Tools and Methodologies for Power Management in Smart Grids

by

Rituka Jaiswal

A dissertation submitted in partial satisfaction of
the requirements for the degree
PHILOSOPHIAE DOCTOR (PhD)

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science
Jan 2022

University of Stavanger
N-4036 Stavanger
NORWAY
www.uis.no

ISBN xxx-xx-xxx-xx
ISSN aaa-zz-bbb-cc

PhD Thesis UiS no. YYY

Preface

This dissertation is submitted in partial fulfillment of the requirement of the degree of Philosophiae Doctor (PhD) at the University of Stavanger, Norway. The research has been conducted at the University of Stavanger, Norway from July 2018 to Dec 2021, including research visits to University of Lille, France and Technical University of Berlin, Germany. This dissertation consists of a collection of 8 Chapters, which are included within the dissertation after required transformations and realignments to adhere to the requisite format. The content of the originally published articles has been kept intact.

Rituka Jaiswal, Jan 2022

Abstract

The power sector is incorporating Smart Grids and Micro Grids. Smart grids uses ICT technologies for two way communication between utility companies and customers and extensively uses sensors along the power pipeline to automate processes and, better power management. With Smart Grids the goal is to manage power demands, minimise power wastage, detect faults in the transmission and distribution lines, and integrate renewable energy resources for clean and safe electricity and, other services. However, the data driven decentralized systems are disrupting the simplex demand supply relationship between producers and consumers. Also, the current electricity Grid infrastructure lacks mechanisms to handle power demands primarily led by Electric Vehicles. Managing the high power demands during peak hours can be done through the integration of renewable energy resources at the household or neighborhoods levels. Methodologies are needs to integrate energy resources and forecast the consumption of power. At the same time, the data produced from the sensors could be exponentially large and traditional Cloud systems will not be able to manage the data processing. Cloud systems also pose security issues as consumers power data is sensitive.

We define managing power consumption and balancing power needs as Smart Community Neighborhood problem. ICT solution using Graph Algorithms and Artificial Intelligence can be used to substantially solve such challenges. With the integration of renewable energy resources(Wind turbines, Solar PV panels) and storage technologies, consumers will be able to produce clean energy to meet their own requirements and sell excess electricity to the Grid. A neighborhood level information intensive energy management solution is proposed. Firstly, for optimal integration of renewable energy resources(wind turbines in our case), we propose a novel Graph Theory based algorithm for the optimal design of wind farm collector system. Thereafter, a Fog Computing based distributed architecture is proposed for real-time processing of Smart Grid data. We also propose approaches for Demand Response management and peak detection using Artificial Intelligence based methodologies. Power forecasting is very crucial for power balance. Therefore, we propose Deep Learning based methodologies for forecasting power consumption. It would enable power companies to make effective decisions about power production and better power management.

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Reggie Davidrajuh, for his outstanding guidance in research throughout my Ph.D. I would like to thank Prof. Chunming Rong for being my co-supervisor and providing excellent support and an exciting domain to work. I would like to thank PACE LUCS for organizing summer school at TU Berlin, Germany and University Of Lille, France on the topics of sustainability and the future of Energy and Smart Grids. I would like to thank Associate Prof. Antorweep Chakravorty for guiding for my initial work and necessary domain knowledge required.

I would also like to thank my PhD colleagues and friends Dr. Cristina Heghedus, Dr. Faraz Barzideh, Dr. Aida Mehdipour, Dr. Nikita Karandikar, Dhanya Jose, Dr. Rahul Mishra, Dr. Jayachander Surbiryala, Dr. Albana Roci, Associate Prof. Mina Farmanbar, Per Jotun for their support in my PhD. I would specially like to thank my mother Pratibha Devi for her inspiration and the dream of educating me despite severe challenges, my sister Dipika for her constant encouragement and my brother Rakesh for his support. Finally, I would like to thank my Head Of Department Dr. Tom Ryen for his constant support and exceptional encouragement. My PhD could not have been possible without his constant encouragement.

Contents

Dedicated to my mother and all the women who lived with the spirit of creating a gender neutral world.

Chapter 1

Introduction

1.1 Smart Grids

A smart grid uses digital technology that allows for two-way communication between the utility provider and its smart home consumers [1]. It is the IoT sensing networks across the generation, transmission, distribution lines and eventually to smart homes within smart neighborhoods that effectively makes the grid "SMART". This two-way communication is represented in Figure 1.1. In Figure 1.1, dashed lines represent the flow of electricity while the solid lines represent the communication channel. Power generators generate electricity which is sent to the transmission lines, from transmission lines it goes to the distribution lines and, finally to the consumers in neighborhoods. The sensor data generated along this electricity channel has the properties of volume, variety, veracity and velocity and, therefore, demands real-time processing for quick control and monitoring of smart grid operations and outage prevention [3].

ICT technologies are integrated into smart grids for effective power management and, reliable and secure supply of power at all times. Smart Grids also ensures clean, sustainable energy by integrating with renewable sources of energy like solar PV panels and wind turbines. This provides consumers with ways to proactively manage the energy which is convenient, cost-effective and environment friendly [**smartgridrenewable**].

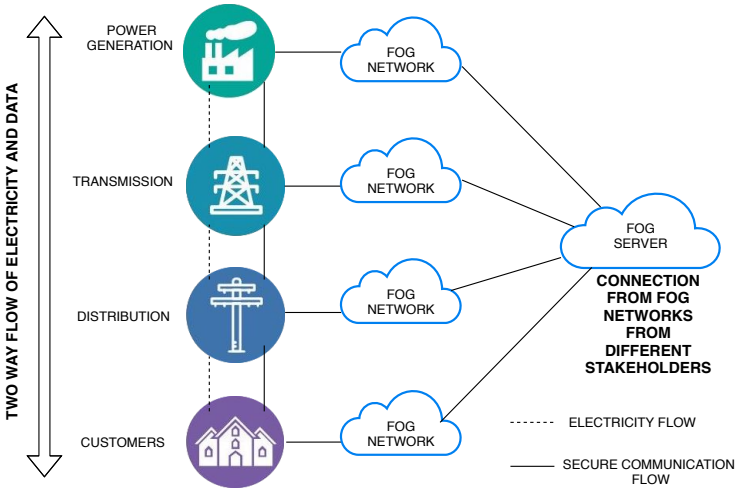

Figure 1.1: Smart Grid flow.

1.2 Smart Grid Concepts

1.2.1 Advanced Metering Infrastructure

It is important to talk about smart meters and the functions they perform to understand smart grid operations. Apart from measuring the power consumption by individual smart appliances, smart meter communicates with other intelligent devices in the home and the grid utility provider. It notifies the consumer of a power outage, consumption behavior and utility provider for system monitoring and customer billing. It monitors the power quality and performs automatic remote turn on and turn off operations of appliances [**ami**]. Smart meters are not working separately but are organized as networks. So, every household smart meter can communicate with each other.

Advanced metering infrastructure (AMI) is an integrated system of smart meters, communication networks and, energy & data management system that aids in two way communication between energy providers and energy consumers [**ami**]. Smart meters in every smart home are connected to form a network which gains perspective about operations mentioned above by processing data on fog nodes. As shown in Figure 1.2, Home area network has smart home appliances sending data to smart their meters. The network of smart meters in a neighborhood is connected to a gateway and send the collective data to the Fog server for that region for processing. The fog server runs the meter data management system.

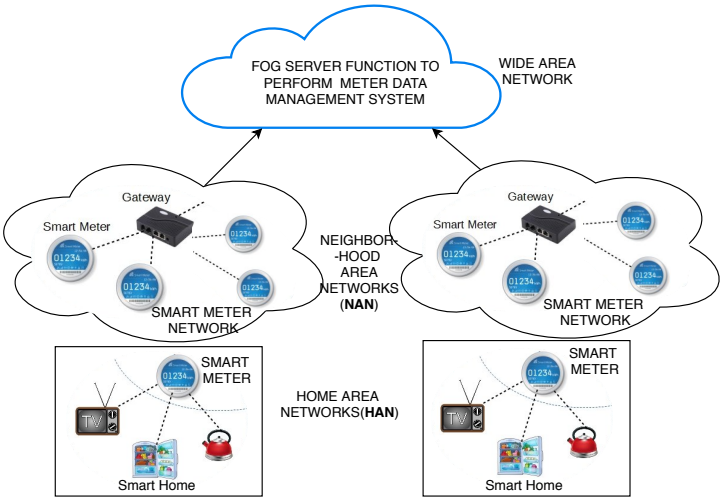

Figure 1.2: Advanced metering infrastructure(AMI).

1.2.2 Vehicle to Grid (V2G) Integration

The transportation sector is undergoing massive changes as more and more vehicles are becoming electricity powered. EVs are one of the feasible solutions to the challenges such as global climate change, energy security and geopolitical concerns on the availability of fossil fuels. The smart grid has essentially no storage (other than its 2.2% capacity in pumped storage [**V2G_intro**]), so generation and transmission must be continuously managed to match fluctuating customer load and effective demand response management. The EVs integration in smart grid serves as an independent distributed energy storage source which can potentially address above challenges. The energy stored in the EV batteries can potentially provide power to grid in needs. This is called Vehicle to Grid technology (V2G). Whenever there is a power request from the utility provider, EVs should provide power within minutes. This near real-time response can be possible with a local processing technology available near the EVs. V2G has to fulfill the needs of the consumer/driver and the grid operator. The driver needs enough stored energy on-board for the driving needs. The grid operator needs power usage to be turned on and off at precise times. Intelligence controls should be available near the EVs for satisfying consumers and the grid operator [**V2G_challenges**]. The typical services that the V2G operation can offer are summarized as follows:

- Store power;

- Load shifting by peak shaving and valley filling;
- Renewable energy resource integration;
- Frequency and voltage regulation;

Integration of a large number of EVs into the smart grid is a also a major challenge which requires an intensive assessment and observation for operation and control of smart grid operations.

1.2.3 Demand Response Management

Demand response management (DRM) is the key component to increase the efficient use of electricity by remote monitoring and control of electricity load by setting efficient electricity prices with the objective to shift the high demand of electricity users to an off-peak period. It effectively reduces power generation costs and consumer bills [**DRM**]. With this, DRM improves energy efficiency and reduces overall electricity wastage for both consumers and producers. Let's understand the need and importance of DRM more with the help of an example. Most of the households in a neighborhood have almost the same electricity consumption patterns for 24 h. During evening times, around 5–6 p.m., users come home from their workplaces and turn on all their appliances like TVs, lights, electric vehicle charging, and so forth. Therefore, power consumption during the evenings is much higher compared to afternoons or night when the consumer is not using most of the appliances. But, power generation companies generate power as per the peak hours of power consumption by the majority of the consumers in the neighborhood. This leads to power wastage and loss for the power service providers because other than a short duration of peak hours, the power generated can not be utilized for non-peak hours. DRM effectively schedules smart home appliances in such a way that distributes the peak hour power consumption to non-peak hours so that power consumption throughout 24 h does not have high spikes and power consumption curve is flattened. This also benefits the consumers because they get to pay less for the same electricity usage.

The real-time monitoring of electricity consumption and scheduling strategies of appliances will implement DRM and improve grid reliability and utilization. Fog Computing will be a key technology enabler to implement DRM.

1.2.4 Renewable Energy Source Integration

Renewable energy sources such as PV panels and Wind turbines are sources of energy integrated in local neighborhoods that can provide a variety of bene-

fits including improved reliability, security, efficient energy system if they are properly operated in the electrical distribution system [**RES_microgrids**]. They cause minimal local environmental damage and net emissions of greenhouse gases (GHGs). RES integration makes the Grid operate as MicroGrid which can function independently from the main grid. Increasing number of consumers are installing PV panels and Wind turbines locally to control their power needs at reduced cost and, also trade power to other consumers. Various studies have found that a large number of utilities as well as consumers that have installed RES at their facilities to realize benefits mentioned above [**RES_microgrids**]. Also, in order for the reliability of MicroGrid and challenge of intermittent power generated by RES, EVs are used. The typical services that the RES integration in MicroGrid plays are as follows:

- Reliable MicroGrid operation.
- Secure and privacy preserving MicroGrid operation.
- Energy supply to store in EVs.
- Energy cost reduction.
- Frequency and voltage regulation.

1.3 Smart Community Neighborhoods

Smart Grid integrates renewable sources of energy like PV panels and wind turbines to provide a sustainable green source of energy [**renewable_energy**]. The electricity generated by PV panels and wind turbines makes the grid to function independently for household power usage. An independent grid makes the smart grid reliable and is commonly called MicroGrid. We focus on MicroGrid at the community level, which is called community MicroGrid. A community MicroGrid is a self-sustained local energy grid where it can disconnect from the traditional smart grid and operate autonomously [**microgrid**]. It provides power back up to the grid in case of emergencies when there is a fault at the power lines or, when the distribution lines are cut off from the households or, due to security issues. To summarize, Community MicroGrid supports the power transmission system by supplying power and, to the customers, it enhances local reliability, reduce feeder losses, support local voltages, perform voltage sag correction and provides uninterruptible power supply [**renewable_microgrid**].

Figure 3.1 shows the renewable energy sources, solar panels and wind turbines and, EVs integration in smart grid at community level. They make it possible to

realize MicroGrids.

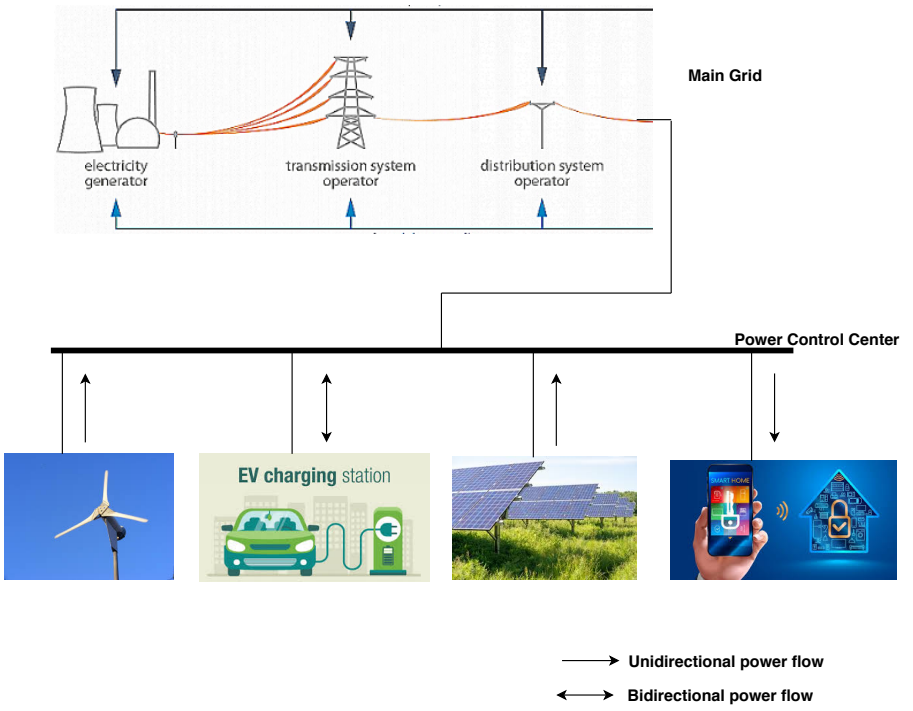

Figure 1.3: V2G and RES integration in smart grids.

We define smart Neighborhood as a data driven decentralized community Micro-Grid infrastructure to achieve community level Grid reliability, demand response management, secure and privacy preserving operations. It integrates local renewable energy resources and electric Vehicles for local production and storage of electricity and use automation, control and optimization, analytic tools for operation.

The objective of this dissertation is to demonstrate how concepts and techniques from graph theory, machine learning can be applied to solve some of the Smart Grid challenges as discussed below. Automatic and optimal design of wind farm collector systems for optimal operation of wind farms has been addressed in Chapter 2. This work is applicable in the planning and installation stages of wind farms and particularly relevant in the current scenario with increasing focus on integration of renewable sources of Energy in Smart Grids. Graph Theoretic algorithms have been useful in proposing the optimal design.

Chapter 2

Challenges in Smart Grids based Smart Neighborhoods

2.1 Challenges in Smart Grids based neighborhoods

In this section, we describe the key challenges for realizing Smart Grid based neighborhoods. We proposed tools and methodologies to address these challenges.

2.1.1 Optimizing the design of wind turbines for RES integration

Derived from Sudiptas work, need editing and work changes

An challenge in Smart Grid is integration of renewable sources of energy such as wind turbines. An important Smart Grid functional characteristic is the ability to accommodate all renewable energy generation sources including wind turbines and PV panels. Also storage options in form of Electric Vehicle batteries. In the United States, installed capacity of wind turbines has been increasing over the years. In Norway also the wind turbines have been installed at a growing pace. The Department of Energy (DOE) has put forward goals of achieving 20% wind power penetration by 2030, a target that would increase the wind power installed in the United States from 47 GW in 2011 to 300 GW in 2030 [2, 3]. This huge growth target is posing challenges of large scale wind turbine integration from planning, design to operations as part of Smart Grid implementation.

Wind farm with wind turbines distributed over a geographical region and a substation which consolidates power generated by the turbines and transmits to

the grid. Suppose that the wind turbines and the substation are already placed based on wind patterns, proximity to the transmission grid etc., and the question is: How should the electrical cables which connect the wind turbines to the substation be laid out in an optimal way? Thus the problem is that of optimal design of the wind farm collector system and is an important part of wind farm design since optimal operation of wind farms depends on it. The difficulty of a collector system design project is that given the wind turbine locations and the substation location, depending on the dimension of the wind farm, there may be thousands of feasible layout configurations to choose from. Selecting an optimal design from these choices can be a challenging task. In addition there are several design constraints to be taken into account. Hence an optimal design method is a critical need. In addition, it is necessary to automate the design process.

2.1.2 Real-time control and monitoring of operations and data

The Smart Grid has to improve regarding efficiency, energy management, reliability by considering its real-time implementation. We need real-time communication between consumers and utility providers to fulfill the demands and supply needs. The power demands are fluctuating rapidly. The consumers have the provision for choosing Real Time Pricing (RTP) schemes. The power data should be analysed in real-time to meet the consumers needs.

According to the US Energy Information Administration, the residential customers utilized around 30% of the total energy in year 2020 [**30_percentenergy**]. These figures will increase in future and lead to rapid growth of smart meters in residential households. According to another data estimation [**bigdatasmartgrids**], 1 million smart meters installed in the smart grid community will generate around 2920 Tb of data with a 15 minute sampling rate. In North America alone, the number of smart meters installed has increased from 6% of households in 2008 to 89% of households in 2012. As the number of smart meters increases from hundreds to thousands to millions, the current Cloud infrastructure will not be sustainable with such big data explosion [**bigdata_issue**]. It will be a huge challenge to manage this data and generate real-time insights from it. At the same time, Google Cloud deployed 900,000 servers and consumed 1.9KWh of electricity according to [**Koomey2011growth**]. To save the energy consumed by the data center, prevent backbone network congestion, maintain security & privacy of consumers data, it is important to process the data close to where the data is generated and where the smart grid generates electricity.

According to an estimate by Cisco, there will be 75 billion connected devices by 2025 [**Internet_revolution**, **Future_iot**]. This exponential growth in the number

of devices per person is due to the proliferation of mobile devices (e.g., wireless sensors, mobile phones tablets, etc.) as more and more consumer services are provided through wireless sensor networks. These numbers will soon be superseded by sensing/acting devices placed virtually everywhere (the so-called Internet of Things and pervasive sensor networks). And therefore, it is estimated that around 90 percent of the data generated by the sensor networks will be stored and processed locally rather than in the cloud [**ref4**]. Smart Grid domain sensors are no exception. According to an estimation, 1 million smart meters installed in the Smart Grid community would generate around 2920 Tb data in quantity with a sampling rate of 4 per hour [**big_data_analytics_smart_grid**]. In North America, the number of smart meters has grown from 6 % of households in 2008 to 89 % in 2012. These numbers continue to grow as more communities and businesses are installing smart meters. Austin Energy in Texas has implemented 50,000 smart meters with a sampling rate of 5 min to produce around 800 TB of data.

In the smart grid system, large amounts of sensors are deployed across a wide area of generation lines, transmission lines, distribution lines, and consumer buildings to implement complex monitoring and control functions. The massive volume of real-time data collected by smart meters will help the grid operators gain a better understanding of the large-scale and highly dynamic power system [**dynamic_power_system**]. As the number of smart meters increases from thousands to millions, the current state-of-the-art centralized data processing architecture will no longer be sustainable with such a big data explosion. In addition, sending the huge smart meter sensor data to the centralized cloud servers will require prohibitively high network bandwidth for the transmission and, high round trip time. If we have a distributed architecture, the communication cost between the service providers and AMI infrastructures will be minimized since they are physically located close to each other. The communication bandwidth needed is mainly for the information exchange between each distributed node and the central server, which will not vary much because it will not be affected by the increasing number of smart meters or the sampling frequency of smart meter data. Data security and privacy are also critical issues when the sensitive smart meter data is aggregated in the centralized cloud server [**AMI_fog**] because the high-frequency metering data which is required for efficient smart grid operations reveals consumers behavioral patterns.

The Electric Vehicle (EV) and Vehicle to Grid (V2G) technology are an important component of smart grid [2]. They provide the grid, support by delivering services such as Demand Response Management (DRM), spinning reserve, voltage and

frequency regulations whenever needed. To the service provider, the EVs are a potential independent distributed power backup source. But on the challenging side, EVs are dynamic storage sources which are difficult to schedule due to their unpredictable usage by the consumers and, their numbers are increasing continuously. EVs store the energy produced by local renewable energy sources (RES). According to an estimation, China has set a goal to install 150–180 GW of wind power and 20 GW of PV solar power by 2020. The integration of large renewable energy sources like wind and photovoltaic (PV) solar energies into the power system will continue to grow for local and global energy requirements. Importantly, the integration of RES in Smart grid makes it possible to meet MicroGrid operations and energy security issues [2]. On the other hand, these RES are intermittent in nature due to their dependency on weather, and therefore, the forecast is quite unpredictable. Therefore, for effective V2G operations, reliable MicroGrid operations and real-time RES energy forecasting, we need a computing paradigm near these sources.

2.1.3 Predictive analysis of power consumption for effective Demand Response Management

Very short-term forecasting (from a few seconds to minutes): Very short-term forecasts can be used for PV and storage control and electricity market clearing, such as 5 minutes. In the smart grid environment, very short-term forecasting of solar power becomes more important than before. Short-term (up to 48–72 hours ahead): Such forecasts are crucial for different decision-making problems involved in the electricity market and power system operation, including economic dispatch, unit commitment, etc. Medium-term (up to one week ahead): Medium-term forecasting would be useful for e.g., maintenance scheduling of power plants, transformers, and transmission lines. Long-term (up to months to years): Long-term prediction/estimation can be applied for long-term assessment of power planning and long large scale decision making.

Prediction is the most effective method to ensure minimum power outages, and demand response management and effective power management. Predictions using machine learning, deep learning and swarm intelligence techniques are used in various domains of study in many fields [**health_prediction**, **manu_pred**, **vehicles_prediction**, **smartgrid_prediction**, **smartcity_pred**]. In our case of community Grids, consumers need a reliable power supply, and any anomaly in the power consumption should be quickly detected. Here predictions techniques are the most effective. Since minute-sampled power consumption data at the appliance level is available through smart meters, predictions of power

consumption for the next couple of days in every smart home can be made. In this manner, the power supplier/service provider will be able to know how much power should be to be generated and help them in making decisions. At the same time, forecasting the generated power by the power producers will help utility providers become aware of the power production in coming days and therefore, inform consumers about the costs based on the production. This will do the scheduling of appliances with customer preferences and cost minimization. With this, demand-side management with minimum energy loss will be effectively achieved through predictions.

For reliability issues or detecting faults like sensor faults, malfunctioned devices or security threats also, forecasting is the most effective tool. We can analyse and forecast future power consumption values from the previous month's data. Large deviations from the predicted consumption could be a potential fault, and customers and service provider will be alerted the customer about the possible faults (after eliminating false alarms). They will be able to locate faults at the appliance level because granular data analysis and real-time fault detection is done.

Potentially high number of use cases can be handled with forecasting techniques as sensor data, and computing resources are available near the sensors. Another use case is predicting power produced by wind turbines and PV panels. With weather data of past weeks and months, we can predict how much power will be produced in the coming weeks, months. This will allow consumers to make informed decisions like, planning how much power they need to buy and decision about selling power to other users at maximum profit. Importantly, renewable sources of energy effectively reduce the carbon footprint by the neighborhood and contribute to the environment.

2.1.4 Reliable power supply and anomaly detection in power consumption

Power outages are common as, the demand side data is not analysed and forecasted for anomalies. There have been numerous incidents of power outages across the world in the past [**blackouts_history**]. During the winter storm in Texas in February 2021, nearly 5 million people lost power [**texas_crisis**]. Also, in June 2021, due to the high power demand and insufficient supply, California grid operators suggested consumers to charge their electric vehicles during off-peak hours. In New York, for the first time power outages hit several neighborhoods due to high temperature rise and officials sent an emergency mobile alert to consumers urging them to save electricity [**ny_poweroutage**]. To avoid

these power outages, it becomes highly important to detect the power anomalies. Preventing and detecting power anomalies has become more necessary in today's global power system. Each year, the economy losses hundreds of millions of dollars caused by power wastage and outages [**power_wastage1**, **power_theft**]. With the growth and popularity of Internet of things, demand and supply complexity, monitoring and forecasting of power is critical for power companies in terms of power generation, scheduling and dispatching. It benefits power consumers by allowing them to enhance their power usage schedules and thus reduce their costs. Additionally, power suppliers can detect abnormal meter readings caused by unforeseen meter failures, intentional meter manipulations, or users' unusual consumption behaviors [**unusual_data**, **unusual_data1**] and thus, reduce their costs. Discovering unusual meter measurements, appliance faults and, unexpected consumer behavior is known as Anomaly detection or outlier detection. Anomaly detection has been widely applied in a variety of applications, such as fraud detection and fault detection in safety-critical systems, medical diagnosis, etc. Anomaly detection can be applied to smart meter data to assist power consumers in identifying abnormal activities, such as faulty appliances, forgotten turned on appliances. Anomaly feedback will alert the consumers to reduce their energy consumption or replace inefficient appliances. Importantly, anomaly detection will assist power suppliers and utility companies in preventing outages, identifying energy wastage and unnoticed meter faults. It will help them establish a baseline for more precise demand-response management [**drm_power**] for their clients.

Smart meter records the power consumed by household appliances. The power consumption of every household is highly dynamic. It is dependent on various factors like outside temperature, Electric Vehicle charging needs, consumers daily, weekly and yearly usage patterns, holiday events, etc. Therefore, the power supply should adapt rapidly as per the changing demands [**Grid_challenge**]. Demand response management (DRM) reduces the power demand by the consumers, by, reducing the peak power demand from the demand side and, thus, preventing power outages and emergencies. Anomaly detection maintains Smart Grid balance and effective Demand Response Management.

2.1.5 Security and Privacy preserving Smart Grid Operations

Although Smart Grids are seeking to become "smarter," its applications raise a challenges in terms of security and privacy. As an information and networking paradigm, the Smart Grid should be able to defend the involved information from unauthorized access, disclosure, disruption, modification, inspection, and

annihilation. Underlying security and privacy requirements, including confidentiality, integrity, non-repudiation, availability, access control, and privacy [5], should be satisfied in the information, communication. Besides these general requirements, securing a Smart Grid still faces a set of unique challenges. On one hand, a smart meter collects granular-scale and privacy-sensitive information from consumer power consumption; on the other hand, it analyses this information, and manipulates and impacts consumers lives. Smart Grids had some major attacks that happened in the past [**cyberattacksmartgrid**]. Smart meter data is the most important component for the service providers. The communication network for sensor networks may not be secure, and communication protocols may not be robust. Also, smart meter readings can be tampered by unauthorized users. Intercepting, tampering, misrepresenting, or forging the smart meter data will cause imbalance to the grid. The attacker can access sensitive information about consumers. The consumers who are the owners of their data have almost no control over their data. They have no control and authority of who can access their data. They have less or no information, whether the party accessing the data is authorized or not [**securityprivacy**]. Grid customers are connected over a vast network of computerized meters and infrastructure; they and the infrastructure itself is vulnerable to scalable network-borne attacks. A number of researchers discussed privacy issues in smart meters [**security_fog**, **security_fog1**].

Due to the RES integration at consumer sites, consumers want to trade the extra electricity produced to other households. They want to trade electricity via a secure channel and to authorized consumers. Therefore, to address the challenges mentioned above and to ensure secure trading of electricity in a community among households, we need a secure and privacy-preserving distributed architecture. Fog Computing based distributed architecture with Blockchain and machine learning can addresses all these challenges.

As a Smart Grid takes uses Cloud servers for data storage and processing, it faces security threats due to the untrusted Cloud servers. If the smart meter data are raw during storage and processing, they are directly revealed to the Cloud server [12]. Fog Computing paradigm as defined later provides a distributed and secure architecture, prevents the untrusted cloud server from directly accessing the collected data.

2.2 Research Questions

The research objectives and research questions of this thesis are identified and set by reckoning the challenges described in section 4. Some research questions have

the direct association with the challenges and some have indirect association.

- How can we manage the high volume data from smart meters and create a energy information solution by pushing the computation to the local community neighborhoods?
- How can we integrate the renewable energy source (wind turbines) in the most optimal way?
- What is the best way to do real-time decision making and data processing of Smart Grid data for effective Demand Response Management?
- How to have effective outage prevention and anomaly detection in Smart meter data?
- How to detect peak power usages and analyse consumers power consumption behavior for achieving power balance?
- How to effectively predict future power consumption for making power supply decisions and achieving power balance?
- How to create a security and privacy preserving system for securing the smart meter data and still analysing consumers behavior for effective power balance?

Chapter 3

Application of Graph Theory for Wind Farm Collector System Design

3.1 Application of Graph Theory for Wind Farm Collector System Design

A wind farm is a collection of wind turbines whose power should be collected and distributed through the existing electrical network. The wind turbines generate the power from the wind energy. The power produced by the wind turbines should be transmitted to a specific substation. The substation then further transmits the collected power to the Smart Grid. The substation transfers the power generated from the turbines to the Smart Grid. The electrical collector system, which consists of cables, transformers, junction boxes, switch-gear, and other electrical equipment's collects the power generated by the turbine units(distributed across the geographical area of the wind farm) to the substation. This chapter addresses the optimal design of wind farm collector systems. The work presented in this chapter is extremely crucial for planning and installation stages of wind farms as shown in Figure 40. The topology of the layout of the collector system is decided based on the factors such as climatic conditions, wind conditions, the position of wind turbines, geography, landowner requirements, aviation restrictions and, construction restrictions[**design_consider**]. We assume that the cable costs are known. There are several works on the optimal design of the collector system [**Dutta**, **MST_selftracking**] to minimize the cable length, but our work solves it

in the minimum time as compared to the others.

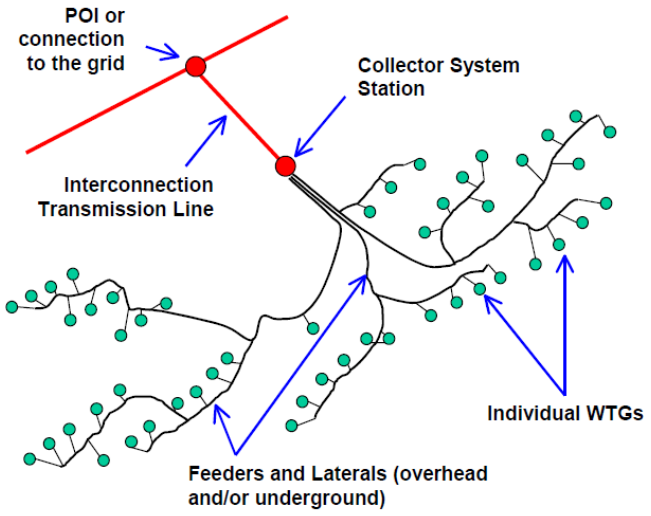

Figure 3.1: Wind Farm Collector System.

Due to the focus on increasing penetration of renewable energy resources in Smart Grids, this topic is particularly relevant at the present time. Based on the extensive study as described in the literature review section, we found that Graph-theoretic algorithms can be extremely useful tools in developing optimal designs along with taking into consideration several design constraints.

3.1.1 Motivation and Prior Art

Wind farm collector systems constitute the single most important element of wind farms after the turbines and the substation. Optimal operation of wind farms depends on optimal designs of the wind farm collector systems. So there is a need to develop sophisticated methodologies to generate these designs. However, optimal design of a wind farm collector system is not an easy problem since this requires consideration of several real-life design constraints. Considerations for laying out cables in a collector system include turbine placement, terrain, reliability, landowner requirements, economics, and expected climatic conditions for the location [99]. Another consideration is the configuration of collector systems, which can be structured as the loop system or radial system depending on the desired level of collector system reliability. Typically designs of collector systems are done manually which is cumbersome and prone to errors. But

with growing sizes of wind farms this manual process needs to be replaced by automatic design processes. Hence, there is a need for algorithms which automatically generate optimal collector system designs.

The motivation for this work is to demonstrate the applicability of graph-theoretic methods and concepts to the optimal wind farm collector system design. The problem of cable layout design for a wind farm collector system can be considered as finding a tree to meet required design characteristics in a graph G = (V, E), where V represents the set of vertices or wind turbines and the substation, and E represents the set of branches or edges connecting the vertices which in this work are the connecting cables. However, in a wind farm with hundreds of turbines, performing an exhaustive search for the optimal tree is computationally expensive since there are numerous trees to be analyzed. In fact, by Cayley's tree formula, the number of non-identical trees of order V is $V^V - 2$. Minimizing the cost of the distribution system can be a considerable challenge, as there are thousands of feasible design options to choose from. For these reasons, a lot of research [101, 23, 102] has been focused on development of optimization algorithms to identify the lowest cost distribution configuration and hence the best design. Even with approximations, such programs can help 64 reduce distribution costs by 5 to 10%

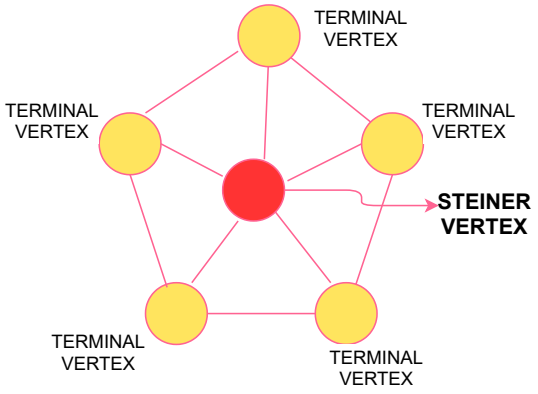

Figure 3.2: Five vertices connected in the minimum possible weight using Steiner vertex.

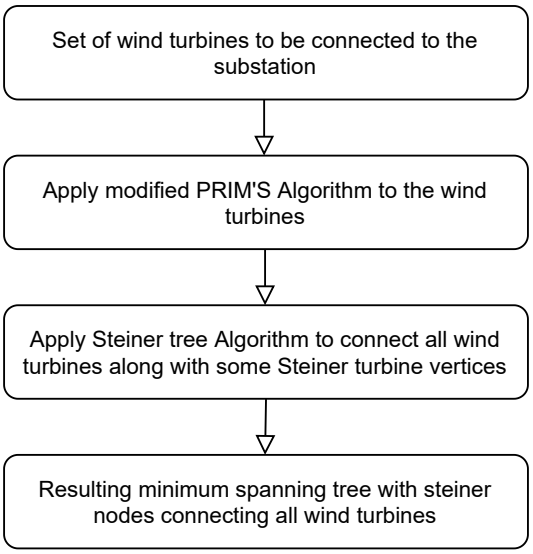

Figure 3.3: Flow diagram of the proposed approach.

3.1.2 Minimum-Weight Spanning Trees (MST)

For an undirected graph of V vertices E edges, we need to choose $|E| - 1$ edges to connect all the vertices together in an acyclic network (a 'tree'). More than $|E| - 1$ edges will induce cycles.

The MST problem can be defined as followingly:

Given an undirected graph $G = (V, E)$, in which each edge $(u, v) \in E$, has a weight $w(u, v)$ specifying the cost to connect u and v. The problem is finding an acyclic subset $T \subseteq E$ that connects all of the vertices so that the total weight of the edges is minimal. $w(T) = \Sigma w(u, v), \forall w(u, v) \in T, w(T)$ is minimal.

Since the network of edges T is acyclic and connects all of the vertices, it is a tree. Hence, it is called minimum- weight spanning tree (or "minimum spanning tree (MST)", for short).

Literature provides two simple algorithms for solving MST problem: Kruskal's algorithm [**kruskal1956shortest**] and Prim's algorithm [**prim1957shortest**]. Both Kruskal's algorithm and Prim's algorithm are "greedy algorithms". As greedy algorithms, these two algorithms make one of several choices at every step, assuming the choice taken at that step will provide a globally optimal solution (greedy approach: local optimization leads to global optimization). Usually, greedy algorithms do not guarantee globally optimal solutions. However, Kruskal's

algorithm and Prim's algorithm do provide a globally optimal solution for the MST.

Both Kruskal's algorithm and Prim's algorithm implement a "generic" minimum-spanning-tree method that grows a spanning tree by adding one edge at a time [**Cormen2009**]. However, the new algorithm proposed in section-3.1.4 is based on modified Prim's algorithm. This is because Prim's algorithm has the following advantage over Kruskal's algorithm:

- Prim starts with a source vertex (or a 'prime' vertex) and build a tree from this vertex. However, Kruskal looks for any light edge at every step and not necessarily connected with the existing tree, thus builds a set of trees (a forest) during the iterations. This means, if we stop the iterations abruptly, Prim's algorithm will give a rudimentary tree of some connected vertices, whereas Kruskal's algorithm will provide a set of rudimentary trees. Thus, Prim's algorithm strictly follows the "incremental approach" and always has a partial solution at the end of every itertation.

The running time of Kruskal's algorithm takes $\mathcal{O}(|E| \times log|V|)$ [**Cormen2009**]. Hence, for dense graphs, Kruskal's algorithm takes $\mathcal{O}(|V|^2 log|V|)$. In contrast, Prims's algorithm take $\mathcal{O}(|E| + |V|log|V|)$ [**Cormen2009**], which becomes $\mathcal{O}(|V|^2 + |V|log|V|)$ [**Cormen2009**]. Therefore, Prim's algorithm runs (somewhat) faster for dense graphs.

3.1.3 Steiner Spanning Trees

A Steiner spanning tree (SST) is a more practical spanning tree, as we usually need not connect every vertex of a graph. We may be interested only in connecting a subset of vertices in a minimum-weight tree of edges. The subset of vertices that are needed to be connected is called the *Terminals*.

There are a large classes of Steiner spanning trees [**cheng2013steiner**, **robins2000improved**, **graham1976remarks**]. However, in this paper, we limit the scope to the elementary Steiner spanning trees. The problem of developing Steiner Spanning Trees are defined as followingly:

Given a connected undirected graph $G = (V, E)$, the weight of each edge $w(e)$ is an integer, and a set of vertices N is known as the terminals, $N \subseteq V$, then the problem is to find a subtree $T \subseteq E$ spanning all the terminals and the total weight of T is minimal.

Let $|N|$ be the number of terminals, and $|V|$ be the total number of vertices. We can classify the Steiner spanning tree problem into three cases:

- $|N| = 1$: Trivial case. There is only one terminal vertex.
- $|N| = 2$: Special case in which only two terminal vertices are needed to be connected.
- $|N| = |V|$: Special case in which all the graph's vertices are terminals.
- $2 < |N| < |V|$: The usual (non-trivial) case.

3.1.3.1 Steiner Spanning Trees: Trivial case, $|N| = 1$

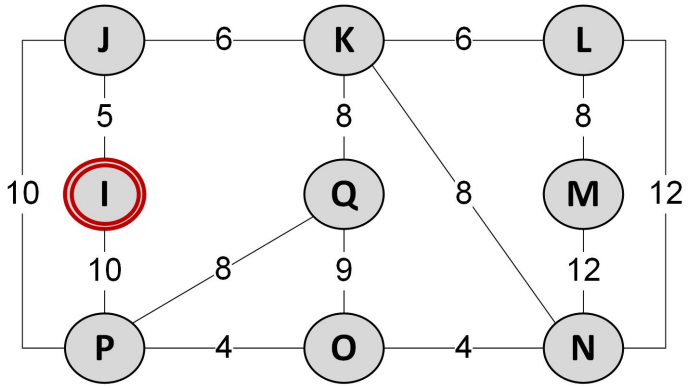

Figure 3.4: Steiner Spanning Trees: Trivial case, $|N| = 1$.

As shown in Fig. 3.4, we are only interested in a single terminal vertex ('I'). Thus, there is no need to establish a tree. The single vertex 'I' becomes the tree.

3.1.3.2 Steiner Spanning Trees: Special case, $|N| = 2$

As shown in Fig. 5.1, we need to establish a tree that connects two terminal vertices ('I' and 'M') only. This is straightforward, as we can run "Single source shortest path" (e.g., Bellman-Ford's algorithm [**bellman1958routing**], [**ford1956network**], or Dijkstra's algorithm [**dijkstra1959note**]) from the source vertex 'I'. Upon completion of the MST, backtracking from the terminal vertex 'M' to the source 'I' will identify the path that becomes the Steiner spanning tree (it is a path than a tree).

In Fig. 5.1, the Steiner spanning tree that spans the terminals 'I' and 'M' and possesses some other vertices such as 'J', 'K', and 'L'. Though the focus is on the terminals ('I' and 'M'), we need some other vertices without which forming a spanning tree might not be possible. *Steiner vertices* are non-terminal vertices that happen to end up in the spanning tree; without the Steiner vertices, creating

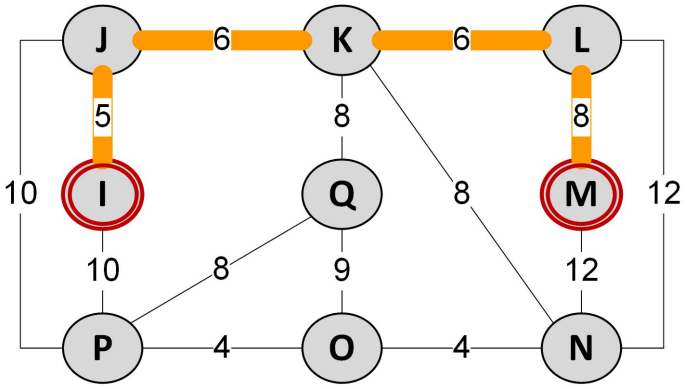

Figure 3.5: Steiner Spanning Trees: Special case, $|N| = 2$.

a Steiner spanning tree is may not be possible as the terminals vertices are not connected with each other directly.

3.1.3.3 Steiner Spanning Trees: Special case, $|N| = |V|$

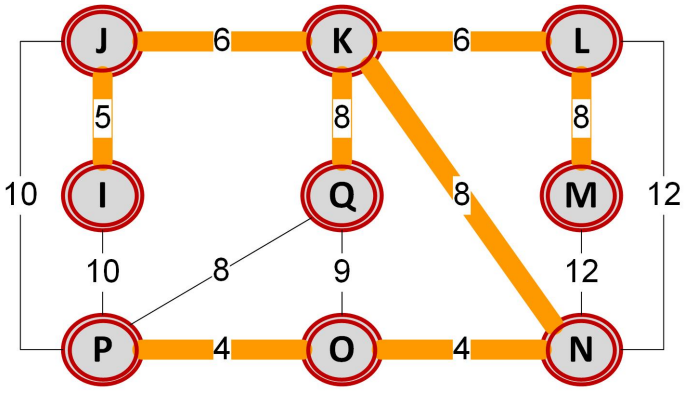

Figure 3.6: Steiner Spanning Trees: Special case, $|N| = |V|$.

As shown in Fig. 5.2, $|N| == |V|$ means all the vertices are Terminal vertices. Thus, the problem becomes the usual "Minimum Weight Spanning Tree (MST)" problem. Hence, Prim's or Kruskal's algorithm can be used. Fig. 5.2 shows the Steiner spanning tree, which is the same as the minimum-weight spanning tree.

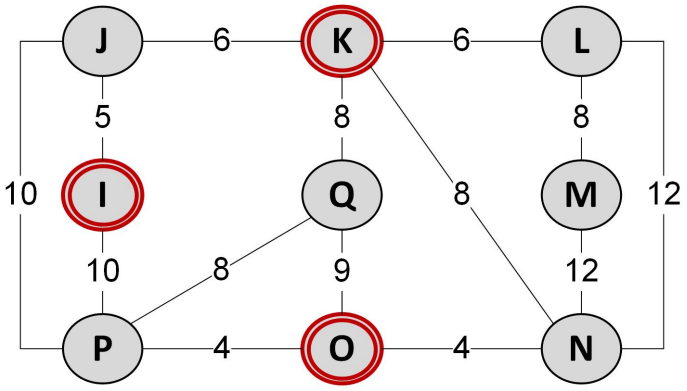

Figure 3.7: Steiner Spanning Trees: Non-trivial case, $|N| = 3$.

3.1.3.4 Steiner Spanning Trees: Special case, $2 < |N| < |V|$

Fig. 5.3 shows a non-trivial case ($|N| = 3$). Providing a solution for this non-trivial case ($2 < |N| < |V|$) is the goal of this paper. The proposed algorithm is given in the following section.

3.1.4 Methodology

The location of wind turbines is considered as the vertices while, the costs of cables is regarded as the weights of the edges connecting the wind turbines, forming a graph. Fig. 1 illustrates the Steiner spanning tree for a graph with 5 vertices. Yellow colored vertices are the vertices which should be connected with total minimum weight. When we introduce an additional vertex represented by red color then, these 5 terminal vertices can be connected in the minimum possible weight. We have wind turbine vertices which should be connected mandatorily and, are called *Terminal* vertices. As explained, the euclidean Steiner tree problem is NP-hard, and hence an optimal solution can not be found using a polynomial time algorithm.

Our proposed algorithm achieves this goal in polynomial time as explained in the next section.

The algorithm, its running time, an application example, and the implementation details are given in the following subsections.

3.1.4.1 The Modified Prim's Algorithm

Prim's algorithm is run from a source vertex. And it will run until all the vertices that can be reached from the source vertex are added to the spanning tree. However, for a Steiner spanning tree, we need not wait until all the possible vertices that are reachable from the source vertex are added. Our primary goal is to add all the possible terminal vertices that are reachable from the source terminal. Hence, Prim's algorithm is modified so that the iterations will stop once all the Terminals are absorbed into the tree. Fig. 3.8 shows the modified Prim's algorithm, in which line-6 is the modification.

If some of the terminals are not reachable from the source terminal, then the iterations will run until no other vertex can be added to the spanning tree, just as in the case of the (original) Prim's algorithm.

PRIM_Modified (*G.V*, *G.E*, *T*)
Input: ***G.V*** – set of vertices
G.E – set of edges
T – set of terminal vertices
Output: ***MST*** - Minimum Spanning Tree

% initialization
1 sort E in nondecreasing order by weight w
2 extract $e_1 = (u_1, v_1)$ from E with minimal w
3 $A = \{u_1, v_1\}$
4 eliminate e_1 from E
5 $V = G.V - A$
% the main loop
6 while $V \neq \emptyset$ or $(V \cup T) \neq \emptyset$
7 | extract $e = (u_i, v_i) \in E$ with minimal w
| such that $u_i \in A$ and $v_i \in (V - A)$
8 | $A = A \cup \{v_i\}$
9 | $V = V - v_i$
10 | eliminate e from E

Figure 3.8: Modified Prim's Algorithm.

Steiner_Spanning_Tree (*G.V*, *G.E*, *T*)
Input: ***G.V*** – set of vertices
G.E – set of edges
T – set of terminal vertices
Output: ***SST*** – Steiner Spanning Tree

% step-1: set T(1) as the source terminal vertex
1 set the source terminal vertex $T(1) = t_S$

% step-2: run Prim's modified algorithm
2 ***MST = Prim_modified(G.V, G.E, T)***

% step-3: the main loop
6 **while** ***NOT(exist(v_i a Steiner_vertex with 1-degree))***
7 | Eliminate v_i from ***MST***

% the result: assign the remaining MST as the
% Steiner spanning tree
8 ***SST = MST***

Figure 3.9: The Algorithm for finding Steiner spanning tree.

3.1.4.2 The Algorithm for finding Steiner Spanning Tree

The algorithm is shown in Fig. 3.9, and it consists of three steps:

- Step-1: In the first step, one of the terminal vertices are taken as the source vertex.

- Step-2: The modified Prim's algorithm is run from the chosen terminal vertex. Prim's algorithm will be terminated once all the Terminals are absorbed into the tree (no need to wait until all the graph's vertices, including the non-terminal vertices, are included in the tree).

- Step-3: This step is also iterative, in which we will delete the Steiner vertices (non-terminal vertices) that have one degree of connection. This step will continue to delete non-terminal vertices with one degree of connection, one by one, until there exist no more non-terminal vertices with one degree of connection.

3.1.4.3 Running Time

- Step-1: Takes $\mathcal{O}(1)$ time as this step only involves a simple assignment (assigning one of the terminal vertices as the source vertex).
- Step-2: This step involves modified Prim's algorithm; thus, take $\mathcal{O}(|E| + |V|log|V|)$ time, which is equivalent to $\mathcal{O}(|V|^2)$ for dense graphs.
- Step-3: This step deletes non-terminal vertices that have one degree of connection. There will be a maximum of $|V|-|N|$ number of non-terminal vertices. Thus, this step takes $\mathcal{O}(|V| - |N|)$ time.

Hence, the running time of the algorithm (RT):
$RT = \mathcal{O}(1) + \mathcal{O}(|V|^2) + \mathcal{O}(|V| - |N|) = \mathcal{O}(|V|^2)$.

3.1.5 Application Example

The algorithm is implemented in the MATLAB platform as a function of the tool known as the General-Purpose Petri Net Simulator (GPenSIM). GPenSIM is developed by the second author of the paper, and is freely available [**Davidrajuh2018-Book**]. As shown in the Fig. 5, the wind farm collector system optimization problem is defined as a graph problem.

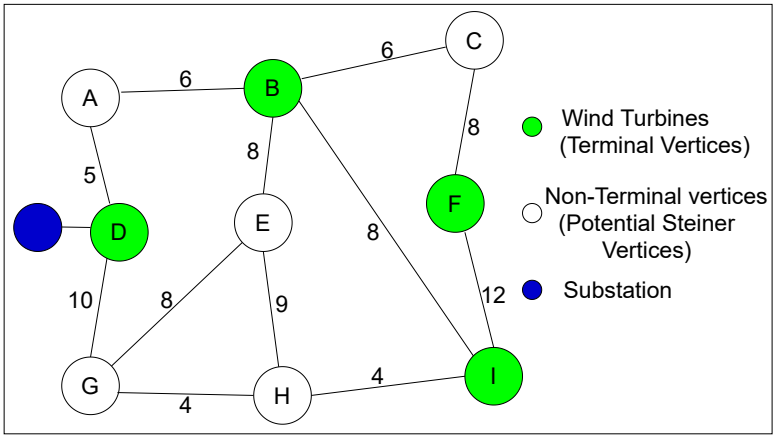

Figure 3.10: Wind turbine collector system optimization problem.

The initial connection and cable length is as shown in the figure 5. The wind turbines which should be connected are shown in green colored vertices and, are also called Terminal vertices. Blue color vertex is the substation. The cable lengths are shown as edge length. The next step is to apply the modified Prim's

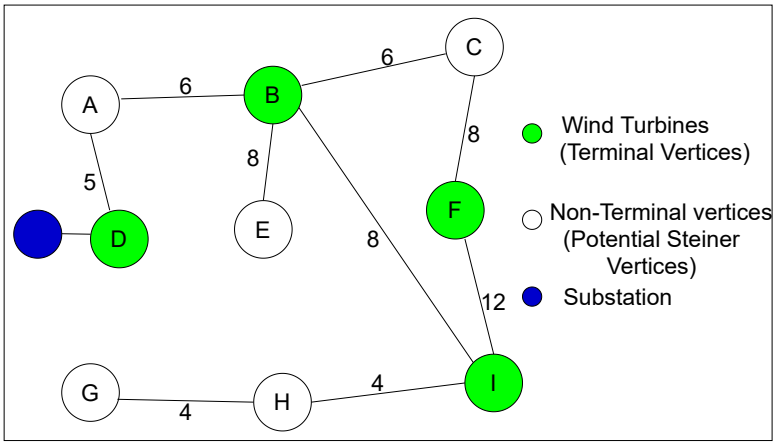

Figure 3.11: Intermediate path after applying the porposed algorithm.

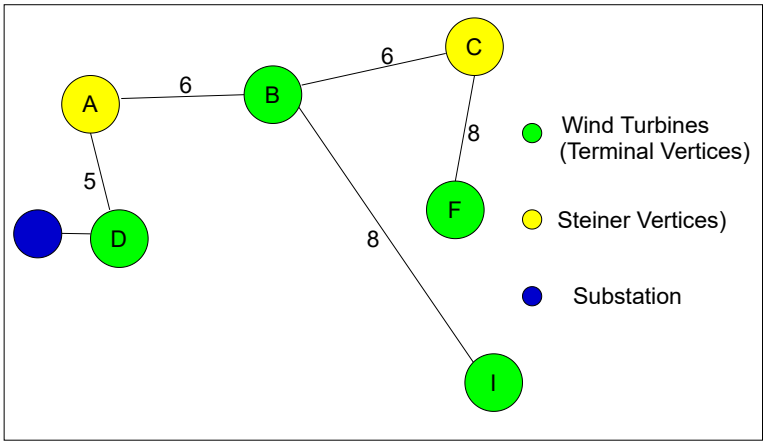

Figure 3.12: Resulting Steiner Spanning tree after running the proposed algorithm with minimum cable length.

algorithm that connects all the turbines with minimum possible cable length. After applying the modified Prim's algorithm, we get the minimum spanning tree with vertices D-A, A-B, B-E, B-I, B-C, C-F, I-H and, H-G. The total weight comes out to be 49. Thereafter, we apply the Steiner tree algorithm. It identifies two Steiner vertices with one-degree of connection G and E, so, these are deleted. In the next iteration, it identifies a vertex H with one degree of connection. And so, vertex H is deleted. The algorithm stops as there are no more Steiner vertices with one degree of connection. The resulting minimum spanning Steiner tree has vertices D-A, A-B, B-I, B-C and, C-F. The minimum weight calculated comes out to be 33. The resulting Steiner tree is shown in Fig. 6.

3.1.6 Conclusion

In this Chapter, we proposed a novel Steiner tree algorithm for generating a design of wind farm collector system with minimum cable length. Our algorithm reduces the construction cost and the maintenance cost. Also, our algorithm solves a NP-hard minimum Steiner tree problem, in polynomial time.

Chapter 4

Application of Machine Learning Prediction techniques for Anomaly Detection and effective Demand Response Management

4.1 Motivation

In Smart Grids, smart meter records the power consumed by household appliances. The power consumption of every household is highly dynamic. It is dependent on various factors like outside temperature, Electric Vehicle charging needs, consumers daily, weekly and yearly usage patterns, holiday events, etc. Therefore, the power supply should adapt rapidly as per the changing demands [**Grid_challenge**]. Demand response management (DRM) try to reduce the power demand by the consumers, by, reducing the peak power demand from the demand side and, thus, preventing power outages and emergencies. Despite that, power outages are common as, the demand side data is not analysed and forecasted for anomalies. There have been numerous incidents of power outages across the world in the past [**blackouts_history**]. During the winter storm in Texas in February 2021, nearly 5 million people lost power [**texas_crisis**]. Also, in June 2021, due to the high power demand and insufficient supply, California grid operators suggested consumers to charge their electric vehicles during off-peak hours. In New York, for the first time power outages hit several neighborhoods due to high temperature rise and officials sent an emergency mobile alert

to consumers urging them to save electricity [**ny_poweroutage**]. To avoid these power outages, it becomes highly important to forecast the power anomalies. It can maintain Smart Grid balance and Demand Response Management can be achieved effectively.

4.2 Prior Art

Statistical, clustering, and data mining approaches are the commonly used techniques for discovering abnormal consumption behaviors [**visual_anomaly**]. Liu and Nielsen [**liu2016regression**] proposed a model for anomaly detection, where they first applied a mixture of supervised learning algorithm called Periodic Auto-Regression with eXogenous variables (PARX) and Gaussian statistical distribution to detect the anomalies on historical consumption data. Even though their results have identified certain anomalies based on temperature change, their suggested method could not detect long term daily and weekly seasonality, and, abnormal holiday behaviors. Fathnia et al.[**Fathnia**] provided a method that combined the Local Outlier Factor LOF index and Ordering Points To Identify the Clustering Structure OPTICS density-based algorithm to detect the unusual nature of the data. Their strategy recognised outliers as transmitted faults in the smart meter data to the control center. They reasoned that cyberattacks caused these failures. While the research findings demonstrated the efficiency of the proposed technique, they have not considered unnatural users activities or inefficient appliances causing anomalies. On the same lines, Zhang et al. [**GMM_LDA_clustering**] introduced an adaptive method to detect abnormalities in the smart meter data. They labelled the data set using the Gaussian Mixture Model Linear Discriminant Analysis GMM-LDA algorithm, which clusters some data sets to obtain the optimal feature representation of normal and abnormal patterns. Thereafter, they used the Particle Swarm Optimization Support Vector Machine (PSO-SVM) classifier to learn from the labelled data and predict future unlabelled data. Their approach also could not detect unexpected patterns due to special events and, could be complex to detect strong seasonal behaviors. In the research [**jakkula2010outlier**], the authors experimented with two approaches for detecting abnormalities in real and generated synthetic datasets. They started with a statistical technique, which uses Standard Deviation to identify extremes. The second technique utilised the K-Nearest Neighbor KNN clustering algorithm to distinguish anomalous from normal data using the point-to-point distance measure. Also, research work by Janetzko et al. [**visual_anomaly**] introduced two methods that adapted to the seasonality of the energy consumption data. The first approach estimated the error rate using weighted prediction, where they gave

more substantial influence of the current measurements than older ones. The latter approach used similarity-based anomaly detection after transforming the daily pattern into the frequency domain by Fourier transformation. Additionally, they provided a range of time series visualisation techniques for resulting anomaly scores that aid in analysing and comprehending energy consumption behaviour. Research work [**rituka**] propose a prediction based approach for finding anomalies in the power consumption data. The drawback of this approach is that their model does not identify all the seasonality patterns and therefore, misses some of the anomalies.

Nevertheless, a combined drawback of all the above papers is that they have not considered some anomalies, by not considering one or more of these factors: all kinds of seasonality impact, holidays effect, long term seasonality and trend change, did not scaled well for the data. This could have significant impact on the grid balance and will effect decision making for power companies. In light of the shortcomings of the mentioned prior works, we present a simple and highly effective technique to detect a wide range of anomalies considering the strong seasonality and could overcome all the shortcomings mentioned above.

4.3 Methodology

We propose a novel approach for anomaly detection by employing Facebook's prophet Algorithm. The power consumption domain knowledge is extracted from exploratory data analysis. We found trend, seasonality components and holiday dates and modelled on the Prophet Algorithm. Prophet, is a Facebook-developed open source library mainly used to analyze time-series data and excels at forecasting highly periodic data [**taylor2018forecasting**]. We identify the anomalies by using the Prophet model's confidence intervals. Any data point that lies outside the interval is considered as an anomaly. Further, since zero power consumption could indicate sensor fault, we classified all zero values in the data as anomaly. After classifying the data into two categories (positive class indicating anomaly, negative class indicating normal data), we get an unbalanced data set. Thus, in the second phase of our study, we propose the best classification model that classifies the power consumption data as either normal or anomaly. We evaluate our models performance using Sensitivity, Specificity, G-mean, F-score, AUC values and model run-time metrics. This approach can be tested on any anomaly labelled dataset. In below subsections, we describe our approaches in detail.

The Prophet is a Generalize Additive Model(GAM) that Facebook has devised

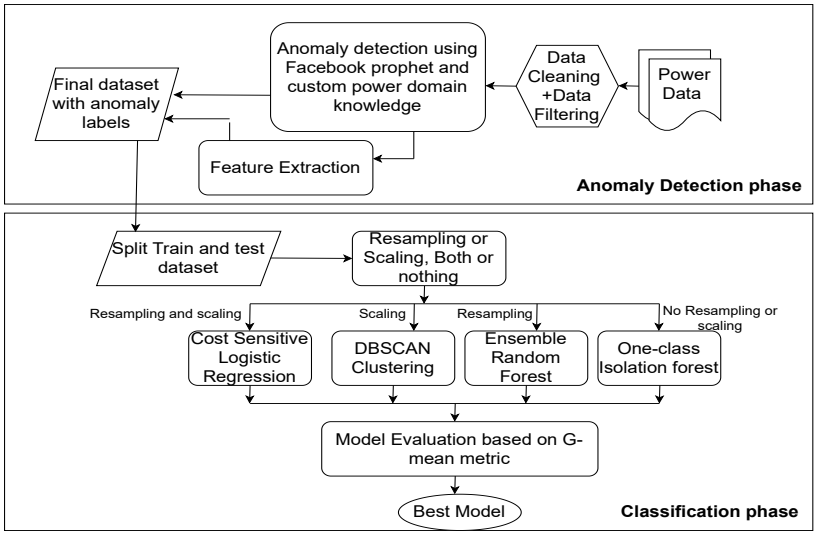

Figure 4.1: Workflow Anomaly detection and classification approach.

and is available as an open-source library. It is a decomposable time series model [**harvey1990estimation**] with three main model components: trend, seasonality, and holidays. It provides with the ability to make fast, time series predictions with good accuracy using simple intuitive parameters and has support for customizing seasonality, holidays and behavior based on the domain knowledge (power domain in our case) [**fang_combine_2019-1**, **kumar_jha_time_2021**].

The Prophet core is an additive regression model that accommodates three components as indicated by the following mathematical notation [**hasan_shawon_forecasting_2020**, **freeman_experimental_2019**, **thiyagarajan_temporal_2020**].

$$y(t) = g(t) + s(t) + h(t) + \epsilon(t) \tag{4.1}$$

where,

- $g(t)$ is the trend function that represents the linear or logistic non-periodic changes in power consumption readings obtained at evenly time-space intervals.
- $s(t)$ is a seasonal component formed using Fourier series, and it captures the historical data periodic changes, which can be daily, weekly, monthly or yearly seasonality. In our case, we identified daily, weekdays, weekends, yearly seasonality, so models regression equation will be

[**vink_build_2018**]:

$$y(t) = g(t)+s(t)_{daily}+s(t)_{weekdays}+s(t)_{weekends}+s(t)_{yearly}+h(t)+\epsilon(t) \tag{4.2}$$

where,

- $h(t)$ represents the holidays during the year and can be provided by the user or unusual expected times that happen irregularly.
- $\epsilon(t)$ indicates an independent error term, which is assumed to be normally distributed.

We considered an uncertainty interval by taking the *99-th* quantile of the posterior predictive distribution, which has a 99% chance of containing the parameter's actual value [**oakley_chapter_2021**]. Besides, any value that lies outside the interval is identified as an anomaly. We define the uncertainty interval ***I*** as:

$$I = [\hat{y}_{lower}, \hat{y}_{upper}] \tag{4.3}$$

We established the following decision rule for an original data sample y [**bellas_anomaly_2014**]:

$$\begin{cases} \text{y is normal, if } y \in I \\ \text{y is anomaly, if } y \notin I \end{cases} \tag{4.4}$$

4.3.1 Anomaly Classification Approach

In the two-class (binary) imbalanced classification problem, the minority (under-represented) group is typically referred to as the positive class, while the dominant group is the negative class [**alberto_fernandez_learning_2018**]. The classification algorithm's performance is often assessed by comparing the predicted class labels to the real ones. The model is trained on the training set and then evaluated on the holdout test set. The standard evaluation metrics, such as Accuracy (which is the percentage of accurately categorised samples), can not be used for the classification problem because in an imbalanced classification problem, the majority class has more data compared to the minority class and therefore, the model is biased towards the majority class [**alberto_fernandez_learning_2018**, **brownlee_imbalanced_2021**].

Before presenting the evaluation metrics for the classification task that we used, we need to understand the Confusion Matrix for binary classification

tasks. The confusion matrix can provide more insight into the model performance and gives information about the correctly and incorrectly predicted labels [**brownlee_imbalanced_2021**]. It has four possibilities: A true positive(TP) is an outcome where the model correctly predicts the positive class. Similarly, a true negative(TN) is an outcome where the model correctly predicts the negative class. A false positive(FP) is an outcome where the model incorrectly predicts the positive class. And a false negative(FN) is an outcome where the model incorrectly predicts the negative class. In our power anomaly context, False Negatives should be focused, since it shows how many true anomalies are predicted as normal samples and, we want to make it as low as possible.

- **Sensitivity** indicates the accuracy with which the positive class was anticipated and it is suitable when the objective is to minimise false negatives.

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.5}$$

- **Specificity** indicates the accuracy with which the negative class was anticipated and it is suitable when the objective is to minimise false positives.

$$Specificity = \frac{TN}{TN + FP} \tag{4.6}$$

- **G-mean** combines Sensitivity and Specificity into a single score that takes both into consideration.

$$G - mean = \sqrt{Sensitivity \times Specificity} \tag{4.7}$$

- **ROC Curves** The Receiver Operating Characteristic (ROC) curve is a graphical evaluation technique that widely used for summarising classifier performance over a range of true positive and false positive error rates trade-offs. ROC demonstrates that the true positive rate cannot be increased without raising the false positive rate for any classifier. AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example. The higher the AUC score, the more accurate the classifier.

4.3.2 Ausgrid Dataset

We used the open dataset by the largest distributor of electricity on Australia's east coast called Ausgrid. The data was collected from 300 randomly selected

consumers between 1 July 2010 and 30 June 2013, sampled at half-hour interval. The Ausgrid corporation used three separate meter recording devices for three different categories Gross Generation GG, General Consumption GC, Controlled Load CL. The first meter (GG) records the solar power generated by the solar PV units placed at the rooftop of each household. The second meter records the daily power consumption (GC) in (KWh) for each consumer. And, the last meter recorded the power consumption with water heating placed in some of the houses by offering a monetary incentive [**ratnam_residential_2017**]. The dataset contains 54 columns. These are columns: consumer IDs, Postcode, Generator Capacity, Consumption Category and Date, followed by 48 columns of half-hour power meter data. We used consumption category GC, since we are interested in finding anomalies in the power consumption.

4.3.2.1 Data Filtering

To demonstrate the model and approach, we picked only ten consumers to represent a small community and minimise the time required for the experimentation for all 300 consumers. We analysed the Postcode feature and selected the area with the most consumers.

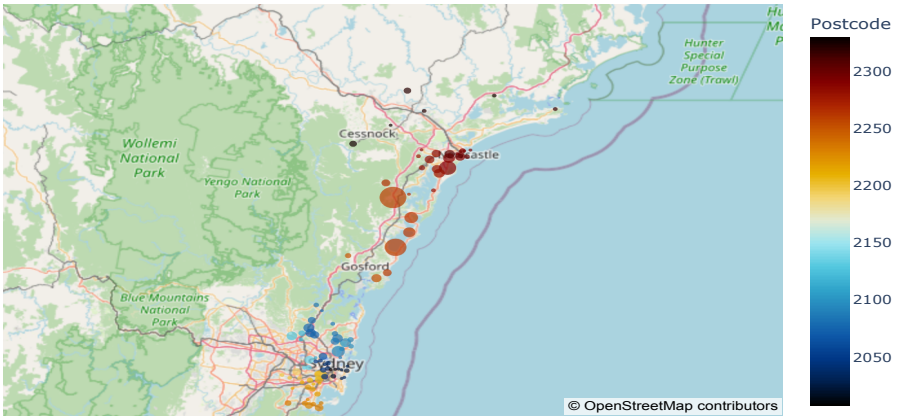

Figure 4.2: The distribution map of Ausgrid. The circles represent postcode regions covered in the 300-consumer dataset.

Our files contain postcodes ranging from 2008 to 2330, which are of New South Wales (NSW) state. Additionally, as seen in Figure 4.6, the largest area is located between Newcastle and Sydney; and its postcode value is 2259. We selected the top ten consumers from this postcode. The ten selected consumers' IDs are 7, 29, 30, 64, 155, 160, 184, 202, 206 and, 215.

4.3.3 Exploratory Data Analysis

To obtain a summary of the energy consumption of the consumers, we started by aggregating the data annually using the summation method for each consumer. Figure 4.4 shows the overall households consumption over two years. It indicates how the individual consumers behaviour varies from other consumers. The power consumption was highest for the year 2012, and this could be due to comparatively low temperatures during winter. This consumption behavior can potentially cause grid imbalance/outages if not forecasted in advance.

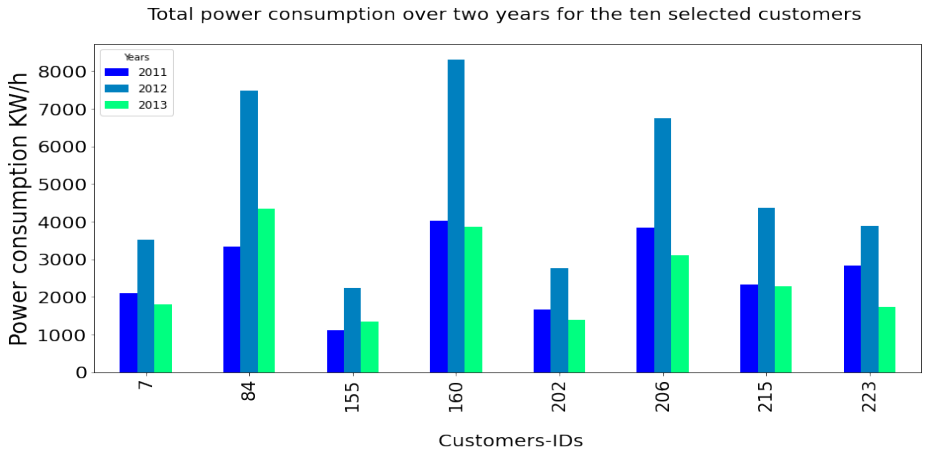

Figure 4.3: Total power consumption over two years for ten consumers.

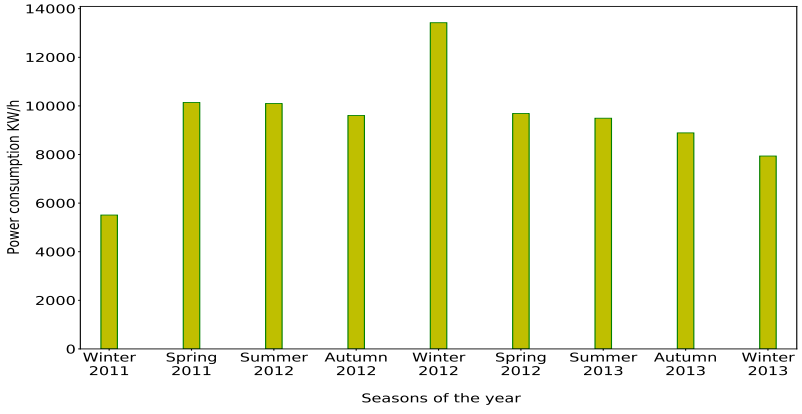

Figure 4.4: Total peak power consumption over the four seasons.

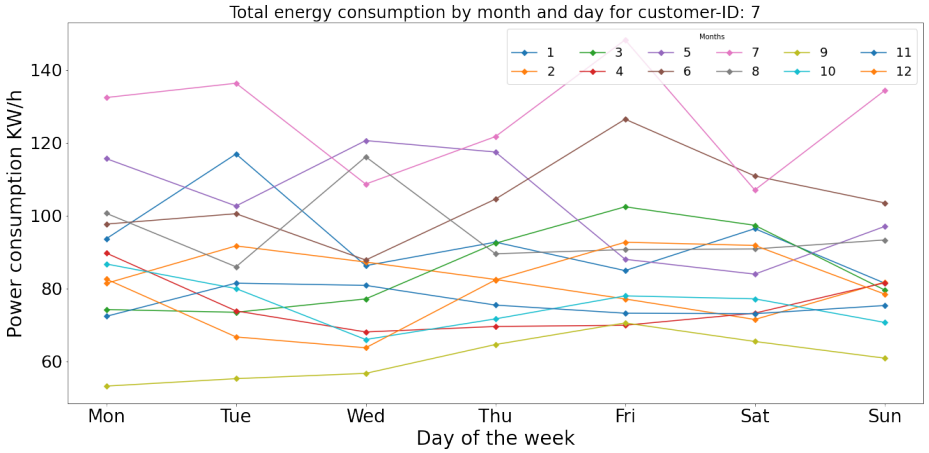

Figure 4.5: Total power consumption aggregated for week for a period of one year for consumer number 7

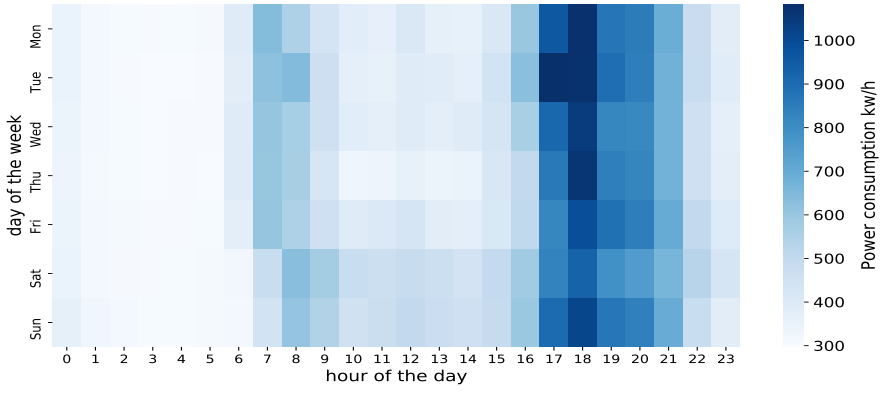

Figure 4.6: Power consumption aggregated for 24 hours for the seven days of a week.

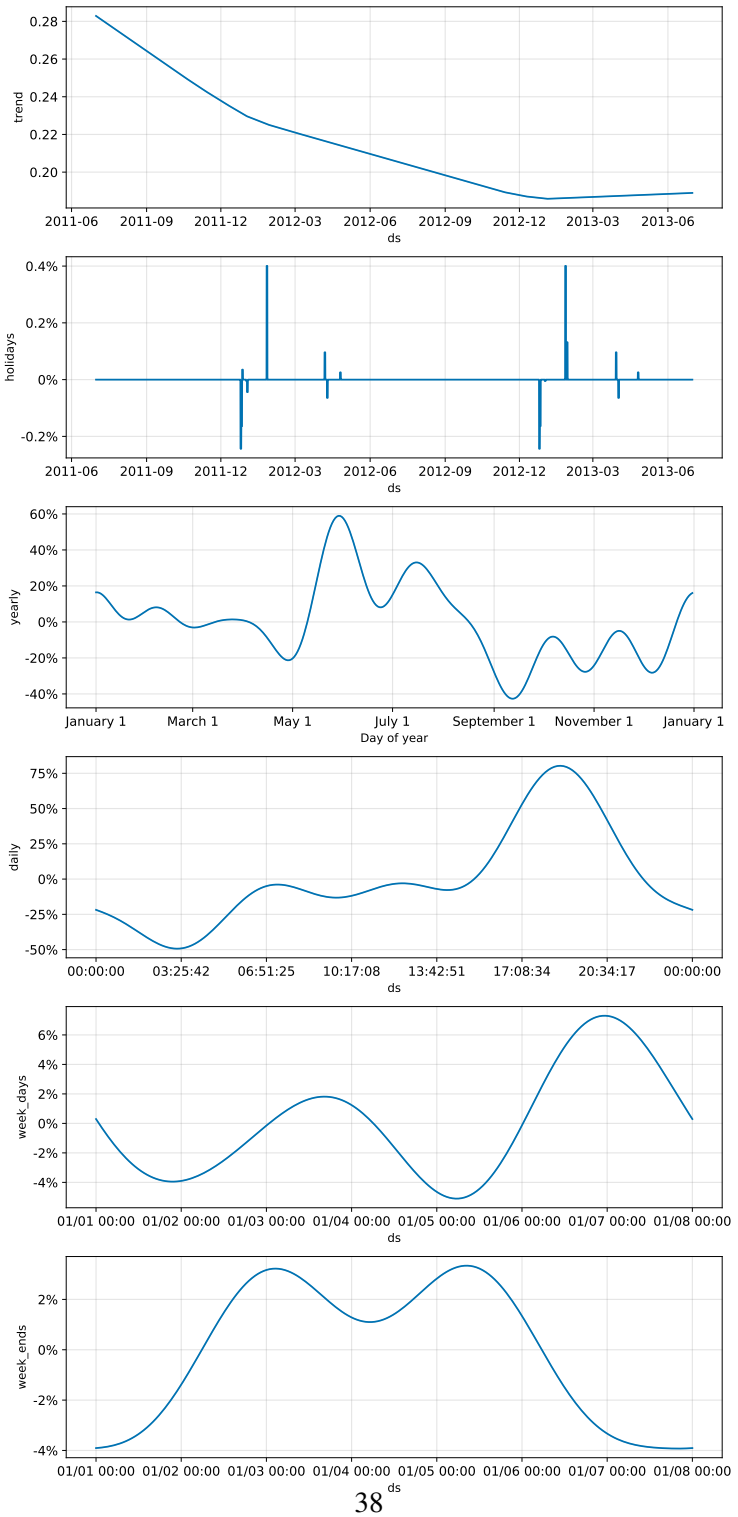

Figure 4.7: Trend, daily, weekdays, weekends and yearly seasonality and, holidays pattern.

Furthermore, we aggregated all the consumers in one and found out the total power consumed for all the four seasons of the year. In Australia, the peak Winter period months are (June-July-August), the Spring period includes (September-October-November), the Summer period includes (December-January-February), and the Autumn period includes (March-April-May). Figure 4.4 shows the peak power consumption in seasons of 2011, 2012 and 2013. People tend to use air conditioning in the summer and heating in the winter. The winter of 2012 recorded the maximum power consumption while the winter of 2011 recorded the lowest consumption value.

Further, the consumer no. 7's total consumption was aggregated for a week for a period of one year. Figure 4.5 shows the plot, which shows that the highest consumption was in month July (winter season), while the lowest demand was in September and October (autumn) months. Figure 4.6 the power consumption aggregated for 24 hours for the seven days of a week for consumer 7. The darker the color blue, the higher the demand.

As seen there is a higher variation in power consumption throughout the 24 hours of the day. Higher power consumption occurs during the evenings between 5 p.m and 9 p.m, reaching more that 1000 KWh or more at 6 p.m; while it decreases at least by 70 per cent (from 1000 to 300 KWh) after midnight and until 6 a.m.

4.3.3.1 Feature Extraction

Feature extraction is essential to ensure that our ML algorithms perform optimally. Through the data exploration and analysis step describe above, we were able to derive the following important features: *hour-of-day*, *day-of-week*, *month-of-year*, and *year*.

4.3.4 Anomaly Identification

We employed the *Prophet* method to analyse and identify anomalies by applying the anomaly decision rules mentioned in previous sections. After fitting the model, predictions are done for a period referred to as the "horizon". The "cutoff" points were defined as follows:

The "initial" training period is chosen to be three times the length of the "horizon" forecasting period, and its size is required to be large enough to capture the seasonality being studied. Since the yearly seasonality is included in the model, the initial period must be at least one year in length [**facebook_quick_2021**]. Cutoffs were made every half a "horizon"[**facebook_quick_2021**]. With the range

of two years data and taking the above considerations, we determined an initial training period as 18 months, a horizon forecasting period as six months and a single cutoff point of "01-12-2012". The Prophet provides a performance metrics tool that can be used to select the hyperparameter combination set with the lowest evaluation measure. This utility analyses each hyper-parameter combination forecast and calculates a valuable statistic on the prediction performance, the root mean squared error (RMSE) score. To tune the hyper-parameters, we created a dictionary of all rational hyper-parameter combinations. The parameters included in the tuning are: **change_point_range**, **changepoint_prior_scale**, **seasonality_prior_scale**, **holiday_prior_scale** and, **seasonality_mode**. After tuning the hyper-parameters, we created custom functions for weekday and weekends seasonalities. Finally, we forecasted the power consumption.

The year wise trend changes is visualised in Fig 4.7, starting with a higher overall power consumption in 2011 and downward growth reaching year 2013. The holidays component is shown in part two of the figure 4.7, while in part 3 of the same figure, the yearly seasonality is shown, part 4 shows the daily seasonality and the weekends and weekdays consumption pattern is shown in part 5 and 6 of the figure 4.7.

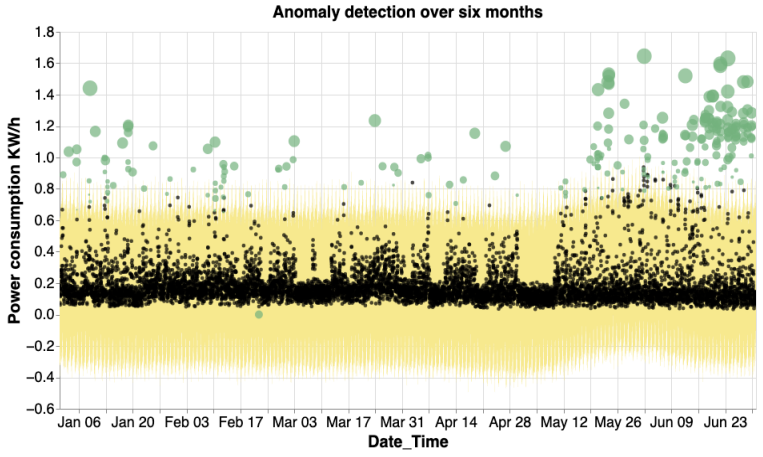

Figure 4.8: Actual and Anomalous power consumption over a period of six months for consumer number 7.

For the Prophet model, we chose a 99% uncertainty interval to account for any unexpected events requiring additional energy. Figure 4.10 shows the result of the Prophet tuned model with anomalies for consumer-ID 7. As seen in the figure the yellow region represents the uncertainty interval, and the black dots

denotes the actual power consumption for consumer-ID 7 during a a period of six months. The green circles represent anomalies with a diameter proportionate to their distance from the interval range [$\hat{y}_{lower}$,$\hat{y}_{upper}$].

We exhibit a subset of the data to illustrate the abnormalities for shorter time periods in greater depth. Figure 4.9 depicts actual consumption data over three months; on the other hand, figure 4.10 depicts anomalies over the same time period.

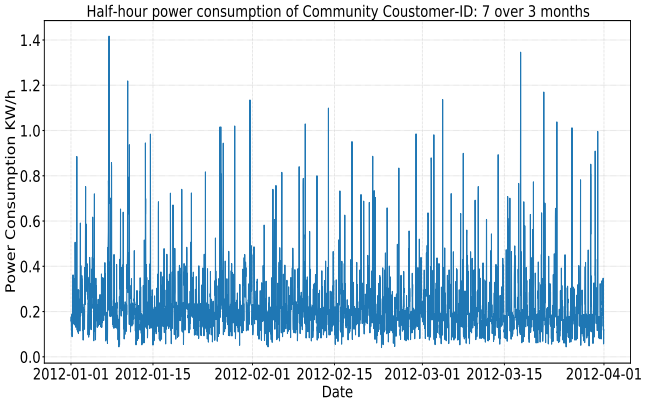

Figure 4.9: Actual power consumption for three months.

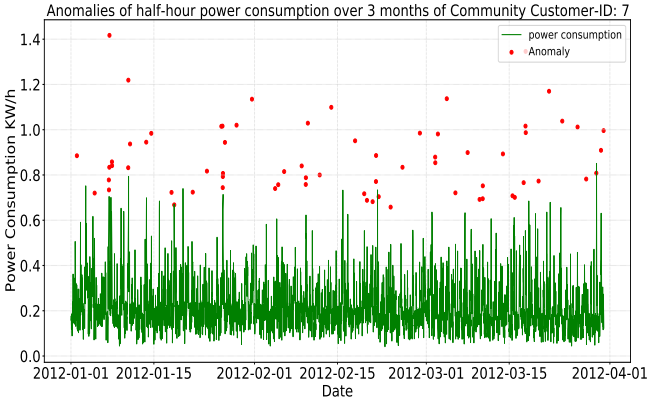

Figure 4.10: Actual and Anomalous power consumption for a period of three months

4.3.5 Anomaly Classification Approach and Experimental results

Figure 4.11 depicts how the data is disproportionately distributed among the two classes.

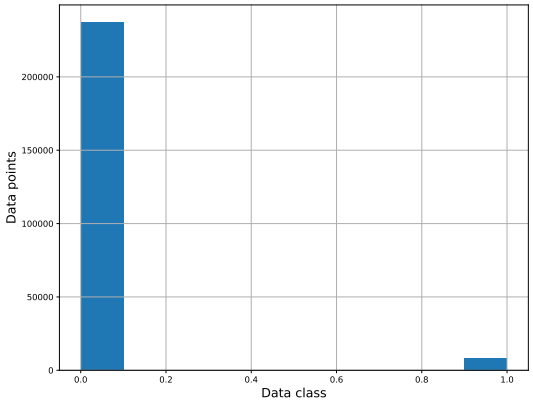

Figure 4.11: Imbalanced data of Actual and Anomalous power consumption.

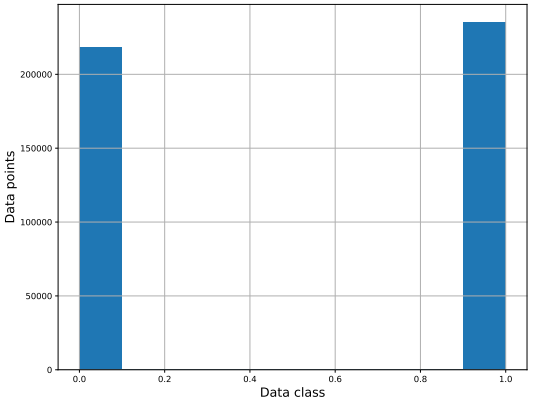

Figure 4.12: Balanced data distribution of Actual and Anomalous power consumption after applying sampling techniques on imbalanced data.

We have 96.6% values for the negative class and 3.4% for the positive class. To sample the data for accurate classification, we used the SMOTE-ENN class from

Scikit-imbalanced-learn library. The SMOTE function generates new instances of the minority class, while ENN reduces noisy points along with class borders. We used the sampling techniques only with the train set and not the test set. After applying the sampling techniques, the balanced distribution is shown in figure 4.12 which shows a significant improvement over the imbalanced dataset.

We examined four effective models with the balanced training set. These are Logistic Regression, DBSCAN, Random forest and Isolation forest. Three runs of 10-fold stratified cross-validation are run to validate the models. The F1-score is computed in each iteration of cross-validation. Subsequently, the four models are compared by the model's G-mean scores and their execution times. DBSCAN model from the Scikit-learn library requires tuning of two critical parameters, namely eps and min_samples. Additionally, the default metric parameter option is set to "euclidean", which uses the Euclidean distance measure to determine the distance between the samples. The RandomForestClassifier class from Scikit-learn is used for implementing the ensemble random forest algorithm. Also, the IsolationForest isolates data points by randomly choosing a feature and a split-value in a range between [minimum value, maximum value] of the selected feature. Further, the number of splitting required to isolate a sample is set to the path's length from the root to the terminal node. Additionally, random partitioning significantly reduces the pathways taken by anomalies, and any specific sample with shorter path lengths is considered to be anomalous [**buitinck_api_2013**]. We applied Grid search and Bayesian optimisation for hyperparameter optimization for our two models: logistic regression and DBSCAN.

4.3.6 Classification Models Evaluation

To evaluate the models, we created a function called evaluation that computes all the evaluation measures discussed in section 3.2. We started by using the Confusion Matrix metric to compute and return the True Negative TN, False Positive FP, False Negative FN, and True Positive TP values. After that, the Sensitivity, Specificity, G-mean, True Positive Rate TPR and, False Positive Rate FPR scores, F1 and F2 scores were computed.

We used the roc_curve and roc_auc_score functions from Scikit-learn library. The first function enables us to visualise the ROC curve of our models, which calculated the probabilities of the estimated class labels (all the models except DBSCAN). The second function computes the area under the ROC curve, whose scores are presented on the ROC plot to have a clear visual evaluation of the models.

The Sensitivity, Specificity, G-mean and F1 scores are shown in table 4.1. As seen, ensemble RF model had the highest Sensitivity, Specificity, and G-mean score and F-1 scores among all the models. Further, the DBSCAN and iForest also produced almost comparable results slightly less than ensemble RF model. Figure 4.13 shows the true positive rate TPR and false positive rates FPR for each of the four models.

Models	Sensitivity	Specificity	G-Mean	F1-score
Logistic Regression	0.911	0.932	0.921	0.946
DBSCAN	0.921	0.896	0.908	0.924
Ensemble RF	**0.972**	**0.975**	**0.973**	**0.978**
One-class iForest	0.926	0.895	0.91	0.924

Table 4.1: Sensitivity, Specificity, G-mean and F1 scores for the four models.

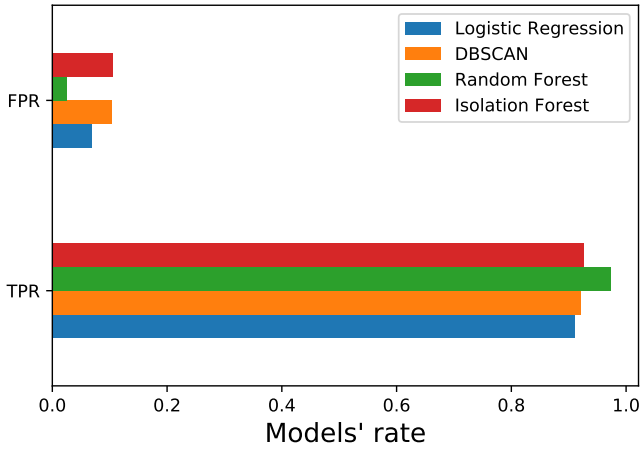

Figure 4.13: Comparison between True Positive Rates and False Positive Rates for the models.

The capability of ensemble RF to detect true positives (TPR) is the highest, while its capability to misclassify negatives (FPR) is the lowest among the four models. Further, seen in the figure, DBSCAN and iForest performed equally and had a higher TPR score than the Logistic regression model. Figure 4.14 demonstrates that ensemble RF outperforms the other models and has the highest F2-score.

The receiver operating characteristic (ROC) curve is utilized to evaluate the

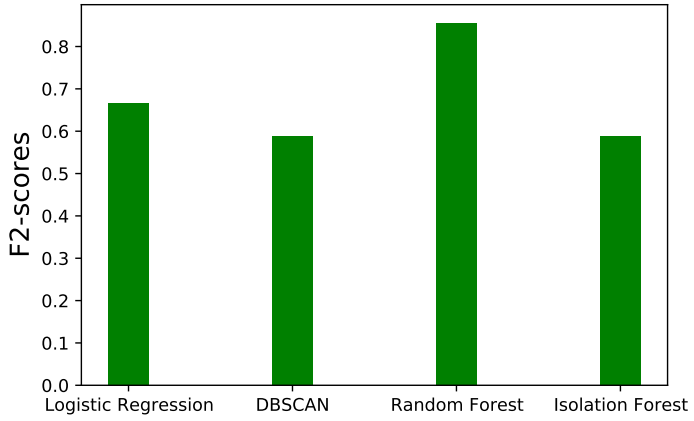

Figure 4.14: F2-scores for the models.

models (except DBSCAN). The ROC curves and AUC values for the Random Forest, iForest, and logistic regression models are shown in figure 4.15. We observe that the RF produces highest score from the figure, whereas logistic regression has the lowest score. We also measured the run time of all these models which is shown in figure 4.29. We notice that logistic regression had the least execution time, while DBSCAN took the longest time to execute. To summarize, as seen by the classification metric results, except for the model run-time, the ensemble random forest model outperformed all other models and had the best classification accuracy.

We performed the experiments on a low computing capacity Fog device and not on high computing Cloud server. Fog device are suitable for processing huge sensor data, as the number of smart meters increases from thousands to millions, the current state-of-the-art centralized data processing architecture will no longer be sustainable with such a big data explosion [**computers9030076**]. For the evaluation of the models, we used a **Fog device** with a capacity of 2 GHz Quad-Core Intel Core i5 processor and 16 GB RAM.

4.3.7 Conclusion and future work

In this paper, we discussed the need of effective power forecasting for anomaly detection to prevent potential power outages and maintain Grid balance. We proposed a effective and simple anomaly detection approach using Facebook

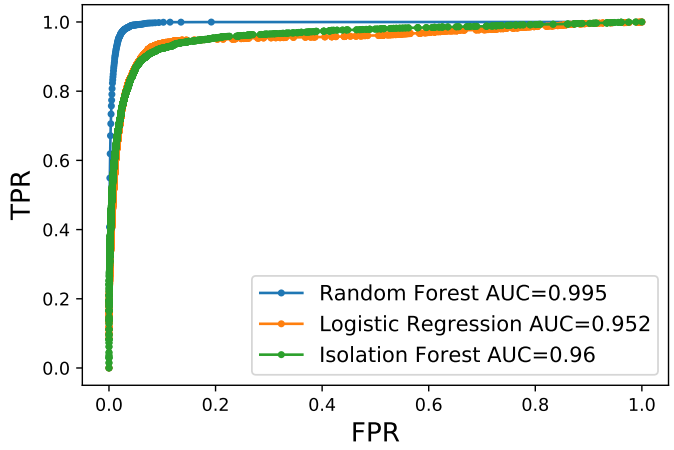

Figure 4.15: ROC curve and AUC values for the models.

Prophet library. The approach is fast, scalable and, the parameters are easily interpretable. This approach is first time used on a real-world power dataset and, anomaly labels are created. We release the anomaly dataset for anyone to use and research. Also, we proposed a machine learning based classification approach to classify power anomalies which achieved a G-mean score of 97.3 percent. For the future work, we plan to utilize and analyze the Ausgrid dataset's solar power generation data since, the houses have the provision of producing clear renewable power through solar panels. We plan to predict the local power produced by the solar panels. with accurate estimate of local power production, the community can sustain better on the power produced and, operate in MicroGrid mode. This can reduce the dependency of the community on the main Grid and help maintain power balance.

4.4 Novel approach for Prediction based Anomaly Detection

4.4.1 Motivation and Prior Art

Survey paper by Zhang et al.[**Zhang**] discuss about Big Data analytics in Smart Grids. They discuss about the applications like fault detection, predictive maintenance, electrical device health monitoring, stability analysis, power quality monitoring, renewable energy forecasting, load forecasting and other areas where

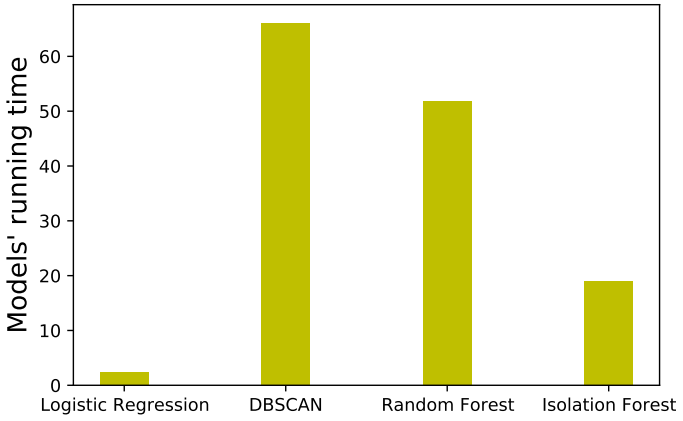

Figure 4.16: Run time on the Fog device for the models.

Big Data analytics is very essential. They discuss about the various machine learning algorithms like supervised, unsupervised algorithms and dimensionality reduction algorithms. Another survey paper by Hare et al.[**Hare**] discuss about fault diagnostics in various components of the Micro Grid. These components are cable and transmission lines, PV panels, wind turbines, fuel cells and, diesel generator systems and engines. They also describe various methods to remove the faults, which are threshold-based, fuzzy logic based, classification based and state estimation based methods. Research work[**Rajasegarar**] by Rajasegarar et al. discuss about anomaly detection in wireless sensor networks. They examine statistical approach, rule based approach, Support Vector Machine based approach, clustering based and density based approach as important methods. Authors in [**Jaradat**] survey about all the smart grid components and propose Big Data management and stream processing techniques for managing load demand, power faults and weather forecasting.

In research by Borthakur et al.[**Borthakur**], Fog Computing is used for real world pathological speech data processing by obtaining data from Smart-watches worn by patients with Parkinson's disease. The Fog device extracts loudness and fundamental frequency features from speech data and processes it with k-means clustering. Even though they use a Fog device but for real-time applications when emergency situation arises, the response will be slower without distributed processing considering the data will be generated at a high speed and will be huge in volume. DeepFog by Rojalina et al.[**Priyadarshini**] propose a Fog based

deep learning model that collects medical data from individuals and predicts the wellness statistics using deep neural network model that can handle heterogeneous and multidimensional data. Even though the proposed solution has good accuracy, neural networks are not suitable for real time applications. They need more data to train and takes more time to process the data, both of which are not suitable for Fog devices. In real situations, physiological parameters like stress and hypertension vary abruptly and recent records should be used for prediction. Therefore, neural networks will not perform well and quick detection of health issues will not be feasible.

Jui-Sheng et al.[**Chou**] use hybrid neural-net ARIMA model for predicting power consumption. The neural-net ARIMA model requires high computation capacity and for that authors have used IBM servers with quad-core processor and 64 GB RAM. Their prediction model has good accuracy but they do not mention the run time of the anomaly detection method to test its efficacy in real-time application and do not address Big Data issues. Another work for detecting anomalies in power consumption comes from authors Jakkula et al.[**Jakkula**]. They use synthetic datasets and inject them with outliers. They use two approaches, first statistical approach with moving window to identify extremes using standard deviation based distance measures and second clustering based approach KNN with Discrete Time Warping for outlier detection.

4.4.2 Power Dataset Details

As discussed in related works section, studies of anomaly detection in power consumption domain used various methods like regression methods, statistical methods, clustering methods, neural network based methods and other methods. Due to the requirement of small training dataset, resource constraints and real-time anomaly detection, we choose to use regression based models that are proved effective by works[**whyRegression**][**ensemble_regression**]. We use an ensemble of four lightweight regression based models to compensate for the deficiencies of a single model. Studies have shown that an ensemble of models reduces the variance and generalization[**ensemble_regression**, **hybridarimann**, **hybrid_models**]. There are several works of ensemble learning that use homogeneous models, heterogeneous models and a combination of both the models for anomaly detection[**SVM_KNN_PSO_ensemble**, **Zhao_ensemble**, **ensemble_amorzegar**, **folino_ensemble**, **anomaly_ensemble**]. Real world time series models have both linear and non-linear components. Therefore, our ensemble method uses both linear and non-linear models as described later in this section.

We use smart meter dataset of individual level household consumption from UCI machine learning repository. The dataset has the following features. Global active power - The minute averaged active power consumed by the household(in KW), Global reactive power - The minute averaged reactive power consumed by the household(in KW), Voltage - The minute averaged voltage consumed by the household(in Volts); Global intensity - The minute averaged current consumed by the household(in Ampere), Sub metering 1 - The minute averaged active energy consumed by the kitchen appliances, containing a dishwasher, an oven and a microwave(in Watt-hour), Sub metering 2 - The minute averaged active energy consumed by laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light(in Watt-hour), Sub metering 3 - The minute averaged active energy consumed by an electric water-heater and an air-conditioner(in Watt-hour). For this work we use one main parameter Global active power for the sake of simplicity and it measures the active power consumed in the house. We use 3 months of data for training the models and predict for future 16 hours using multi-step methodology and test our real-time anomaly detection algorithm. Likewise, we predict for future 24 and 48 hours, and test the efficacy of our method. In Table I, we describe the statistical properties of the dataset:

Mean(KW)	Standard Deviation(KW)	Min(KW)	Max(KW)	Median(KW)
89.191	69.213	14.640	381.832	81.030

Table 4.2: Statistical properties of 3 months hourly data(Global active power)

4.4.3 Machine Learning Approaches

Now we describe about the regression models we are using for our prediction-based anomaly detection:

- Linear regression(LR) : The main idea of Linear Regression is to obtain a line that best fits the data. The best fit line is the one for which total prediction error (for all data points) is as small as possible. Prediction error is the distance between the point to the regression line [**Linear**].
- Support Vector Regression(SVR) : Support Vector Regression is a model where, rather than minimising the error rate as is done in Linear Regression, we try to fit the error within a certain threshold. A kernel function is used to map a lower dimensional data to a higher dimensional data. In our work, we choose a non-linear function which is also called Radial Basis Function (RBF)[**SVR**], as the kernel function.

- Random Forest Regression(RFR) : Random forest regression is an ensemble learning approach capable of performing both regression and classification tasks using multiple decision trees and a effective statistical technique called bagging. For more details of the model please refer [**RFR**].
- Gradient Boosting Regression(GBR) : Gradient boosting is machine learning algorithm widely used for regression, classification and ranking problems. It is a combination of Gradient Descent and Boosting techniques. Gradient boosting builds an additive model in a forward stage-wise fashion. It allows for the optimisation of arbitrary differentiable loss functions. In each stage, a regression tree is fit on the negative gradient of the given loss function[**friedmanGBR**][**GBR**].

4.4.4 Methodology

Our Distributed Fog architecture uses a prediction based approach with two-sigma rule for threshold calculation and majority voting to detect anomalous patterns in household power consumption in real-time. The process is depicted conceptually as shown in Figure 4.17. Below we describe the workflow in detail and later in the section describe the prediction method and the anomaly detection algorithm:

(1) As shown in the Figure 4.17, we first train all the four regression models on 3 months data on the Core Layer. Predictions are made for future 16 hours using multi-step forecasting method with sliding window. We calculate the prediction error as the difference of actual data and the predicted data.

(2) We calculate the mean and standard deviation of the prediction errors. We define threshold as data values greater than two times the standard deviation of the prediction errors. Threshold is defined based on the empirical rule of normal distribution that 95% values lie roughly within 2 times the standard deviation of the mean and therefore, remaining 5% values can be considered as anomalies.

(3) We find thresholds for all the four models. Now we predict for future 16 hours again for all the four models. This prediction is done for the real sensor data that is yet not generated and on which anomaly is to be detected in real-time. These prediction results and the thresholds are now sent to the Edge layer device.

(4) On Edge layer device, we have the predictions for the sensor data that is yet not generated. As data from the sensor is generated, we find out the

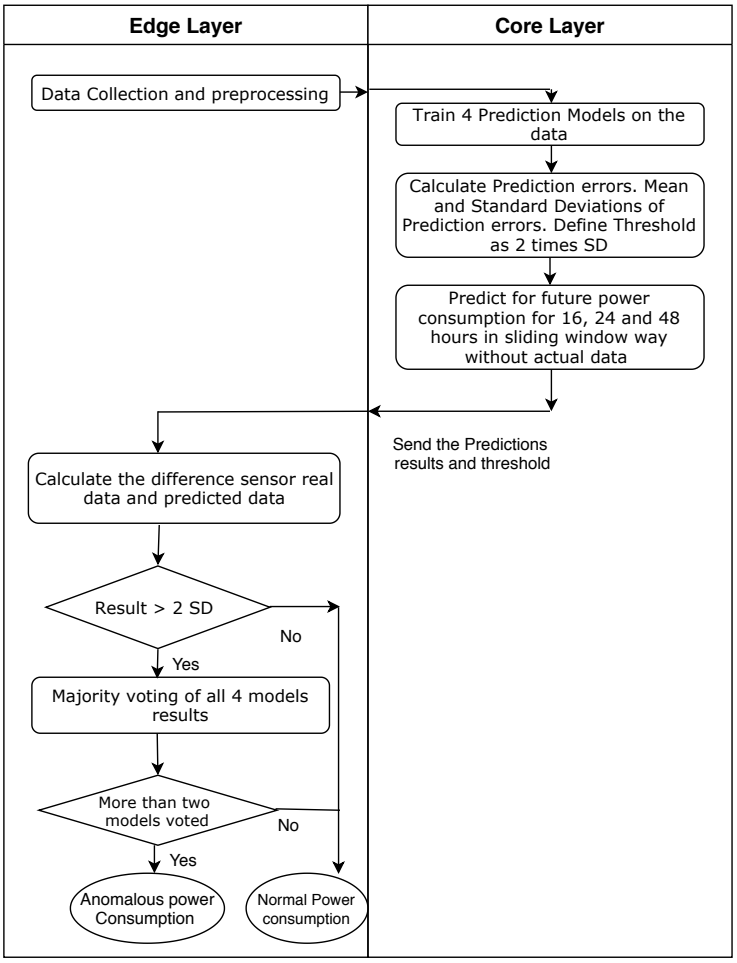

Figure 4.17: Workflow Of the Anomaly Detection process

difference of the actual data and the predicted data and find out the actual values greater than the threshold. These are potential anomalies. Likewise, we find out potential anomalies by all the four models.

(5) To find out actual anomalies and minimize the false alarms, we use majority voting method. When more than 2 models vote for the anomalies values, these anomalies are considered real anomalies.

(6) We calculate the run time for training the four prediction models and the time taken to detect anomalies on the real per hour data. This gives us the estimate of the amount of time taken by model to train and report

anomalies. In actual scenario where there will be more houses and smart meter sensors, we can test the efficacy of our method.

(7) After finding the anomalies on the incoming real sensor data, we raise alarms or take any preventive action. We can repeat the the process to collect the sensor data for a number of hours on a storage device at the Edge layer and keep sending periodically to the Core layer device for prediction and threshold calculation on the latest data. Therefore, we repeat the process from step 1 for finding anomalies on newly generated data.

It is worth mentioning that the threshold on the power consumption is calculated dynamically. The threshold depends on prediction errors, and the actual power consumption which keeps on varying. Therefore, if we use static threshold, it will fail to calculate anomalies in the power consumption at different times. Since we keep updating the models with new data, the thresholds also keeps changing based on the prediction errors. We also predict for future 24 hours and 48 hours and find anomalies in real-time by following the same procedure from step 1.

Algorithm 1 Real-Time Anomaly Detection

UnionUnion FindCompressFindCompress InputInput OutputOutput Predicted data for 4 models $S_p(i)$, Threshold for 4 models Th_p, Actual sensor data S_a Anomaly A_a All models $E_a(i)$ = $|S_p(i)$ -$S_a(i)| > Th_p(i)$; Find $A_a = Find$ $\cap$E$_a(i)$ for all combinations of 3 different models
A_a

Here $S_p(i)$ is the predicted data by our four regression models. S_a is the actual sensor data for that particular hour. Th_p is the threshold values that are greater than 2 times Standard Deviation of prediction errors. Four models generate four thresholds values. $E_a(i)$ are the potential anomalies greater than the threshold for all four models. A_a is the actual anomaly values voted by three or more models. Likewise, we repeat the same steps for finding anomalies for 24 hours and 48 hours sensor data.

4.4.5 Experimental Results and Discussion

The predictions for future 16, 24 and 48 hours are done using four models. We plot the prediction plots for all the models against the actual values in Figure **??**. The prediction accuracy is calculated with RMSE scores as they are more punishing to forecast errors as compared to MAPE or MAE. RMSE scores for

all the models are shown in Figure **??**. We predict recursively multiple hours based on previous predicted values as actual data is not generated. The higher prediction errors are compensated with the ensemble technique. The RMSE scores vary at different hours of the day. It can be seen that almost all the models give a high RMSE error at hour 19, 21 and 43 with highest error at hour 21. The average RMSE scores for all the regression models for prediction of future 16, 24 and 48 hours is shown in Table II. The average RMSE errors for all the models shows that Support Vector Regression performs better for predicting 16 hours while Random Forest Regression performs better for 24 hours prediction and Linear Regression performs better when predicting for future 48 hours. Figure **??** shows the anomalies detected by the anomaly detection algorithm. We also evaluate time taken to train models by plotting a bar graph as shown in Figure **??**.

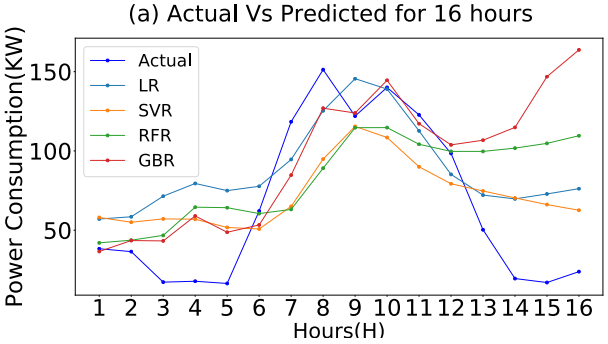

Figure 4.18: Run time on the Fog device for the models.

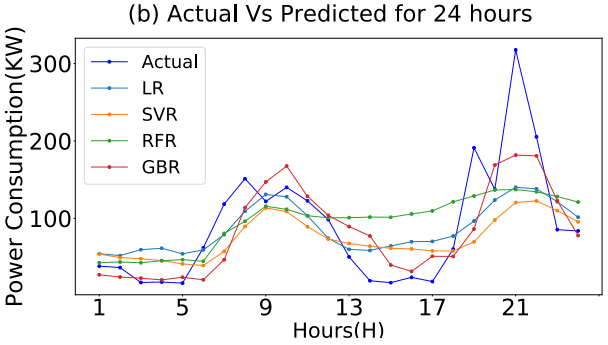

Figure 4.19: Run time on the Fog device for the models.

We perform model validation and interpret that Linear Regression, Gradient Boosting Regression performed better compared to the other two models. Based on two sigma approach, thresholds are defined for each model which are shown in

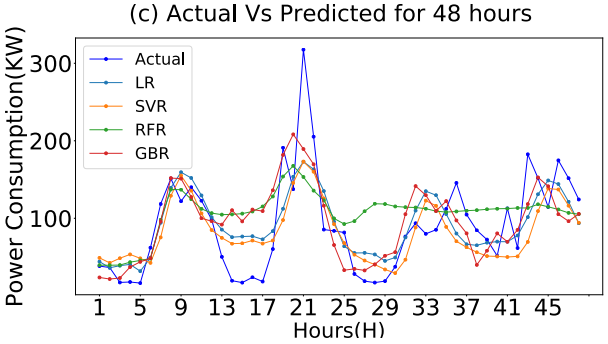

Figure 4.20: Run time on the Fog device for the models.

Table III. Anomalies for different models are calculated based on these thresholds.

We do a majority voting by three or more models on the potential anomalies reported by the models. The final anomalies are plotted in Figure 5. Different models report anomaly indexes based on threshold. For example, considering prediction for future 48 hours, Linear Regression reported indexes [18, 20, 35, 42] as anomalies. Support Vector Regression reported indexes [18, 20, 42], Random Forest Regression reports [13, 14, 15, 16, 20, 26, 27, 28, 29] indexes and Gradient Boosting Regression reported [13, 16, 20, 45] indexes as anomalies. By Doing a majority voting of three or more than three models, we conclude, index 20th as anomalous. This is the power consumption at hour 21 and it is raised as an anomaly and plotted in Figure 5. The plots show that there is no anomaly in the first 16 hours, there is an anomaly at hour 21 and then till 48 hours there is no anomaly. We plot a bar graph comparing run time of different models. They are shown in Figure **??**. It can be seen that Linear Regression is the most time efficient while Gradient Boosting Regression takes the most time for training.

We measure the run time of the anomaly detection algorithm and it comes out 126 milliseconds. This ensures real-time performance for anomaly detection. We use Core layer device with hardware capacity of 2.3GHz dual-core Intel Core i5, 64-bit processor and 8GB RAM and for the Edge layer also we use the same configurations.

4.4.6 Conclusions

In this paper, we proposed a Distributed Fog Computing architecture for Big Data analytics in Smart Grids for finding anomalous power consumption at

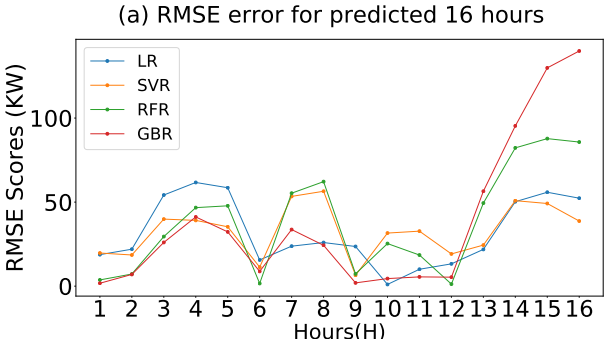

Figure 4.21: Run time on the Fog device for the models.

household level. This architecture is more effective and efficient as compared to Cloud computing architecture because of Big Data, location awareness and low latency requirements. We used an ensemble of prediction based models, dynamic thresholds and majority voting technique for finding anomalies in actual sensor data in real-time. We used machine learning algorithms for predictions. Then threshold was defined based on two sigma approach on prediction errors. Ensemble technique of majority voting was successfully used to reduce the false anomalies which were reported by individual models. The performance of our approach for detecting anomalies indicates its substantial potential as a practical method to be used for future Smart Grid monitoring and real-time anomaly detection.

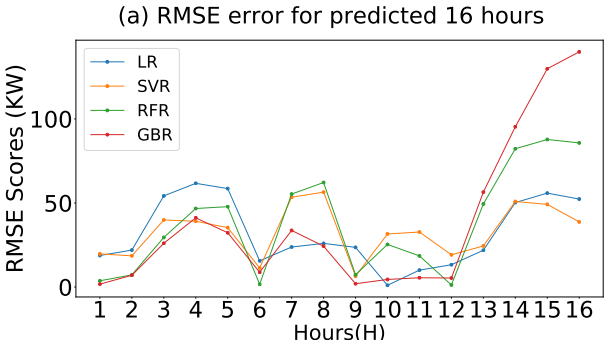

Figure 4.22: Run time on the Fog device for the models.

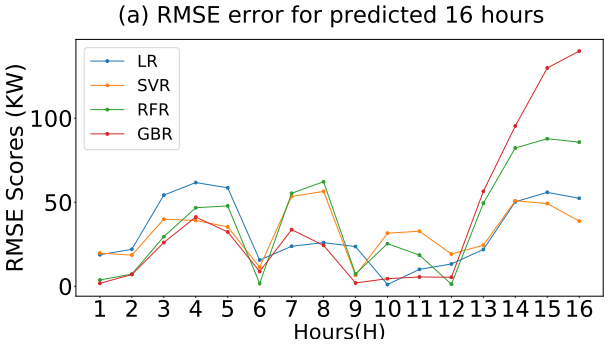

Figure 4.23: Run time on the Fog device for the models.

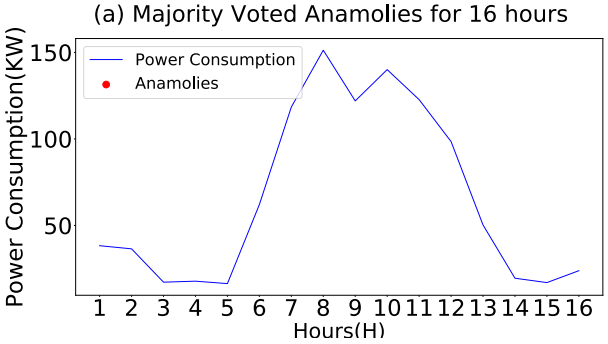

Figure 4.24: Run time on the Fog device for the models.

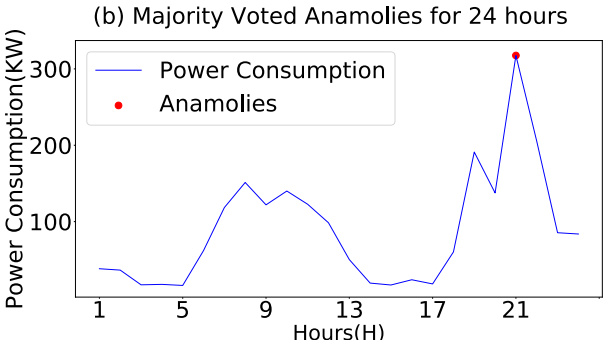

Figure 4.25: Run time on the Fog device for the models.

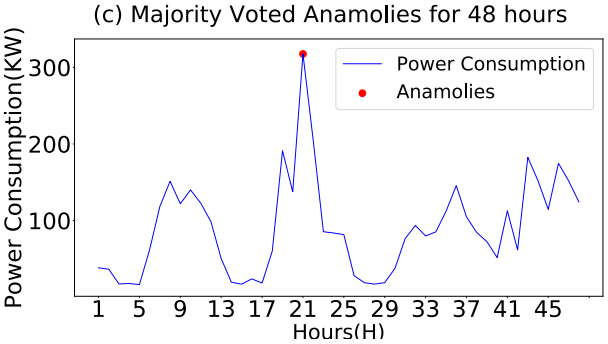

Figure 4.26: Run time on the Fog device for the models.

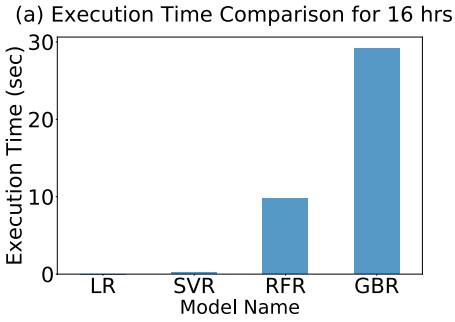

Figure 4.27: Run time on the Fog device for the models.

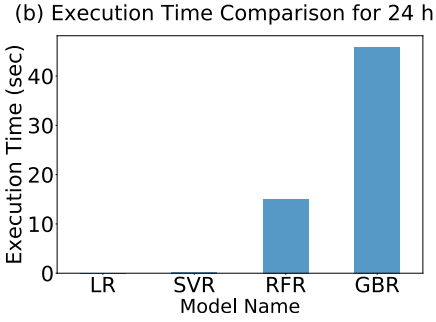

Figure 4.28: Run time on the Fog device for the models.

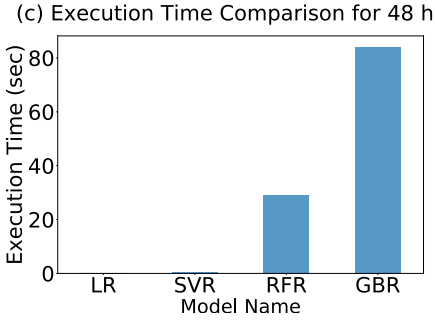

Figure 4.29: Run time on the Fog device for the models.

Table 4.3: Average RMSE scores for different models(KW)

PredictionHours	LinearRegression	SupportVectorRegression	RandomForestRegression	GradientBoostingRegression
16 Hours	37.24	36.10	48.50	52.24
24 Hours	53.43	56.32	50.99	79.20
48 Hours	49.90	59.81	63.32	60.31

Table 4.4: Prediction based Anomaly thresholds for different models(KW)

PredictionHours	LinearRegression	SupportVectorRegression	RandomForestRegression	GradientBoostingRegression
16 Hours	83.35	107.59	66.86	92.85
24 Hours	66.63	72.61	63.41	99.56
48 Hours	63.71	77.72	71.62	73.57

Chapter 5

Deep Learning based Power forecasting techniques for achieving Power Balance

5.1 Why Deep Learning Techniques?

5.2 Methodology

We performed data pre-processing, model development and selection of the most accurate models for forecasting. In the data pre-processing step, we did visual exploration of the data, data cleaning, feature extraction, feature transformation and feature selection. Thereafter, we did input/output modelling for final feeding into our selected algorithms. Visual examination of the data found seasonality which were daily, weekly and yearly. During winter months, the power consumption was high compared to the summer. We found various missing values in the data. These missing values could be due to various reasons like smart meter sensor fault or other data transmission issues. The missing values were filled using exponential weighted moving average (EWMA) technique. This method takes the arithmetic mean of the data surrounding the missing data value while giving emphasis on the latest data, as the present data is dependent on the previous latest data.

For input/output modelling, we took latest twenty four hours data as input features and the next hour data as the output feature due to the weekly seasonality of the

data and moved the time step to t+1, and, repeated the process for the whole data. For transformation of the data, minmax scalar function was used as power data values had different scales. This transformed the data values of each input and output variable in between the range of 0 to 1. For training, two years data ranging from from 01/01/2007 to 31/12/2008, is used. For testing, a data set with date range from 01/01/2009 to 31/01/2009, which is one month's data is used.

Now, we discuss the machine learning models with the best accuracy used for forecasting :

- **Support Vector Regression :** Support Vector Regression is a model where rather than minimizing the error rate as is done in Linear Regression, we try to fit the error within a certain threshold. A kernel function is used to map a lower dimensional data to a higher dimensional data. In our work, we choose a non-linear function which is also called the Radial Basis Function(RBF), as the kernel function. We used GridSearchCV for hyper-parameter optimization. We found out the values of C=100, gamma=0.01 and epsilon=0.1 as the optimal hyper-parameters for tuning our model.
- **Artificial Neural Network :** An Artificial neural network or ANN is system of processing units which are called neurons that are linked together in different ways to learn the non-linear patterns in the data that the classical machine learning algorithms can not detect or, it is too expensive to detect such patterns [**ann**]. We used 1 hidden layer in our network. The output of neurons in the hidden layer is determined by activation function. We used "relu" as the activation function in the input and hidden layers. The number of neurons in the input layer is 10, number of neurons in hidden layer as 20. At the output layer, we use "adam" optimizer to minimize the weight and bias learning error.
- **Recurrent Neural Network :** Recurrent Neural Network or RNN is an advanced form of ANN with a capability to learn from previous weight computations [**rnn**]. RNN layer uses a for loop to iterate over the timesteps of a sequence, while also maintaining an internal state that encodes the information about the timesteps it has seen so far. We used SimpleRNN function from the tensorflow keras library. 1 hidden layer is used. "relu" activation function is used in both input and hidden layer. The number of neurons at the input and hidden layer is 32 and 8 respectively. At the output layer "adam" optimizer is used to minimize the learning error.
- **Long Short Term Memory :** A Long Short Term memory or LSTM is an advanced form of RNN where it learns long term dependencies along with

Table 5.1: System Specifications of Cloud and Fog devices for running prediction models

eviceMemory (RAM)ProcessorFog 12 GB 2.70 GHz dual-core Intel core i5 64-bitCloud 25 GB 2.30 GHz

short term dependencies of RNN [**lstm**]. Moreover, unlike RNNs, LSTM resolve the disadvantage of vanishing and exploding gradients. Here we use one hidden layer with 216 neurons. The activation function used in hidden layer is "relu" and, the optimizer used at the output layer is "adam" optimizer.

5.3 Experimental Results

In this section, we discuss the forecasting results on Fog and Cloud platforms, their memory consumption and CPU runtime. Figure 2,3, 4 and 5 shows the comparison between real and predicted power consumption for the four models we chose. The visual inspection shows that the predictions results are good and our models are trained well to capture data patterns.

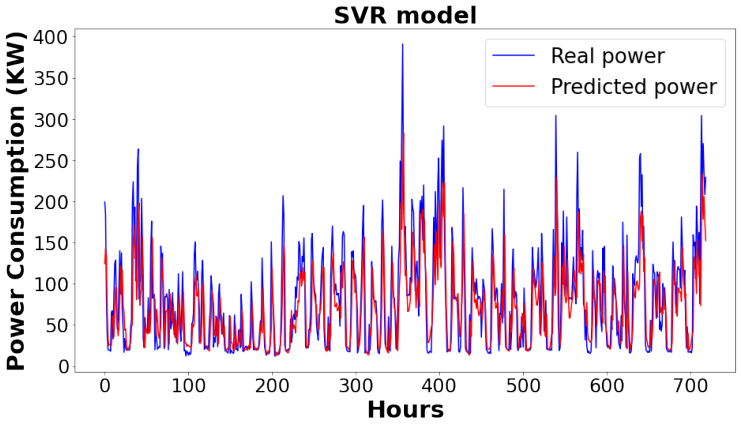

Figure 5.1: Real vs predicted power consumption using SVR model.

All the models have been implemented using the scikit-learn open source library for Machine Learning and Keras library for Deep Learning model training. The predictions were performed on Fog and Cloud devices. The RMSE scores of the prediction results models is shown in Table 2. The results shows that ANN model performs best with the RMSE score of 39.50. All the other models also have almost same low RMSE scores. The SVR model has the least performance among all. We measured the training loss of Deep neural networks as follows,

Table 5.2: Root Mean Square Error scores of the prediction models.

Model	RMSE Scores (KW)
Support Vector Regression	40.89
Artificial Neural Network	39.50
Recurrent Neural Network	39.58
Long Short Term Memory	40.80

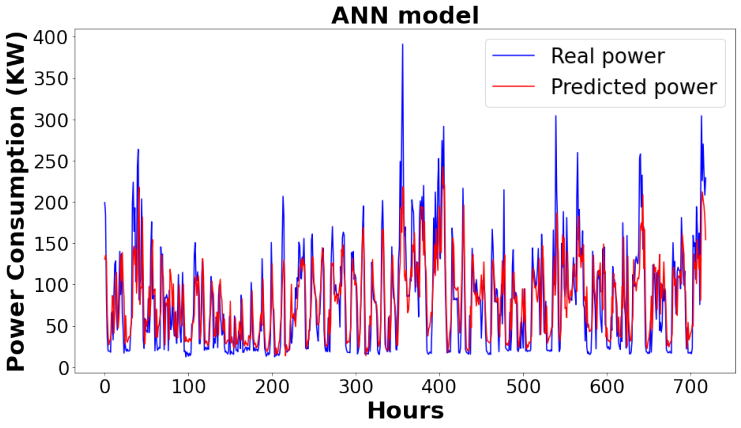

Figure 5.2: Real vs predicted power consumption using ANN model.

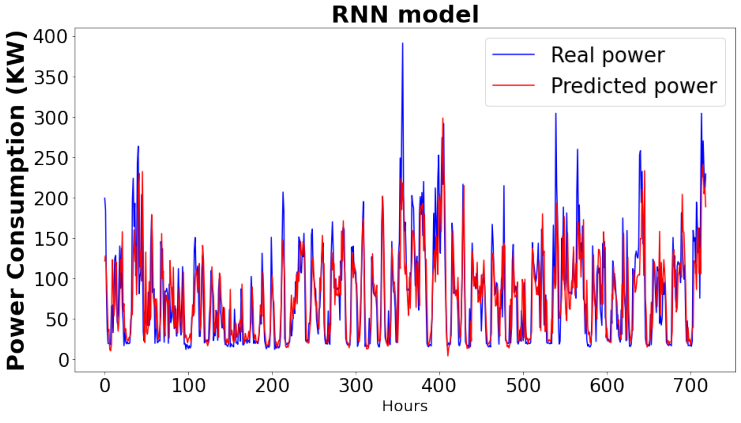

Figure 5.3: Real vs predicted power consumption using RNN model.

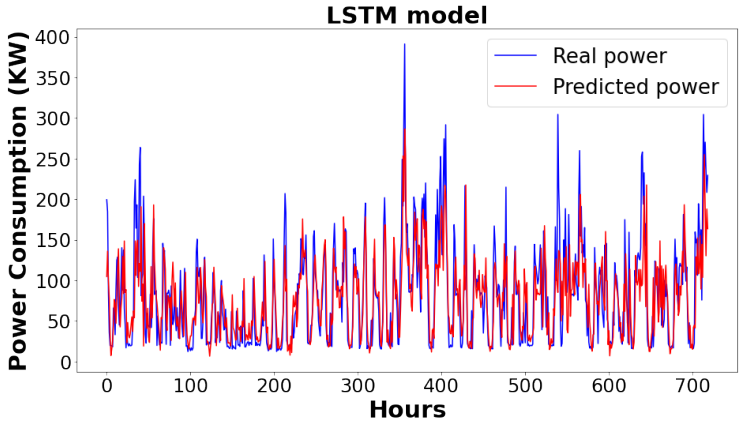

Figure 5.4: Real vs predicted power consumption using LSTM model.

ANN: 0.0091, RNN: 0.0084 and LSTM: 0.0079. This shows that the training loss of LSTM is the lowest and, it performed slightly better compared to RNN and ANN.

In Figure 6 and 7, we have shown the CPU runtime for the application on Fog and Cloud environments. Here Cloud performs better than Fog but we did not measured the application latency of actual data transfer through the communication networks from the sensor networks to the Cloud server. To measure the communication delay of sending the sensor data on the Cloud and getting results back to the access network, we should consider the network bandwidth of the Core networks which is typically 1 Gbps. Hence, to transport the smart meter data of size 1000 Terabytes from sensor network to the Cloud will take around 101 Days 19 Hours, 21 Minutes. This is clearly a very high latency and therefore, not practical for the Grid application. The data produced by sensors will continue to grow exponentially as already estimated by various companies. Despite the high processing capacity of Cloud servers, the network bandwidth is adding high latency that can be avoided with Fog processing.

The task of using of Fog Computing to compliment Cloud Computing is still challenging. The network infrastructure of Fog Computing is still developing for handling the huge big data and its applications. These applications need ultra low latency, dynamic scalability, and efficient in-network processing to provide the services. Fog computing promises the computing, networking and storage capabilities anywhere between the Edge and Cloud continuum. The Fog network even though hierarchical and geographically distributed in nature, requires soft-

Table 5.3: Root Mean Square Error scores of the prediction models.

Model	RMSE Scores (KW)
Support Vector Regression	40.89
Artificial Neural Network	39.50
Recurrent Neural Network	39.58
Long Short Term Memory	40.80

ware models in addition to network architecture to be built to compliment the Cloud. With high performance Fog devices using Tensor processing units, high speed 5G networks, and big tech companies like Nvidia, Google and Qualcomm producing high performance edge devices, we will soon be able to elevate Fog to become next generation Cloud Computing.

Fog and Cloud are interdependent ideally support each other to fulfill the application needs. We suggest that city level pre-analysed data with higher level representation could be send for further data analysis on the Cloud for large city scale decision making.

5.4 Discussion, Conclusion and future work

In Figure 6 and 7, we have shown the CPU runtime for the application on Fog and Cloud environments. Here Cloud performs better than Fog but we did not measured the application latency of actual data transfer through the communication networks from the sensor networks to the Cloud server. To measure the communication delay of sending the sensor data on the Cloud and getting results back to the access network, we should consider the network bandwidth of the Core networks which is typically 1 Gbps. Hence, to transport the smart meter data of size 1000 Terabytes from sensor network to the Cloud will take around 101 Days 19 Hours, 21 Minutes. This is clearly a very high latency and therefore, not practical for the Grid application. The data produced by sensors will continue to grow exponentially as already estimated by various companies. Despite the high processing capacity of Cloud servers, the network bandwidth is adding high latency that can be avoided with Fog processing.

The task of using of Fog Computing to compliment Cloud Computing is still challenging. The network infrastructure of Fog Computing is still developing for handling the huge big data and its applications. These applications need ultra low latency, dynamic scalability, and efficient in-network processing to provide

the services. Fog computing promises the computing, networking and storage capabilities anywhere between the Edge and Cloud continuum. The Fog network even though hierarchical and geographically distributed in nature, requires software models in addition to network architecture to be built to compliment the Cloud. With high performance Fog devices using Tensor processing units, high speed 5G networks, and big tech companies like Nvidia, Google and Qualcomm producing high performance edge devices, we will soon be able to elevate Fog to become next generation Cloud Computing.

Fog and Cloud are interdependent ideally support each other to fulfill the application needs. We suggest that city level pre-analysed data with higher level representation could be send for further data analysis on the Cloud for large city scale decision making.

In this paper, we discussed the dynamic nature of the Smart Grid and the need for real-time power forecasting for the Grid balance. We proposed a Fog Computing architecture for real-time power consumption forecasting using smart meter data. Fog devices are located at the edge of the network and therefore, sensor data do not need to be transferred from the access network to the Cloud servers via the core network. As the sensor data is increasing exponentially, it becomes a huge performance bottleneck for critical smart grid applications and, therefore, can lead to power hazards. We discussed that processing sensor data locally also leads to less energy consumption with Fog Computing. We used four efficient performing machine learning algorithms for power consumption forecasting. The prediction results shows that artificial neural networks performs best among all the models. The prediction task was performed on both Fog and Cloud environments. We measured the CPU run time and memory consumption on Fog and Cloud devices and showed the efficacy of Fog processing.

To the best of our knowledge, in smart grid domain our work is the first work where we concretely apply machine learning for power consumption forecasting on real world smart meter sensor data in Fog environment and, measure the memory and CPU runtime of the forecasting application. We also compare these performance metrics with the analysis on the Cloud platform.

For future work, we plan to measure the network latency of transferring large Smart Grid sensor data to the Cloud server and, measure the actual latency of a forecasting application. We plan to measure the actual energy consumption for this process. We plan to use Nvidia Jetson Nano and Google Coral with Tensor Processing Units Fog device for analysing smart meter data. Finally, we plan to the analyse smart meter data for a big community with Terabytes size.

Chapter 6

Generative Adversarial Networks based Techniques for Anomaly Detection in Smart Meter data

6.1 Motivation and Prior Art

As per Yann LeCun(ACM Turing Award Laurate), Adversarial training(also called GAN for Generative Adversarial Networks), and the variations that are now being proposed, is the most interesting idea in the last 10 years in the field of Machine Learning. GANs are generative models that generate samples similar to the training dataset by learning the true data distribution. It consists of a Generator network and a Discriminator network, which learn a data distribution through a two-player game. The generator tries to capture the distribution of some arbitrary data, while the discriminator tries to estimate the probability of some data-sample originating from the training data instead of the generator. When the generator is able to fool the discriminator about half of the time, we know it is generating plausible examples that the discriminator fails to distinguish from training data. GANs can be used for a multitude of tasks, examples including generation of songs, art or even photo-realistic images of people that is indistinguishable from real human-made work.

For Anomaly detection, we use GAN as follows. The generator G that produces

fake samples. The discriminator D that that receives samples from both G and the dataset. During training the two networks have competing goals. The generator tries to fool the discriminator by outputting values that resemble real data and the discriminator tries to become better at distinguishing between the real and fake data.

Mathematically, this means that the Generator's weights are optimized to maximize the probability that fake data is classified as belonging to the real data. The discriminators's weights are optimized to maximize the probability that the real input data is classified as real while minimizing the probability of fake input data being classified as real. Optimality is reached when the generator produces an output that the discriminator cannot concretely label as real or fake and this, happens when either of the networks cannot improve anymore.

In the first formulation of GAN training Goodfellow et al., the loss function quantified the Jenson-Shanon divergence between the training data distribution Pr and the generator sample distribution Pg defined by x=G(z), with z p(z). This formulation suffers from training instability and is prone to mode collapse, and thus GANs were hard to train. Prior GAN-related work has rarely involved time series, because of the complex temporal correlations within this type of data having multiple seasonalities pose challenges for generative modelling. First, to use GANs for anomaly detection in time series, Li et al. [7] propose a vanilla GAN model to capture the distribution of a multivariate time series, and using the Critic to detect anomalies. In BeatGAN [31], a Encoder and Decoder GAN architecture is used that allows for the use of the reconstruction error for anomaly detection in the heartbeat signals. Yoon et al. [14] propose a time series GAN which uses temporal embeddings to assist network training. However, their work is designed for time series representation learning instead of anomaly detection. To the best of our knowledge, we are the first to introduce cycle-consistent GAN architectures for time series data, such that Generators can be directly used for time series reconstructions. In addition, we systematically investigate how to utilize Critic and Generator outputs for anomaly score computation. we propose a simple and complete framework for time series anomaly detection with GANs.

6.2 Methodology

Bidirectional LSTM. Bidirectional LSTMs (biLSTM) [15] is a sequence processing model that consists of two LSTMs. One LSTM handles the input in a forward direction and one LSTM handles it in a backwards direction. This in turn increases the amount of information that is available to the network at a

given time. For example knowing from a given sentence what word will follow and whats words that will precede that word. The final architecture has a design as shown in figure 6.1

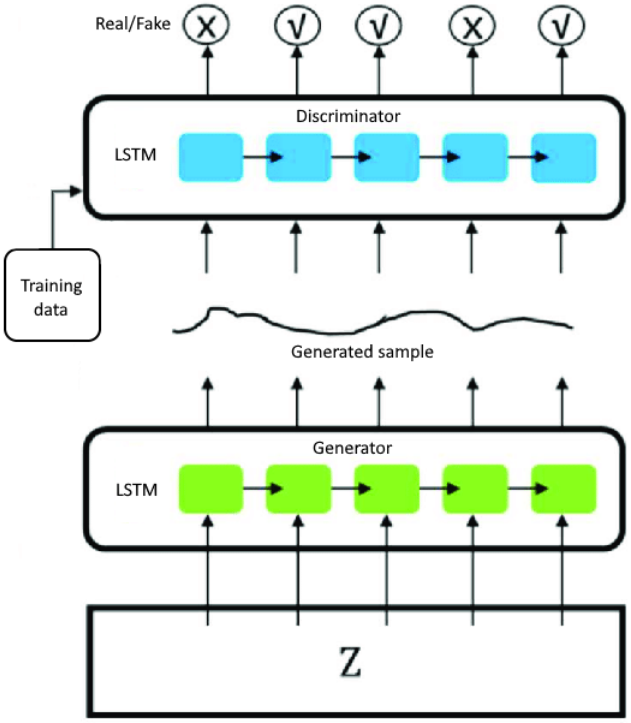

Figure 6.1: GAN based anomaly detection architecture.

There were three main parameters to be improved upon regarding the BiLSTM GAN implementation; learning-rate, number of epochs and batch size. A simple grid-search implementation crudely found the best parameters by extensively testing the network with different combinations of parameters. Each parameter combination is used to train a GAN before subsequently detecting anomalies using the trained discriminator. The resulting predictions are used to evaluate the GAN, where mainly the accuracy score is checked. When this rather time consuming function has completed running, the best parameters are returned. The Ausgrid Solar Home Electricity Dataset is rather large, with data spanning several years. While conducting the work for this project, initial tests where made using data from small time periods, and when satisfying results were reached testing on larger time periods were conducted. By doing this, time spent on waiting for

models to train in the earlier stages of the project could be cut drastically.

6.3 BiLSTM Network

Keras has been used to implement the BiLSTM network. The generator and discriminator of the GAN has been implemented slightly differently, although they are both based on BiLSTM. Figure 6.2 shows the types of layers and their output shapes used to implement the discriminator.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_6 (Bidirection (None, 8, 256)            133120
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 8, 256)            0
_________________________________________________________________
bidirectional_7 (Bidirection (None, 256)               394240
_________________________________________________________________
leaky_re_lu_10 (LeakyReLU)   (None, 256)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
repeat_vector_1 (RepeatVecto (None, 1, 256)            0
_________________________________________________________________
time_distributed_3 (TimeDist (None, 1, 128)            32896
_________________________________________________________________
leaky_re_lu_11 (LeakyReLU)   (None, 1, 128)            0
_________________________________________________________________
dropout_4 (Dropout)          (None, 1, 128)            0
_________________________________________________________________
time_distributed_4 (TimeDist (None, 1, 128)            16512
_________________________________________________________________
leaky_re_lu_12 (LeakyReLU)   (None, 1, 128)            0
_________________________________________________________________
dropout_5 (Dropout)          (None, 1, 128)            0
_________________________________________________________________
time_distributed_5 (TimeDist (None, 1, 1)              129
=================================================================
Total params: 576,897
Trainable params: 576,897
Non-trainable params: 0
```

Figure 6.2: Discriminator Model

Figure 6.3 shows how the generator has been implemented regarding to the types of layers and their output shapes.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional (Bidirectional (None, 8, 256)            133120
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 8, 256)            0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 8, 256)            394240
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 8, 256)            0
_________________________________________________________________
bidirectional_2 (Bidirection (None, 256)               394240
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 256)               0
_________________________________________________________________
repeat_vector (RepeatVector) (None, 8, 256)            0
_________________________________________________________________
bidirectional_3 (Bidirection (None, 8, 256)            394240
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 8, 256)            0
_________________________________________________________________
bidirectional_4 (Bidirection (None, 8, 256)            394240
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 8, 256)            0
_________________________________________________________________
bidirectional_5 (Bidirection (None, 8, 256)            394240
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 8, 256)            0
_________________________________________________________________
dropout (Dropout)            (None, 8, 256)            0
_________________________________________________________________
time_distributed (TimeDistri (None, 8, 128)            32896
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 8, 128)            0
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 128)            0
_________________________________________________________________
time_distributed_1 (TimeDist (None, 8, 128)            16512
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 8, 128)            0
_________________________________________________________________
dropout_2 (Dropout)          (None, 8, 128)            0
_________________________________________________________________
time_distributed_2 (TimeDist (None, 8, 1)              129
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 8, 1)              0
=================================================================
Total params: 2,153,857
Trainable params: 2,153,857
Non-trainable params: 0
```

Figure 6.3: Generator Model

6.4 Experimental Results

Results from experimentation have been split into results from running tests on a single day of data capture, and from running tests on an entire week. Everything shown is from a single household from the Ausgrid dataset.

The following is the results of training the model using data from one day with

a batch size of *128*, a learning rate of *0.0001* through *24* epochs. Running the training with these parameters took 2 minutes and 35 seconds.

When the GAN has finished training using the above parameters we can plot the resulting loss of the two models, and by comparing the loss of the discriminator to the loss of the generator we can see that they are slowly converging towards an optimal state.

6.4.1 One day data

Figure 6.4: Training Loss

By storing the generated data from each epoch, we can make plots showcasing the evolution of the generated data. Figures 6.5, 6.6, 6.7 and 6.8 below shows how the generator is adapting to the training data throughout epochs.

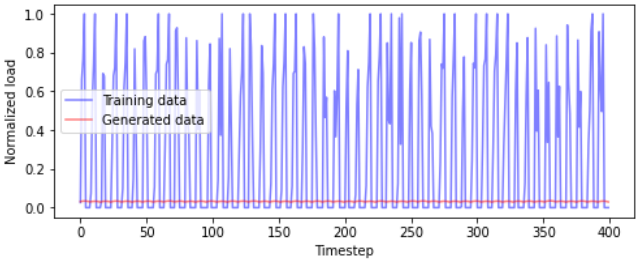

Figure 6.5: Epoch 3

As we can see from the above figures, it is possible to follow how the generator early on makes minor adjustments to its generated data, to an overshoot in the

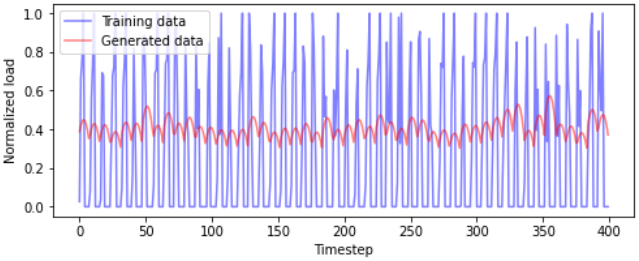

Figure 6.6: Epoch 12

middle epochs, before later presenting a closer approximation to the training data.

Next by running the anomaly detection using the trained discriminator we get a resulting set of predicted y-values, i.e. the predicted anomalies. This is a list of boolean values where a 0 represent good measurements and 1 represent anomalies. By evaluating the predicted y-values compared to the true y-values we get an accuracy of *90.27* percent.

To further visualize the results of the anomaly detection we can plot the time-series data and overlay the true anomalies on one plot and the predicted anomalies on another. By doing this it becomes simple to see if the predicted results are satisfying.

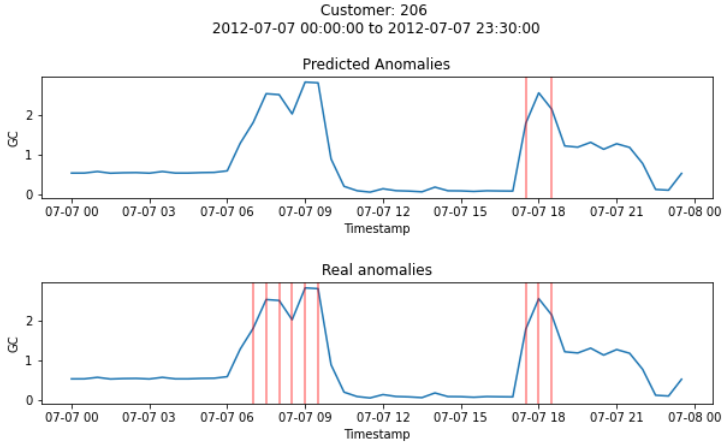

Figure 6.9: Time series data with predicted vs real anomalies for a given customer, spanning one day

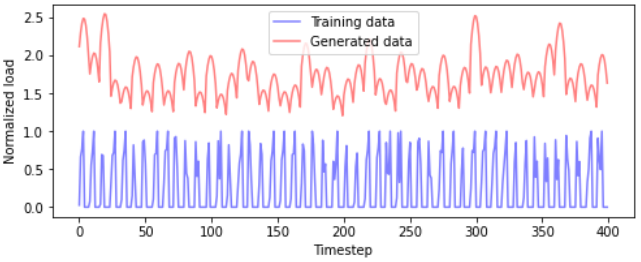

Figure 6.7: Epoch 15

6.4.2 One week training results

Running the same experiments as above for data from an entire week yielded results with an accuracy of 90.27 percent, which is exactly the same as when running for a single day. The following plots show the

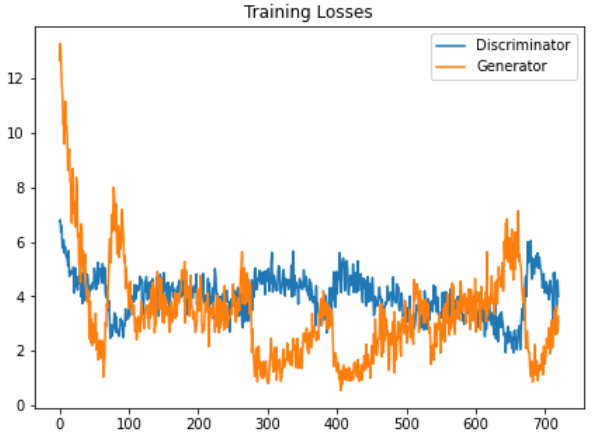

Figure 6.10: Training Loss

By plotting the loss after training on the week long time series we see that both the loss of the discriminator and the generator has a different pattern than before where it does not converge as satisfyingly as it did on the one-day data.

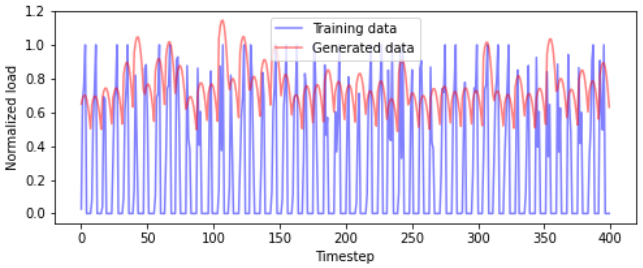

Figure 6.8: Epoch 20

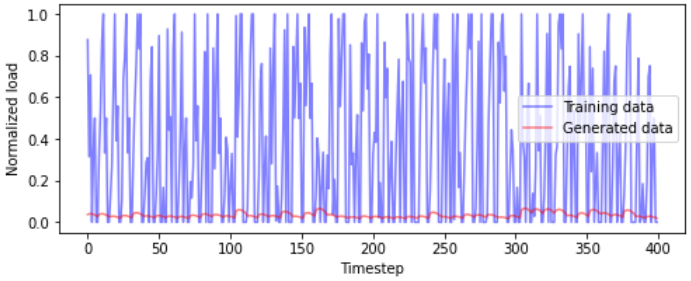

Figure 6.11: Epoch 5

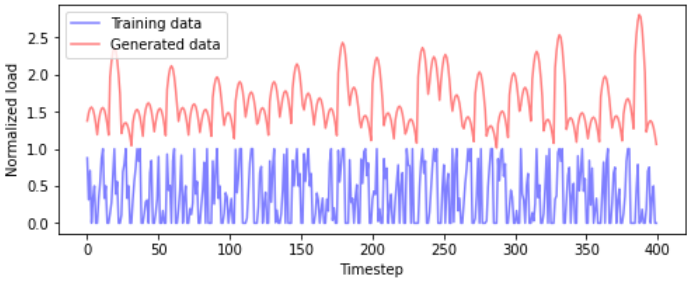

Figure 6.12: Epoch 15

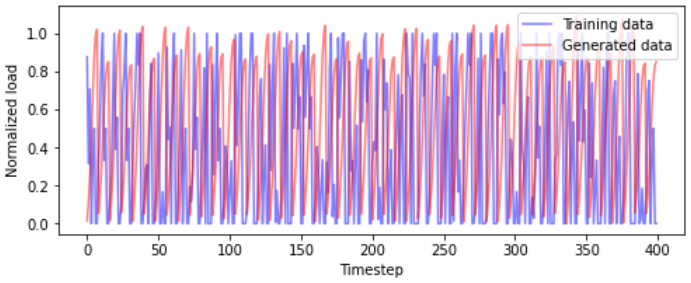

Figure 6.13: Epoch 24

Figures 6.11, 6.12 and 6.13 shows a big change from the one-day dataset. Now it can be observed that at the final epoch, the generator is producing data that is quite similar to the training data. Figure 6.14 shows the rolling average of the generated data, and it can clearly be seen to slowly converge towards the training data.

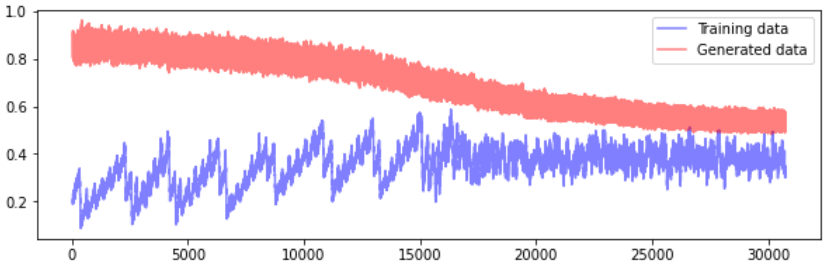

Figure 6.14: Rolling average of generated data

Figure 6.15 shows the result of the anomaly detection using the trained discriminator. It can be seen that a number of anomalies are not being picked up by the anomaly detection, although the ones that make the cut are correct classifications.

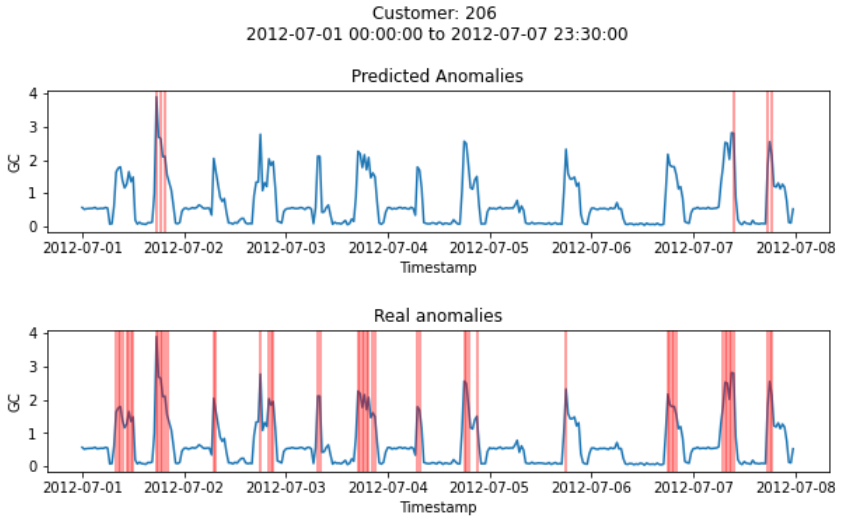

Figure 6.15: Time series data with predicted vs real anomalies for a given customer, spanning one week.

6.5 Conclusions and Further Work

We saw that by utilizing generative adversarial networks in conjunction with bidirectional Long Short-Term Memory networks, we could indeed generate fake electricity data that could fool a discriminator trained on actual electricity data. By taking advantage of this we could build an anomaly detection algorithm capable of locating anomalies with an accuracy of 90.27 percent.

To further improve the work done in this project, there are several things that could be done.

(1) More hyperparameter testing.

- Because of time restrictions as well as hardware restrictions, I was simply not capable of performing satisfying tests regarding discovering the optimal set of hyperparameters. I believe that the results outlined in the experimental evaluation section could see improvements given enough time dedicated to finding better parameters.

(2) Testing more networks

- I managed to implement two different types of networks, a multilayer perceptron and a bidirectional LSTM. Other networks should be

tested to see if the anomaly detection results can be improved, for example with an encoder-decoder architecture.

(3) Testing with more data

- For this project, again because of time and hardware restrictions, I was only capable of testing with small amounts of data. Future work should attempt to perform anomaly detection on the entirety of the labeled dataset used while developing the model. When the model has been proven to give good results on the entire dataset, it will be feasible to perform anomaly detection on actual unlabeled data. This could produce interesting results.

Chapter 7

Fog Computing for Real-Time Data Analytics in Smart Grid

7.1 Motivation and Prior Art

The main challenge is to build a scalable and secure distributed architecture for real-time data processing for reliable smart grid operations. We need a mechanism to store and process data near to where it is produced. An architecture is proposed recently as Fog Computing architecture which is promising for distributed real-time, scalable and securing & privacy preserving IoT applications. The term "Fog" was first introduced by Cisco [**1_arch1**]. Fog is like a cloud near the ground, so this term aptly described cloud services near sensors networks.

We discuss in detail the challenges of smart grids and how Fog Computing is a promising solution for realizing Smart Neighborhoods. To summarize, Fog Computing is crucial in Smart Grid domain to ensure:

- Real-time control and monitoring of smart grid operations.
- Scalable and distributed smart grid architecture.
- Security and privacy-preserving smart grid operations.
- Network bandwidth and traffic management.
- Reliable smart grid operations and power outage prevention.
- Effective distributed power generation and RES integration.

- Optimized smart grid operations and minimized losses.

- Effective MicroGrid operations and V2G integration.

Together with the support of Cloud Computing, applications require both fog localization and Cloud centralization. Fog Computing plays a vital role in at least the following fields: smart grids, smart healthcare, smart manufacturing, smart connected vehicles and smart agriculture. Fog Computing provides proximity and location awareness, geo-distribution, and hierarchical organization, which makes it a suitable platform for resource constraint wireless sensor and actuator networks.

Figure 7.1 shows Fog Computing application domains which are smart grids, smart cities, smart healthcare, smart manufacturing, smart connected vehicles, smart agriculture and, the high-level architecture of Fog Computing. In Table 7.1, we compare Cloud and Fog Computing.

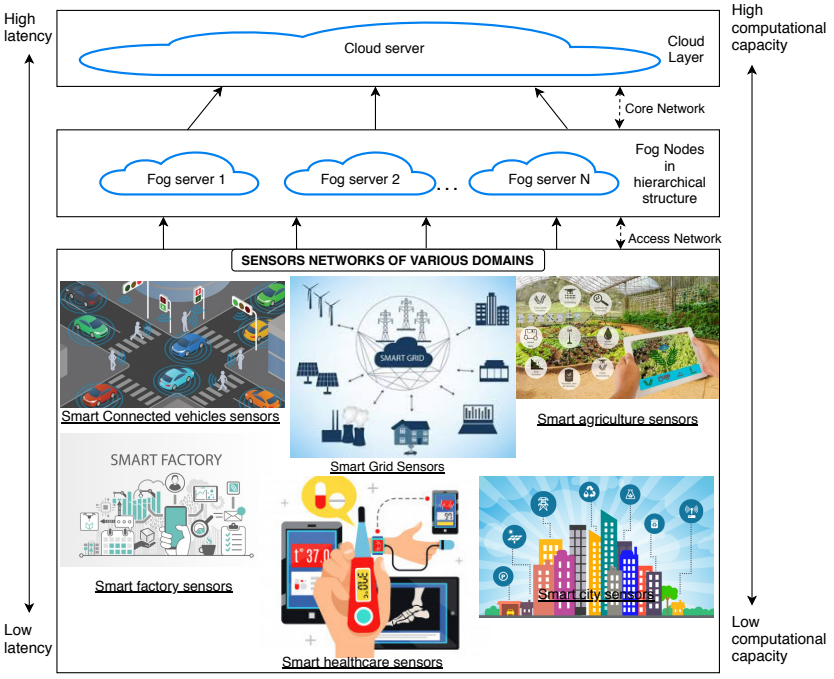

Figure 7.1: Fog Computing high-level architecture and application in various domains.

Table 7.1: Comparison between Cloud Computing and Fog Computing.

Attributes	**Cloud Computing**	**Fog Computing**
Definition	Ondemandcompute,storageandnetworkingservicesoncentralizedCloudserverslocatedfarfromIoTdevicesandusersaccessedviacore network	Compute,storage,and networkingservicesincontinuumbetweenIoTsensornetworksandCloud servers
Applications	Applicationswhicharenotlatencysensitivelikesoftwareservices,platformservices,e-commerce applications,webapplications,andsoforth	IoTapplicationswhicharelatency-sensitivelikemedicalservices,smarthealthcare,smartgrid,smartconnectedvehicles,augmentedreality, andso forth.
Architecture	Centralized	Decentralizedandhierarchical
Programming	Distributedprogramming	Distributedandparallel programming
Storage capacity	Highstoragedataservers	DatahubsnearIoTsensornetworks
Reliability	Lessreliableintermsofconnectivity,performanceandsecurity	Morereliableintermsofconnectivity,performance,and security
Security & Privacy	Low	High.ProvidedonFognetworksthroughBlockchainandothertechnologiesandalongcloud-to-things continuum
Communication mode	IPnetworks	WLAN,WiFi,LAN,WAN,ZigBee,wiredcommunication,cellularnetworks
Computing capacity	Highcomputingcapacity	Moderatecomputingcapacity
Mobility support	Low	High
Geographical distribution	Global,centralized	Lessglobal,geographicallydistributed
Application latency	High	Low

7.2 Fog Computing Based Architecture for Smart Neighborhoods

As shown in Figure 7.2, we propose a distributed Fog Computing architecture for processing smart meter data and to address the challenges mentioned above. We introduce an intermediate layer between smart meter networks and cloud server, which is a hierarchical distributed network of fog nodes attached with fog servers for individual smart neighborhoods. These networks are geographically distributed rather than being centralized. The data from smart meters from one neighborhood is sent to fog network 1. Whereas, the data from smart meters from the second neighborhood is sent to fog network 2. More than one smart neighborhoods are connected to the same fog server. Smart homes in one neighborhood have smart appliances that have sensors which measure the power consumed. These appliances communicate through Bluetooth, Zigbee, Wifi and other short-range protocols with other appliances. Streaming minute-sampled data appliance level data is generated recorded by the smart meters. As clear, real-time analytics of power data is the most important requirement to ensure availability of smart grid services.

A neighborhood will have a number of houses depending upon the processing capacity of the fog network it is attached to. Fog server will perform the following operations:

- Data storage
- Distributed and scalable Fog Computing architecture
- Real-time service
- Resource management, provisioning and control
- Resilient and Reliable Grid operations
- Security and Privacy preserving architecture
- Coordination between Cloud and grid sensor networks

The fog server can be analogous to a cloud server with comparatively less computation and storage capacity that receives data from individual fog networks connected to it. The fog server is a highly virtualized environment where different VMs are meant for executing various fog applications. The fog server also sends relevant data to the centralized cloud server for large scale data processing and service provisioning. It is important to know that Fog Computing will not replace

Cloud Computing altogether; rather, it will act as a vital supplement to the Cloud Computing [**fogcloudsupliment**].

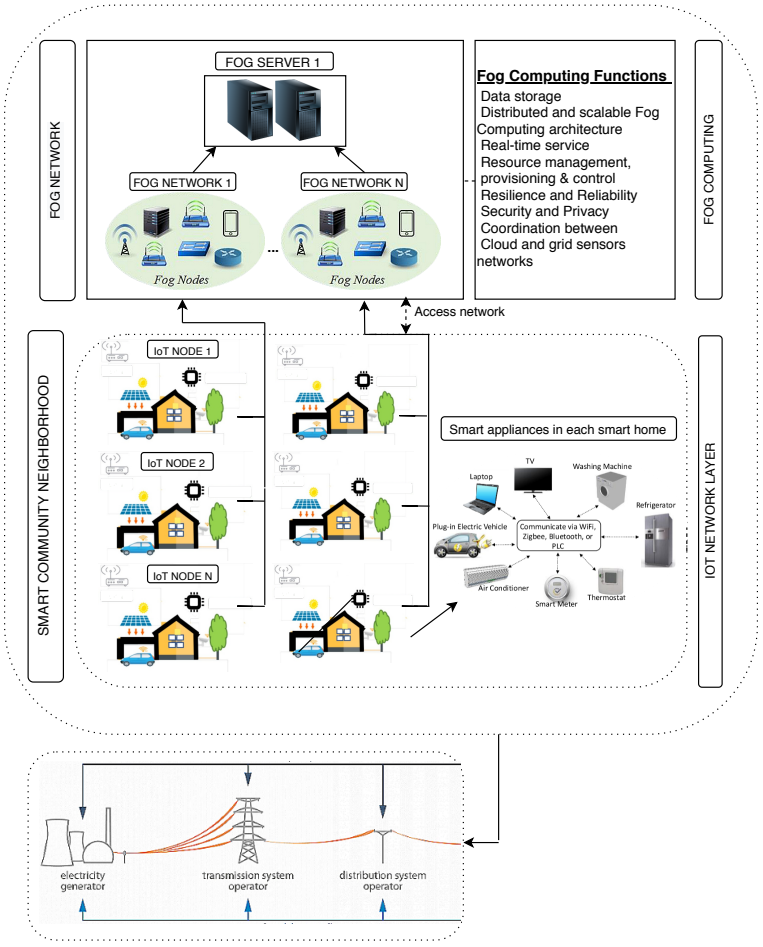

Figure 7.2: Fog Computing architecture for smart neighborhoods.

Chapter 8

Conclusions and Future Directions

The major highlights and takeaways of this dissertation are as follows.

- We proposed a novel Steiner tree algorithm for generating a design of wind farm collector system with minimum cable length. Our algorithm reduces the construction cost and the maintenance cost. Also, our algorithm solves a NP-hard minimum Steiner tree problem, in polynomial time.
- We proposed an Artificial Intelligence based approach for anomaly detection in Smart meter data that identified anomalous patterns based on consumers usage patterns and historical patterns. The approach is fast, scalable and, the parameters are easily interpretable. This approach is first time used on a real-world power dataset and, anomaly labels are created. We released the anomaly labelled dataset for anyone to use and research. Also, we proposed a machine learning based classification approach to classify power anomalies which achieved a G-mean score of 97.3 percent.
- We proposed a prediction and dynamic threshold based approach for real-time anomaly detection in smart meter data to balance power and avoiding any potential hazards. Also we proposed a distributed fog Computing architecture for handling the big data needs which is a better alternative to Cloud Computing.
- We propose an Generative adversarial networks based methodology for anomaly detection with an accuracy of 90.27 percent.

Bibliography

[1] **Abdullah Hamed Al-Badi et al.** "Survey of smart grid concepts and technological demonstrations worldwide emphasizing on the Oman perspective." In: *Applied System Innovation* 3.1 (2020), p. 5.

[2] **Francis Mwasilu et al.** "Electric vehicles and smart grid interaction: A review on vehicle to grid and renewable energy sources integration." In: *Renewable and sustainable energy reviews* 34 (2014), pp. 501–516.

[3] **Pierluigi Siano**. "Demand response and smart grids—A survey." In: *Renewable and Sustainable Energy Reviews* 30 (2014), pp. 461–478. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2013.10.022. URL: http://www.sciencedirect.com/science/article/pii/S1364032113007211.