DG) units are fully harnessed. Some of these challenges arise from assumptions typically applied to conventional distribution systems that are no longer valid, while others result of stability issues formerly observed only at transmission system level. The presence of distributed generation units in the network at low voltage levels can cause reverse power flows that may lead to complications in protection coordination, undesirable power flow patterns, fault current distributions and voltage control. Interactions between control system of DG units may create local oscillations requiring a thorough small-disturbance stability analysis. Moreover, transition activities between the grid-connected and islanding modes of operation in a microgrid can create transient instability. Many characteristics of traditional schemes such as the prevalence of three-phase balanced conditions, primarily inductive transmission lines and constant power loads do not necessarily hold true for microgrids and consequently models need to be revised. Microgrids exhibit a low-inertia characteristic that makes them different to bulk power systems, where a large number of synchronous generators ensures a relatively large inertia. Especially if there is a significant proportion of power electronic-interfaced DG units in the microgrid, this phenomenon is more evident. The low inertia in the system can lead to severe frequency deviations in island mode operation if a proper control mechanism is not implemented [8]. The operation of microgrids involves addressing much uncertainty, which is something the economical and reliable operation of microgrids relies on. Load profile and the weather are two of there uncertainties that make this coordination more challenging in isolated microgrids, where critical demand-supply balance and typically higher component failure rates require solving a strongly coupled problem over an extended time horizon. This uncertainty is higher than those in bulk power systems, due to the reduced number of loads and highly correlated variations of available energy resources (the averaging effect is much more limited).

1.2 Motivation of Study

Since the inception of data collection activities in the power and energy industry, the amount of data generated or recorded is far too much for the operator to make any sense about it. Tools must be created to process the data and provide useful information to system planners, operators and protection engineers. Now, with all this information, the question we seek the answer to is "How do we use this data to improve our situational intelligence and awareness?" [15]. As shown in figure 1.5, there are intelligent sensors installed which produce high resolution raw data to derive insights from and answer questions like What, When, Where, How and What-If. This results in improved operations, performance and system planning.



Intelligent Sensors
•Sensors produce high resolution high fidelity raw and derivative data (current, voltage, temperature, vibration, etc.)
Insights
•Advanced edge and cloud analytics produce descriptive, predictive, and prescriptive models
Decision Making
•These models are used to derive the intelligence needed to address the What, When, Where, How, and What-if questions
Business Transformation
•Reduced O&M cost
•Improved operations
•Improved performance
•Improve system planning

Figure 1.5: From data to business transformation using Intelligent Sensing, Control and Analysis

1.3 Literature Review

Microgrid data is very seldom utilized for analysis. The data is often recorded but there is not enough investment going into it for further analysis of the data. The few basic components of a microgrid as explained in the earlier sections mention generation, consumption, energy storage and point of common coupling devices. Various devices form a part of these different components and measure different quantities ranging from voltages and currents to the frequency. As devices modernize, they acquire the capability to generate data which can be transmitted to a central repository for storage and further analysis. This is called the time scale database since, the quantities measured by the devices are in respect with the time. Sometimes the zone, is set to Coordinated Universal Time to keep the time zone consistent while performing the analysis in different regions of the world. With the amount of data generated and the advancements in Big Data Analytics, there is a lot of valuable information which can be used for deriving useful insights ranging from prediction for the future (for example; load forecasting) to anomaly detection in the data. As described by the authors in [16], the pervasive deployment of advanced ICT (Information and Communication Systems), especially the smart metering will generate big energy data in terms of volume, velocity and variety. The generated big data can bring huge benefits to the better energy planning, efficient energy generation, and distribution. This article discusses the far reaching effects of anomalous behaviour of a smart grid. Since, there could be existing multiple sources leading to observed abnormal data, it is not always easy to determine the source of the data. Thus research efforts are made to design general energy big data anomaly detection algorithms. In [15] a scalable method is proposed to observe anomaly behaviour over energy big data. Supervised learning classifiers have been applied to finding anomaly data. It can be used to detect electricity theft, fault, and cyber intrusions. This approach has utilized some ad hoc intuitions in addressing the big data challenge.

Wireless sensor networks have become integral components of the monitoring systems for critical infrastructures such as the power grid or residential microgrids. Therefore, implementation of robust Intrusion Detection Systems (IDS) at the sensory data aggregation stage has become of paramount importance. Key performance targets for IDS in these environments involve accuracy, precision, and the receiver operating characteristics which is a function of the sensitivity and the ratio of false alarms. Furthermore, the interplay between machine learning and networked systems has led to promising opportunities, particularly for the system level security of wireless sensor networks as described in [17]. In [18] a mathematical model of smart loads in demand-response schemes is presented, which is integrated into centralized unit commitment with optimal power flow coupled energy management systems for isolated microgrids for optimal generation and peak load dispatch. The smart loads were modelled as a function of neural network load estimator. To train this neural network estimator, realistic data from actual energy hub management system was used for supervised training. In [19] a supervised machine learning approach is introduced to predict and schedule the real-time operation mode of the next operation interval for residential battery systems controlled by mode-based controllers. The optimal operation mode for each control interval was first derived from the historical data used as the training set. Then, four ML algorithms of neural network [20], support vector machine [21], logistic regression [22], and random forest [23] were applied.

The difference between supervised [24] and unsupervised learning techniques for anomaly detection is labelled data. Since, the anomalies or anomalous behaviour is rare in real life situations, there is a dearth of labelled data about anomalies and more so in the power and energy industry. This is also because of the frequency at which data points are recorded. 30 or 60 data points per second is a high frequency rate and finding out instantaneous anomalies at these frequencies is difficult. Hence, the lack of labelled data for anomalies. Which consequently leads to using unsupervised learning techniques [25]. In [26], a comparison between supervised and unsupervised learning techniques has been done for fraud detection in telecommunication networks. The advantage of using unsupervised learning techniques for anomaly detection is that the models do not need labelled dataset and with the advent of deep learning [27] methodologies, the raw data can be used as is. In unsupervised learning techniques, the network tries to model the correct or the non-anomalous data which is the majority of the entire dataset and then a threshold is set according to how well the network learns the good data. Based on this, and using the threshold any deviation from the normal or good data beyond the threshold is classified as an anomaly. For the purpose of this thesis, this is the mechanism we have used as well. In [28] a review of faults and fault diagnosis in microgrids in electrical energy infrastructure is presented. With the advancements of sensing, communication, and control technologies, the existing power systems have evolved with the development of Smart Micro-grids. Smart micro-grids integrate information technology, communication technology and power generation systems into one unified micro power system for robust and reliable power. A critical problem in power systems is the cascading effect of faults leading to severe failures and blackouts unless timely protective actions are taken. As a recovery mechanism, smart micro-grids are envisioned to detect these critical changes and switch into island mode for continual power generation and system stability. The paper essentially discusses two methods for fault diagnosis. One is model based and the other is a data driven approach. In the model based approach, the training data is generated for different scenarios and classification algorithms can be implemented for detection and diagnosis. In a data driven approach, the data is mined from the physical system instead of being generated. In island modes smart grids utilize measurements generated from meters and non invasive sensing devices to perform diagnostics and maintain performance. Some of the methods mentioned in the paper are Support Vector Machine [20], K-Means Clustering [29], Artificial neural networks [21], fuzzy logic [30] and many others. As concluded in the paper [28], fault detection and diagnosis is critical to improve performance and reliability of power distribution. Early diagnosis of failures of components within the distribution system facilitates condition based monitoring and autonomous reconfiguration to reduce repair time and costs while minimizing cascading failures.

1.4 Organization of thesis

This thesis is organized as per the following chapters:

- **Chapter 2:** The second chapter provides an overview of the dataset. We explore the features present in the dataset, the features we use for analysis, the time frame of the data collected and the significance of the features. We also look in depth at the cleaning and preprocessing methodologies used on the dataset to make it suitable for model training.

- **Chapter 3:** The third chapter covers an extensive review of time series data, univariate and multivariate time series. We go further into identifying the outliers in such time series data sets and the methodologies to do so. We discuss supervised and unsupervised learning approaches and why unsupervised learning approaches are the future in anomaly or outlier detection tasks in time series data

- **Chapter 4:** The fourth chapter focuses on the Vector Auto Regression algorithm for anomaly detection in time series data. This chapter covers the algorithm in detail including its history and working as a statistical model that can be used for anomaly detection.

- **Chapter 5:** The fifth chapter covers the Long Short Term Memory Network Autoencoder model for anomaly detection. We discuss the LSTM Network architecture, its success in time series tasks and the reasons behind the same. We also look at autoencoders and their importance in unsupervised anomaly detection tasks.

- **Chapter 6:** The sixth chapter covers the novel Conv-LSTM (Convolution Long Short Term Memory Network) model for anomaly detection in time series tasks. We discuss in this chapter, the disadvantages of the LSTM model and why using the Convolutions from the Convolution Neural Network model would be of significance.

- **Chapter 7:** This chapter is the conclusion of the thesis. The results generated from the algorithms mentioned above are compared and we look at the best performing model at the unsupervised anomaly detection tasks using various parameters.

CHAPTER 2: OVERVIEW OF THE DATASET

The data utilized for the purpose of this thesis is acquired from Duke Energy's microgrid facility at Mt. Holly, NC. Thus, by nature the dataset is a most likely representation of the data quality in the industry. That means the data is not curated for research purposes and a primary data cleansing process is absolutely necessary before any analysis is done on the data. Of the multiple devices installed at the microgrid facility we make use of data from three of those devices at different points in the microgrid. For the sake of confidentiality, we shall call the devices as "Device A", "Device B" and "Device C" respectively. Device A is connected to a battery in the microgrid and sees output from the battery. Device B is connected to a solar farm (renewable energy component) in the microgrid and Device C is connected to a point of common coupling between the microgrid and the main grid. For the purpose of anomaly detection in this thesis, a entire week was identified when the microgrid was functional to its optimum capacity. This week falls between the dates 2019-10-30 to 2019-11-05. This raw dataset has over an entire week and has roughly 2 million instances. Also, the frequency at which data is measured by the three devices is sixty hertz (60 Hz) 2.1. As shown in 2.2 the raw dataset has 47 unique columns. Out of these 47, the first 5 columns are identifiers, profile names and timestamps. Of these 5, "timestamp" is the only column that is of interest to us for the purpose of this analysis. The rest 42 columns are measurement columns. There is a pattern in these column names. It follows, "measurement_type_phase (if necessary)_magnitude/angle" pattern. For example, "a_net_mag" means that measurement is for the net current magnitude. Based of the measurement type, there are 8 distinct categories of measurements [31] that are recorded which are as follows:

1. a: Current, One Ampere of current is defined as one coulomb of charge passing through a unique point in one second.

2. hz: Frequency, the rate at which something occurs or is repeated over a particular period of time in a given sample.

3. pf: Power factor, the ratio of the actual electrical power dissipated by an (Alternating Current) AC circuit to the product of the root mean square values of current and voltage. The difference between the two is caused by the reactance in the circuit and represents power that does no useful work.

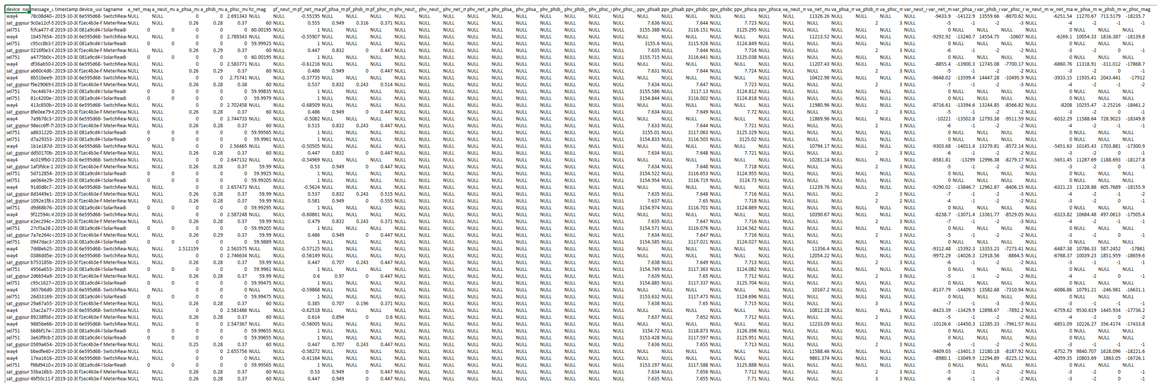
4. phv: Phase voltage, Voltage measured between line of the conductor to the neutral (at zero potential).

5. ppv: Per Phase Voltage, Line to line voltage is the relative voltage between the lines.

6. va: Apparent power, Measure of power by multiplying root mean square values of current and voltage.

7. var: Reactive Power, Reactive power is the resultant power in watts of an AC circuit when the current waveform is out of phase with the waveform of the voltage, usually by 90 degrees if the load is purely reactive, and is the result of either capacitive or inductive loads

8. w: Real Power, The actual amount of power being used or dissipated in a circuit and is measured in Watts.



device_na	message_u	timestamp	device_uui	tagname	a_net_ma	a_neut_ma	a_phsa_ma	a_phsb_ma	a_phsc_ma	hz_mag	pf_neut_m	pf_net_ma	pf_phsa_m	pf_phsb_m	pf_phsc_m	phv_neut_	phv_neut_	phv_net_m	phv_net_a	phv_phsa_	phv_phsa_	phv_phsb_	phv_phsb_	phv_phsc_	phv_phsc_	ppv_phsab	ppv_phsab	ppv_phsbc	ppv_phsbc	ppv_phsca	ppv_phsca	va_neut_m	va_net_ma	va_phsa_m	va_phsb_m	va_phsc_m	var_neut_	var_net_m	var_phsa_	var_phsb_	var_phsc_	w_neut_m	w_net_ma	w_phsa_m	w_phsb_m	w_phsc_mag
way4	78c08d40-	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	2.691343	NULL	NULL	-0.55195	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	11326.26	NULL	NULL	NULL	NULL	-9433.9	-14122.9	13559.66	-8870.62	NULL	-6251.54	11270.67	713.5179	-18235.7
sat_gypsur	9c0a12cf-5	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.37	60	NULL	0.555	0.949	0.316	0.371	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.636	NULL	7.644	NULL	7.723	NULL	NULL	NULL	NULL	2	3	NULL	-5	-1	-2	-3	NULL	-4	-2	-1	-1
sel751	fcfca477-d	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	60.00195	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3155.388	NULL	3116.151	NULL	3125.295	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
way4	1b457654-	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	2.789343	NULL	NULL	-0.55907	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	11213.52	NULL	NULL	NULL	NULL	-9292.92	-13240.7	14554.75	-10607	NULL	-6269.1	10054.33	1816.387	-18139.8
sel751	c95cc8b3-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.99925	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3155.6	NULL	3115.926	NULL	3124.849	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
sat_gypsur	0216f0e3-	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.29	0.37	60	NULL	0.447	0.832	0	0.447	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.635	NULL	7.644	NULL	7.724	NULL	NULL	NULL	NULL	2	3	NULL	-6	-1	-2	-2	NULL	-3	-2	0	-1
sel751	a4775b0c-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	60.00195	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3155.715	NULL	3116.641	NULL	3125.038	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
way4	df36ab50-8	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	2.583771	NULL	NULL	-0.61216	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	11207.43	NULL	NULL	NULL	NULL	-8855.4	-13900.3	12745.08	-7700.17	NULL	-6860.76	11318.91	-311.012	-17868.7
sat_gypsur	ab60c4d6-	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.29	0.37	60	NULL	0.486	0.949	0	0.447	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.631	NULL	7.644	NULL	7.724	NULL	NULL	NULL	NULL	2	3	NULL	-5	-1	-2	-2	NULL	-3	-2	0	-1
way4	8b51bee9-	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	2.75741	NULL	NULL	-0.37735	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	10422.98	NULL	NULL	NULL	NULL	-9648.02	-13599.4	14447.28	-10495.9	NULL	-3933.15	11935.41	2043.441	-17912
sat_gypsur	f9e29069-8	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.38	60	NULL	0.537	0.832	0.243	0.514	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.634	NULL	7.647	NULL	7.723	NULL	NULL	NULL	NULL	2	3	NULL	-7	-1	-2	-3	NULL	-4	-2	-1	-2
sel751	7ec44b74-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.99835	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3155.586	NULL	3117.13	NULL	3124.812	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
sel751	81c4200e-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.9979	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3154.844	NULL	3116.002	NULL	3124.818	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
way4	413c850b-	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	2.702458	NULL	NULL	-0.68509	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	11980.96	NULL	NULL	NULL	NULL	-8716.61	-13394.6	13244.85	-8566.82	NULL	-8208	10255.47	-2.25216	-18461.2
sat_gypsur	8fe0ee79-6	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.37	60	NULL	0.486	0.949	0	0.447	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.634	NULL	7.649	NULL	7.72	NULL	NULL	NULL	NULL	2	3	NULL	-5	-1	-2	-2	NULL	-3	-2	0	-1
way4	7a9b78c3-	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	2.744733	NULL	NULL	-0.5082	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	11869.96	NULL	NULL	NULL	NULL	-10221	-13502.8	12793.38	-9511.59	NULL	-6032.29	11588.64	728.9023	-18349.8
sat_gypsur	98ecc6ff-7	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.37	60	NULL	0.515	0.832	0.243	0.447	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.633	NULL	7.644	NULL	7.721	NULL	NULL	NULL	NULL	2	3	NULL	-6	-1	-2	-2	NULL	-4	-2	-1	-1
sel751	a8631220-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.99565	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3155.01	NULL	3117.063	NULL	3125.329	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
sel751	d7a29353-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.9961	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3154.833	NULL	3116.503	NULL	3125.02	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
way4	1b1e187d-	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	2.56465	NULL	NULL	-0.50505	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	10794.17	NULL	NULL	NULL	NULL	-9303.68	-14011.4	13279.81	-8572.14	NULL	-5451.63	10145.43	1703.881	-17300.9
sat_gypsur	dd50170b-	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.37	60	NULL	0.447	0.832	0	0.447	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.634	NULL	7.648	NULL	7.721	NULL	NULL	NULL	NULL	2	3	NULL	-6	-1	-2	-2	NULL	-3	-2	0	-1
way4	4c019fb0-1	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	2.647132	NULL	NULL	-0.54969	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	10281.14	NULL	NULL	NULL	NULL	-8581.81	-13299	12996.38	-8279.17	NULL	-5651.45	11287.69	1188.693	-18127.8
sat_gypsur	1af3fdce-1	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.37	59.99	NULL	0.53	0.949	0	0.447	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.634	NULL	7.648	NULL	7.718	NULL	NULL	NULL	NULL	2	3	NULL	-5	-1	-2	-2	NULL	-3	-2	0	-1
sel751	5d712854-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.9925	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3154.522	NULL	3116.653	NULL	3124.955	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
sel751	ae064e29-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.99205	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3154.954	NULL	3116.719	NULL	3124.73	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
way4	91d0d8c7-	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	2.657472	NULL	NULL	-0.5624	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	11239.76	NULL	NULL	NULL	NULL	-9290.02	-13846.7	12962.87	-8406.15	NULL	-6321.23	11228.88	605.7689	-18155.9
sat_gypsur	8d3449e1-	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.37	59.99	NULL	0.537	0.832	0.243	0.515	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.635	NULL	7.648	NULL	7.716	NULL	NULL	NULL	NULL	2	3	NULL	-7	-1	-2	-3	NULL	-4	-2	-1	-2
sat_gypsur	1092e1f8-	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.37	59.99	NULL	0.581	0.949	0	0.555	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.637	NULL	7.65	NULL	7.718	NULL	NULL	NULL	NULL	2	2	NULL	-4	-1	-2	-2	NULL	-3	-2	0	-1
sel751	d9d68b76-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.99295	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3154.974	NULL	3116.701	NULL	3124.869	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
way4	9f22594c-4	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	2.587248	NULL	NULL	-0.60861	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	10390.67	NULL	NULL	NULL	NULL	-8238.7	-13071.4	13361.77	-8529.05	NULL	-6323.82	10684.48	497.0613	-17505.4
sat_gypsur	e2ec294c-	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.37	59.99	NULL	0.479	0.832	0.243	0.371	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.635	NULL	7.647	NULL	7.716	NULL	NULL	NULL	NULL	2	3	NULL	-7	-1	-2	-3	NULL	-4	-2	-1	-1
sel751	27cf3a28-2	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.99205	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3154.571	NULL	3116.076	NULL	3124.562	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
sat_gypsur	7a7e264c-	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.29	0.37	59.99	NULL	0.486	0.949	0	0.447	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.634	NULL	7.647	NULL	7.716	NULL	NULL	NULL	NULL	2	3	NULL	-5	-1	-2	-2	NULL	-3	-2	0	-1
sel751	0947dac3-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.9889	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3154.585	NULL	3117.021	NULL	3124.027	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
way4	7dd8eb25-	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	2.512159	0	2.563575	NULL	NULL	-0.57125	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	11356.4	NULL	NULL	NULL	NULL	-9312.48	-15392.3	13353.23	-7273.41	NULL	-6487.38	10786.33	587.2452	-17861
way4	0386dd5e-	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	2.746634	NULL	NULL	-0.56149	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	12054.22	NULL	NULL	NULL	NULL	-9972.29	-14026.3	12918.56	-8864.5	NULL	-6768.37	10039.23	1851.959	-18659.6
sat_gypsur	b753185b-	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.37	59.99	NULL	0.447	0.707	0.243	0.447	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.638	NULL	7.649	NULL	7.713	NULL	NULL	NULL	NULL	2	3	NULL	-6	-1	-2	-2	NULL	-3	-1	-1	-1
sel751	4956a653-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.9961	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3154.749	NULL	3117.363	NULL	3124.082	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
sat_gypsur	2d6b54a9-	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.37	59.99	NULL	0.6	0.97	0	0.447	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.639	NULL	7.65	NULL	7.712	NULL	NULL	NULL	NULL	2	3	NULL	-5	-1	-2	-2	NULL	-4	-2	0	-1
sel751	c95c1627-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.99475	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3154.885	NULL	3117.337	NULL	3125.704	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
way4	3657b6d0-	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	0	NULL	NULL	-0.59868	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	10167.2	NULL	NULL	NULL	NULL	-8137.79	-14409.5	13582.68	-7310.94	NULL	-6086.86	10791.21	-246.981	-16631.1
sel751	26d33169-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.99475	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3153.632	NULL	3117.473	NULL	3124.696	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
sat_gypsur	29a67a55-	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.37	60	NULL	0.385	0.707	0.196	0.371	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.638	NULL	7.65	NULL	7.715	NULL	NULL	NULL	NULL	3	3	NULL	-7	-1	-3	-3	NULL	-3	-1	-1	-1
way4	15ac2a77-	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	2.581486	NULL	NULL	-0.62518	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	10812.28	NULL	NULL	NULL	NULL	-8423.39	-13429.9	12898.67	-7892.2	NULL	-6759.62	9530.619	1445.934	-17736.2
sat_gypsur	89238f0d-	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.37	60	NULL	0.614	0.894	0	0.6	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.637	NULL	7.652	NULL	7.712	NULL	NULL	NULL	NULL	2	3	NULL	-5	-1	-2	-2	NULL	-4	-2	0	-2
way4	9885be66-	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	2.547367	NULL	NULL	-0.56005	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	12233.09	NULL	NULL	NULL	NULL	-10126.6	-14450.3	12285.33	-7961.57	NULL	-6851.09	10226.27	356.4174	-17433.8
sel751	bb8bf17e-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.99655	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3154.72	NULL	3118.873	NULL	3126.398	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
sel751	3e63f9cb-5	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.99655	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3153.428	NULL	3117.597	NULL	3125.951	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
sat_gypsur	0589a654-	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.25	0.28	0.37	60	NULL	0.447	0.707	0.243	0.447	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.636	NULL	7.655	NULL	7.713	NULL	NULL	NULL	NULL	2	3	NULL	-6	-1	-2	-2	NULL	-3	-1	-1	-1
way4	bbedfe40-	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	2.655756	NULL	NULL	-0.58272	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	11588.46	NULL	NULL	NULL	NULL	-9409.03	-13401.3	12180.18	-8187.92	NULL	-6752.79	9840.707	1628.096	-18221.6
way4	17ea1616-	2019-10-3	6e595d68-	SwitchRea	NULL	NULL	0	0	0	NULL	NULL	-0.41164	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	9861.374	NULL	NULL	NULL	NULL	-8980.1	-13049.9	12294.89	-8225.12	NULL	-4059.35	10803.69	1863.05	-16726.1
sel751	f98d9410-	2019-10-3	081a9cd4-	SolarReadi	0	0	0	0	0	59.99565	NULL	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	3153.197	NULL	3117.588	NULL	3125.888	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL	NULL	0	NULL	NULL	NULL
sat_gypsur	55ba16b3-	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.37	60	NULL	0.53	0.949	0	0.447	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.634	NULL	7.656	NULL	7.712	NULL	NULL	NULL	NULL	2	3	NULL	-5	-1	-2	-2	NULL	-3	-2	0	-1
sat_gypsur	4bf50c11-f	2019-10-3	f1ec4b3e-f	MeterRea	NULL	NULL	0.26	0.28	0.37	60	NULL	0.447	0.949	0	0.447	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	7.635	NULL	7.655	NULL	7.71	NULL	NULL	NULL	NULL	3	3	NULL	-6	-1	-3	-2	NULL	-3	-2	0	-1

Figure 2.1: Snapshot of the raw data from 2019-10-30

```
Index(['device_name', 'message_uuid', 'timestamp', 'device_uuid', 'tagname',
       'a_net_mag', 'a_neut_mag', 'a_phsa_mag', 'a_phsb_mag', 'a_phsc_mag',
       'hz_mag', 'pf_neut_mag', 'pf_net_mag', 'pf_phsa_mag', 'pf_phsb_mag',
       'pf_phsc_mag', 'phv_neut_mag', 'phv_neut_ang', 'phv_net_mag',
       'phv_net_ang', 'phv_phsa_mag', 'phv_phsa_ang', 'phv_phsb_mag',
       'phv_phsb_ang', 'phv_phsc_mag', 'phv_phsc_ang', 'ppv_phsab_mag',
       'ppv_phsab_ang', 'ppv_phsbc_mag', 'ppv_phsbc_ang', 'ppv_phsca_mag',
       'ppv_phsca_ang', 'va_neut_mag', 'va_net_mag', 'va_phsa_mag',
       'va_phsb_mag', 'va_phsc_mag', 'var_neut_mag', 'var_net_mag',
       'var_phsa_mag', 'var_phsb_mag', 'var_phsc_mag', 'w_neut_mag',
       'w_net_mag', 'w_phsa_mag', 'w_phsb_mag', 'w_phsc_mag'],
      dtype='object')
```

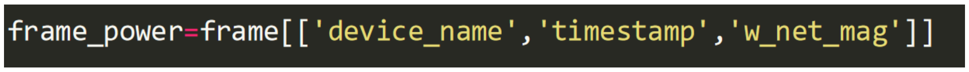
Figure 2.2: Columns in raw data from 2019-10-30

2.1 Data Cleansing

From the Figure 2.1 we can clearly see that the raw data is not really of use to for any direct analysis. Hence, the need for cleansing and preprocessing. This section focuses on the data cleansing portion while the next discusses the preprocessing before making the data ready to be fed to a statistical or deep learning model. Data Cleansing followed here is essentially a three step process described in order of the sections below.

2.1.1 Removal of Null values or NaN's

This is a very important step in any data analysis task. The raw data is always analyzed for having null values in the same first. Since, this is a data set coming from a real microgrid it is not curated for analysis and thus, we have to go through the process. After analysis of all the columns where null values are present, we identify that "w_net_mag" is the most appropriate column for our analysis, since it has no missing data that is null values and is a good measure for the case of anomaly detection [32]. The code snippet shown in Figure 2.3 creates a new data frame in Python based on the device names, timestamps and the true power measurement for all the devices.

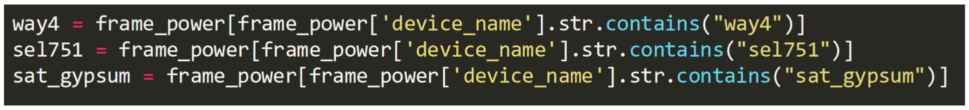


```
frame_power=frame[['device_name','timestamp','w_net_mag']]
```

Figure 2.3: Code snippet to remove NaN's

2.1.2 3 point segregation

After removal of the null values, we have a dataframe consisting of the device names, timestamps and the real power measurement for the three devices. So, to have the values, device-by-device we segregate the dataframe into three different dataframes for each of the individual devices. The code snippet shown in Figure 2.4 describes this process.

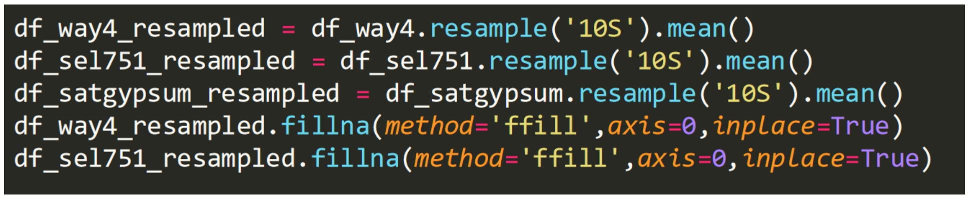


```
way4 = frame_power[frame_power['device_name'].str.contains("way4")]
sel751 = frame_power[frame_power['device_name'].str.contains("sel751")]
sat_gypsum = frame_power[frame_power['device_name'].str.contains("sat_gypsum")]
```

Figure 2.4: Code snippet to segregate individual devices

2.1.3 Resampling

After the segregation of the original dataframe into 3 different dataframes for each of the three devices, there is a problem of inconsistent timestamps for each of the three devices. Thus, to get rid of the inconsistency in the timestamps, we go by sampling the values to every 10 seconds instead of every 1 second by taking the mean of the values from the original dataframe as shown in Figure 2.5. The method "ffill" as seen in the Figure 2.5 is called the forward fill method. This method essentially propagates the last valid observation forward to fill up the remaining NaN's. This reduces the number of data instances but is a trade-off and is essential to be carried out for the analysis.



```
df_way4_resampled = df_way4.resample('10S').mean()
df_sel751_resampled = df_sel751.resample('10S').mean()
df_satgypsum_resampled = df_satgypsum.resample('10S').mean()
df_way4_resampled.fillna(method='ffill',axis=0,inplace=True)
df_sel751_resampled.fillna(method='ffill',axis=0,inplace=True)
```

Figure 2.5: Code snippet for resampling data points to every 10 seconds

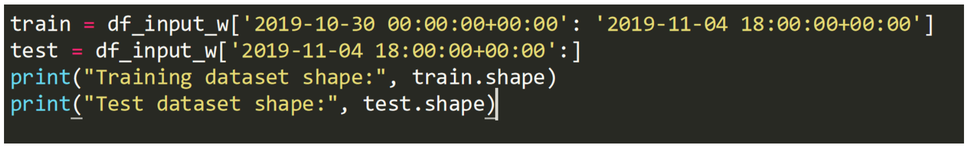
2.2 Data Preprocessing

After the data cleansing techniques performed in the previous section we have data of real power measured at three different devices at different points in the microgrid. The idea behind using one particular quantity at three different devices for the purpose of anomaly detection and treating this as a multi-variate problem is to understand whether the devices in the microgrid have any effects on each other and consequently trying to answer the question of "Can we understand the behaviour of a microgrid and model it to detect probable anomalies?" The total number of instances are now amounting to 58,658 data spanning 7 days. It is believed that this data is the ideal or expected way of the functioning of a microgrid. That means essentially the chances of anomalies occurring in this data are very small. That makes this dataset ideal for the purpose of this thesis as we discuss a methodology for unsupervised anomaly detection [26]. This section discusses, the preprocessing of the dataset which involves splitting the dataset into training and testing set, manipulating the skew values for the training set and data normalization.

2.2.1 Splitting between training and testing set

The dataset we have is a time series and hence the training and testing set are defined accordingly. Data in the interval of 2019-10-30 00:00:00+00:00 to 2019-11-04 18:00:00+00:00 comes under the training set while data in the interval 2019-11-04

18:00:00+00:00 to 2019-11-05 18:56:00+00:00 comes under the testing set. The code snippet shown in Figure 2.6 implements this behaviour.



```
train = df_input_w['2019-10-30 00:00:00+00:00': '2019-11-04 18:00:00+00:00']
test = df_input_w['2019-11-04 18:00:00+00:00':]
print("Training dataset shape:", train.shape)
print("Test dataset shape:", test.shape)
```

Figure 2.6: Code snippet for splitting data into train and test set

After splitting the training data for three devices at three different points in the microgrid is plotted in the Figure 2.7

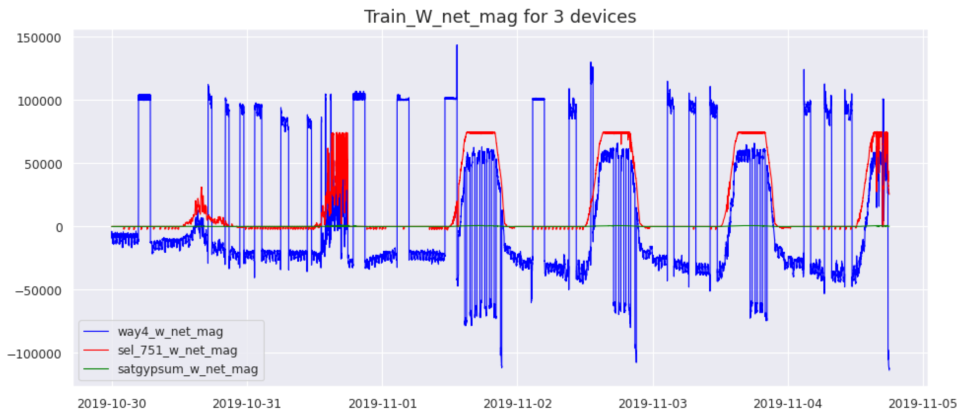
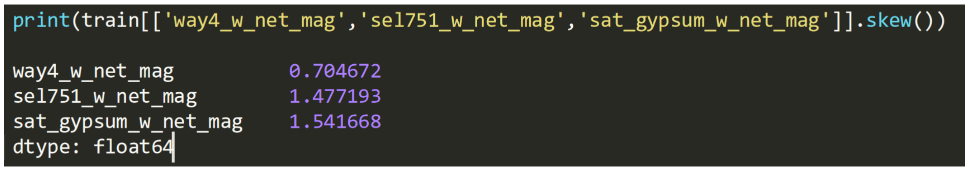


Figure 2.7: Real power measured plotted for the interval of training data

2.2.2 Skew values of the data

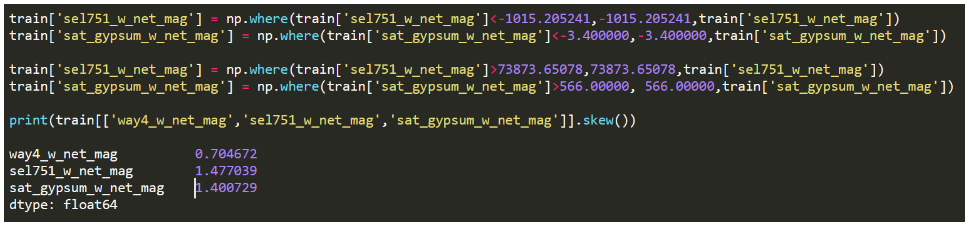
Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean [33]. Skewness value can be positive, negative, zero or undefined. Based of literature review within statistics, for a normally distributed data the skew value should be between -1 to +1. The code snippet shown in Figure 2.8 calculates the skew values for all three measurements.



```
print(train[['way4_w_net_mag','sel751_w_net_mag','sat_gypsum_w_net_mag']].skew())

way4_w_net_mag          0.704672
sel751_w_net_mag        1.477193
sat_gypsum_w_net_mag    1.541668
dtype: float64
```

Figure 2.8: Skew values for the original training data

Skew values higher than +1 and lower than -1 indicate the presence of outliers in the data. We do not want outliers in the training data. Thus, in this particular case to preserve the total number of records, we substitute those outliers identified as being lower than the tenth quantile and higher than the ninetieth quantile with the values of the tenth and ninetieth quantile respectively. The Figure 2.9 implements this behaviour and we can see the skew values modified for the three measurements.

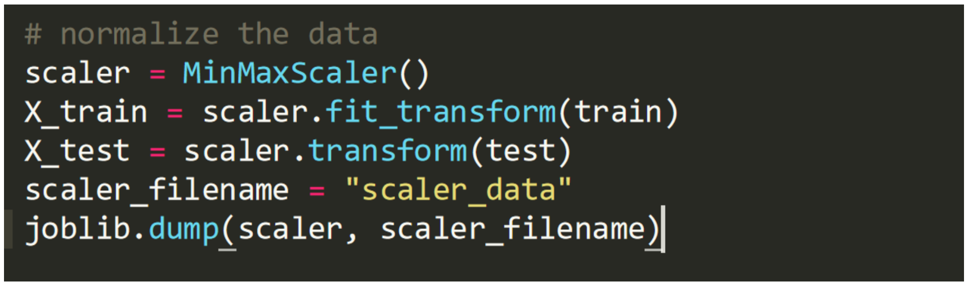


```
train['sel751_w_net_mag'] = np.where(train['sel751_w_net_mag']<-1015.205241,-1015.205241,train['sel751_w_net_mag'])
train['sat_gypsum_w_net_mag'] = np.where(train['sat_gypsum_w_net_mag']<-3.400000,-3.400000,train['sat_gypsum_w_net_mag'])

train['sel751_w_net_mag'] = np.where(train['sel751_w_net_mag']>73873.65078,73873.65078,train['sel751_w_net_mag'])
train['sat_gypsum_w_net_mag'] = np.where(train['sat_gypsum_w_net_mag']>566.00000, 566.00000,train['sat_gypsum_w_net_mag'])

print(train[['way4_w_net_mag','sel751_w_net_mag','sat_gypsum_w_net_mag']].skew())

way4_w_net_mag          0.704672
sel751_w_net_mag        1.477039
sat_gypsum_w_net_mag    1.400729
dtype: float64
```

Figure 2.9: Modified skew values after manipulating training data

2.2.3 Normalizing the data

By definition normalizing a vector often means dividing by the norm of the vector. It also often refers to rescaling by the minimum range of the vector, to make all elements lie between 0 and 1 thus bringing all numeric columns in a dataset to a common scale [34]. The goal of normalization is to change values of numeric columns in a dataset to a common scale, without distorting differences in the range of values. Normalization is a good technique to use when we do not know the distribution of our data or when we know the distribution is not Gaussian [35] (a bell curve). Normalization is useful when data has varying scales and the algorithm trying to be implemented does not make any assumptions about the distribution of the data such as deep neural networks. The Min-Max Scalar used here, is useful as it is sensitive to outliers. Essentially converting the data into bounds of 0 and 1 results in smaller standard deviations which suppress the effect of outliers. The code snippet show in the Figure 2.10 implements this behaviour.



```
# normalize the data
scaler = MinMaxScaler()
X_train = scaler.fit_transform(train)
X_test = scaler.transform(test)
scaler_filename = "scaler_data"
joblib.dump(scaler, scaler_filename)
```

Figure 2.10: Min-Max Scalar implemented from scikit-learn library

CHAPTER 3: ANOMALY DETECTION IN TIME SERIES

Observations that have been recorded in an orderly fashion and which are correlated in time constitute a time series. Time series data mining aims to extract all meaningful knowledge from this data like clustering, forecasting and outlier detection. Outlier detection has become a field of interest for many researchers and practitioners and is now one of the tasks of time series data mining [36]. Applications such as credit card fraud detection, intrusion detection in cybersecurity or fault diagnosis in the industry. From a classical point of view, a widely used definition of an outlier has been "An observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism." Therefore, outliers can be thought of as observations that do not follow expected behaviour [37]. The most popular and intuitive definition for the concept of a point outlier is a point that significantly deviates from its expected value. Therefore, given a univariate time series, a point at time t can be declared an outlier if the the distance to its expected value is higher than a predefined threshold τ:

$$|x_t - \hat{x}_t| > \tau$$

where x_t is the observed data point and $\hat{x}_t$ is its expected value. This problem is graphically depicted in the Figure 3.1 where observed values within the shadowed area are at most at distance τ from their expected values. Clearly, O3 is the point that differs the most from its expected value, although O1, O2, and O4 are also farther than distance τ, so all four points are declared outliers.

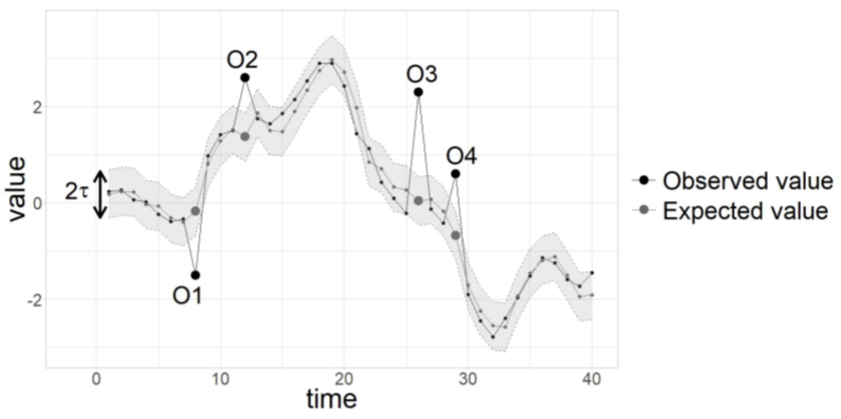


Figure 3.1: Point outlier detection in univariate time series based on comparison between expected and observed values

Outliers in time series can have two different meanings, and the semantic distinction between them is mainly based on the interest of the analyst or the particular scenario considered. These observations have been related to noise, erroneous, or unwanted data, which by themselves are not interesting to the analyst. In these cases, outliers should be deleted or corrected to improve the data quality and generate a cleaner dataset that can be used by other data mining algorithms. For example, sensor transmission errors are eliminated to obtain more accurate predictions because the principal aim is to make predictions. Nevertheless, in recent years and, especially in the area of time series data, many researchers have aimed to detect and analyze unusual but interesting phenomena. Fraud detection [38] is an example of this because the main objective is to detect and analyze the outlier itself. These observations are often referred to as anomalies.

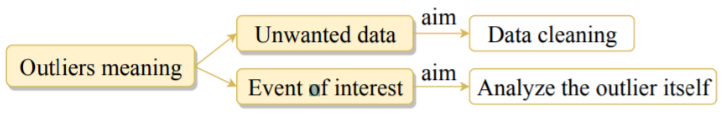


Figure 3.2: Meaning of outliers in time series data depending on aim of the analyst

Outlier detection techniques in time series data vary depending on the input data type, the outlier type, and the nature of the method as seen in Figure 3.3

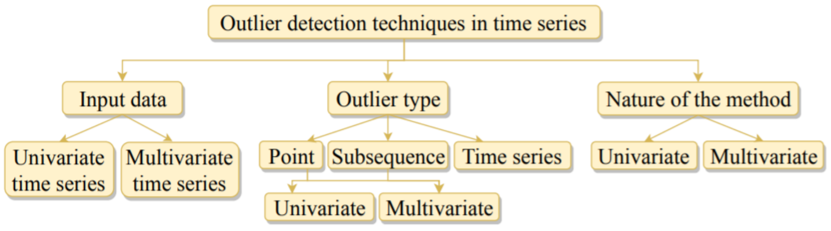
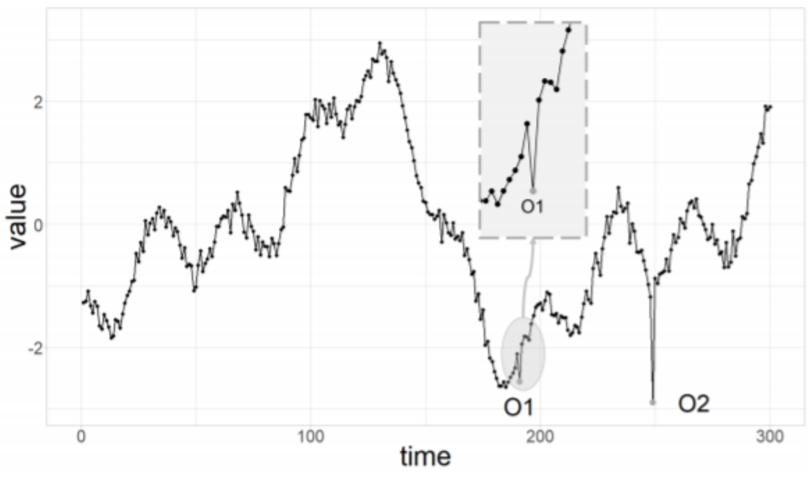


Figure 3.3: Outlier detection techniques

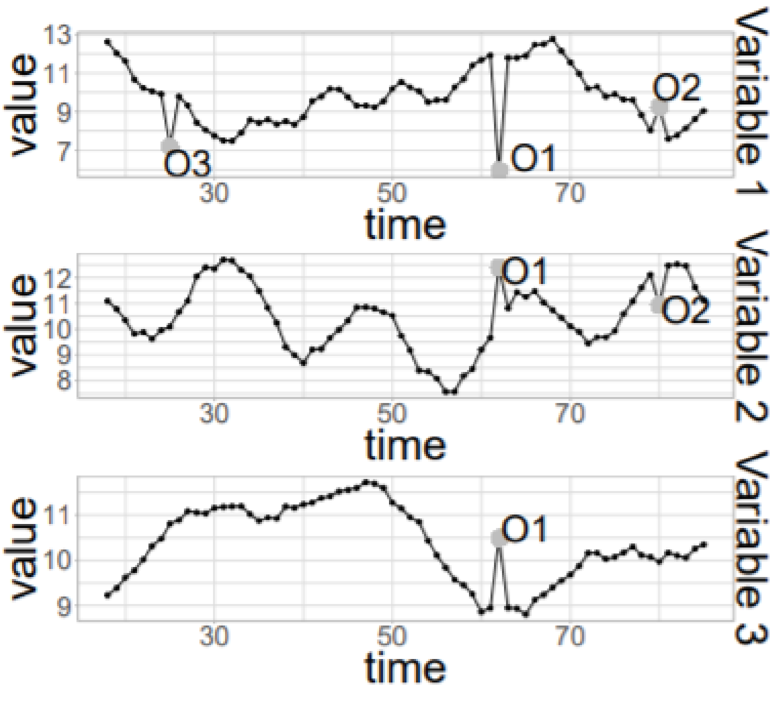
1. The first axis represents the type of input data that the detection method is able to deal with (univariate or multivariate time series).

   A univariate time series $X = \{x_t\}_{t \epsilon T}$ is an ordered set of real-valued observations where each observation is recorded at a specific time $t \epsilon T$. Then x_t is the point or observation collected at time t and $S = x_p, x_{p+1}, \cdot\cdot\cdot, x_{p+n-1}$ the subsequence of length $n \leq |T|$ starting at position p of the time series X, for $p, t \epsilon T$ and $p \leq |T| - n + 1$.

   A multivariate time series $X = \{x_t\}_{t \epsilon T}$ is defined as an ordered set of k-dimensional vectors, each of which is recorded at a specific time $t \epsilon T$ and consists of k real valued observations $x_t = (x_{1t}, \cdot\cdot\cdot, x_{kt})$. Then x_t is said to be a point and $S = x_p, x_{p+1}, \cdot\cdot\cdot, x_{p+n-1}$ the subsequence of length $n \leq |T|$ of the multivariate time series X, for $p, t \epsilon T$ and $p \leq |T| - n + 1$. For each dimension $j \epsilon 1, \cdot\cdot\cdot, K, X_j = \{x_{jt}\}_{t \epsilon T}$ is a univariate time series and each observation x_{jt} in the vector x_t is a realized value of a random time dependent variable X_{jt} in $X_t = (X1t, \cdot\cdot\cdot, X_{kt})$.



(a) Univariate time series



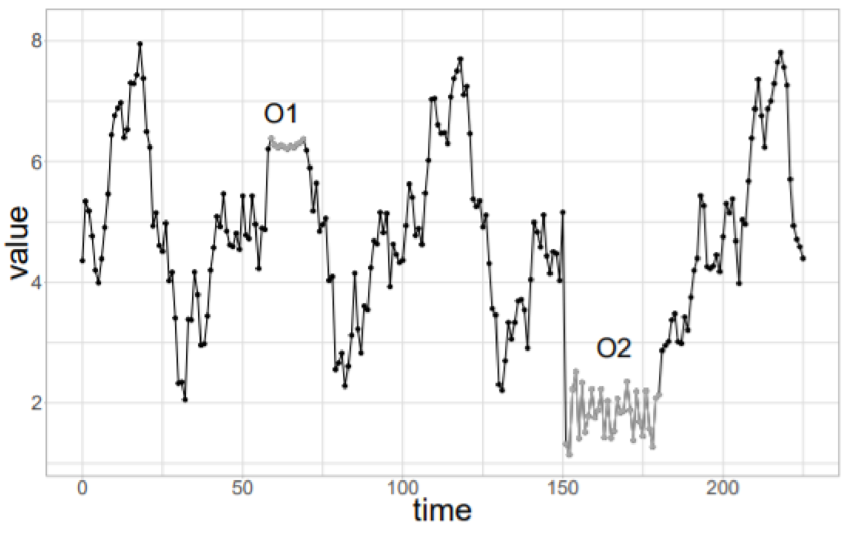
(b) Multivariate time series

Figure 3.4: Time series with univariate and multivariate input data

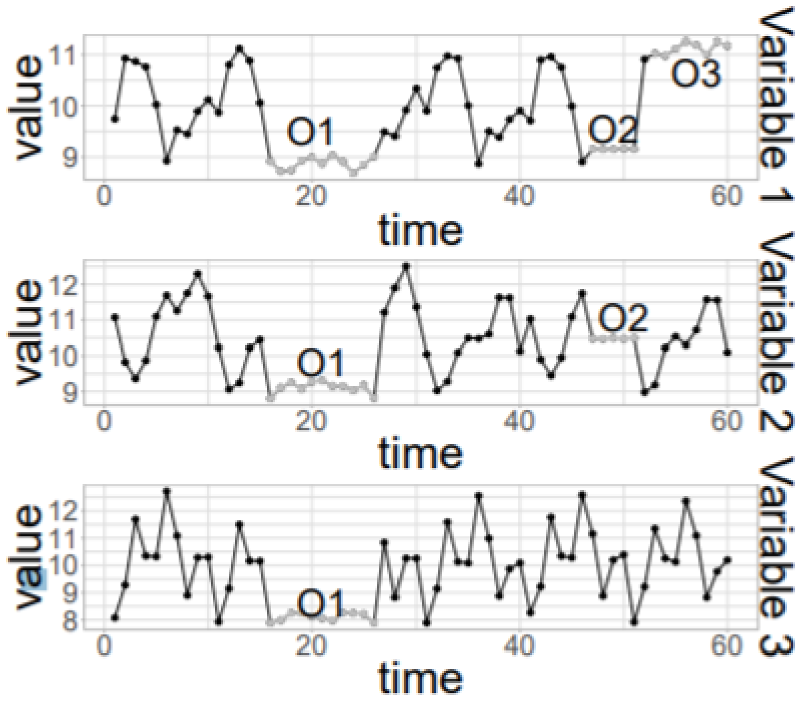
2. The second axis describes the outlier type that the method aims to detect (a point, a subsequence or a time series). Point outliers are the ones that behave

unusually in a specific time instance when compared to the other values in the time series (global outlier) or to its neighbouring points (local outlier). Point outliers can be univariate or multivariate depending on whether they affect one or more time-dependent variables, respectively. For example in Figure 3.4 in the univariate time series we can see two point outliers, O1 and O2 whereas in the multivariate time series composed of three variables has both univariate O3 and multivariate (O1 and O2) point outliers.

Subsequence outliers refer to consecutive points in time whose joint behaviour is unusual although each observation individually is not necessarily a point outlier. Subsequence outliers can be global or local and can affect one (univariate subsequence outlier) or more (multivariate subsequence outlier) time-dependent variables. Figure 3.5 provides an example of univariate (O1 and O2) and multivariate (O1 and O2) subsequence outliers.



(a) Univariate time series with subsequence outliers



(b) Multivariate time series subsequence outliers

Figure 3.5: Time series with univariate and multivariate input data and subsequence outliers

As an additional type of outliers, an entire time series can also be outliers. But they can only be detected when input data is a multivariate time series. As seen in Figure 3.6 the outlier time series corresponds to variable 4 as its behaviour is significantly different from the rest of the variables over time.

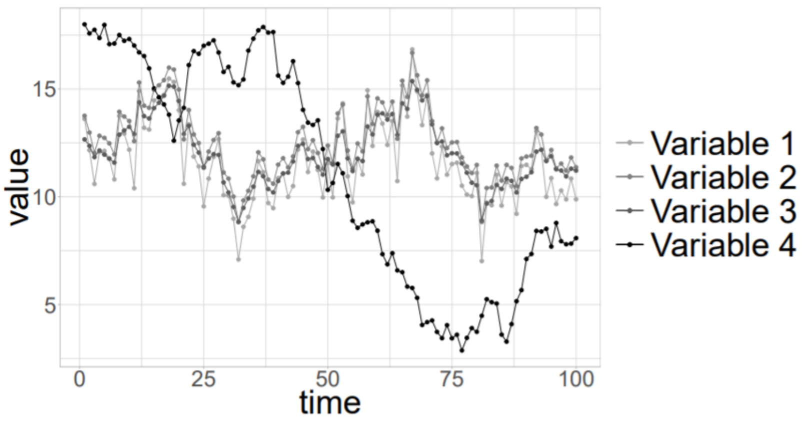


Figure 3.6: Entire time series as outlier

3. The third axis analyzes the nature of the detection method employed (if the detection method is univariate or multivariate). A univariate detection method only considers a single time dependent variable, whereas a multivariate detection method is able to simultaneously work with more than one time dependent variable. Note that the detection method can be univariate, even if the input data is a multivariate time series, because an individual analysis can be performed on each time-dependent variable without considering the dependencies that may exist between the variables. In contrast, a multivariate technique cannot be used if the input data is a univariate time series.

3.1 Anomaly detection in time series using supervised learning

This technique hinges on the prior labelling of data as "normal" or "anomalous". The algorithm is trained using existing current or historical data, and is then deployed to predict outcomes on new data. Though this technique finds application in fraud detection in the banking/ financial technology space, it can only be applied to predict known anomalies such as previously identified fraud/ misappropriations.

3.2 Unsupervised anomaly detection

In this technique, anomalies can be identified from unlabelled data by assuming a majority of the data points to be normal. Deviating instances that are statistically significant on either side of the established normal are regarded as anomalies. The more powerful the algorithm, the higher the accuracy of the anomaly detection. This method may be used for detecting anomalies in time series data and also to predict and flag future anomalies. In case of raw, unlabelled data unsupervised anomaly detection is feasible. It has a variety of applications across industries - from identifying abnormalities in ECG data to finding glitches in aircraft sensor data.

Also, we normally know 20% of the anomalies or those which are expected. The remaining 80% are new or unpredictable. Unsupervised anomaly detection is the only technique that is capable of identifying these hidden signals or anomalies - and flagging them early enough to fix them before they occur as seen in Figure 3.7

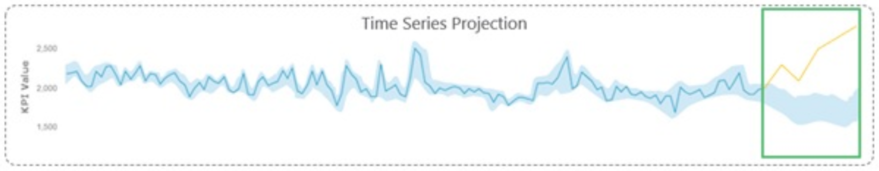


Figure 3.7: Unsupervised anomaly detection use case in flagging anomalies before they occur

One of the most effective ways of detecting anomalies in time series data is via deep learning. This technique involves the following steps:

1. Deep neural networks are applied to a series of input and output sets to establish the normal and accordingly predict the time series. This process is repeated until the predictions achieve a high level of accuracy. However, the models need

to be updated regularly to accommodate changing trends and ensure accuracy and relevance. Long short-term memory (LSTM) neural networks are great at remembering seasonal and other trends.

2. Once the model has been trained, it can predict the next series based on real-time explanatory variables.

3. Once the values are predicted, the algorithm creates upper and lower limits at a specified confidence level. For instance, a 95% confidence level means that limits need to be at a 1.96 * standard deviation with respect to the mean on both sides for a normal distribution.

4. Whenever an actual perceived value falls beyond the predicted normal range, anomalies are marked and scored based on their magnitude of deviation. A simple scoring methodology could be:

$$Score = \frac{(predicted - minimum * 100)}{(maximum - minimum)}$$

   Anomaly scores help users filter out anomalies that are less than a set threshold value and also to prioritise them so that they can focus on more serious anomalies first and then move on to less serious ones. In case of critical metrics that involve huge expenses, the threshold value can be set to zero so that the tiniest of anomalies with the lowest of scores can be scrutinised for relevant action.

Anomaly detection in industrial data is by no means a simple process given the scale at which it needs to happen, and also the highly dynamic nature of business in this world. However, its still imperative to get it right, as no digital business can hope to stay relevant and competitive in an increasingly tough economy without the power of meaningful data analytics to back its growth.

CHAPTER 4: VECTOR AUTO REGRESSION

4.1 The algorithm

The vector autoregression (VAR) model is one of the most successful, flexible, and easy to use models for the analysis of multivariate time series. It is a natural extension of the univariate autoregressive model to dynamic multivariate time series. The VAR model has proven to be especially useful for describing the dynamic behavior of economic and financial time series and for forecasting [39]. It often provides superior forecasts to those from univariate time series models and elaborate theory-based simultaneous equations models. Forecasts from VAR models are quite flexible because they can be made conditional on the potential future paths of specified variables in the model. In addition to data description and forecasting, the VAR model is also used for structural inference and policy analysis. In structural analysis, certain assumptions about the causal structure of the data under investigation are imposed, and the resulting causal impacts of unexpected shocks or innovations to specified variables on the variables in the model are summarized. These causal impacts are usually summarized with impulse response functions and forecast error variance decompositions. Vector autoregression (VAR) is a stochastic process model used to capture the linear inter dependencies among multiple time series. VAR models generalize the univariate autoregressive model (AR model) by allowing for more than one evolving variable. All variables in a VAR enter the model in the same way: each variable has an equation explaining its evolution based on its own lagged values, the lagged values of the other model variables, and an error term [40]. VAR modeling does not require as much knowledge about the forces influencing a variable as do structural models with simultaneous equations. The only prior knowledge required is a list of variables which can be hypothesized to affect each other inter-temporally. By definition, a VAR model describes the evolution of k variables called endogenous variables in over the same sample period $t = 1, \cdot, \cdot, \cdot, \cdot, T$ as a linear function of only their past values. A p^{th} order VAR denoted by VAR(p) is given by

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + e_t$$

where the observation y_{t-i} is called i^{th} lag of y, c is a k-vector of constants, A_i is a time invariant kxk matrix and e_t is a k-vector of error terms satisfying $E(e_t) = 0$ meaning every error term has mean 0. A p^{th} order VAR is also called a VAR with p-lags. The process of choosing maximum lags in VAR requires special attention because inference depends on the correctness of the selected order. One can stack the vectors in order to write a VAR(p) as a stochastic matrix difference equation, with a concise matrix notation as

$$Y = BZ + U$$

For a general example of a VAR(p) with k variables, A VAR(1) in two variables can be written in matrix form (more compact notation) as

$$\begin{bmatrix} y_{1,t} \\ y_{2,t} \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \begin{bmatrix} y_{1,t-1} \\ y_{2,t-1} \end{bmatrix} + \begin{bmatrix} e_{1,t} \\ e_{2,t} \end{bmatrix}$$

(in which only a single A matrix appears because this example has a maximum lag p equal to 1), or, equivalently, as the following system of two equations

$$y_{1,t} = c_1 + a_{1,1} y_{1,t-1} + a_{1,2} y_{2,t-1} + e_{1,t}$$

$$y_{2,t} = c_2 + a_{2,1} y_{1,t-1} + a_{2,2} y_{2,t-1} + e_{2,t}$$

Each variable in the model has one equation. The current (time t) observation of each variable depends on its own lagged values as well as on the lagged values of each other variable in the VAR. A VAR with p lags can always be equivalently rewritten as a VAR with only one lag by appropriately redefining the dependent variable. The transformation amounts to stacking the lags of the VAR(p) variable in the new VAR(1) dependent variable and appending identities to complete the number of equations. For example, the VAR(2) model

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + e_t$$

can be recast as the VAR(1) model

$$\begin{bmatrix} y_t \\ y_{t-1} \end{bmatrix} = \begin{bmatrix} c \\ 0 \end{bmatrix} + \begin{bmatrix} A_1 & A_2 \\ I & 0 \end{bmatrix} \begin{bmatrix} y_{t-1} \\ y_{t-2} \end{bmatrix} + \begin{bmatrix} e_t \\ 0 \end{bmatrix}$$

where I is the identity matrix. The equivalent VAR(1) form is more convenient for analytical derivations and allows more compact statements. An estimated VAR model can be used for forecasting, and the quality of the forecasts can be judged, in ways that are completely analogous to the methods used in univariate autoregressive modelling. Christopher Sims has advocated VAR models, criticizing the claims and performance of earlier modeling in macroeconomic econometrics. He recommended VAR models, which had previously appeared in time series statistics and in system identification, a statistical specialty in control theory. Sims advocated VAR models as providing a theory-free method to estimate economic relationships, thus being an alternative to the "incredible identification restrictions" in structural models. VAR models are also increasingly used in health research for automatic analyses of diary data or sensor data [41].

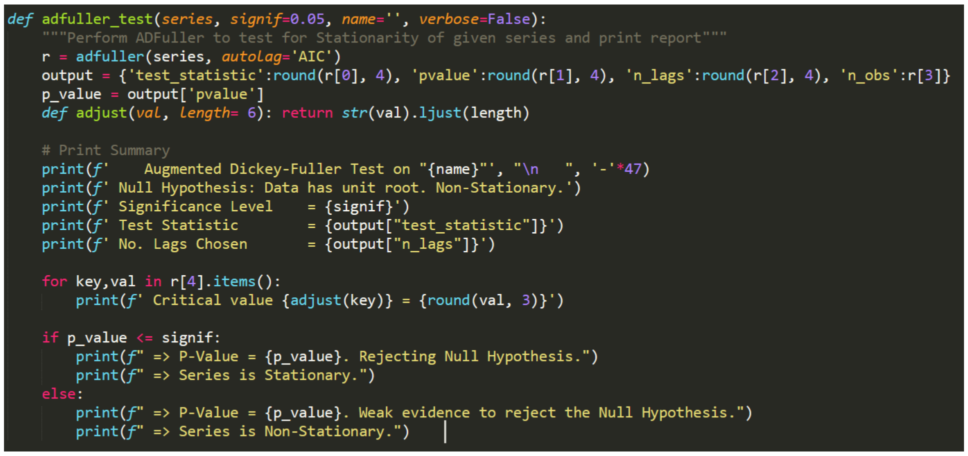
In statistics, the Dickey Fuller test tests the null hypothesis that a unit root is present in an autoregressive model. The alternative hypothesis is different depending on which version of the test is used, but is usually stationarity or trend-stationarity. It is named after the statisticians David Dickey and Wayne Fuller, who developed the test in 1979. The augmented Dickey Fuller test (ADF) tests the null hypothesis that a unit root is present in a time series sample [42]. The alternative hypothesis is different depending on which version of the test is used, but is usually stationarity or trend-stationarity. It is an augmented version of the Dickey Fuller test for a larger and more complicated set of time series models. The augmented Dickey Fuller (ADF) statistic, used in the test, is a negative number. The more negative it is, the stronger the rejection of the hypothesis that there is a unit root at some level of confidence [43]. The intuition behind the test is that if the series is characterised by a unit root process then the lagged level of the series y_{t-1} will provide no relevant information in predicting the change in y_t besides the one obtained in the lagged changes Δy_{t-k}. In this case the $\gamma = 0$ and null hypothesis is not rejected. In contrast, when the process has no unit root, it is stationary and hence exhibits reversion to the mean - so the lagged level will provide relevant information in predicting the change of the series and the null of a unit root will be rejected. The testing procedure for the ADF test is the same as for the Dickey Fuller test but it is applied to the model

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \cdots + \delta_{p-1} \Delta y_{t-p+1} + \varepsilon_t$$

where α is a constant, β the coefficient on a time trend and p the lag order of the autoregressive process. Imposing the constraints $\alpha = 0$ and $\beta = 0$ corresponds to modelling a random walk and using the constraint $\beta = 0$ corresponds to modeling a random walk with a drift. Consequently, there are three main versions of the test, analogous to the ones discussed on Dickey Fuller test. By including lags of the order p the ADF formulation allows for higher-order autoregressive processes. This means that the lag length p has to be determined when applying the test. One possible approach is to test down from high orders and examine the t-values on coefficients. An alternative approach is to examine information criteria such as the Akaike information criterion, Bayesian information criterion or the Hannan Quinn information criterion. The unit root test is then carried out under the null hypothesis $\gamma = 0$ against the alternative hypothesis of $\gamma < 0$. Once a value for the test statistic is computed it can be compared to the relevant critical value for the Dickey Fuller test [44]. As this test is asymmetrical, we are only concerned with negative values of our test statistic DF_τ. If the calculated test statistic is less (more negative) than the critical value, then the null hypothesis of $\gamma = 0$ is rejected and no unit root is present.

$$DF_\tau = \frac{\hat{\gamma}}{SE(\hat{\gamma})}$$

The Augmented Dickey Fuller test is performed as shown in the code snippet shown in Figure 4.1



```python
def adfuller_test(series, signif=0.05, name='', verbose=False):
    """Perform ADFuller to test for Stationarity of given series and print report"""
    r = adfuller(series, autolag='AIC')
    output = {'test_statistic':round(r[0], 4), 'pvalue':round(r[1], 4), 'n_lags':round(r[2], 4), 'n_obs':r[3]}
    p_value = output['pvalue']
    def adjust(val, length= 6): return str(val).ljust(length)

    # Print Summary
    print(f'    Augmented Dickey-Fuller Test on "{name}"', "\n   ", '-'*47)
    print(f' Null Hypothesis: Data has unit root. Non-Stationary.')
    print(f' Significance Level    = {signif}')
    print(f' Test Statistic        = {output["test_statistic"]}')
    print(f' No. Lags Chosen       = {output["n_lags"]}')

    for key,val in r[4].items():
        print(f' Critical value {adjust(key)} = {round(val, 3)}')

    if p_value <= signif:
        print(f" => P-Value = {p_value}. Rejecting Null Hypothesis.")
        print(f" => Series is Stationary.")
    else:
        print(f" => P-Value = {p_value}. Weak evidence to reject the Null Hypothesis.")
        print(f" => Series is Non-Stationary.")
```

Figure 4.1: Code snippet to perform the Augmented Dickey Fuller test

The Granger causality test is a statistical hypothesis test for determining whether one time series is useful in forecasting another, first proposed in 1969. Ordinarily, regressions reflect "mere" correlations, but Clive Granger argued that causality in economics could be tested for by measuring the ability to predict the future values of a time series using prior values of another time series. Since the question of "true causality" is deeply philosophical, and because of the post hoc ergo propter hoc fallacy of assuming that one thing preceding another can be used as a proof of causation, econometricians assert that the Granger test finds only "predictive causality". Using the term "causality" alone is a misnomer, as Granger-causality is better described as "precedence", or, as Granger himself later claimed in 1977, "temporally related". Rather than testing whether Y causes X, the Granger causality tests whether Y forecasts X [45].

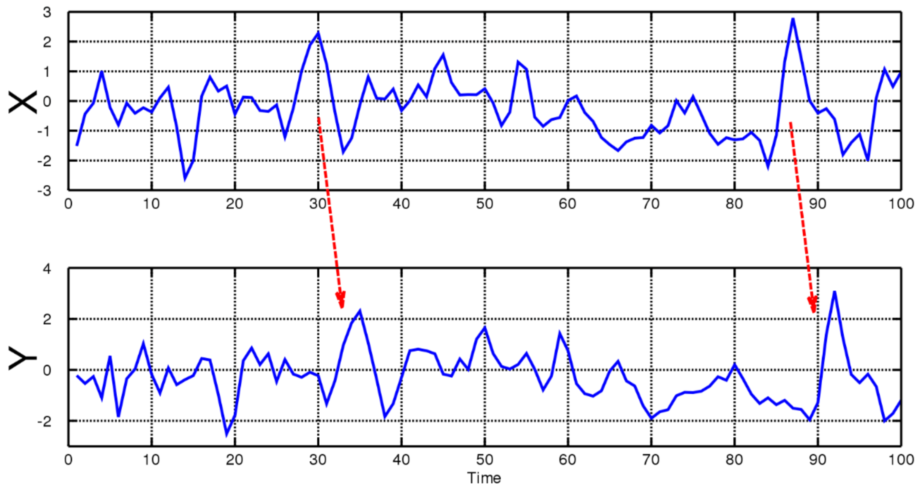


Figure 4.2: When time series X Granger-causes time series Y, the patterns in X are approximately repeated in Y after some time lag

A time series X is said to Granger-cause Y if it can be shown, usually through a series of t-tests and F-tests on lagged values of X (and with lagged values of Y also included), that those X values provide statistically significant information about future values of Y. Granger also stressed that some studies using "Granger causality" testing in areas outside economics reached "ridiculous" conclusions. "Of course, many ridiculous papers appeared", he said in his Nobel lecture. However, it remains a popular method for causality analysis in time series due to its computational simplicity. The original definition of Granger causality does not account for latent confounding effects and does not capture instantaneous and non-linear causal relationships, though several extensions have been proposed to address these issues. We say that a variable X that evolves over time Granger-causes another evolving variable Y if predictions of the value of Y based on its own past values and on the past values of X are better than predictions of Y based only on Y's own past values. Granger defined the causality relationship based on two principles:

1. The cause happens prior to its effect.
2. The cause has unique information about the future values of its effect.

Given these two assumptions about causality, Granger proposed to test the following hypothesis for identification of a causal effect of X on Y:

$$\mathbb{P}[Y(t+1) \in A \mid \mathcal{I}(t)] \neq \mathbb{P}[Y(t+1) \in A \mid \mathcal{I}_{-X}(t)]$$

where $\mathbb{P}$ refers to probability, A is an arbitrary non-empty set, and $\mathcal{I}(t)$ and $\mathcal{I}_{-X}(t)$ respectively denote the information available as of time t in the entire universe, and that in the modified universe in which X is excluded. If the above hypothesis is accepted, we say that X Granger-causes Y. If a time series is a stationary process, the test is performed using the level values of two (or more) variables. If the variables are non-stationary, then the test is done using first (or higher) differences. The number of lags to be included is usually chosen using an information criterion, such as the Akaike information criterion or the Schwarz information criterion. Any particular lagged value of one of the variables is retained in the regression if (1) it is significant according to a t-test, and (2) it and the other lagged values of the variable jointly add explanatory power to the model according to an F-test. Then the null hypothesis of no Granger causality is not rejected if and only if no lagged values of an explanatory variable have been retained in the regression [46]. In practice it may be found that neither variable Granger-causes the other, or that each of the two variables Granger-causes the other. Let y and x be stationary time series. To test the null hypothesis that x does not Granger-cause y, one first finds the proper lagged values of y to include in a univariate autoregression of y:

$$y_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_m y_{t-m} + \text{error}_t$$

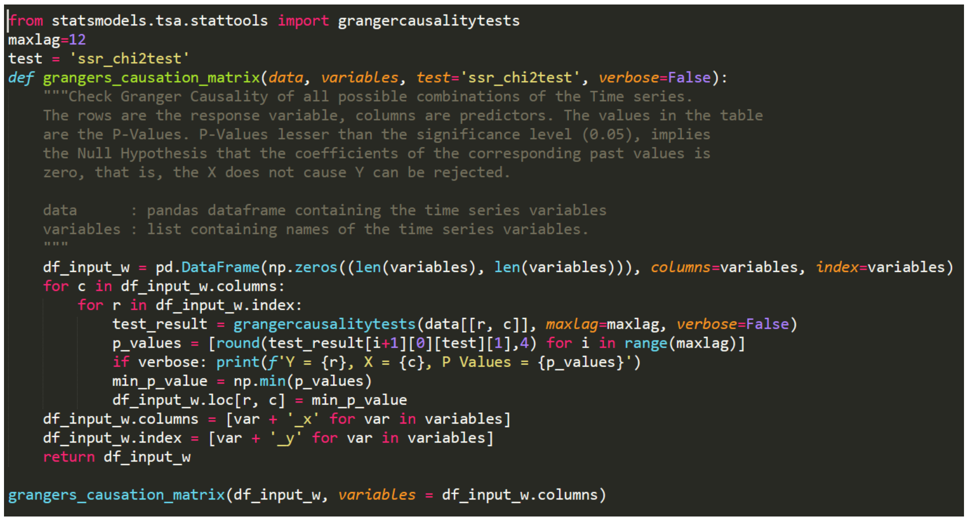
Next, the autoregression is augmented by including lagged values of x:

$$y_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_m y_{t-m} + b_p x_{t-p} + \cdots + b_q x_{t-q} + \text{error}_t$$

One retains in this regression all lagged values of x that are individually significant according to their t-statistics, provided that collectively they add explanatory power to the regression according to an F-test (whose null hypothesis is no explanatory power jointly added by the x's). In the notation of the above augmented regression, p is the shortest, and q is the longest, lag length for which the lagged value of x is significant. The null hypothesis that x does not Granger-cause y is accepted if and only if no lagged values of x are retained in the regression. Multivariate Granger causality analysis is usually performed by fitting a vector autoregressive model (VAR) to the time series. In particular, let $X(t) \in \mathbb{R}^{d \times 1}$ for $t = 1, \ldots, T$ be a d-dimensional multivariate time series. Granger causality is performed by fitting a VAR model with L time lags as follows:

$$X(t) = \sum_{\tau=1}^{L} A_\tau X(t-\tau) + \varepsilon(t)$$

where $\varepsilon(t)$ is a white Gaussian random vector, and A_τ is a matrix for every τ . A time series X_i is called a Granger cause of another time series X_j, if at least one of the elements $A_\tau(j,i)$ for $\tau = 1, \ldots, L$ is significantly larger than zero (in absolute value). The Granger Causality test is can be performed as shown by the code snippet in Figure 4.3



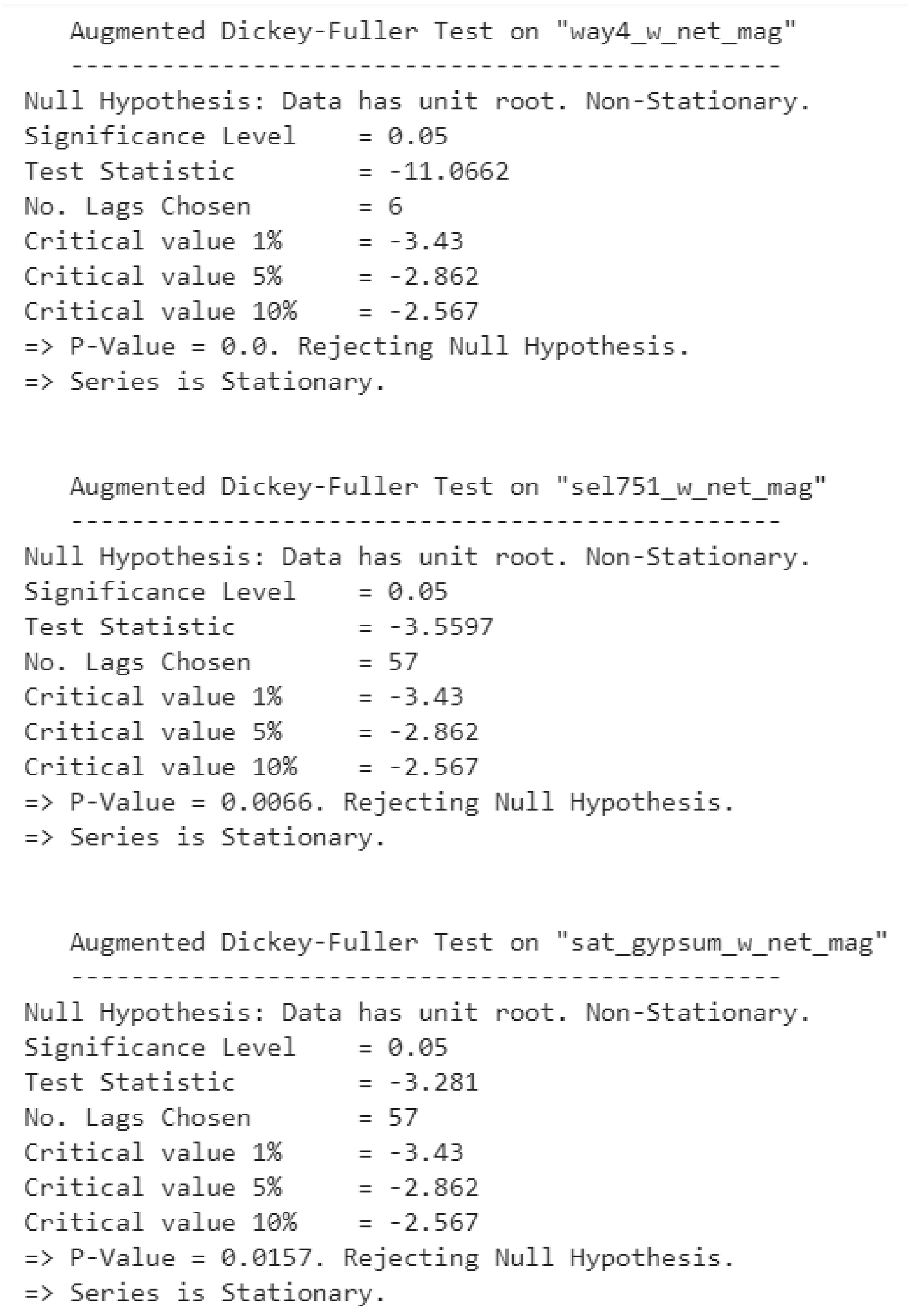
```python
from statsmodels.tsa.stattools import grangercausalitytests
maxlag=12
test = 'ssr_chi2test'
def grangers_causation_matrix(data, variables, test='ssr_chi2test', verbose=False):
    """Check Granger Causality of all possible combinations of the Time series.
    The rows are the response variable, columns are predictors. The values in the table
    are the P-Values. P-Values lesser than the significance level (0.05), implies
    the Null Hypothesis that the coefficients of the corresponding past values is
    zero, that is, the X does not cause Y can be rejected.

    data      : pandas dataframe containing the time series variables
    variables : list containing names of the time series variables.
    """
    df_input_w = pd.DataFrame(np.zeros((len(variables), len(variables))), columns=variables, index=variables)
    for c in df_input_w.columns:
        for r in df_input_w.index:
            test_result = grangercausalitytests(data[[r, c]], maxlag=maxlag, verbose=False)
            p_values = [round(test_result[i+1][0][test][1],4) for i in range(maxlag)]
            if verbose: print(f'Y = {r}, X = {c}, P Values = {p_values}')
            min_p_value = np.min(p_values)
            df_input_w.loc[r, c] = min_p_value
    df_input_w.columns = [var + '_x' for var in variables]
    df_input_w.index = [var + '_y' for var in variables]
    return df_input_w

grangers_causation_matrix(df_input_w, variables = df_input_w.columns)
```

Figure 4.3: Code snippet to perform the Granger Causality test

4.2 Anomaly Detection

This section describes the anomaly detection algorithm using the statistical vector auto regression model. The problems with the dataset has been discussed in chapter 2 in detail. The data that we are using is from an real world microgrid. It is the net power values from three different points in the microgrid at the frequency of one second. But because of inconsistent timestamps, we resampled it to have a frequency of every 10 seconds. Also, the scaling of the data is an issue more so because this is a statistical model and not a deep learning model which can work with mostly raw data. We have to scale the three variables taken into consideration onto a common scale. We use a MinMax scaler from the Scikit learn package library in Python to scale our data values between 0 and 1.

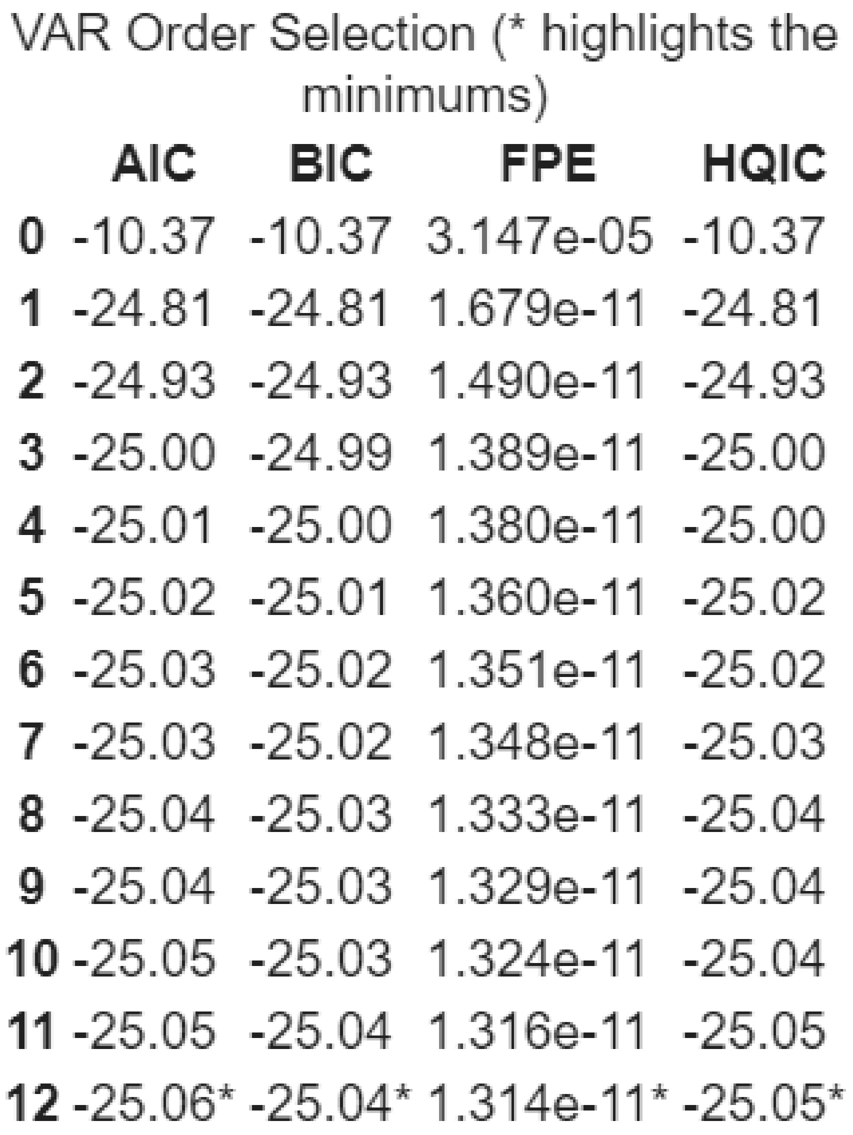
Now before we go any further, we need to check for stationarity in the time series data that we have. Checking for stationarity is important because it is one type of dependence structure. Suppose we have a data X from 1 to n. The basic assumption is that one is independent and hence is a sample. This property of independence is nice as it can be used to derive a lot of useful results. But sometimes, this property does not hold true. Two random variables can be independent only in one way, but they can be dependent in various ways. So stationarity is one way of modeling the dependence structure. Hence, here we check for stationarity of all the three variables using the Augmented Dickey Fuller test as shown in the figure 4.4.



```
   Augmented Dickey-Fuller Test on "way4_w_net_mag"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -11.0662
No. Lags Chosen       = 6
Critical value 1%     = -3.43
Critical value 5%     = -2.862
Critical value 10%    = -2.567
=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.


   Augmented Dickey-Fuller Test on "sel751_w_net_mag"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -3.5597
No. Lags Chosen       = 57
Critical value 1%     = -3.43
Critical value 5%     = -2.862
Critical value 10%    = -2.567
=> P-Value = 0.0066. Rejecting Null Hypothesis.
=> Series is Stationary.


   Augmented Dickey-Fuller Test on "sat_gypsum_w_net_mag"
   -----------------------------------------------
Null Hypothesis: Data has unit root. Non-Stationary.
Significance Level    = 0.05
Test Statistic        = -3.281
No. Lags Chosen       = 57
Critical value 1%     = -3.43
Critical value 5%     = -2.862
Critical value 10%    = -2.567
=> P-Value = 0.0157. Rejecting Null Hypothesis.
=> Series is Stationary.
```

Figure 4.4: Augmented Dickey Fuller test results showing the series as stationary

After this we look for the lag order. For forecasting purposes the best statistical criterion to choose is the AIC. It tends to select the model from a pool of models that yields the smallest squared forecast error one step ahead. That is as shown in the figure 4.5. We choose the AIC criterion. For a maximum of 12 lags the different AIC values are calculated and we choose the lag number which has the minimum AIC value.



VAR Order Selection (* highlights the minimums)

	AIC	BIC	FPE	HQIC
0	-10.37	-10.37	3.147e-05	-10.37
1	-24.81	-24.81	1.679e-11	-24.81
2	-24.93	-24.93	1.490e-11	-24.93
3	-25.00	-24.99	1.389e-11	-25.00
4	-25.01	-25.00	1.380e-11	-25.00
5	-25.02	-25.01	1.360e-11	-25.02
6	-25.03	-25.02	1.351e-11	-25.02
7	-25.03	-25.02	1.348e-11	-25.03
8	-25.04	-25.03	1.333e-11	-25.04
9	-25.04	-25.03	1.329e-11	-25.04
10	-25.05	-25.03	1.324e-11	-25.04
11	-25.05	-25.04	1.316e-11	-25.05
12	-25.06*	-25.04*	1.314e-11*	-25.05*

Figure 4.5: Order selection in VAR using various criterion

The model is fit on lag order of 12. We use this fitted model to forecast for the test data and then use the mean absolute error function to compute the average loss. Plotting a loss distribution helps to identify a threshold value beyond which anomalies can be classified as "True". The plotted loss distribution can be seen in the figure 4.6.

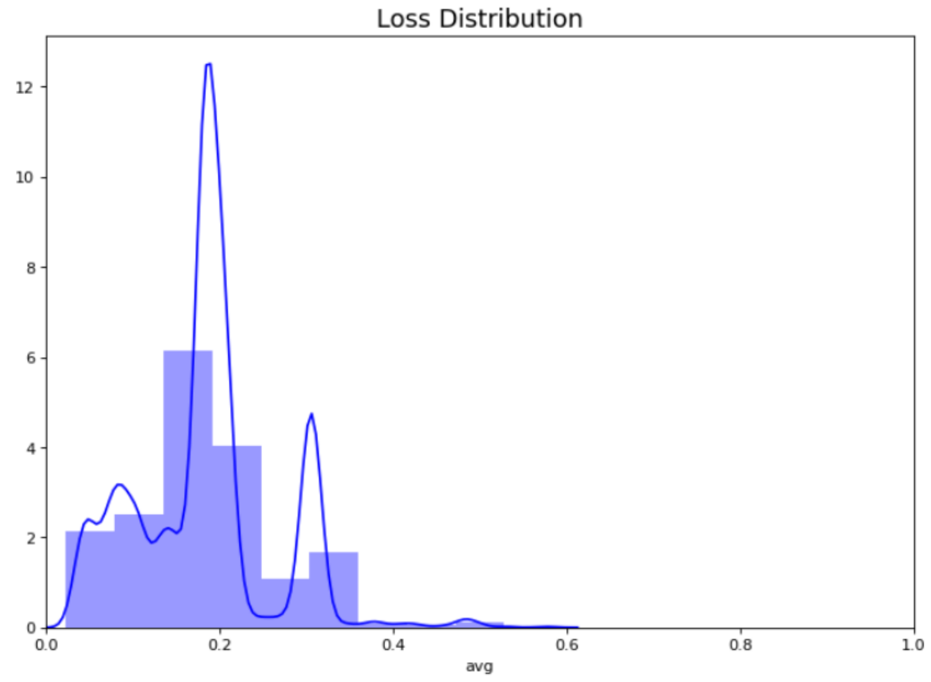


Figure 4.6: VAR Loss distribution plot

Looking at this plot, we can appropriate a threshold value of "0.35". Therefore, any loss which goes beyond this threshold is classified as a true anomaly. The test data anomaly plot is as shown in the figure 4.7. The red line is the threshold line and the blue line plot is the test data loss values. Based on this plot, the timestamps can be identified which have been classified as having anomalous data and further investigation can be done. By digging deeper and reverse engineering our way through the timestamp values, first sampled at 10 seconds and then at each second which is the original data we can identify if the values were truly anomalous using the help of subject matter experts.

The data we have is from a real world microgrid and the data points taken in time when it was in "normal" or expected operations. Hence, the chances of anomalies occurring are reduced greatly. The VAR model took 1 minute to run in Google Colaboratory environment on 2vCPU at 2.2GHz and 13 GB RAM. It successfully detected 145 anomalies from a total of 8,977 data points in the test data which is 1.65% of the total test data. Although, this makes sense the sheer number of the anomalies leads to believe the presence of false positives. Also the fact that this is a statistical algorithm and not a deep neural network, the chances of false positives occurring are a lot more and cannot be neglected.

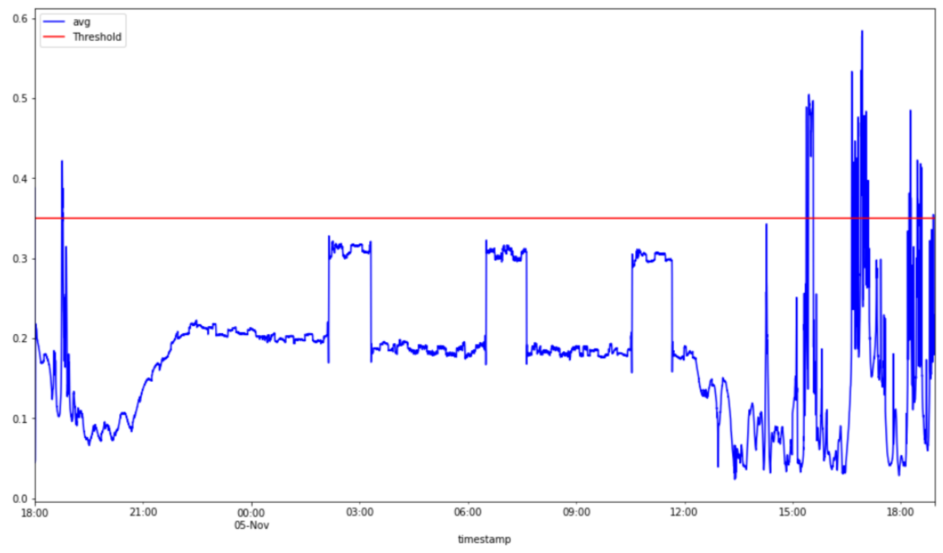


Figure 4.7: Anomaly detection in test data set using VAR

CHAPTER 5: LSTM AUTOENCODER

5.1 The algorithm

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feed forward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video) [47]. For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDS's (intrusion detection systems). A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate as shown in Figure 5.1. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTM's were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNN's. Relative insensitivity to gap length is an advantage of LSTM over RNN's, hidden Markov models and other sequence learning methods in numerous applications. In theory, classic (or "vanilla") RNN's can keep track of arbitrary long-term dependencies in the input sequences. The problem of vanilla RNN's is computational (or practical) in nature: when training a vanilla RNN using back-propagation, the gradients which are back-propagated can "vanish" (that is, they can tend to zero) or "explode" (that is, they can tend to infinity), because of the computations involved in the process, which use finite-precision numbers [48]. RNN's using LSTM units partially solve the vanishing gradient problem, because LSTM units allow gradients to also flow unchanged.

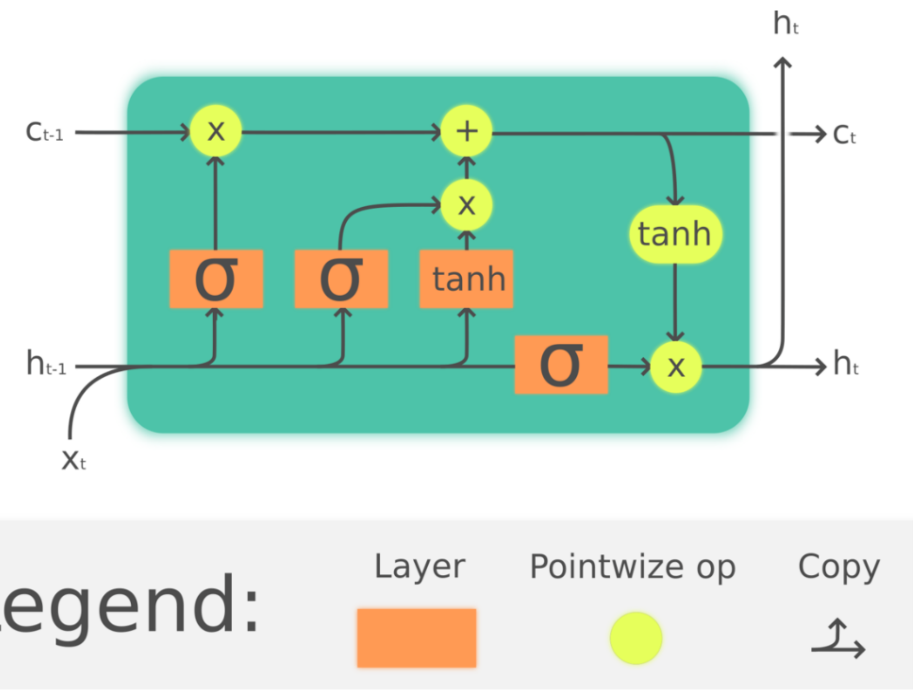


Figure 5.1: LSTM cell can process data sequentially and keep its hidden state through time

In 1997, LSTM was proposed by Sepp Hochreiter and Jurgen Schmidhuber. By introducing Constant Error Carousel (CEC) units, LSTM deals with vanishing gradient problem. The initial version of LSTM block included cells, input and output gates [48]. In 1999, Felix Gers and his advisor Jurgen Schmidhuber and Fred Cummins introduced the forget gate (also called the "keep gate") into the LSTM architecture, enabling the LSTM to reset its own state. Gers and Schmidhuber and Cummins added peephole connections in the year 2000 (connections from the cell to gate) into the architecture. Additionally, the output activation function was omitted. In the early 2009, an LSTM based model won the ICDAR connected handwriting recognition competition. Three such models were submitted by team lead Alex Graves, One was the most accurate model in the competition and the other was the fastest. LSTM networks were a major component of the network that achieved a record 17.7% phoneme error rate on the classic TIMIT natural speech dataset in the year 2013. In

2014, a simplified variant called the Gated Recurrent Unit (GRU) was put forward. Google started using an LSTM for speech recognition on Google voice in the year 2015. According to the official blog post, the new model cut transcription errors by 49%. In the year 2016, Google started using an LSTM to suggest messages in the Allo conversation app. In the same year, Google released the Google Neural Machine Translation system for Google Translate which used LSTM's to reduce translation errors by 60%. Apple announced in its Worldwide Developer Conference that it would start using the LSTM for quick type in the iPhone and for Siri. Amazon released Polly, which generates the voices behind Alexa, using a bidirectional LSTM for the text-to-speech technology in the same year. Facebook performed around 4.5 billion automatic translations every day using long short-term memory networks in the year 2017. Researchers from the Michigan State University, IBM Research and Cornell University published a study in the Knowledge Discovery and Data Mining (KDD) conference which described a novel neural network that performs better on certain data sets than the widely used long short-term memory networks. Microsoft reported reaching 94.9% recognition accuracy on the Switchboard corpus, incorporating a vocabulary of 165,000 words. The approach used "dialog session-based long-short-term memory". In 2019, Researchers form the University of Waterloo proposed a related RNN architecture which represents continuous windows of time. It was derived using the Legendre polynomials and outperforms the LSTM on some memory-related benchmarks. An LSTM model climbed to the third place on the Large text compression benchmark.

There are several architectures of LSTM units. A common architecture is composed of a cell (the memory part of the LSTM unit) and three "regulators", usually called gates, of the flow of information inside the LSTM unit: an input gate, an output gate and a forget gate. Some variations of the LSTM unit do not have one or more of these gates or maybe have other gates. For example, gated recurrent units (GRU's) do not have an output gate as shown in the Figure 5.2.

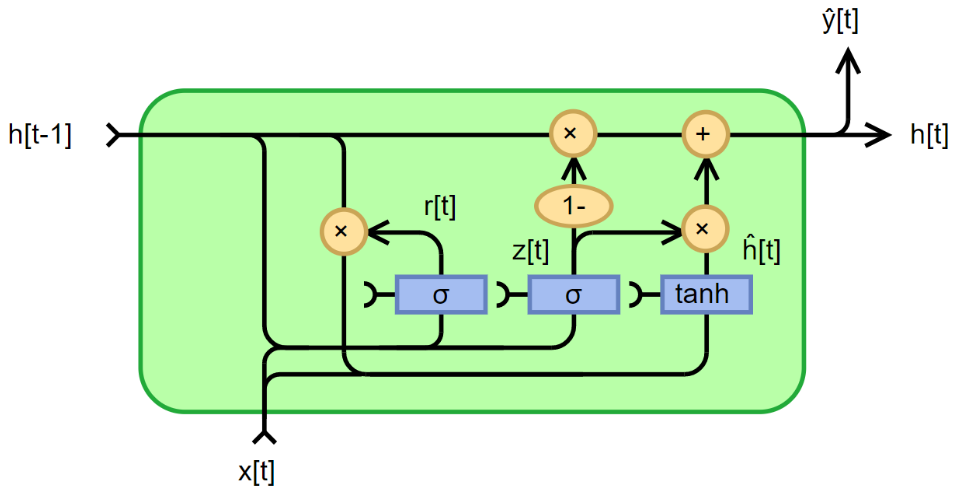


Figure 5.2: A fully gated version of Gated Recurrent Unit

Intuitively, the cell is responsible for keeping track of the dependencies between the elements in the input sequence. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. The activation function of the LSTM gates is often the logistic sigmoid function. There are connections into and out of the LSTM gates, a few of which are recurrent. The weights of these connections, which need to be learned during training, determine how the gates operate.

An RNN using LSTM units can be trained in a supervised fashion, on a set of training sequences, using an optimization algorithm, like gradient descent, combined with back propagation through time to compute the gradients needed during the optimization process, in order to change each weight of the LSTM network in proportion to the derivative of the error (at the output layer of the LSTM network) with respect to corresponding weight. A problem with using gradient descent for standard RNNs is that error gradients vanish exponentially quickly with the size of the time lag between important events. This is due to $\lim_{n\to\infty} W^n = 0$ if the spectral radius of W is smaller than 1. However, with LSTM units, when error values are back-propagated from the output layer, the error remains in the LSTM unit's cell. This "error carousel" continuously feeds error back to each of the LSTM unit's gates, until they learn to cut off the value. Many applications use stacks of LSTM RNN's and train them by connection temporal classification (CTC) to find an RNN weight matrix that maximizes the probability of the label sequences in a training set, given the corresponding input sequences. CTC achieves both alignment and recognition. Sometimes, it can be advantageous to train (parts of) an LSTM by neuro-evolution or by policy gradient methods, especially when there is no "teacher" (that is, training labels).

There have been several successful stories of training, in a non-supervised fashion, RNN's with LSTM units. In 2018, Bill Gates called it a "huge milestone in advancing artificial intelligence" when bots developed by OpenAI were able to beat humans in the game of Dota 2 [49]. OpenAI Five consists of five independent but coordinated neural networks. Each network is trained by a policy gradient method without supervising teacher and contains a single-layer, 1024-unit Long-Short-Term-Memory that sees the current game state and emits actions through several possible action heads. In 2018, OpenAI also trained a similar LSTM by policy gradients to control a human-like robot hand that manipulates physical objects with unprecedented dexterity. In 2019, DeepMind's program AlphaStar used a deep LSTM core to excel at the complex video game Starcraft II [50]. This was viewed as significant progress towards Artificial General Intelligence.

An LSTM Autoencoder is an implementation of an autoencoder for sequence data using an Encoder-Decoder LSTM Architecture. Once fit, the encoder part of the model can be used to encode or compress sequence data that in turn may be used in data visualizations or as a feature vector input to a supervised learning model. An autoencoder is a neural network model that seeks to learn a compressed representation of an input. They are an unsupervised learning method, although they are trained using supervised and thus referred to as self-supervised. They are typically trained as part of a broader model that attempts to recreate the input. The design of autoencoder model purposefully makes this challenging by restricting the architecture to a bottleneck at the midpoint of the model from which the reconstruction of the input data is performed. A problem with sequences is that the length of the sequence can vary. This is challenging because machine learning algorithms are designed to work with fixed length inputs. Another challenge with sequence data is that the temporal ordering of the observations can make it challenging to extract features suitable for use as input to supervised learning models, often requiring deep expertise in the domain or in the field of signal processing. Recurrent neural networks such as the Long Short-Term Memory networks are specifically designed to support sequences of input data [51]. They are capable of learning the complex dynamics within the temporal ordering of input sequences as well as use an internal memory to remember or use information across long input sequences. The LSTM network can be organized into an architecture called the Encoder-Decoder LSTM that allows the model to be used to both support variable length input sequences and to predict or output variable length output sequences. This architecture is the basis for many advances in complex sequence prediction problems such as speech recognition and text translation. An encoder LSTM model reads the input sequence step-by-step. After reading the entire input sequence, the hidden state or output of this model represents an internal learned representation of the entire input sequence as a fixed length vector. This vector is then provided as an input to the decoder model that interprets it as each step in the output sequence is generated. For a given dataset of sequences, an encoder-decoder LSTM is configured to read the input sequence, encode it, decode it and recreate it. The performance of the model is evaluated based on the model's ability to recreate the input sequence. Once the model achieves a desired level of performance recreating the sequence, the decoder part of the model may be removed, leaving just the encoder model. This model can then be used to encode input sequences to a fixed length vector. The resulting vector can then be used in a variety of applications, not least as a compressed representation of the sequence as an input to another supervised learning model.

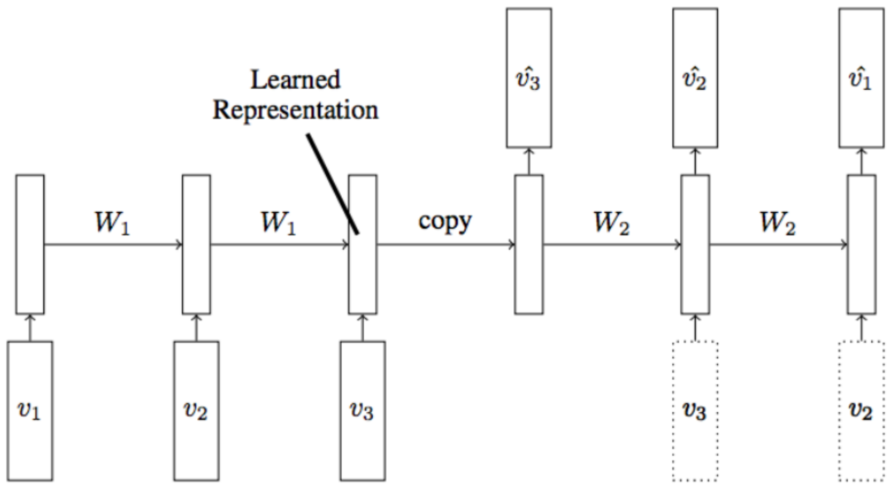


Figure 5.3: An LSTM Autoencoder model

5.2 Anomaly detection

This section discusses the anomaly detection from the time series data using the LSTM Autoencoder model explained in the previous section. The data that we are looking at is the net power values at different points in a microgrid and is generated at 1 second intervals. Because of the inconsistent timestamps, it had to be resampled for every 10 seconds and is discussed in detail in the second chapter of this thesis. Another problem with this dataset is the scaling. As can be seen from the figure 5.4, the three different variables that we are considering have different scales thus different maximum and minimum values. Training the network on this data would give erroneous results.

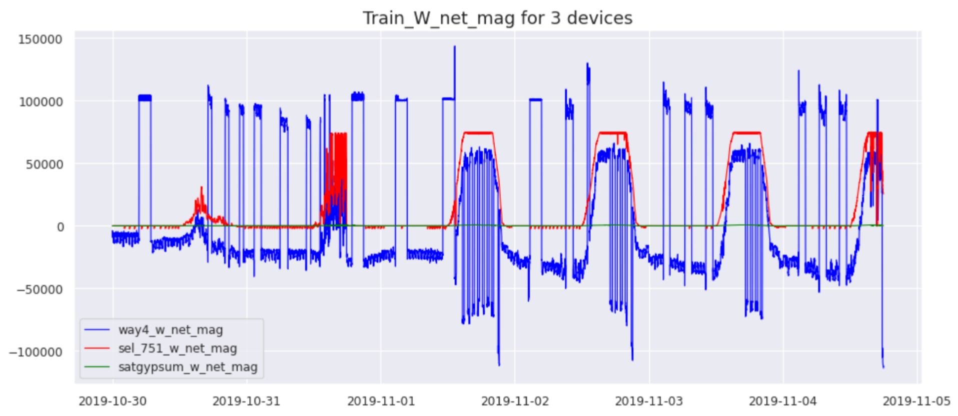


Figure 5.4: Training data from 3 different devices at different points in the microgrid sampled at 10 second intervals

Hence, we use a MinMax scaler from the python package Scikit-learn, to scale this data into the range of 0 to 1. Now, all the three variables have the same scale as can be seen, from the figure 5.5. Now that we have the scaled data, we can perform further model training on this data. Anomaly detection tasks are simplified if the data points from different variables have the same scale.

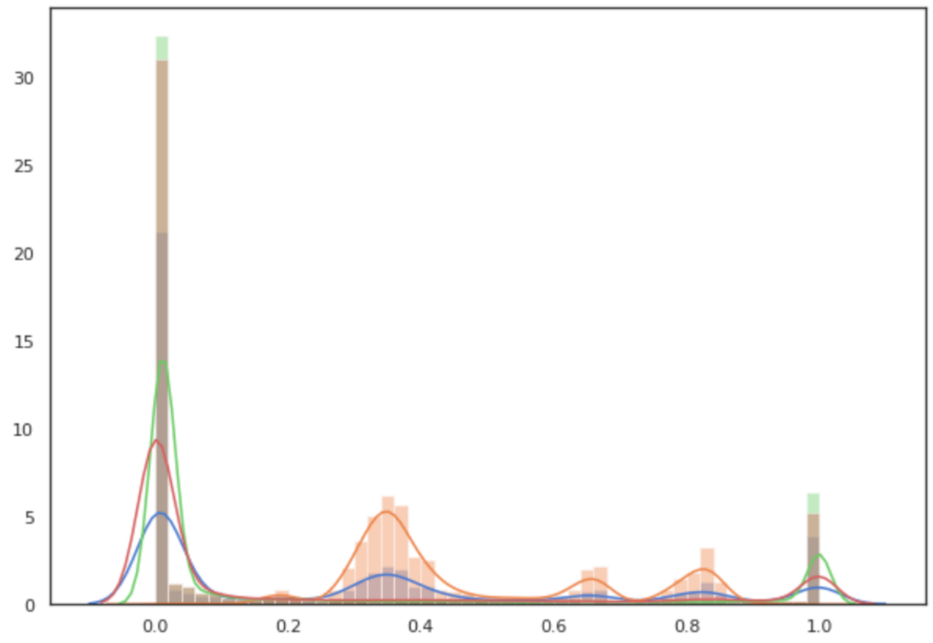


Figure 5.5: Training data from 3 different devices at different points in the microgrid sampled at 10 second intervals scaled between 0 to 1

As mentioned in the previous section, the LSTM Autoencoder features an encoder and decoder section which essentially attempts to reconstruct the inputs. The figure 5.6 shows the model architecture. As can be seen from the model architecture, the input data is reshaped into the format of [samples, timesteps, features]. The input is then given to an LSTM layer with 32 filters with a rectified linear unit as the activation function. This layer has an L2 regularizer to penalize the layer's kernel. Each LSTM cell will output one hidden state for each input. The return sequences parameter is set to "True" here so that the hidden states return the hidden state output for each time step. Then, the output from this layer is given to another LSTM layer with 8 filters with the same activation function. Here, the return sequences parameter is set to "False" to reduce the dimension as can be seen in the model architecture. This portion forms the encoder part of the model. Thus, the output of this layer is given to a Repeat vector layer. This layer repeats the input specified number of times. After this layer, comes the decoder part which is essentially flipping the encoder network. Thus, the output from the repeat vector layer is given to another LSTM layer with 8 filters with the same activation function of rectified linear unit. The output of this layer is given to an LSTM layer with 32 filters and the same activation function. For both these layers, the return sequences parameter has to be set to "True" for the dimensions to stay intact. After this decoder part, the final output layer is a Time Distributed Dense layer which brings down the output dimensions and matches it to that of the input layer as we can see from the figure. This wrapper allows to apply a temporal layer to every slice of an input. It basically applies a specific layer such as Dense, to every sample it receives as input.

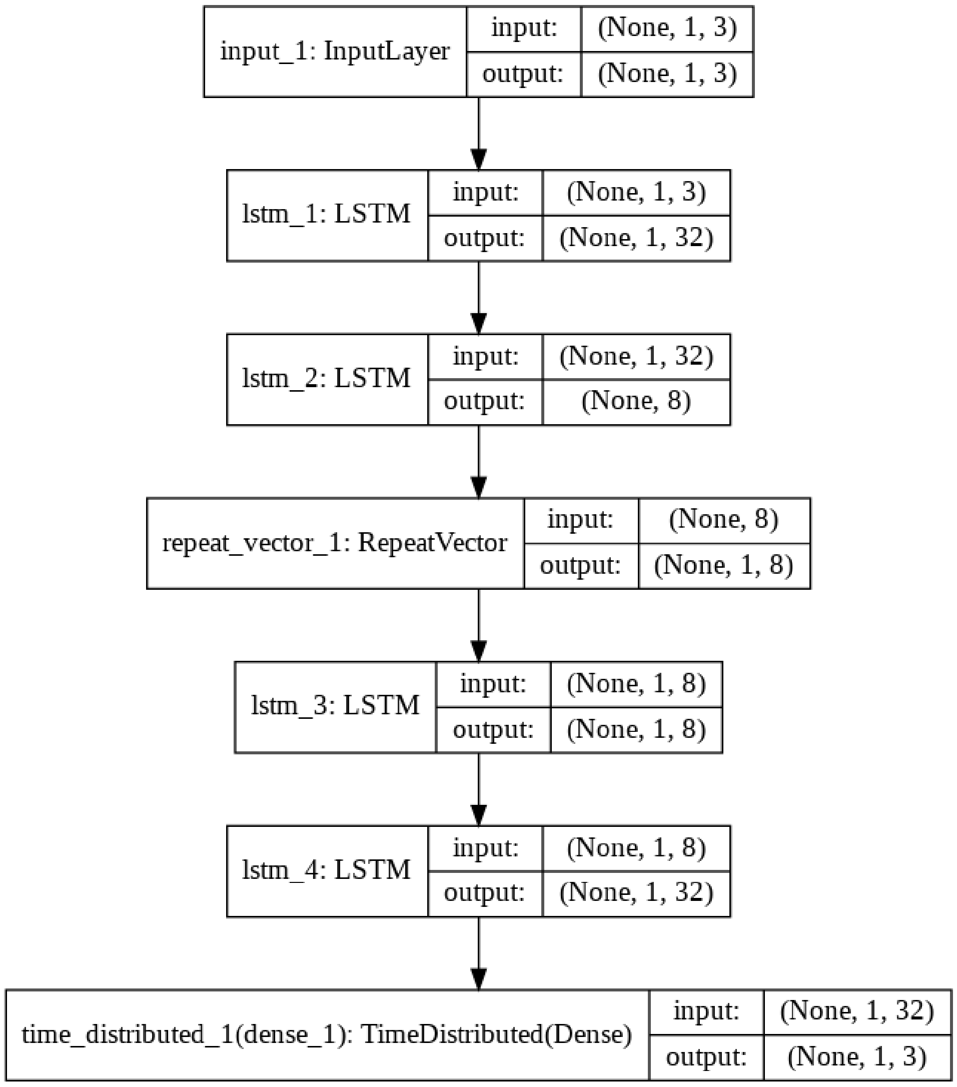


Figure 5.6: The constructed LSTM Autoencoder model

This autoencoder model was fit on training data of size of 45,706 and validated on a sample size of 3,975 samples for 50 epochs with a batch size of 10 and took 25 minutes to complete on the Google Colaboratory platform with 2vCPU at 2.2 GHz and 13 GB RAM. We use the Mean Absolute error loss for evaluation of the model. After training, we make predictions on the training data and calculate the loss distribution on the training data. This gives us a way of setting threshold values to detect anomalies in the test set. The loss distribution plot can be seen from the figure 5.7.

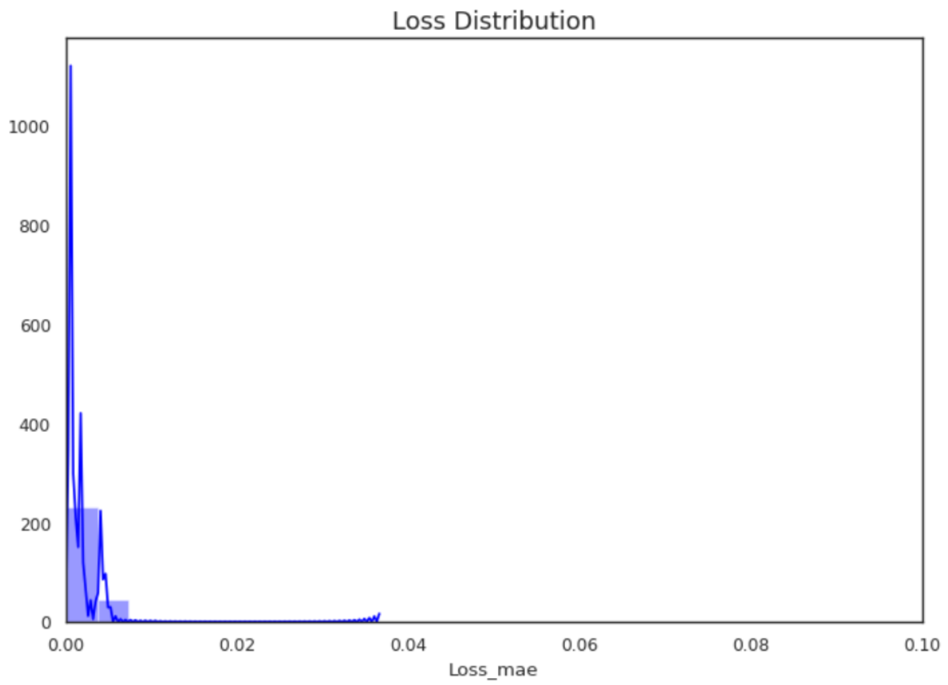


Figure 5.7: Loss distribution of LSTM Autoencoder model

Looking at the plot, we take "0.015" as an appropriate threshold value. Any loss higher than this value would be classified as an anomaly. Now we make predictions on the test data and calculate the same mean absolute error loss for each time step in the test data. Comparing these loss values with the threshold value, we classify which values are anomalies and which are not. The final plot of detected values based on these loss values looks as shown in the figure 5.8. This plot is plotted using the entire training as well as the test data. The red line on the plot is the threshold line. Loss values that exceed that line are classified as anomalies. Based on this plot, the timestamps can be identified which have been classified as having anomalous data and further investigation can be done. By digging deeper and reverse engineering our way through the timestamp values, first sampled at 10 seconds and then at each second which is the original data we can identify if the values were truly anomalous using the help of subject matter experts.

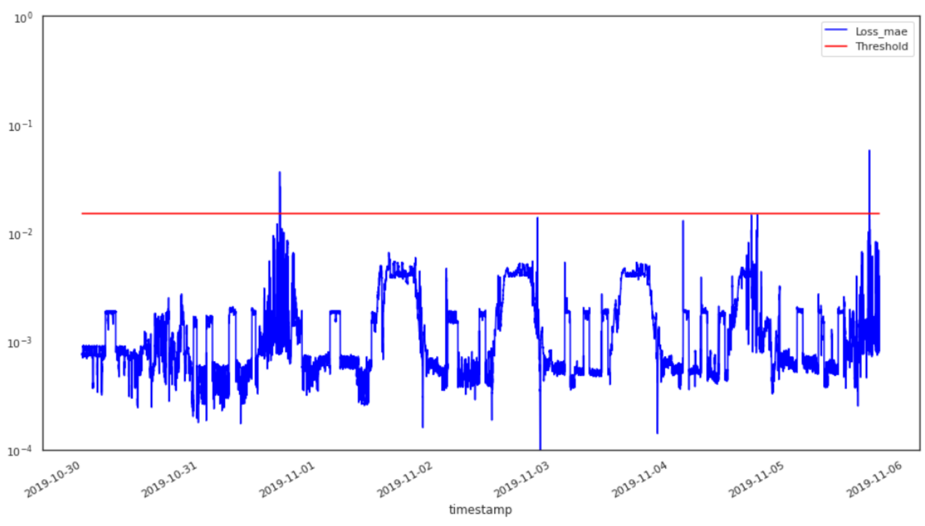


Figure 5.8: Anomaly detection using LSTM Autoencoder model

Since, the data we received is the actual operation data of a real microgrid, chances of anomalous data coming to fore is extremely rare and besides, we have chosen a time period where the microgrid was operating normally or as expected. The LSTM Autoencoder detected 5 anomalies which form 0.05% of the entire test data of 8977 data points. This is possible and quite probable because of certain other anomalies going undetected as the LSTM layers learn the temporal sequences or are good with temporal data but may have missed certain relation between the features we have selected. To overcome this problem, we look at using Convolution layers along with LSTM layers as the convolution layers are good at interpreting relation between features. For this, the model developed in the section is a Convoluted LSTM model and for that to be successful, ideally it should return more number of anomalies and at the same time it should be able to detect the same anomalies as detected by this LSTM Autoencoder model. We look at that in the next chapter.

CHAPTER 6: CONVOLUTIONAL LSTM

6.1 The algorithm

Data nowadays is found in the form of sequence of images and is quite common. The most typical example is video at social networks such as YouTube, Facebook or Instagram. Data collected over successive periods of time are characterised as Time Series. In such cases, an interesting approach is to use a model based on LSTM (Long Short-Term Memory), a Recurrent Neural Network architecture [52]. In this kind of an architecture, the model passes the previous hidden state to the next step of the sequence. Therefore holding information on previous data the network has seen before and using it to make decisions. In other words, the data order is extremely important.

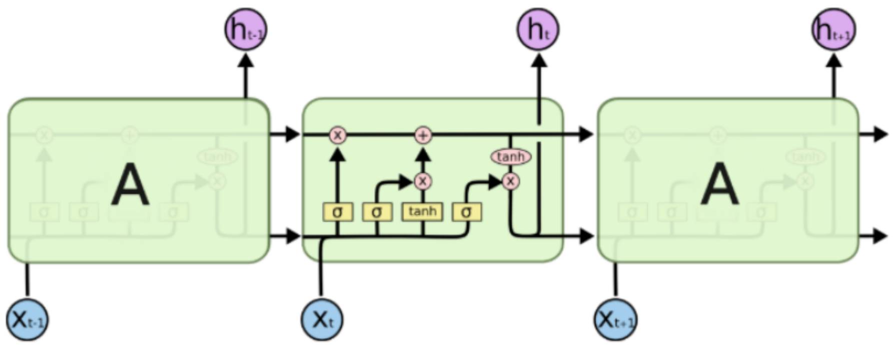


Figure 6.1: An LSTM Cell

When working with images, the best approach is a CNN (Convolutional Neural Network) architecture. The image passes through the Convolutional Layers [53] in which several filters extract important features. After passing some convolutional layers in sequence, the output is connected to a fully connected dense network.

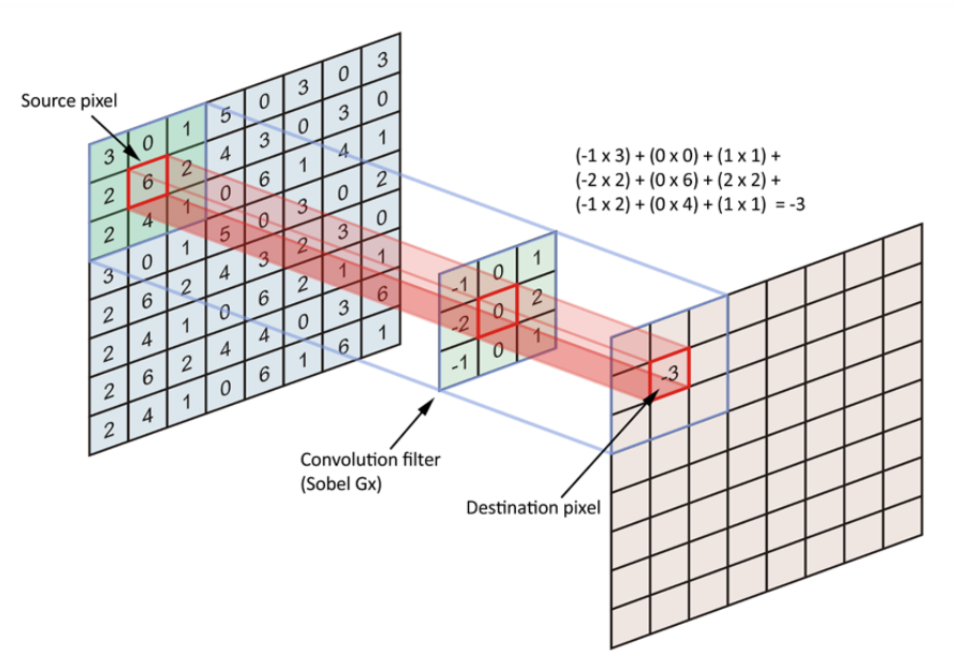

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

Figure 6.2: Convolution of image with one filter

In sequential images, one approach is using ConvLSTM layers. It is a recurrent layer, but internal matrix multiplications are exchanged with convolution operations. As a result, the data that flows through the ConvLSTM cells keeps the input dimension, instead of being just 1-dimensional vector with features. A different approach of a ConvLSTM is a Convolutional-LSTM model, in which the image passes through the convolution layers and its result is a set flattened to a 1-dimensional array with obtained features. When repeating this process to all images in the time set, the result is a set of features over time, and this is the LSTM layer input.

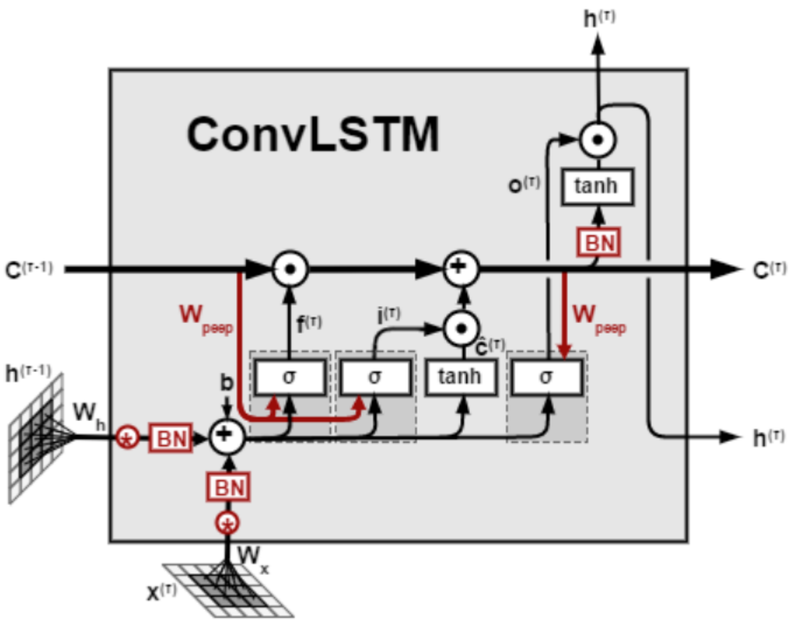


Figure 6.3: A ConvLSTM Cell

ConvLSTM is a variant of LSTM containing a convolution operation inside the LSTM cell. Both the models are a special kind of RNN, capable of learning long-term dependencies. ConvLSTM replaces matrix multiplication with convolution operation at each gate in the LSTM cell. By doing so, it captures underlying spatial features by convolution operations in multiple-dimension data. The main difference between ConvLSTM and LSTM is the number of input dimensions. As LSTM input data is one-dimensional, it is not suitable for spatial sequence data such as video, satellite, radar image and such. On the contrary, a CNN-LSTM is an integration of a CNN (Convolutional Neural Network) with LSTM. First, the CNN part of the model processes the data and one-dimensional result is fed to the LSTM model.

6.2 Anomaly detection

This section discusses the anomaly detection algorithm using the Convoluted LSTM neural network. As mentioned in the previous chapter, the raw training data shown in the figure 5.4 is the data of net power values coming from a real microgrid sampled at every one second. Because of inconsistent timestamps, we have to resample it to every 10 seconds and is discussed in detail in the second chapter of this thesis. Another problem with the data, which was scaling the values of all the variables in one common scale which is very important before using the data as input to a neural network. After scaling the raw data, the data looks something as shown in the figure 5.5. We use a MinMax scaler from the python package Scikit-learn, to scale this data into the range of 0 to 1. Now, all the three variables have the same scale as can be seen, from the figure 5.5. Now that we have the scaled data, we can perform further model training on this data. Anomaly detection tasks are simplified if the data points from different variables have the same scale.

As the name suggests this is a convolutional LSTM neural network meaning a combination of convolution and the LSTM layers in one single layer. The term convolution refers to a mathematical combination of two functions to produce a third function. It merges two sets of information. In the case of a Convolutional neural network, the convolution is performed on the input data with the use of a filter or kernel to then produce a feature map. A convolution is executed by sliding the filter over the input. At every location, a matrix multiplication is performed and sums the result onto the feature map. Numerous convolutions are performed on the input, where each operation uses a different filter. This results in different feature maps. In the end, all these feature maps are taken together and put together as the final output of the convolution layer. This systematic application of the same filter across an image is a powerful idea. If the filter is designed to detect a specific type of feature in the input, then the application of that filter systematically across the entire input image allows the filter an opportunity to discover that feature anywhere in the image. This capability is commonly referred to as translation invariance meaning, the general interest in whether the feature is present rather than where it was present. This is why convolutional layers are good at identifying features and hence, we are using that power to leverage in this anomaly detection application. The LSTM layer as a combination is to provide the strength of temporal sequence remembrance which is the primary strength of the LSTM layer. That is, it performs well on temporal data. Combining that with convolution layer within each LSTM layer so that the internal matrix operations of an LSTM layer are replaced by convolution operations to help the network identify relationships between the features is a novel idea. The model architecture leveraging this concept is as shown in the figure 6.4. We reshape inputs first into the required dimensions which is [samples, timesteps, rows, columns, channels]. As can be seen from the model architecture after, the inputs are given to the network where the first layer is the "ConvLSTM2D" using the Keras API in Python. We choose a filter size of 64 which is appropriated after a few trial and error of different permutations and combinations. The kernel size is taken to be [1,1] so that the kernel is applied to each data point in the data set individually. The activation function used here is the rectified linear unit and as mentioned in the previous chapter, we have the return sequences parameter set to "True" to keep the dimensions intact and so that each LSTM hidden state output is passed on to the next layer. We use four such layers stacked on top of each other with the same parameters. The number of layers we appropriated after a lot of trial and error and finally optimizing 4 layers based on the best results. After these 4 layers, we have two Dense layers and finally a Reshape layer to reshape the output in the input dimensions for interpretation. This ConvLSTM model was fit on training data of size of 46,700 and validated on a sample size of 2,981 samples for 40 epochs with a batch size of 10 and took 40 minutes to complete on the Google Colaboratory platform with 2vCPU at 2.2 GHz and 13 GB RAM. We use the Mean Absolute error loss for evaluation of the model. After training, we make predictions on the training data and calculate the loss distribution on the training data.

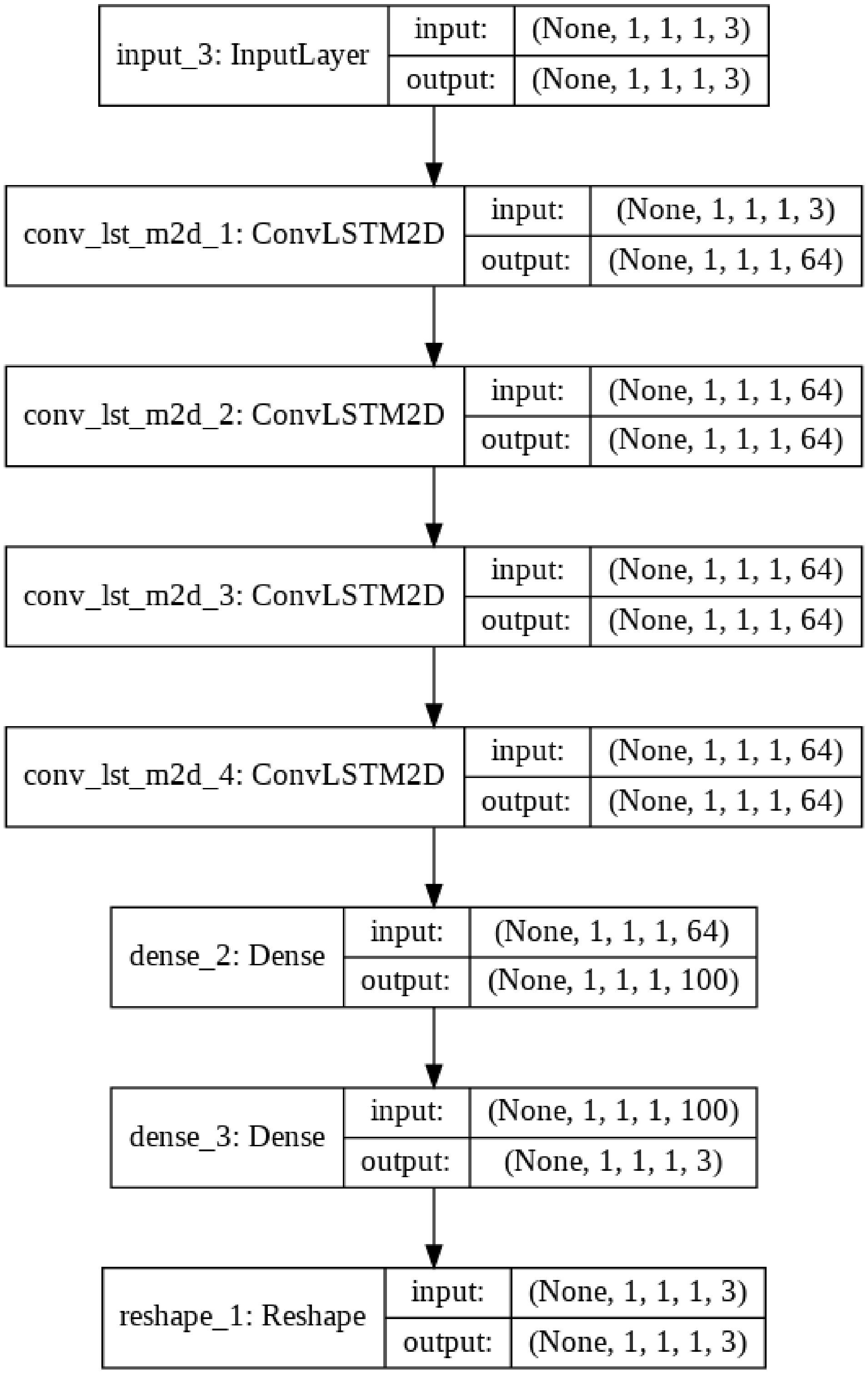


Figure 6.4: Convolutional LSTM Neural network architecture

This loss distribution plot helps us decide an appropriate threshold to detect anomalies in the test data. The model is used to make predictions on the training data and the mean absolute error loss is computed and thus the loss distribution plot is plotted. It looks as show in the figure 6.5.

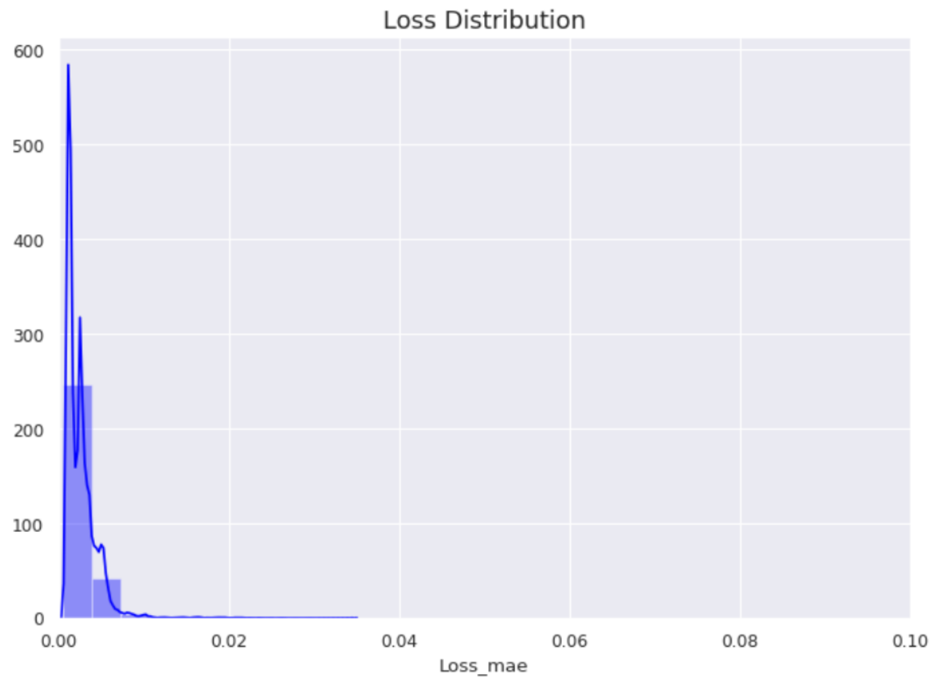


Figure 6.5: Convolutional LSTM Loss distribution plot

Looking at the plot, we appropriated a threshold value of "0.02" to a good threshold value for detecting the anomalies. Any loss higher than this value would be classified as an anomaly. Now we make predictions on the test data and calculate the same mean absolute error loss for each time step in the test data. Comparing these loss values with the threshold value, we classify which values are anomalies and which are not. The final plot of detected values based on these loss values looks as shown in the figure 6.6. This plot is plotted using the entire training as well as the test data. The red line on the plot is the threshold line. Loss values that exceed that line are classified as anomalies. Based on this plot, the timestamps can be identified which have been classified as having anomalous data and further investigation can be done. By digging deeper and reverse engineering our way through the timestamp values, first sampled at 10 seconds and then at each second which is the original data we can identify if the values were truly anomalous using the help of subject matter experts.

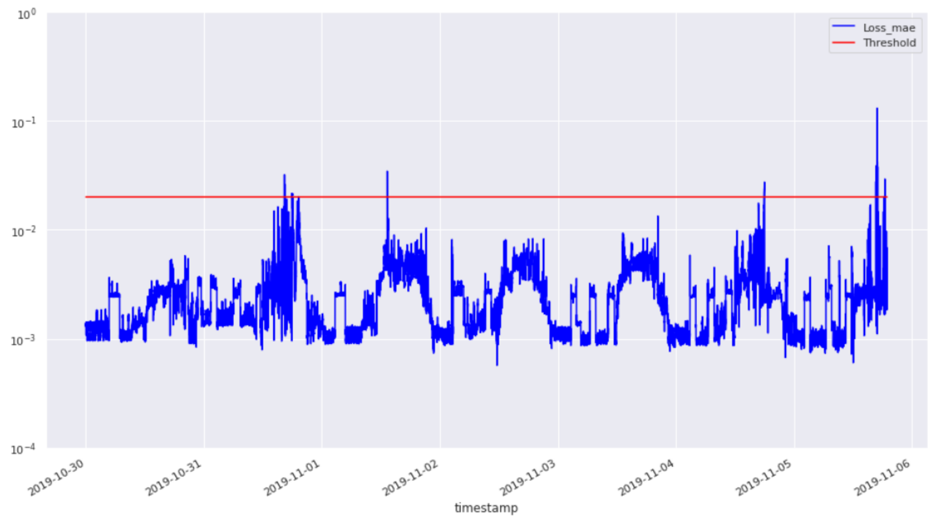


Figure 6.6: Anomaly detection using Convolutional LSTM model

Because of the fact, that we are using data from a real world functioning microgrid and have also selected data points range where the microgrid was working as expected or "normally", the chances of anomalies occurring is very rare. This Convolutional LSTM neural network was able to detect 23 anomalies from a test data set of 8,977 data points which is 0.2% of the entire test data. Due to the convolution operations happening within the LSTM layers this network has a unique quality to learn hidden interdependencies. Also, this network was able to detect the same anomalies as detected by the LSTM autoencoder network but at the same time able to find more thus proving that the LSTM layer on its own is good but convolution operations within them give it more strength as network and can become more sensitive to the anomalous data which is specially useful in unsupervised learning since there is a lack of labelled dataset. This network is not as bad with the false positives as the VAR model but at the same time sensitive enough than the LSTM autoencoder network to detect more probable anomalies. This sort of approach or network is always better in high risk conditions where the occurrence of anomalous data can be a big problem.

Then we need a network sensitive enough to detect probable anomalies but at the same time not have too many false positives which is precisely the area where this novel ConvLSTM neural network excels.

CHAPTER 7: CONCLUSIONS AND FUTURE SCOPE

Time series forecasting tasks are important in various industries, for forecasting tasks, spotting trends, analysis of the trends, predictions for the future and also for anomaly detection in time series data coming from sensors in an industrial or commercial setting. The statistical methodologies such Moving Averages (MA), Auto Regressive Moving Average (ARMA), Vector Auto Regression (VAR) and such, suffice for most of the tasks until there is good data present. Anomaly detection tasks present a problem for detecting anomalous behaviour in the available data. For this to happen, accurate labels need to be present on which data points are anomalies. This becomes a supervised learning problem. But there are a lot of scenarios where the anomaly labels are not recorded. This becomes an unsupervised learning problem and is the situation where machine learning or deep learning can come in handy. In supervised learning, where any model developed and trained on available data looks for labels of the anomalies and detects patterns which contribute to the anomaly. On the contrary in unsupervised learning because of the lack of labelled data, the models learn patterns which form a normal behaviour and any deviation from this normal pattern beyond a certain threshold is classified as an anomaly. This makes the model susceptible to potential false positives but over the course of time, the model has the capability to learn from previous mistaken classifications and can improve in accurately detecting the anomalies.

The purpose of this thesis was to investigate the novel approach of Deep Learning methodologies over statistical approaches in the analysis of time series. For this, we use the data at three different points in a real world microgrid at the frequency of every one second. Since, this is an industrial dataset and is not curated for academic purposes, we can safely say that the chances of anomalies happening are drastically low. The dates for which we are performing the analysis on the data has been the period of "normal" or expected behaviour in the functioning of the microgrid. There is also a lack of labelled data in the classification of anomalies, which makes this an unsupervised learning task. We chose to compare a statistical method of Vector Auto Regression (VAR) and two deep learning approaches. One is an LSTM Autoencoder which has been proven to perform well for anomalous detection tasks. The other is a novel Convolutional LSTM neural network developed to investigate how the addition of the convolutional operations within the LSTM layers affect its performance. As we saw from the chapters 4 to 6 where each algorithm and their anomaly detection results are mentioned in detail, the Vector Auto Regression was able to detect the most number of anomalies which are 145 in number. Aside from the usual data cleaning and preprocessing tasks, one additional task to perform for this model was to check the stationarity in the time series data. For most statistical analysis it is important we remove trends from the time series and then perform the analysis. Although this model took only 1 minute to provide the results, the number of 145 anomalies out of the total 8977 test data, leads to believe the presence of false positives. False positives are those anomalies which are not anomalies in reality but have been classified by the model as such. Then we used the LSTM Autoencoder model for this task. Essentially an autoencoder model tries to reconstruct the input after passing through an encoder and a decoder. This model was able to detect 5 anomalies from the 8977 test data set. This model took 25 minutes to complete which understandably is a lot longer than the VAR model but in terms of removing the false positives, is a much better candidate. Then we look at the novel Convolutional LSTM model for this task. This is a novel approach and is performed on an experimental basis. The addition of convolution layers within the LSTM layers should help the model in identifying the interdependencies between the features consequently being more accurate in the anomaly detection tasks. The model was able to detect 23 anomalies from the 8977 test data points and took 40 minutes to complete. This model detected the same anomalies as the LSTM Autoencoder and at the same time was able to detect 18 more. Of course, because of the lack of labelled anomaly data, we can not know sure which one is an actual anomaly. This model provides timestamps where it has detected the probable anomalies and these can be further investigated with help from subject matter experts into identifying the true anomalies.

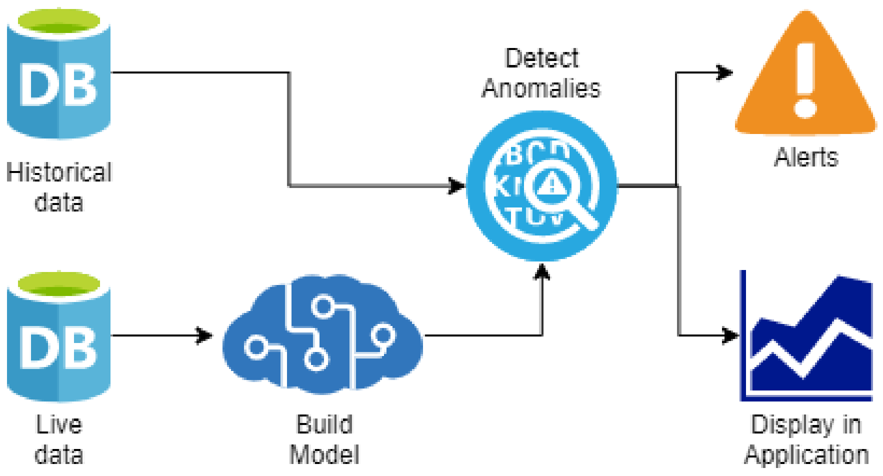


Figure 7.1: Proposed framework for real time anomaly detection

Over the course of time and making the wrong classifications, the model can eventually learn the right pattern and would no longer require labelled data for classification and can perform extremely well in areas where labelled data cannot be acquired or is too difficult. That is when, such techniques as discussed in these thesis come in handy. Since, we are looking at data from three different devices in the microgrid, this anomaly detection task also gives us insight into how the devices function with respect to each other and further investigation of why the anomalies occur can lead to interesting discoveries either in the device configuration or the microgrid. Once the models get better at classifying the anomalous data, they no longer need to be as standalone models where data is fed as chunks separately. The entire configuration can at this point be in the form of a streaming format, where the input data is processed and fed to the model and the results can be provided in real-time 7.1. This can be possible because of the availability of big data tools like Apache Spark and cloud technologies which does not require the organization to have on-premise hardware for such high computation tasks. The use of cloud servers would also yield the results quickly thus helping the near real-time approach. This would be of immense use to any organization wanting to perfect their systems and situations where the occurrence of anomalies is either a high-risk area or can cause high losses. This real-time approach and going forward predicting anomalies in the future can help organizations investigate well in advance what is wrong, trigger warning systems and thus prevent an eventual catastrophe.

REFERENCES

[1] D. T. Ton and M. A. Smith, “The u.s. department of energy’s microgrid initiative,” *The Electricity Journal*, vol. 25, no. 8, pp. 84 – 94, 2012.

[2] N. Hatziargyriou, A. Anastasiadis, J. Vasiljevska, and A. Tsikalakis, “Quantification of economic, environmental and operational benefits of microgrids,” in *2009 IEEE Bucharest PowerTech*, pp. 1–8, IEEE, 2009.

[3] S. Bossart, “Renewable and distributed systems integration demonstration projects,” in *EPRI Smart Grid Demonstration Advisory Meeting. http://www. smartgrid. epri. com/doc/15% 20DOE% 20RDSI% 20Project% 20Update. pdf*, 2009.

[4] R. Act, “The american recovery and reinvestment act of 2009,” *Public Law*, vol. 111, no. 5, pp. 5–30, 2009.

[5] F. P. Ultra-Wideband, “Environmental security technology certification program,” *Environmental security*, 2005.

[6] P. Basak, S. Chowdhury, S. H. nee Dey, and S. Chowdhury, “A literature review on integration of distributed energy resources in the perspective of control, protection and stability of microgrid,” *Renewable and Sustainable Energy Reviews*, vol. 16, no. 8, pp. 5545–5556, 2012.

[7] A. Hirsch, Y. Parag, and J. Guerrero, “Microgrids: A review of technologies, key drivers, and outstanding issues,” *Renewable and Sustainable Energy Reviews*, vol. 90, pp. 402–411, 2018.

[8] N. Hatziargyriou, H. Asano, R. Iravani, and C. Marnay, “Microgrids,” *IEEE power and energy magazine*, vol. 5, no. 4, pp. 78–94, 2007.

[9] H. Zeineldin, E. El-Saadany, and M. Salama, “Distributed generation microgrid operation: Control and protection,” in *2006 Power Systems Conference: Advanced Metering, Protection, Control, Communication, and Distributed Resources*, pp. 105–111, IEEE, 2006.

[10] M. Shahidehpour and J. F. Clair, “A functional microgrid for enhancing reliability, sustainability, and energy efficiency,” *The Electricity Journal*, vol. 25, no. 8, pp. 21–28, 2012.

[11] K. Ravindra and P. P. Iyer, “Decentralized demand–supply matching using community microgrids and consumer demand response: A scenario analysis,” *Energy*, vol. 76, pp. 32–41, 2014.

[12] M. E. Khodayar, “Rural electrification and expansion planning of off-grid microgrids,” *The Electricity Journal*, vol. 30, no. 4, pp. 68–74, 2017.

[13] T. Abdallah, R. Ducey, C. A. Feickert, R. S. Balog, W. Weaver, A. Akhil, and D. Menicucci, "Control dynamics of adaptive and scalable power and energy systems for military micro grids," tech. rep., CONSTRUCTION ENGINEERING RESEARCH LAB (ARMY) CHAMPAIGN IL, 2006.

[14] R. M. Kamel, A. Chaouachi, and K. Nagasaka, "Carbon emissions reduction and power losses saving besides voltage profiles improvement using micro grids," *Low Carbon Economy*, vol. 1, no. 1, p. 1, 2010.

[15] W. Hurst, M. Merabti, and P. Fergus, "Big data analysis techniques for cyber-threat detection in critical infrastructures," in *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pp. 916–921, 2014.

[16] J. Hu and A. V. Vasilakos, "Energy big data analytics and security: Challenges and opportunities," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2423–2436, 2016.

[17] S. Otoum, B. Kantarci, and H. Mouftah, "Adaptively supervised and intrusion-aware data aggregation for wireless sensor clusters in critical infrastructures," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2018.

[18] B. V. Solanki, A. Raghurajan, K. Bhattacharya, and C. A. Cañizares, "Including smart loads for optimal demand response in integrated energy management systems for isolated microgrids," *IEEE Transactions on Smart Grid*, vol. 8, no. 4, pp. 1739–1748, 2015.

[19] G. Henri and N. Lu, "A supervised machine learning approach to control energy storage devices," *IEEE Transactions on Smart Grid*, vol. 10, no. 6, pp. 5910–5919, 2019.

[20] Xin Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[21] K. P. Bennett and A. Demiriz, "Semi-supervised support vector machines," in *Advances in Neural Information processing systems*, pp. 368–374, 1999.

[22] R. E. Wright, "Logistic regression.," *Machine learning*, 1995.

[23] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[24] T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection," *Information Sciences*, vol. 177, no. 18, pp. 3799–3821, 2007.

[25] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection," in *Applications of data mining in computer security*, pp. 77–101, Springer, 2002.

[26] C. S. Hilas and P. A. Mastorocostas, "An application of supervised and unsupervised learning approaches to telecommunications fraud detection," *Knowledge-Based Systems*, vol. 21, no. 7, pp. 721–726, 2008.

[27] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1285–1298, 2017.

[28] J. Hare, X. Shi, S. Gupta, and A. Bazzi, "A review of faults and fault diagnosis in micro-grids electrical energy infrastructure," in *2014 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 3325–3332, IEEE, 2014.

[29] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[30] L. A. Zadeh, "Fuzzy logic," *Computer*, vol. 21, no. 4, pp. 83–93, 1988.

[31] K. Heumann, *Basic principles of power electronics*. Springer Science & Business Media, 2012.

[32] W. McKinney *et al.*, "pandas: a foundational python library for data analysis and statistics," *Python for High Performance and Scientific Computing*, vol. 14, no. 9, 2011.

[33] G. Brys, M. Hubert, and A. Struyf, "A robust measure of skewness," *Journal of Computational and Graphical Statistics*, vol. 13, no. 4, pp. 996–1017, 2004.

[34] J. Quackenbush, "Microarray data normalization and transformation," *Nature genetics*, vol. 32, no. 4, pp. 496–501, 2002.

[35] N. R. Goodman, "Statistical analysis based on a certain multivariate complex gaussian distribution (an introduction)," *The Annals of mathematical statistics*, vol. 34, no. 1, pp. 152–177, 1963.

[36] B. Abraham and A. Chuang, "Outlier detection and time series modeling," *Technometrics*, vol. 31, no. 2, pp. 241–248, 1989.

[37] D. M. Hawkins, *Identification of outliers*, vol. 11. Springer, 1980.

[38] Y. Kou, C.-T. Lu, S. Sirwongwattana, and Y.-P. Huang, "Survey of fraud detection techniques," in *IEEE International Conference on Networking, Sensing and Control, 2004*, vol. 2, pp. 749–754, IEEE, 2004.

[39] W. Härdle, A. Tsybakov, and L. Yang, "Nonparametric vector autoregression," *Journal of Statistical Planning and Inference*, vol. 68, no. 2, pp. 221–245, 1998.

[40] D. E. Spencer, "Developing a bayesian vector autoregression forecasting model," *International Journal of Forecasting*, vol. 9, no. 3, pp. 407–421, 1993.

[41] K. Holden, "Vector auto regression modeling and forecasting," *Journal of Forecasting*, vol. 14, no. 3, pp. 159–166, 1995.

[42] Y.-W. Cheung and K. S. Lai, "Lag order and critical values of the augmented dickey–fuller test," *Journal of Business & Economic Statistics*, vol. 13, no. 3, pp. 277–280, 1995.

[43] R. I. Harris, "Testing for unit roots using the augmented dickey-fuller test: Some issues relating to the size, power and the lag structure of the test," *Economics letters*, vol. 38, no. 4, pp. 381–386, 1992.

[44] C. Agiakloglou and P. Newbold, "Empirical evidence on dickey-fuller-type tests," *Journal of Time Series Analysis*, vol. 13, no. 6, pp. 471–483, 1992.

[45] G. Shukur and P. Mantalos, "A simple investigation of the granger-causality test in integrated-cointegrated var systems," *Journal of Applied Statistics*, vol. 27, no. 8, pp. 1021–1031, 2000.

[46] C. Diks and V. Panchenko, "A new statistic and practical guidelines for nonparametric granger causality testing," *Journal of Economic Dynamics and Control*, vol. 30, no. 9-10, pp. 1647–1669, 2006.

[47] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[48] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[49] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[50] S. Stanford, "Deepmind's ai alphastar showcases significant progress towards agi," *ML Memoirs*, 2019.

[51] A. Gensler, J. Henze, B. Sick, and N. Raabe, "Deep learning for solar power forecasting, an approach using autoencoder and lstm neural networks," in *2016 IEEE international conference on systems, man, and cybernetics (SMC)*, pp. 002858–002865, IEEE, 2016.

[52] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 240–254, 1994.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.